applied sciences

MDPI

Article

NowDeepN: An Ensemble of Deep Learning Models for Weather Nowcasting Based on Radar Products' Values Prediction

Gabriela Czibula [1,*,†], Andrei Mihai [1,†] and Eugen Mihuleţ [2]

[1] Department of Computer Science, Babeş-Bolyai University, 400084 Cluj-Napoca, Romania; mihai.andrei@cs.ubbcluj.ro
[2] Romanian National Meteorological Administration, 013686 Bucharest, Romania; eugen.mihulet@meteoromania.ro
* Correspondence:gabis@cs.ubbcluj.ro; Tel.: +40-264-405327
† These authors contributed equally to this work.

Abstract: One of the hottest topics in today's meteorological research is *weather nowcasting*, which is the weather forecast for a short time period such as one to six hours. Radar is an important data source used by operational meteorologists for issuing nowcasting warnings. With the main goal of helping meteorologists in analysing radar data for issuing nowcasting warnings, we propose *NowDeepN*, a supervised learning based regression model which uses an ensemble of *deep artificial neural networks* for predicting the values for radar products at a certain time moment. The values predicted by *NowDeepN* may be used by meteorologists in estimating the future development of potential severe phenomena and would replace the time consuming process of extrapolating the radar echoes. *NowDeepN* is intended to be a proof of concept for the effectiveness of learning from radar data relevant patterns that would be useful for predicting future values for radar products based on their historical values. For assessing the performance of *NowDeepN*, a set of experiments on real radar data provided by the Romanian National Meteorological Administration is conducted. The impact of a *data cleaning* step introduced for correcting the erroneous radar products' values is investigated both from the computational and meteorological perspectives. The experimental results also indicate the relevance of the features considered in the supervised learning task, highlighting that the radar products' values at a certain geographical location at a time moment may be predicted from the products' values from a neighboring area of that location at previous time moments. An overall *Normalized Root Mean Squared Error* less than 4% was obtained for *NowDeepN* on the cleaned radar data. Compared to similar related work from the nowcasting literature, *NowDeepN* outperforms several approaches and this emphasizes the performance of our proposal.

Keywords: weather nowcasting; machine learning; deep neural networks; autoencoders; Principal Component Analysis

Citation: Czibula, G.; Mihai, A.; Mihuleţ, E. *NowDeepN*: An Ensemble of Deep Learning Models for Weather Nowcasting Based on Radar Products' Values Prediction. *Appl. Sci.* **2021**, *11*, 125. https://dx.doi.org/10.3390/app11010125

Received: 28 October 2020
Accepted: 18 December 2020
Published: 24 December 2020

1. Introduction

Weather nowcasting [1,2] refers to short-time weather prediction, namely weather analysis and forecast for the next 0 to 6 h. Nowadays, the role of nowcasting in crisis management and risk prevention is increasing, as more and more severe weather events are expected [3]. Large volumes of meteorological data, including radar, satellite and weather stations' observations, are held by meteorological institutes and available for analysis. Radars and weather stations are constantly collecting real-time data, while data about cloud patterns, winds, temperature are continuously gathered by weather-focused satellites. Thus, there is large amount of meteorological related data available to be analyzed using *machine learning* (ML) based algorithms for improving the accuracy of short-term weather-prediction techniques.

The World Meteorological Organization (WMO) [4] mentions that "nowcasting plays an increasing role in crisis management and risk prevention, but its realization is a highly complex task", the highest difficulties being related to the small-scale nature of convective weather phenomena. This implies that the Numerical Weather Prediction approach (NWP) [5] is not feasible and that the forecast has to rely mainly on the extrapolation of known weather parameters. As meteorological institutes worldwide expect climate changes including extreme rain phenomena [3], there is an increasing need for accurate and early warning of severe weather events. Considering the increased number and intensity of severe meteorological phenomena, predicting them in due time to avoid disasters becomes highly demanding for meteorologists.

Mainly due to the extremely large volume of data that has to be analyzed in a short period of time, issuing a nowcasting warning is a complex and difficult task. For operative meteorologists it is difficult to issue nowcasting warnings, as there is a large volume of meteorological data (radar, satellite, or other weather stations' observations) that has to be analyzed in order to take an appropriate decision. Besides, given the stochastic and chaotic character of the atmosphere, the evolution of certain weather phenomena are difficult to predict by human experts. Thus, *machine learning* (ML) and *deep learning* [1,2] techniques are useful for assisting meteorologists in the decision-making process, offering solutions for nowcasting by learning relevant patterns from large amount of weather data. Most of the existing operational and semi-operational methods for nowcasting are using the extrapolation of radar data and algorithms mainly based on cell tracking. Existing nowcasting techniques use various data sources which may be relevant for accurate nowcasting, such as: meteorological data (radar, satellite, meteorological observations) and geographical data (elevation, exposure, vegetation, hydrological features, anthropic features).

In the current study we used real radar data provided by one of the WSR-98D weather radars [6] of the Romanian National Weather Administration. Given its capability to determine the location, size, direction and speed of water droplets, the weather radar is an essential tool used by meteorologists for nowcasting. For this type of forecast, fast decisions are imposed, so with the aim of facilitating the analysis in an operative environment, the data retrieved by the radar is supplied to the meteorologist in the form of coloured maps, which are easy to assess on a brief overview. On such a map, each pixel corresponds to a geographical location and its colour represents a certain value of the displayed product. In a short period of time, commonly under one minute, the meteorologist can locate potential dangerous storm cells, analyse their vertical structure, relative speed and direction, the dimension of hail, top of the clouds, and other relevant parameters. At this point, the operator should appreciate the current phase of the storm and by extrapolating those values in time, usually up to one hour, predicts the future development (intensity and area affected) of the storm. Although extrapolating the radar echoes is one of the main techniques used in nowcasting, it presents weaknesses in terms of processing times and precision, all related to the skills and experience of the meteorologist.

Most of the currently existing operational and semi-operational methods for nowcasting use the extrapolation of radar data and algorithms mainly based on cell tracking [7–9]. However, an important limitation of existing centroid cell-tracking algorithms lies in the detection of storms segments having irregular shapes or with variable wind speeds, resulting in identification and tracking errors.

With the goal of helping meteorologists in analysing radar data for issuing nowcasting warnings, we are introducing in this paper a supervised learning model *NowDeepN* based on an ensemble of *deep neural network* (DNN) regressors for predicting the values for radar products which may be used for weather nowcasting. As an additional goal, we aim to empirically validate the hypothesis that similar values for the radar products at a given time moment for a certain geographical region are encoded in similar neighborhoods of that region at previous time moments. The values predicted by *NowDeepN* for the radar products at certain time moments may be used by the meteorologist in estimating the future

development of potential severe phenomena and thus *NowDeepN* would replace the time consuming process for the operational meteorologists of extrapolating the radar echoes.

As a proof of concept, *NowDeepN* is proposed for learning to approximate a function between past values of the radar products extracted from radar observations and their future values. Experiments will be performed on real radar data provided by the Romanian National Meteorological Administration and collected on the Central Transylvania region. Our experimental goal is to obtain an empirical evidence that in both normal and severe weather conditions the values for a radar product at a given moment in a certain location are predictable from the values of the neighboring locations from previous time moments. If this stands, the study may be further improved and extended on a larger scale. In addition, we are proposing a *data cleaning* step useful for correcting the erroneous input radar data. To the best of our knowledge it is the first time an approach such as *NowDeepN* is introduced in the nowcasting literature.

To summarize the contribution of this paper, our goal is to answer the following research questions:

RQ1 Are *deep neural networks* able to predict the values for a radar product at a given moment in a certain geographical location from the values of its neighboring locations from previous time moments? To what extent this holds both for both normal and severe weather conditions? For answering this research question, we are introducing a supervised learning model *NowDeepN* consisting of an ensemble of DNN-based regressors.

RQ2 How does the data cleaning step introduced for correcting the erroneous input data impact the predictive performance of the *NowDeepN* model? In addition, to what extent is the proposed data cleaning step correlated with the meteorological perspective?

RQ3 How relevant are the features considered in the supervised learning task? More specifically, are the radar products' values from the neighboring area of a certain geographical location l at time $t-1$ relevant for predicting the radar products' values on location l at time t?

The remainder of the paper is structured as follows. The problem of weather nowcasting, its importance and main challenges are discussed in Section 2. Section 3 discusses about the fundamental concepts regarding the used machine learning models, whilst a literature review on supervised learning based weather nowcasting is presented in Section 4. Section 5 introduces our data model and the *NowDeepN* hybrid model for predicting the values for radar meteorological products using *deep neural networks*. Section 6 presents the experimental setting and results, while an analysis of the obtained results and their comparison to related work is provided in Section 7. The conclusions of our paper and future improvements and enhancements of *NowDeepN* are outlined in Section 8.

2. Weather Nowcasting

The World Meteorological Organization (WMO) [4] mentions that "nowcasting plays an increasing role in crisis management and risk prevention, but its realization is a highly complex task", the highest difficulties being related to the small-scale nature of convective weather phenomena. This implies that the Numerical Weather Prediction approach is not feasible and that the forecast has to rely mainly on the extrapolation of known weather parameters. In this context, there is a high need for automated tools and systems supporting the nowcasting meteorologist, so significant research and development have been carried out on the topic of nowcasting. In spite of those efforts, both the operative personnel in nowcasting and the beneficiaries (population, relevant public institutions) are in demand of even more efficient products and services concerning the weather forecast itself and the alerting system.

Most of the currently existing operational and semi-operational methods for nowcasting are using the extrapolation of radar data and algorithms mainly based on cell tracking.

Dixon and Wiener developed a real-time cell tracker named TITAN [10] which is useful for single cells. A centroid method is used in Reference [10] to identify storm entities in consecutive radar scans and estimate future movement of the storm centroid by minimizing a cost function. SCIT was proposed by Johnson et al. [11] as a cell-tracking algorithm with a more more complex cell detection method. SCIT used reflectivity thresholds and a distance function between cells for tracking and estimating future positions. An operational nowcasting tool called TRT was developed by Hering et al. [7] and used by Meteo Swiss. TRT is a centroid cell-tracking method using radar data and, in addition to previously mentioned methods, a multiple sensor system. Jung and Lee [12] introduced a cell-tracking algorithm using fuzzy logic based on a large set of historic radar data, with the goal of minimizing errors induced by single features. Germany's National Meteorological Service DWD uses a nowcasting system called NowCastMIX developed by James et al. [8]. NowCastMIX employes remote and ground observations analyzed using fuzzy logic rules. AROME [13] is a small scale numerical prediction model, which is used by Meteo-France since 2008. Measurements used by AROME include measurements of the precipitation systems, of low level humidity, low-level wind, upper level wind and temperature and it provides a forecast for up to 30 h. AROME-NWC [9] was developed in 2015 on top of AROME, for nowcasting in the range of 0–6 h. The INCA system [14] was developed specifically for mountainous regions with the goal of forecasting for precipitation amounts, temperature, wind and convective parameters.

An important limitation of existing centroid cell-tracking algorithms lies in the detection of storms segments having irregular shapes or with variable wind speeds, resulting in identification and tracking errors. Errors also occur when storm cells are clustered together and the algorithm detects them as a single larger cell. In other cases, a single storm cell is detected as two or more cells at the same horizontal location but on different altitude levels. Given the high spatial and temporal resolution, NowCastMIX [8] estimates of severe storms change quickly with the assimilation of new observations, leading to difficulties in reasoning for meteorologists. It is also prone to an overestimation of the likelihood of severe convection. According to Reference [14], the INCA system provides high accuracy for temperature but comparatively low for wind and precipitation given the relatively low distribution of relevant stations in mountainous regions. The systems described in References [8,9,14], which are considered to be some of the most performant automated nowcasting systems at the present time are highly customized for their respective location and infrastructure, thus their performance and adaptability to other contexts is unclear.

Several industrial players have also shown an interest in the problem of weather forecasting and have developed various solutions in this direction: IBM Deep Thunder [15] aims to provide high-resolution weather prediction for a variety of applications, Panasonic Global 4D Weather [16] is a weather forecasting platform that uses Panasonic patented atmospheric TADMAR sensor collected data, ClimaCell company [17] offers weather forecasting solutions tailored for various weather-sensitive industries while TempoQuest [18] is a company putting on the market a proprietary forecasting software for commercial users and government agencies.

3. Machine Learning Models Used

This section reviews the fundamental concepts regarding the machine learning models used in this paper: *deep neural networks*, *autoencoders* and *t-Distributed Stochastic Neighbor Embedding*.

Supervised learning is a subfield of machine learning, dealing with the task of approximating a mapping from some input domain to some output domain based on input-output example pairs. The data set consisting of the pre-existing examples of input-output pairs is called the *training data set*. A *supervised learning algorithm* generalizes the training data, producing a function that, given an input outside of the training data set, can return a close enough approximation of the correct output. What can be considered a "good enough approximation" is dependent on the specific problem. The vast quantity of data available

nowadays as well as the increasing power of computation make the supervised learning methods an extremely useful tool that can be used in a wide range of domains.

3.1. Deep Neural Networks

Neural network learning methods provide a robust approach to approximating real-valued, discrete-valued or vector-valued target functions [19]. As a biological motivation, neural networks have been modeled to be similar to learning systems that we can find in humans and animals, namely complex networks of neurons. This morphology has been adopted in computer science, by building densely interconnected systems that have as building blocks basic units, that take as input a series of real-valued numbers and produce a single real-valued output [19]. These basic units are called artificial neurons. Neural networks are suited for problems that deal with noisy, complex data, such as camera, microphone or sensor data. Their success is due to their similarity to effective biological systems, that are able to generalize and associate data that has not been explicitly trained upon during the training phase, and correlate that data to a class where it belongs. Each neuron of the network has an array of parameters, based on which it processes the input data, called weights. The weights are adjusted during the training phase, based on the error of the network. The error represents the difference between the correct output and the network output. The learning algorithm used for adjusting the weights based on the error is the *backpropagation* algorithm.

Unlike classical neural networks, *deep neural networks* (DNNs) [20] contain multiple hidden layers and have a large number of parameters which makes them able to express complicated target functions, that is, complex mappings between their input and outputs [21]. Nowadays, DNNs are powerful models in the *machine learning* literature applied for complex classification and regression problems from various domains. Machine learning models, including deep ones, have been successfully applied for developing forecasting models such as for bank failures [22], prediction markets [23] or gambling [24]. Due to their complexity, large networks are slow to use and are prone to *overfitting*, which is a serious problem in DNNs. Overfitting is a major problem for supervised learning models, in which the model learns "by heart" the training data, but it does not have the capability to generalize well on the testing data. An overfit model is discovered through a very good performance on the training data, but a much lower performance on the testing set. A possible cause for overfitting in DNNs is the limited training data, as in such cases the relationships learned by the networks may be the result of sampling noise. Thus, these complex relationships will exist in the training data but not in real test data [21]. There are various methods for addressing overfitting and reducing it, such as—(1) stopping the training when the performance on a validation set starts decreasing; (2) introducing weight penalties through regularization techniques soft weight sharing [25]; (3) applying cross-validation; (4) extending the data set to include more training examples; and (5) *dropout* by randomly dropping neurons and their connections during training.

Regularization stands for an ensemble of techniques that have as purpose the simplification of the model, in order to avoid overfitting. Dropout is the regularization technique applied for neural networks. This process consists in deactivating some neurons during the training process, forcing the network to achieve the result by using a reduced (and simpler) neuron configuration. During prediction phase, the neurons will be reactivated. The selection of which neurons to keep active is done by a probability p, chosen arbitrarily, and dropped out by a probability of 1-p [21].

3.2. Autoencoders and Principal Component Analysis

Autoencoders (AEs) [20] are deep feed forward neural networks which aim to learn to reconstruct the input, being known in the machine learning literature as *self-supervised* learning systems. An AE has two main components: an *encoder* and a *decoder*. Assuming that the input space is $\mathbb{R}^n$, the encoder part learns a mapping $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, while the decoder learns the function $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$. If $m < n$, the network learns to compress

the input data into a lower dimensional latent space and to reconstruct it based on the latent representation, by learning the essential characteristics of the data. For avoiding the overfitting symptom (i.e., simply copying the input to the output), L1 regularization is applied on the encoded state and the model is called a *sparse* one. Autoencoders have been successfully applied in various tasks ranging from image analysis [26] and speech processing [27] to protein analysis and classification [28,29] .

Principal Component Analysis (PCA) is a dimensionality reduction statistical technique heavily used in the ML field for tasks such as descriptive data analysis, data visualization or data preprocessing. It can be seen as an unsupervised learning technique that learns to represent the data in a new, lower-dimensional, space. Given a data set with n samples $(x_1, \ldots, x_n)$ and each sample having p attributes $(a_1, \ldots, a_p)$, the PCA algorithm will search for linear combinations of the variables (i.e., $\sum_{i=1}^{p} a_i \cdot c_i$ with $c_1, \ldots, c_p$ constants) such that they are linearly uncorrelated and that the first linear combination has the largest possible variance, the second has the largest possible variance while being orthogonal on the first linear combination and so on. These linear combinations are called *principal components*. The principal components can be found by computing the eigenvectors and eigenvalues of the covariance matrix of the data set, where the constants of each linear combination are given by the eigenvectors and the first linear combination is the one given by the eigenvector associated with the biggest eigenvalue, and so on [30].

4. Literature Review on Supervised Learning Based Weather Nowcasting

The literature contains various machine learning-based approaches for weather nowcasting. Relevant results obtained recently in predicting short-term weather are summarized in the following and the limitations of the existing solutions are emphasized.

Han et al. [31] use Support Vector Machines (SVM) are trained on box-based features in order to classify whether or not a radar echo >35 dBZ will appear on the radar within 30 min. The approach uses both temporal and spatial features, derived from vertical wind, and perturbation temperature. Greedy feature selection was used in order to finally select these features. The obtained results were around 0.61 Probability Of Detection (POD), 0.52 False Alarm Ratio (FAR), 0.36 Critical Success Index (CSI), which outperformed the Logistic Regression, J48, Adaboost and Maxent approaches compared against on the same data set.

Beusch et al. [32] present the COALITION-3 ("Context and Scale Oriented Thunderstorm Satellite Predictors Development") algorithm developed by MeteoSwiss. Its goal is to identify, track and nowcast the position and development of storms in a robust manner. In order to do this, it employs optical flow methods in combination with winds predicted by COSMO . In order to separate the movement of the storm from its temporal evolution, Lagrangian translation is used. Following this, the algorithm estimates future intensification and quantifies the probability and severity of the different risks associated with thunderstorms. This information is extracted by applying machine learning techniques to a data archive containing observations from the MeteoSwiss dual polarized Doppler radar system and information from other available systems.

Shi et al. [33] introduce an extension of the Long-Short Term Memory Network (LSTM), named ConvLSTM. Through experiments done on Moving MNIST and Radar Echo Dataset, the authors proved their method suitable for spatiotemporal data, having all the perks which come with a simple LSTM, preserving spatiotemporal features due to the inherited convolutional structure. Besides the above mentioned aspect other advantages of ConvLSTM over a basic LSTM are: transitions input-to-state and state-to-state are made in a convolutional manner, deeper models can produce better results with a smaller number of parameters. Their proposed architecture contains 2 networks, an encoder composed of 2 ConvLSTM layers and a forecasting network containing the same number of ConvLSTM layers and an additional 1x1 convolutional layer to generate final predictions.

Kim et al. [34] have also proposed a ConvLSTM-based model for precipitation prediction, but they used three-dimensional and four-channel data unlike Shi et al. [33], who used

three-dimensional and only one-channel data. The four channels correspond to four altitudes. The proposed model, called DeepRain, predicts, based on radar reflectivity data, the amount of rainfall on a large scale. The experimental evaluation has proved a decrease of root mean square error (RMSE) with 23% when compared to linear regression and also a superior performance as compared to a fully connected LSTM. For rainfall prediction a RMSE value of 11.31 has been obtained on the test set.

Heye et al. [35] present a practical solution for leveraging the precipitation nowcasting problem starting from how the data is stored and preprocessed, the deep learning model used and up to the necessary frameworks and hardware. They used the ConvLSTM with peepholes cells described by Shi et al. [33] in an architecture inspired from those used for machine translation. The model is composed of an encoder and a decoder each made out of four ConvLSTM layers. They experimented with both taking the last step of the encoder and using attention for transferring information to the decoder, with the former producing better results in terms of Probability of Detection and Critical Success Rate and worse for the False Alarm Rate. The decoder uses both the encoded actual reflectivity and the predicted reflectivity. For accelerating learning, during training the ground truths are fed back into the decoder, while for evaluation the prior predictions are used. Additionally they noticed that using as start symbol a tensor of ones which represents high reflectivity when data is scaled to [0, 1] improves Probability of Detection and Critical Success Rate.

Shi et al. [36] proposed a benchmark for the nowcasting prediction problem and a new model. The benchmark includes a new data set, 2 testing protocols and 2 measurements for training, Balanced Mean Squared Error and Balanced Mean Absolute Error. Those measurements were necessary due to an imbalancement between the small amount of times where potentially dangerous events occured and normal amount of rainfall. The proposed model, Trajectory Gated Recurrent Unit (TrajGRU), deals better than Convolutional GRU, with the representation of location variant relationships. TrajGRU allows aggregation of the state along a learned trajectories. The architecture consist of an encoder and a forecaster part, inserting between RNN layers downsampling or upsampling operations depending upon region of the architecture. Their proposal show to be flexible and superior than other methods. The experiments done on Moving MNIST and precipitation nowcasting HKO-7 dataset emphasize the ability of the model to capture the spatiotemporal correlation.

Narejo and Pasero [37] have proposed a hybrid model combining Deep Belief Networks (DBN) with Restricted Boltzmann Machine (RBM) to predict different parameters of weather—air temperature, pressure and relative humidity—at a local level (i.e., restricted to a particular geographical area). Initially, the RBMs are trained unsupervisedly. Subsequently, the already trained RBMs are stacked to create a DBN. The DBN is supervisedly trained so as to predict the weather parameters.

Sprenger et al. [38] approached foehn prediction using AdaBoost machine learning algorithm. The authors motivate the choice of the model through the reasonableness of the balance between the predictive power, the computational speed and the interpretability of the results. Being trained with three years of hourly simulations data and using a modified decision stumps as weaker learners, the model achieved, on a validation data set, 0.88 sensitivity (or probability of detection) and 0.29 probability of false detection.

Yan Ji [39] approached the problem of short-term precipitation prediction from radar observations, using *artificial neural networks*. The nowcasting of the rain intensity was carried out on radar raw data and rain gauge data collected from China from 2010 to 2012. The reflectivity values were extracted from the raw data and were interpolated a 3D rectangular lattice grid of 1km*1km in horizontal direction at the height of 1.5 and 3 km [39]. The collected data set was afterwards used for training the predictive model. A *root mean squared error* less than 5 (a minimum of 0.97 and a maximum of 4.7) was obtained. The experiments have shown a correlation coefficient R in the radar-rainfall estimation more than 0.6 and indicate that the accuracy rate of 36mins forecast is higher than 50% [39].

In a recent approach, Tran and Song [40] proposed a new loss function for the convolutional neural network (CNN) based models used for weather nowcasting. Using a

computer vision perspective, the authors used Image Quality Assessment Metrics as loss functions in training, finding that the Structural Similarity function performed better than both MSE and MAE , especially by improving the quality of the images (i.e., the output images were much less blurry). However, the best performance was achieved by combining the Structural Similarity with the MAE and MSE loss functions.

Han et al. [41] proposed a CNN model for convective storms nowcasting based on radar data. The proposed model predicted whether radar echo values will be higher than 35 dBZ in 30 min, thus modelling the problem as a classification problem. The authors modeled the input radar data as multi-channel 3D images and proposed a CNN model that performs cross-channel 3-D convolution. In their model, the output is also an 3D-image, where each point of the image is a 0 if radar echo is predicted to be $\leq$ 35dBZ in 30 min and 1 otherwise. A Critical Success Index (CSI) score of 0.44 was obtained.

Yan et al. [42] proposed a model for precipitation nowcasting employing a convolutional architecture using multihead attention and residual connections (MAR-CNN). The data fed into the network consists of radar reflectivity images on three elevation levels and other numerical features, such as cloud movement speed. In order to deal with the unbalanced classes of precipitations, extreme meteorological events have been oversampled. The proposed model outperformed several deep learning baselines obtained by using only several of the MAR-CNN components—a dual channel convolutional attention module, a dual channel convolutional model, a single channel convolutional model, as well as a GBDT and an SVM.

4.1. Limitations of Existing Approaches

Han et al. [31] highlight that support vector machines require feature reductions in order to be more accurate, so they might not be able to make use of all the information that a data set could provide. They also argue for the importance of collecting more data in order to train the machine learning models better. Another concern is that of feature selection: while other informative features do exist in the literature, it is not always straightforward to get them to a form usable for machine learning.

As most works [31,32] show, each system is mostly tested with some form of local real world data. This does not allow to accurately extrapolate a system's ability to adapt to other locations and their specific meteorological events and trends.

A general concern which later on can morph into a limitation is the possible data imbalance. The performance of the proposed methods can be diminished by the small number of risky weather conditions compared with normal rain [35,36]. The results from Reference [33] might not reflect the overall competence of ConvLSTM due to the fact that Shi et al. used a rather small data set for testing and and a low threshold for rain-rate. Another limitation mentioned [35] is the fact that predictions tend to lower values of precipitations in time due to previously less confident predictions being fed back into the recurrent network. The architecture proposed by Reference [35] is limited to the train data and does not learn from new data. A concern experienced by others when using the proposed TrajGRU cell [36] is the speed. The implementation of this operator turns out to be slower than a simple ConvGRU.

A common limitation of the solutions proposed by Reference [34] and Reference [37] is compromising the interpretability of the prediction results. The boosting-based solution [38] alleviate the issue of interpretability, but Dietterich has highlighted in one of his studies [43] the sensitivity to outliers as a particular disadvantage of boosting. Outliers can negatively affect the final predictions since they might be excessively weighted during the boosting steps. An additional limitation of most of the existing solutions is that they do not combine multiple data sources thus being deprived by an expected enhancement of the predictive capability [44].

5. Methodology

We further introduce our *NowDeepN* approach for weather nowcasting using *deep neural networks*. We start by describing in Section 5.1 the raw radar data used in our experiments. Section 5.2 introduces a theoretical model on which *NowDeepN* is based on, then Section 5.3 presents our proposal. The section ends with the evaluation measures we will use for assessing the predictive performance of *NowDeepN*.

5.1. Radar Data

The experiments in the current study were conducted on real radar data provided by one of the WSR-98D weather radars [6] of the Romanian National Weather Administration. The WSR-98D is a type of doppler radar used by meteorological administrations for weather surveillance, capable of remote detection of water droplets in the atmosphere, that is, clouds and precipitations, retrieving data on their location, size and motion. The WSR-98D scans the volume of air above an area over 70,000 square kilometers, and about every 6 min a complete set of about 30 base and derived products for 7 different elevations is being collected. The base products are *particle reflectivity* (R), providing information on particle location, size and type, and *particle velocity* (V), supplying information on particle motion, that is, direction and speed relative to the radar. Both products are available for several elevation angles of the radar antenna, and for each time step a set of seven data products, R01-R07 and V01-V07, is delivered, each of them corresponding to a certain tilt of the antenna. Among the derived products, of particular interest for this study is VIL (vertically integrated liquid), an estimation of the total mass of precipitation above a certain unit of area. The data in the NEXRAD Level III files is stored in a gridded format, each point of the grid corresponding to a geographical location and containing the value of a certain product at the respective time frame. In the data grid provided by the WSR-98D radar, the OX axis contains the longitude values, while the OY axis contains the latitude values.

5.2. Data Model

As shown in Section 1, the raw data provided by the radar scans during one day (24h) on a certain geographic region was exported in the form of a sequence of matrices of mxn dimensional matrices, one matrix corresponding to a certain time moment t and a certain meteorological product p (i.e., each element from the matrix represents the value for the product p at a certain location from the map). As a set $Prod$ of multiple meteorological products are provided by the radar, the radar data collected at a time t may be visualized as a 3D data grid in which the OZ axis corresponds to the radar products.

For instance, Figure 1 depicts a sample 3D grid with $m = 2$ rows, $n = 2$ columns and three products ($Prod = \{R01, R02, R03\}$) recorded at a certain time stamp t. In the figure, the values for $R01$ are in the front matrix, $R02$ values are in the middle one and $R03$ in the matrix behind.

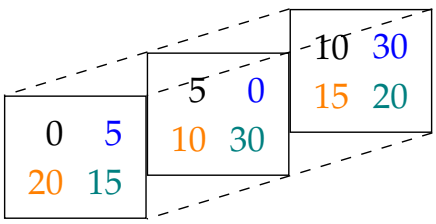

Figure 1. A sample 3D data grid.

During one day, a sequence of 3D data grids (as shown in Figure 1) corresponding to various time stamps is provided by the radar. Assuming that the radar records data every 6 min, 240 3D data grids are provided. For a certain location (i,j) on the map, a time moment t and a set $Prod$ of radar products, we are denoting by $V_t(i, j, l, Prod)$ the vector representing the linearized 3D data subgrid containing the radar products' values for a neighboring area of a certain length l surrounding the point (i,j), at time moment t.

As an example, let us consider a 5×5 dimensional data grid, the set $Prod = \{R01, R02\}$ of radar products, a time stamp t, the location (3, 3) ($i = 3, j = 3$) and a length l of 3 for the neighboring subgrid. Figure 2 depicts the 3D data grid containing values for $R01$ (front) and $R02$ (behind) for each cell from the data grid surrounding the point (3, 3), at time stamp t. The point of interest as well as the 3D data subgrid of length 3 surrounding the point (3, 3) are highlighted.

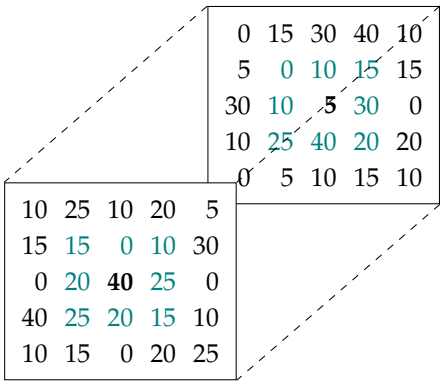

Figure 2. The 3D data grid at time stamp.

The linearized 3D data subgrid (highlighted in Figure 2) is the 18-dimensional vector $V_t(3, 3, 3, Prod) = (15, 0, 0, 10, 10, 15, 20, 10, 40, 5, 25, 30, 25, 25, 20, 40, 15, 20)$.

5.3. NowDeepN Approach

The regression problem we are focusing on is the following—to predict a sequence of values for a set *Prod* of radar products at a given time moment t on a certain location (i,j) on the map, considering the values for the neighboring locations of (i,j) at time moment $t-1$. *NowDeepN* approach consists of three main stages which will be further detailed: *data collection and cleaning*, *training* (building the learning model) and *testing*. *NowDeepN* uses an ensemble of DNNs for learning to predict the values of the radar products from the set *Prod* based on their historical values. The ensemble consists of np DNNs ($np = |Prod|$), one DNN for each radar product. We started from the intuition that using one network for each product would be more effective than using only one network for predicting all np values, as the mapping learned by the model should be specific to each radar product. Thus, we consider that the effectiveness of the learning process will be increased by using a DNN for each radar product and this will be empirically sustained by the experimental results (Section 6).

5.3.1. Data Collection and Cleaning

We mention that in the data set used in our case study, values for R and V products are available for only six elevations (i.e., R01-R04 and R06-R07, V01-V04 and V06-V07). The other three elevations delivered by the radar are missing since they are not regularly used in operational services, thus they are not stored in the same format as the rest of the elevations. The data gathered by the radar and exported as shown in Section 5.1 contains a special value that represents "No Data". This value is usually represented by -999 but we decided to replace it with 0 as in most cases this value refers to air particles with 0 reflectivity (i.e., no significant water droplets). "No data" may also represent air volumes which have returned no signal, for example if a sector with high reflectivity is between the radar and the respective location. In this case, replacing it with 0 is also correct, since the entire region is obturated and the data is not relevant for the learning process [45]. The radar data is also prone to different type of errors, meteorological and technical, which implicitly are to be found in the output data matrix. Meteorological errors (e.g., the underestimation of a particle's reflectivity) are difficult to identify and eliminate, but some errors occurring during the data conversion have

been identified for V. For instance, the product V should only contain values from -33 to 33 but we found *invalid* values which are outside the range $[-33, 33]$, such as -100. From a meteorological point of view, those erroneous values correspond to radar uncertainties in evaluating the direction and/or the speed of the particle, and are not taken into account in operative service since they are punctiform values and are irrelevant to the characteristics of a region.

We note that the cleaning process which is further described will be applied for the V values at each degree of elevation (i.e., V01-V04 and V06-V07). Thus, when we are using V in the following, we refer to the V value at a certain degree of elevation.

For reducing the noise that the invalid values of V represent, a *data cleaning* step is proposed. The underlying idea behind the cleaning step is to replace the invalid values of V on a certain point (i, j) with the weighted average of the valid V values from a neighborhood of length 13 surrounding the point. The weight associated to a certain neighbor of the point is inverse proportional to the Euclidian distance between the neighbor and the point, such that the closest neighbors' values have more importance in estimating the value of point. The reason for this cleaning step is that, from a meteorological viewpoint, V determines the direction and speed of air volumes, thus indicating neighbours that are more relevant for future value of points. The length 13 surrounding the point represents about 5 km in the physical world, a distance which commonly determines small gradients of the meteorological parameters.

Let us consider that (i, j) is the point having an erroneous value for V (e.g., -100) and this value has to be replaced with an approximation of its real value. We are denoting by $V(x, y)$ the value of the product V for the point (x,y) and by $\mathcal{N}_l(i, j)$ the 2D data subgrid representing the neighborhood of length l surrounding (i, j). For instance, the neighborhood $\mathcal{N}_3(3, 3)$ of length 3 surrounding the point (3, 3) from the data matrix from Figure 3 is depicted in Figure 4.

0	15	30	40	10
5	−66	10	30	15
30	10	−100	15	0
10	25	−100	20	20
0	5	10	15	10

Figure 3. The sample 2D data grid. The values for the product V are displayed.

−66	10	30
10	−100	15
25	−100	20

Figure 4. The 2D data subgrid representing $\mathcal{N}_3(3, 3)$.

For a certain point $(x,y) \in \mathcal{N}_l(i, j)$, $(x, y) \neq (i, j)$, we denote by $sim_{ij}(x, y) = \frac{1}{\sqrt{(x-i)^2+(y-j)^2}}$ the "similarity" between (i,j) and its neighbor (x,y). Certainly, the data points closer to (i,j) have a higher similarity degree and their value is more relevant in the cleaning process. Thus, the value $V(i, j)$ will be approximated with the weighted average of its valid neighbors, as shown in Formula (1)

$$V(i,j) = \sum_{\substack{(x,y)\in\mathcal{N}_l(i,j)\\(x,y)\ valid}} (w_{ij}(x,y)\cdot V(x,y)), \tag{1}$$

where $w_{ij}(x,y)$ represent the weight of point (x,y) and is computed as shown in Formula (2) by normalizing the similarity values $sim_{ij}(x,y)$ such that $\sum_{\substack{(x,y)\in\mathcal{N}_l(i,j)\\(x,y)valid}} w_{ij}(x,y) = 1$. We note that this normalization assures that the approximated V values represent valid ones, that is, ranging in the interval $[-33, 33]$.

$$w_{ij}(x,y) = \frac{sim_{ij}(x,y)}{\sum_{\substack{(x',y')\in\mathcal{N}_l(i,j)\\(x',y')\ valid}} sim_{ij}(x',y')}. \tag{2}$$

As previously mentioned, in the cleaning process we considered a length $l = 13$ for the neighborhood. But, if all data points from the neighborhood of length l ($\mathcal{N}_l(i,j)$) are invalid, we are incrementally increasing l until the neighboring area will contain at least one valid point. In this case, Formula (1) is applied again using the new length l for estimating the value $V(i,j)$.

After the data was cleaned as previously shown, the data set is prepared for further training the *NowDeepN* regressor, using the representation described in Section 5.2. We denote, in the following, by $Prod = \{p_1, p_2, \ldots, p_{np}\}$ the set of radar products we are using in our approach. The radar data set cleaned as previously described is split into np subsets $D_k,\ 1 \le k \le np$, a data set corresponding to each radar product. A DNN will be afterwards trained on each D_k, for learning to predict the value of the radar product p_k at a time moment t on a certain geographical location l, based on the radar products' values from the neighborhood of l at time $t-1$.

A training example from a data set $D_k,\ 1 \le k \le np$ is in the form $< x_k, y_k >$, where:

- an instance x_k is the vector $V_{t-1}(i,j,l,Prod)$ (see Section 5.2) representing the linearized 3D data grid expressing the neighborhood of length l surrounding a point (*i*,*j*) at a time moment $t-1$;
- the label y_k of the instance x is the value for the radar product p_k for the location (*i*,*j*) from the map and time moment t (the time moment following the timestamp corresponding to instance x).

Each data set D_k will contain, for each data point from the analyzed map, examples in the form $< x_k, y_k >$ (as previously described). As a preprocessing step before training, the data sets D_k are normalized to [0,1], using the *min-max* normalization.

5.3.2. Building the *NowDeepN* Model

Using the data modelling proposed in Section 5.2 and previously described, we aim to build a supervised learning model *NowdDeepN* consisting of an ensemble of DNNs for expressing np functions (hypotheses) $h_k,\ 1 \le k \le np$ such that $h(x_k) \approx y_k$.

One of the difficulties regarding the regression problem previously formulated is that the training data sets D_k built as shown in Section 5.3.1 are highly *imbalanced*. More specifically, there are a lot of training instances labeled with zero (i.e., $y_k = 0$) corresponding to points on the map without specific weather events and a much smaller number of instances with a non-zero label (i.e., corresponding to a severe meteorological phenomenon). The imbalanced nature of the data may lead to a regressor which is biased to predict zero values, as the majority of the training examples used for building the regressor were zero-labeled. A number of np DNN regressors $N_k,\ 1 \le k \le np$ will be trained on the data sets D_k, such that the model M_k will learn to provide estimates for the radar product p_k.

Each of these DNN regressors outputs a single value which represents the prediction for the next time step for that regressor's product. The output value is given by an linear activation function, while the hidden neurons use the ReLU activation function [46]. The loss we used was the mean squared error and the optimizer used was Adam [47]. For regularization we used one drop-out layer with the default Keras parameters.

5.3.3. Testing

For assessing the performance of *NowDeepN*, a *cross-validation* testing methodology is applied on each of the data sets D_k. The data sets D_k are randomly splitted in 5 folds. Subsequently, 4 folds will be used for training and the remaining fold for testing and this is repeated for each fold (5 times).

For each training-testing split, two evaluation measures are used and computed for each training-testing split: *Root mean squared error* (RMSE) and *Normalized root mean squared error* (NRMSE) [48]. The RMSE computes the square root of the average of squared errors obtained for the testing instances. The NRMSE represents the normalized RMSE, obtained by dividing the RMSE value to the range of the output and is usually expressed as a percentage. The regression related literature indicates NRMSE as a good measure for estimating the predictive performance of a regressor. Lower values for RMSE and NRMSE (closer to zero) indicate better regressors. For a more precise evaluation of the results, the values for the evaluation measures (RMSE and NRMSE) are also computed for the non zero-labeled instances ($\text{RMSE}_{non-zero}$, $\text{NRMSE}_{non-zero}$).

The RMSE and NRMSE values are computed for each data point from the grid (geographical area) and then are averaged over all grid points. As multiple experiments (training-testing data splits) are performed, the values for the evaluation measures were averaged over the 5 runs and a 95% *confidence interval* (CI) [49] of the mean value is computed.

6. Experimental Results

Experiments were performed by applying *NowDeepN* model on real data sets provided by the Romanian National Meteorological Administration and collected on the Central Transylvania region, using the methodology introduced in Section 5.

6.1. Data Set

This study uses data provided by the WSR-98D weather radar [6] located in Bobohalma, Romania and stored in the NEXRAD Level III format, as described in Section 5.1. The day used as case study is the 5th of June 2017, a day with moderate atmospheric instability manifested through thunderstorms accompanied by heavy rain and medium-size hail. In our study we selected an area from the central Transylvania region (parts of Mureş, Cluj, Alba and Sibiu counties) representing a grid having the geographical coordinates (46.076N, 46.725N, 23.540E and 25.064E). In the chosen geographical area, there were two distinct episodes with intense meteorological events in 5 June 2017: the first one between approximately 09:00 and 11:00 UTC, and the second one between approximately 12:00 and 17:00 UTC, with the most severe events taking place between 14:00 and 15:00 UTC. Concerning these phenomena, the National Meteorological Administration issued five severe weather warnings, code yellow.

The data grid provided by the radar for the selected geographical area at a given time moment is fit to a matrix. The radar provides one data matrix for each radar product. As stated in Section 5.1, the radar data is split into multiple time stamps, each time stamp representing data gathered by the radar every 6 min (the radar takes 6 min to gather the data for the area). The radar data used in our case study has been recorded between 00:04:04 UTC and 23:54:02 UTC.

In the current study, we are using only 13 products (i.e., $np = 13$): base *reflectivity* (R) of particles on six elevations (R01-R04, R06-R07) *velocity* (V) on six elevation (V01-V04, V06-V07) and the estimated quantity of water (VIL) contained by a one square meter column

of air. Thus, the set of considered radar products is $Prod = \{$R01, R02, R03, R04, R06, R07, V01, V02, V03, V04, V06, V07, VIL$\}$. Accordingly, *NowDeepN* ensemble of regressors will predict 13 values, corresponding to the products previously enumerated. Our study uses only R, V and VIL products, as they are mostly used by meteorologists for weather nowcasting.

As mentioned in Section 5.2, we consider each point in the grid an instance. For each point we consider its neighbours in a certain radius. We decided to select a value of 13 for the length l of the neighborhood surrounding each point (see Section 5.3.1). More exactly, the 2D data subgrid representing the neighborhood for a point is a 13 by 13 matrix. The reason for choosing 13 as the dimensionality of the neighborhood is that it represents about 5 km in the physical world, which, from a meteorological view, is a common distance to determine small gradients. Thus for each instance we are considering a matrix of 13 by 13 points, each of which have 13 products. Therefore, for each instance we have $13 \times 13 \times 13 = 2197$ attributes. For each timestamp we have a grid of the size 400×312. We only used the instances for which we could get the entire neighbourhood (i.e., where the neighbourhood matrix would not exceed the limit of the grid), thus obtaining 116.400 instances per timestamp. The day used as a case study contains 231 timestamps, thus of our data set consists of 26.888.400 instances. The data used in the experiments are publicly available at http://www.cs.ubbcluj.ro/~mihai.andrei/datasets/nowdeepn/.

6.2. Data Analysis

In order to estimate the impact of the data cleaning step, we analyzed the data set before and after cleaning. For each of the data products V01, V02, V03, V04, V06 and V07 (which were possibly cleaned) and each time stamp t, $1 \leq t \leq 231$, we computed the average values of the radar products for all the cells from the analyzed grid. Additionally, the average of the mean values for all V products and all the cells from the grid were calculated for each time stamp.

Figure 5 comparatively depicts the variation of the mean V value with respect to each time stamp (ranging from 1 to 231) for three cases: (1) *before the cleaning* step; (2) before cleaning but *ignoring the invalid values*; and (3) *after the cleaning*. A step of 10 was selected on OX axis (i.e., an hour). Graphical representations similar to the ones from Figure 5 were created for V01-V07, as well. The time series plots for V01 and V06 are illustrated in Figures 6 and 7, respectively.

Analyzing the plots from Figures 5–7 and comparing the evolution of values *before cleaning* (red coloured), before cleaning but ignoring *invalid values* (blue coloured) and *after cleaning* (green coloured) we observe the following. At lower degrees of elevations (see the plot for V01 from Figure 6) there are much more invalid values than at higher degrees of elevation (see the graph for V06 from Figure 7). Besides, from a meteorological viewpoint, higher degrees of elevation are related to higher altitudes, implying a less chaotic air circulation, leading to more precise radar soundings. The impact of the data cleaning step is noticeable from the time series plots, as for V01 the graph before data cleaning significantly differs from the graph after cleaning. The two graphs have very different shapes and this suggests that the noise introduced in data after the cleaning step is considerably smaller than the noise existing in data before replacing the invalid V values. We note that a similar situation has been observed for V02-V04 as well.

Much more, the time series plots before data cleaning but ignoring invalid values (blue coloured) resembles to the graph after the invalid values were cleaned (green coloured). Figure 8 depicts a zoomed-in version of the graphs from the upper side of Figure 6 (the time series before cleaning but ignoring the invalid values and time series after cleaning). At higher degrees of elevation (V06), there is no significant difference between the evolution of V values before and after cleaning (the red coloured and green coloured plots from Figure 7). The shapes of the two plots are very similar for V06 and this was also observed for V07.

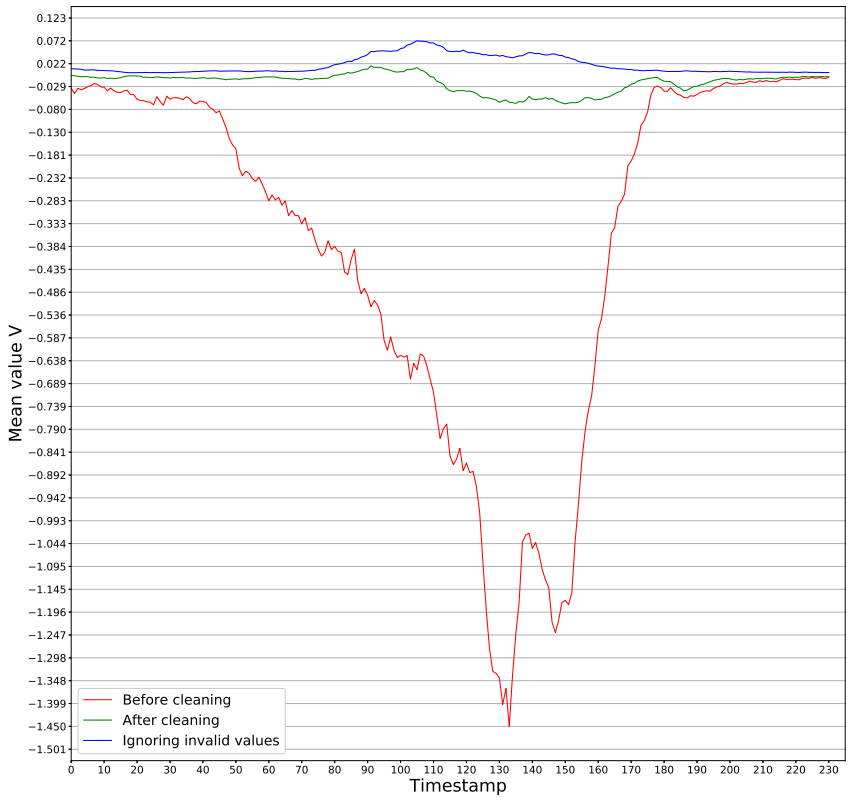

Figure 5. Time series plot for mean V values: ignoring invalid values, before and after cleaning.

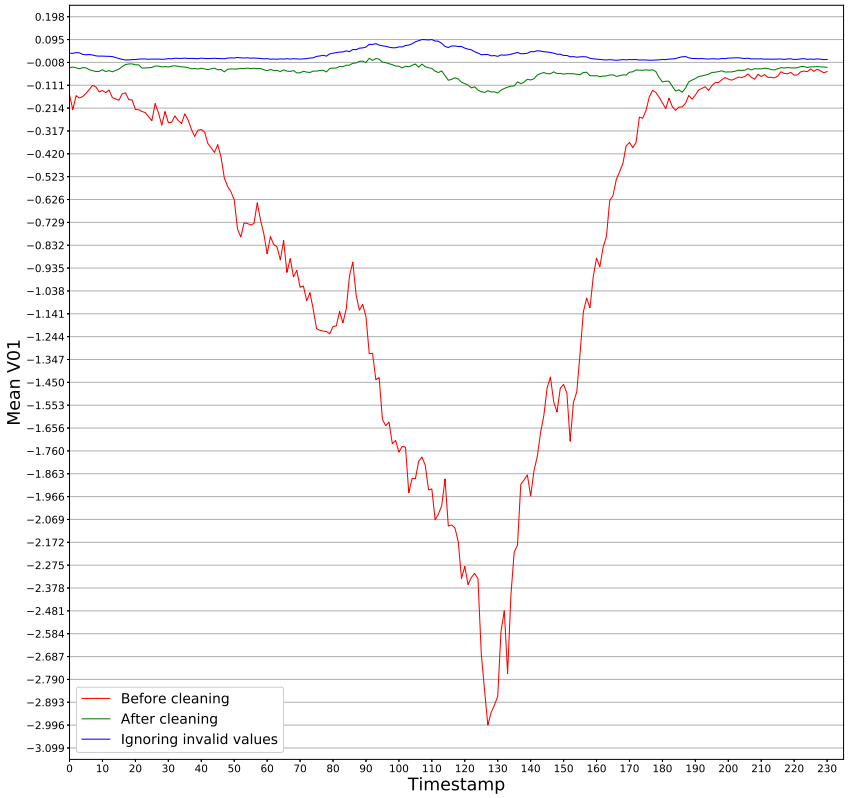

Figure 6. Time series plot for average V01 values: ignoring invalid values, before and after cleaning.

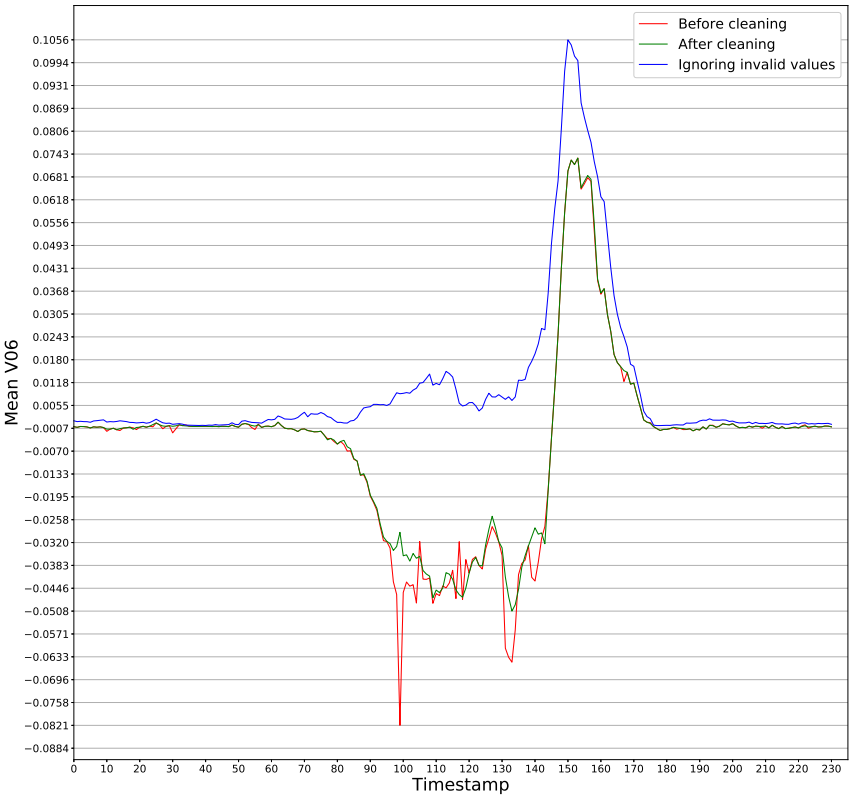

Figure 7. Time series plots for average V06 values: ignoring invalid values, before and after cleaning.

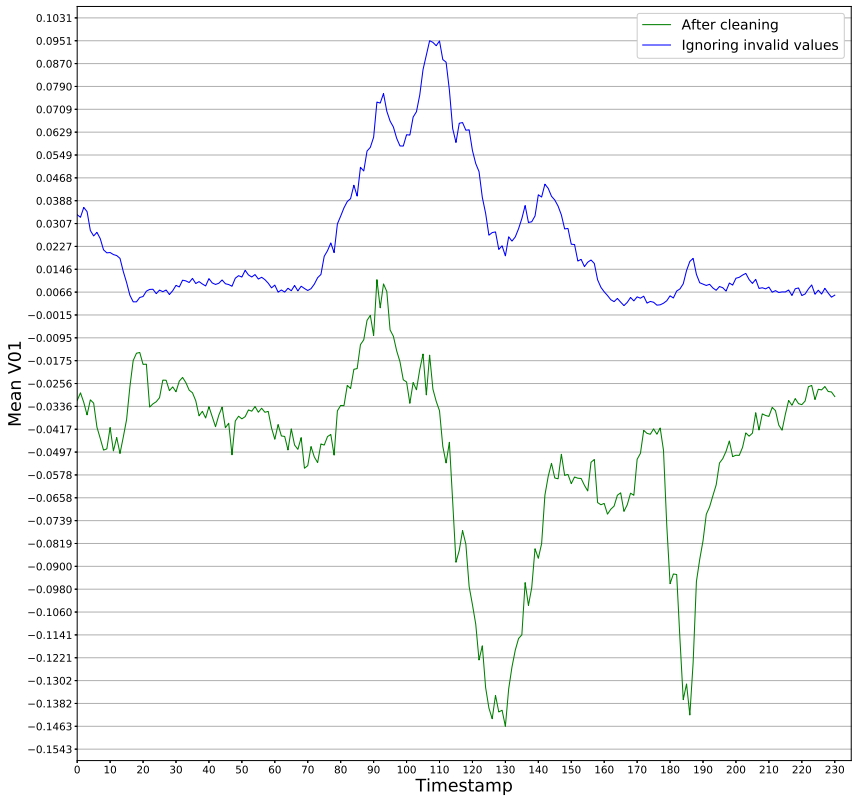

Figure 8. Time series plot for average V01 values: ignoring invalid values and after cleaning.

All the previous observations lead us to the hypothesis that the cleaning step would impact the overall performance of *NowdDeepN*, and this should be visible at least at lower degrees of elevations for V.

6.3. Results

This section presents the experimental results obtained by applying *NowDeepN* approach on the data set described in Section 6.1. For the DNNs used in our experiments, the implementation from the Keras deep learning API [50] using the Tensorflow neural networks framework was employed. The code is publicly available at Reference [51]. Given the fact that our data was quite high-dimensional, as mentioned in Section 6.1, we needed a relatively complex neural network. Each of The DNN regressors in the ensemble contains 12 hidden layer, with the following number of neurons: one layer with 200 neurons, one layer with 2000 neurons, 5 layers with 500 neurons and 5 layers with 100 neurons. These networks were trained for 30 epochs using 1024 instances in a training batch.

As stated at the beginning of the paper, we aim to answer research question RQ1 by assessing the ability of *NowDeepN* to predict the values for the radar products at a given moment in a certain geographical location from the values of its neighboring locations from previous time moments. Besides, we intend to analyze how correlated are our computational findings with the meteorological evidence. Table 1 depicts the obtained results together with their 95% CI. The columns of the table illustrate the evaluation measures computed for all 13 products (second column), the average values computed for all six R products (third column), the average values computed for all six V products (fourth column) as well as for VIL (fifth column). In order to allow an easier interpretation of the results from a meteorological perspective, we also illustrate in Table 1 the *Mean of Absolute Errors* (the average of the absolute errors obtained for the testing instances) for all instances (MAE), as well as only for the non-zero labeled instances ($MAE_{non-zero}$).

Table 1. Experimental results obtained using *NowDeepN*. 95% CI are depicted.

Evaluation Measure	All 13 Products	All R Products	All V Products	VIL
MAE	0.58 ± 0.02	0.76 ± 0.03	0.41 ± 0.02	0.53 ± 0.02
RMSE	2.25 ± 0.12	2.73 ± 0.17	1.44 ± 0.07	1.62 ± 0.10
NRMSE	$3.27\% \pm 0.17\%$	$3.91\% \pm 0.24\%$	$2.15\% \pm 0.11\%$	$2.32\% \pm 0.14\%$
$MAE_{non-zero}$	4.02 ± 0.12	5.51 ± 0.17	2.73 ± 0.12	2.89 ± 0.04
$RMSE_{non-zero}$	5.93 ± 0.14	7.63 ± 0.15	3.50 ± 0.15	3.9 ± 0.18
$NRMSE_{non-zero}$	$8.60\% \pm 0.21\%$	$10.91\% \pm 0.22\%$	$5.22\% \pm 0.22\%$	$5.63\% \pm 0.26\%$

From a meteorological point of view, the MAE for both all and non-zero instances is a satisfactory one, meaning that the predicted value is on the same level or on a neighbouring level on the product value scale.

The following two figures depict a comparison between the real data gathered by the radar (Figure 9) and the prediction made by *NowDeepN* (Figure 10). The timestamp of the comparison was chosen so that there was significant meteorological activity (14:37 UTC is in the middle of the meteorological event, as described in the data set used in Section 6.1). The figures depict product R01, chosen because it is one of the most relevant radar products for nowcasting, as it provides information on the precipitation and usually contains more non-zero data than upper levels. Figure 9 represents the real data collected by the radar, while Figure 10 represents the predicted data, given the real data at 14:31:15 UTC. The figures represent the (real or predicted) values of R01 *over a geographical area*. Each pixel in the images represents a geographical location of roughly 1 km^2. The values on axes OX and OY represent the coordinates *of a pixel* inside the image, relative to the (0,0) origin in the upper-left corner. While the values along the axes do not represent actual

longitude and latitude values, the OY axis runs along the latitudes and OX axis runs along the longitudes (the (0,0) origin of the images being the most north-western point of the geographical area the image represents).

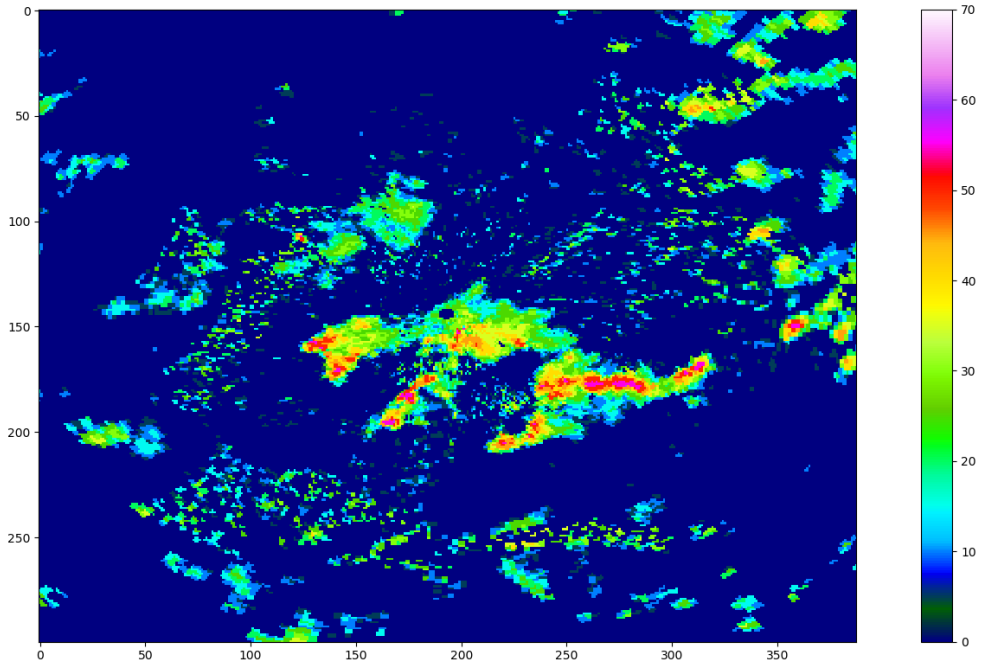

Figure 9. Real data for product R01 (reflectivity at the lowest elevation angle, measured in dBZ with values ranging form 0 to 70) at 14:37:22 UTC.

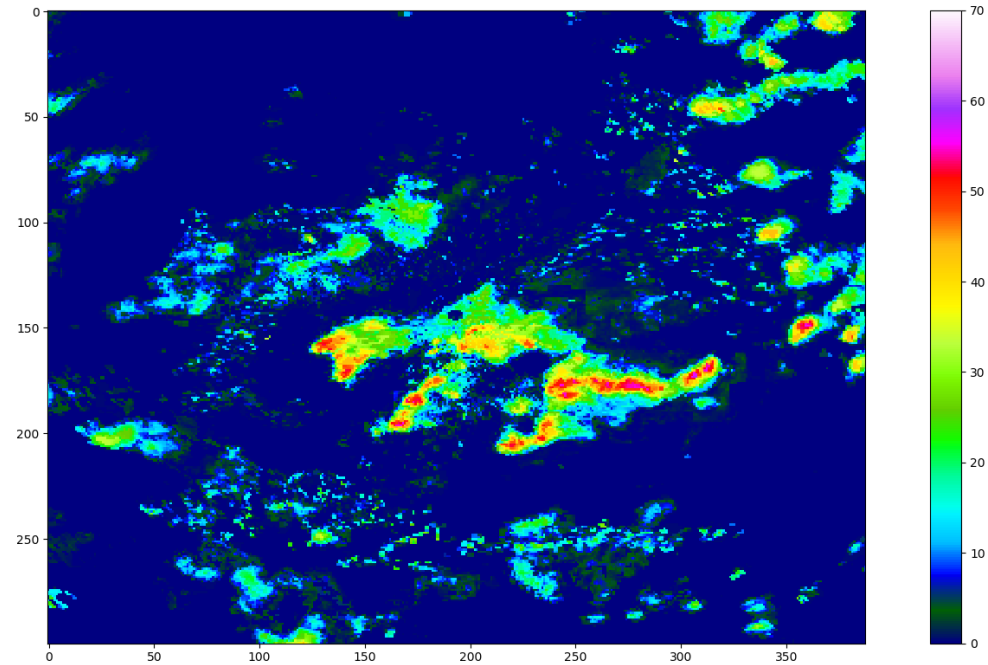

Figure 10. Predicted data for product R01 (reflectivity at the lowest elevation angle, measured in dBZ with values ranging form 0 to 70) at 14:37:22 UTC.

In Table 1 an average NRMSE of less than 4% is reported for the R products, which would entail a close resemblance between the predicted data and the real data, resemblance which can be observed in the comparison between the Figures 9 and 10. It can be observed

that the predicted pattern closely follows the real pattern, closely matching the shape and intensity of R01 in the studied area. There are, however, visible limitations of the prediction. There is a clear smoothing effect present, very clearly seen around areas where there are scattered points with non-zero values, the prediction tends to smooth out the area between them, resulting in a much larger areas of close to 0 non-zero values (colored in dark green) than in the real data. This effect can be also seen on areas with points with higher values, for example, around the point at roughly (170, 230), where in the real data present some ragged shape of higher intensity points, but the predicted data would present the area as a smooth shape, much less ragged, the space between the higher intensity points being predicted with a similar intensity. Another example can be seen at (110, 120), where, in the real data, there is a small shape with extremely high values; but in the predicted data, while the real shape is largely retained, the intensity is greatly diminished. Still, these kinds of effects are on a small scale relative to the entire area represented by the figure. Smoothing may be responsible for the higher NRMSE obtained for non-zero values, as areas with higher values are more affected by smoothing than areas with zero values.

7. Discussion

In this section we are analysing the performance of *NowDeepN* approach in order to answer research questions RQ2 and RQ3. First, we are going to assess the impact of the data cleaning step on the performance of *NowDeepN* (Section 7.1) and to estimate the relevance of the manually engineered features used in the training process (Section 7.2). Then, we continue in Section 7.3 with comparing our results with similar results obtained in the literature.

7.1. Impact of the Data Cleaning Step

As previously shown in Section 6.1, the training data set contains instances with errors, particularly at lower degrees of elevation. Obviously, the noisy training data may affect the performance of the learning task. In this regard, a data cleaning step motivated by the meteorological perspective has been introduced in Section 5.3.1 for replacing the invalid values in the data set which were identified mostly for the product V at lower degrees of elevation (V01-V04). The analysis we performed on data after it was cleaned (Section 6.2) led us to the hypothesis that the cleaning step would impact the overall performance of *NowdDeepN*, and this should be visible at least at lower degrees of elevations for V. Intuitively, as the data cleaning is correlated with the meteorological evidence, we expect a better performance of *NowDeepN*, particularly at lower degrees of elevation.

In order to empirically validate the hypothesis that the cleaning step improves the predictive performance of *NowDeepN*, we have evaluated the model trained on the uncleaned data set, using the same methodology introduced in Section 5.

Table 2 depicts the results obtained applying *NowDeepN* on the uncleaned data. Comparing the results with those obtained on the cleaned data (Table 1) we observe an improvement in the predictive performance of *NowDeepN* achieved on the cleaned data. The last column from the Table 2 illustrates the improvement obtained using the cleaning step, computed for each evaluation measure E (i.e., RMSE, NRMSE, $\text{RMSE}_{non-zero}$, $\text{NRMSE}_{non-zero}$) for all 13 radar products. The improvement is computed as $\frac{E_{uncleaned}-E_{cleaned}}{E_{uncleaned}}$.

Figure 11 depicts a very similar situation as Figure 10, the only difference being that the image represents values for R01 predicted by *NowDeepN* trained on *uncleaned* data. Similar to Figures 9 and 10, the image in Figure 11 represents a geographical area with each pixel representing a geographical location of roughly 1 km^2. Again, the values on axes OX and OY represent the coordinates of pixels inside the image relative to the (0, 0) origin in the upper-left corner, while axis OX runs along the longitudes and axis OY runs along the latitudes, with the (0, 0) origin being the most north-western point in the geographical area represented by the image. The most erroneous values are on the V products, yet the errors can still greatly affect the other products such as R01, presented in the figure. While in the predicted data the shapes and intensities are largely retained as

in the real data, some significant anomalies can be observed. For example, around the point at (140, 250) in the predicted data can be seen an area of higher valued points that is completely absent int the real data, with no means to attribute it to the smoothing effect. Also, in the upper right corner at around (10, 380), there can be seen in the real data a significant shape with quite high values at the interior that is simply removed in the predicted data, although the surrounding shapes are much better represented. There are other such anomalies that cannot be explained by the smoothing effect that appear in the data predicted by *NowDeepN* trained on uncleaned data. None of these anomalies appear on the data predicted *NowDeepN* trained on cleaned data. This suggests that there were some erroneous values of V in those areas that also affected how *NowDeepN* predicted the other products, and thus, cleaning the data yields much better results.

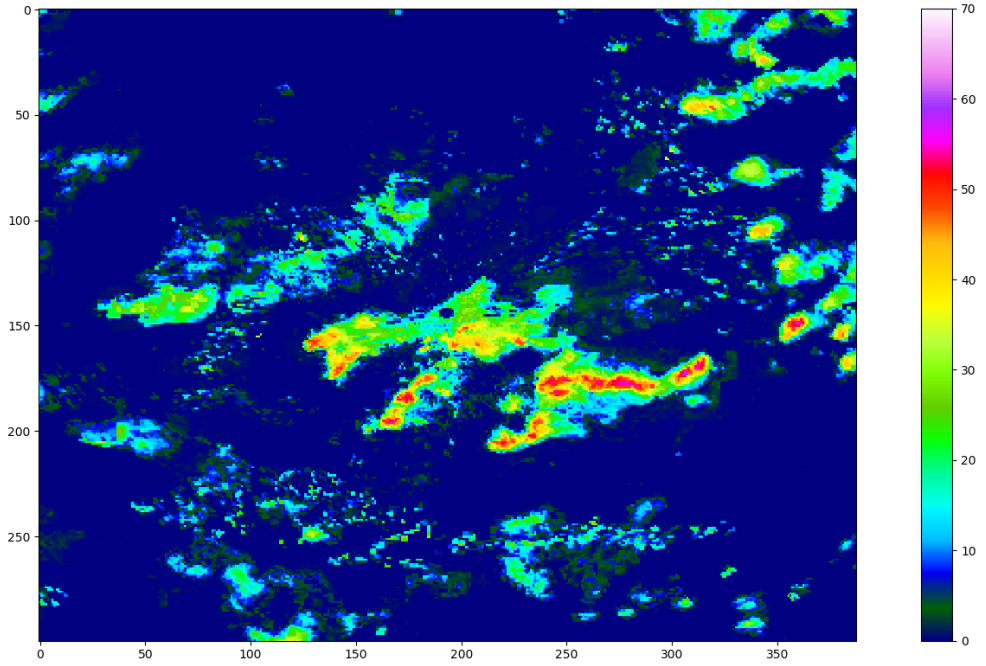

Figure 11. Predicted data, using the model trained on uncleaned data, for product R01 (reflectivity at the lowest elevation angle, measured in dBZ with values ranging form 0 to 70) at 14:37:22 UTC.

7.2. Relevance of the Used Features

We aim to further analyze the *NowDeepN* approach by determining the relevance of the features used in the training process. More precisely, our goal is to examine if the radar products' values from a neighborhood of a certain location at time t are suitable for predicting the products' values at time t+1, for the same location. The analysis from this section is conducted for answering question RQ3.

For determining the significance of the features, we are comparing the results of *NowDeepN* using the original set of features with those obtained by applying *NowDeepN* after the prior application of a feature extraction step. Two feature extractors will be applied on the original set of features, for reducing the dimensionality of the input data.

Table 2. Experimental results obtained using *NowDeepN* on the uncleaned data. 95% CI are used for the results. The last column illustrates the improvement achieved applying *NowDeepN* on the cleaned data, considering all 13 radar products.

Evaluation Measure	All 13 Products	All R Products	All V Products	VIL	Improvement (%) (Cleaning Step)
RMSE	4.98 ± 0.06	4.97 ± 0.10	3.99 ± 0.07	5.24 ± 0.17	55%
NRMSE	7.27% ± 0.09%	7.10%± 0.15%	5.95% ± 0.10%	7.49% ± 0.24%	55%
$RMSE_{non-zero}$	10.05± 0.40	9.38 ± 0.23	10.88± 0.71	9.10 ± 0.31	41%
$NRMSE_{non-zero}$	14.68%± 0.59%	13.40%± 0.33%	13.24%± 1.06%	13.00% ± 0.44%	41%

1. A sparse denoising AE is applied on the original data using a dimensionality of 250 for the hidden state. The AE's latent space representation (i.e., the result of the encoding part) will be further used (as a lower-dimensional representation of the input data) by *NowDeepN* for the prediction task. For the AE implementation we have used the following model: 7 hidden layers, 3 hidden layers for encoding (1000, 750, 500 neurons respectively), 250 neurons on the encoding layer and 3 hidden layers for decoding (500, 750, 1000 neurons respectively), all of these layers using the ReLU activation function, with the exception of the encoding layer, which used the linear activation function; the output size is equal to the input size (2197 neurons), with the linear activation function; using mean squared error loss and the Adam optimizer; the network was trained for 30 epochs using a batch of 1024 instances.
2. The PCA algorithm is applied for a linear dimensionality reduction of the input data into a 250 dimensional space. For PCA we have used the existing scikit-learn Python implementation of the algorithm using 250 principal components and the default values for the other parameters [52].

Table 3 depicts the results obtained applying *NowDeepN* with a previous feature extraction step (AE/PCA). Comparing the results with those obtained without applying a feature extraction step (Table 1) we observe an improvement in the predictive performance of *NowDeepN* achieved without a prior feature extraction step. The last column from the table illustrate the improvement obtained by *NowDeepN* on the original data (computed as the difference between the measure on the original and on the reduced data divided to the value obtained on the reduced data).

The results depicted in the last column from Table 3 empirically demonstrate that the features (i.e., the radar products' values from a neighborhood of a certain location at time *t*) used in training *NowDeepN* are relevant for predicting the radar products' values at time *t*+1, for the same location. The relevance of the features is validated by the fact that a dimensionality reduction technique (AE/PCA) applied prior to the classification using *NowDeepN* does not improve the learning performance. The last column from Table 3 also reveals that AEs preserve better than PCA the characteristics of the data when reducing its dimensionality, which is expectable as AEs perform a non-linear mapping whilst PCA a linear one.

7.3. Comparison to Related Work

The literature review from Section 4 revealed various approaches developed in the nowcasting literature using machine learning methods.

Table 3. Experimental results obtained using *NowDeepN* with a previous feature extraction step. 95% CI are used for the results. The last column illustrates the improvement achieved applying *NowDeepN* on the original data, considering all 13 radar products.

Feature Extraction	Evaluation Measure	All 13 Products	All R Products	All V Products	VIL	Improvement (%) without Feature Extraction
AE	RMSE	2.40 ± 0.07	2.93 ± 0.08	1.51 ± 0.05	1.76 ± 0.11	6%
	NRMSE	3.49% ± 0.10%	4.19% ± 0.12%	2.26% ± 0.08%	2.51% ± 0.16%	6%
	$RMSE_{non-zero}$	6.75± 0.29	8.81 ± 0.41	3.87 ± 0.14	4.55± 0.49	12%
	$NRMSE_{non-zero}$	9.79% ± 0.41%	12.58% ± 0.59%	5.78% ± 0.20%	6.49% ± 0.70%	12%
PCA	RMSE	2.76 ± 0.03	3.40 ±0.15	2.16 ±0.09	2.52 ±0.13	18.50%
	NRMSE	4.01% ±0.05%	4.86%± 0.22%	3.22%±0.14%	3.60% ±0.19%	18.47%
	$RMSE_{non-zero}$	7.56 ± 0.31	9.80± 0.79	5.58 ±0.19	5.94 ±0.37	21.55%
	$NRMSE_{non-zero}$	10.96% ±0.45%	14.01% ±1.13%	8.33%± 0.29%	8.49% ±0.53%	21.54%

We start the comparison between *NowDeepN* and related work by comparing our model to a simple baseline model, the *linear regression* (LR). For an exact comparison, the data model used for *NowDeepN* (Section 5.2) was used for the LR model as well. By applying the LR on the dataset described in Section 6.1, an overall RMSE for the non-zero values ($RMSE_{non-zero}$) of 6.094 was obtained. Surprisingly, there is only 3% improvement achieved by *NowDeeP* on our data, with a higher improvement on V (about 12%). From a meteorological point of view, the minor improvement in reflectivity nowcasting (compared to the LR model) is probably determined by the fact that the model is predicting the values of the radar products for one time step, and on the particular day used in this study the convective structures detected by the radar display a relatively slow evolution due to the light to moderate wind, thus no rapid modifications between two radar scans can be observed in terms of R value or location. Bigger improvements can be observed in V product values predictions, since the evolution of this product has a more stochastic character. Future work, dealing with prediction over more time steps, should display greater improvements compared to the benchmark LR model.

Even if there are numerous machine learning-based methods developed for nowcasting purposes, there are few methods focused on radar base products' values nowcasting, such as reflectivity nowcasting. Most of the related work focus on the precipitation nowcasting problem. We found four approaches having similar goal to our paper, that of predicting the future values of the radar products' values based on their historical values. The approaches from the literature which are the most similar to ours are those proposed by Yan Ji [39], Han et al. [31,41] and Yan et al. [42]. Even if the data sets used in the previously mentioned papers and the evaluation methodology differs from ours, we computed the evaluation measures reported in literature, trying to reproduce as accurately as possible the experiments from the related work.

The work of Ji [39] is focused on predicting only the reflectivity values which are further used for precipitation prediction. Experiments are performed only on radar data collected only for time stamps when storms occurred, disregarding the periods with normal weather (i.e., for which the R value is 0). Besides the minimum and the maximum values for the RMSE, the *Hit rate* (HR) is reported in Reference [39] as the percentage of instances for which the absolute error (between the predicted and the real value) is less than or equal to 5. For an accurate comparison with the work of Yan Ji [39], we trained our *NowDeepN* model only on the instances labeled with non-zero values for R and was tested only on non-zero instances. We also note that the evaluation from Reference [39] is performed only once, without using a cross-validation. Han et al. [31,41] focused on their works on predicting the R values (using SVM and CNN classifiers), more specifically if they exceed 35 dBZ. Considering that the positive class is the one for which the R values are larger than 35, the authors used three evaluation measures: (1) *critical success index* (CSI), computed as

$CSI = \frac{TP}{TP+FN+FP}$; (2) *probability of detection* (POD), $POD = \frac{TP}{TP+FN}$; and (3) *false alarm rate* (FAR), $FAR = \frac{FP}{FP+TP}$. The MAR-CNN model proposed for precipitation nowcasting by Yan et al. [42] used radar reflectivity images on three elevation levels and other numerical features and provided a RMSE of 7.90 for the predicted reflectivity values.

Table 4 summarizes the comparison between *NowDeepN* and the related work. The best values for the evaluation measures are highlighted.

Table 4. Comparison to the work of Yan Ji [39], Han et al. [31,41] and Yan et al. [42].

Model	RMSE	HR	CSI	POD	FAR
Our *NowDeepN*	5.34	0.75	**0.64**	**0.83**	**0.27**
ANN [39]	[**0.97**, **4.7**]	**0.89**	–	–	–
CNN [41]	–	–	0.44	–	–
SVM [31]	–	–	0.36	0.61	0.52
MAR-CNN [42]	7.90	–	–	–	–

Table 4 reveals that *NowDeepN* outperforms the approaches proposed by Han et al. [31,41] in terms of CSI, POD and FAR evaluation measures. Overall, in 71% of the cases (5 out of 7 comparisons), the comparison is favorable to *NowDeepN*. Our proposal is outperformed only by the work of Yan Ji [39] which reported a better HR and a maximum RMSE slightly better than ours. This difference may occur due to the following: (1) the data sets used (both as training and and testing) are different and have particularities due to the geographic area (country) on which were collected (i.e., China [39] and Romania); (2) the testing data set from Reference [39] contains data collected on a relatively small area which may lead to a biased evaluation and an overestimated performance.

As previously mentioned, a lot of work has been carried out in the literature for precipitation nowcasting. Tran and Song [40] tackled the precipitation nowcasting problem from a computer vision perspective, by applying certain thresholds on the reflectivity values (5/20/40 dBZ). In order to measure the performance of *NowDeepN* (in terms of CSI, POD and FAR) for the aforementioned thresholds we transformed the predicted values to predicted classes, by denoting each predicted value lower or equal to a threshold as being in the negative class and each predicted value higher than the threshold as being in the positive class. We then applied the same transformation on the ground truth and computed the measures; following this process for each of the three thresholds (5/20/40 dBZ). Table 5 illustrates the comparison between *NowDeepN* and the model proposed by Tran and Song [40]. We mention that Tran and Song [40] provide for each evaluation measure ranges of values. Since an exact comparison cannot be provided (the datasets used for evaluation and the input data models are different) our comparison relies only on the magnitude of CSI, POD and FAR evaluation metrics. The best values obtained for each reflectivity threshold and evaluation metric are highlighted.

The comparative results from Table 5 highlight that *NowDeepN* obtained better results than the model proposed by Tran and Song [40] in 77.7% of the cases (7 out of 9 comparisons). We note the good performance of *NowDeepN* at higher values for the reflectivity threshold, which indicate the ability of our model to detect moderate and heavy precipitation and medium and large hail.

Table 5. Comparison to the work of Tran and Song [40].

Model	Reflectivity Threshold	CSI	POD	FAR
Our *NowDeepN*	5	**0.6475**	**0.8055**	**0.2326**
	20	**0.5386**	**0.7646**	**0.3543**
	40	**0.4290**	**0.6277**	**0.4245**
TrajGRU [40]	5	0.6729–0.7069	0.7646–0.8053	0.1579–0.1812
	20	0.2994–0.3208	0.3949 –0.4296	0.4443–0.4815
	40	0.0411–0.0549	0.0568–0.0859	0.7539–0.79

8. Conclusions and Future Work

We introduced in this paper a supervised learning based regression model *NowDeepN*, which used an ensemble of *deep artificial neural network* for predicting the values for meteorological products at a certain time moment based on their historical values. *NowDeepN* was intended to be a proof of concept for the feasibility of learning to approximate a function between past values of the radar products extracted from radar observations and their future values.

The *NowDeepN* model consisted of an ensemble of DNNs for radar products' values nowcasting. In this ensemble, a DNN model was used for learning to approximate the value of each radar product at a given time moment and a certain geographical location from the radar products' values from the neighborhood of that location at previous time moments.

Experiments were conducted on real radar data provided by the Romanian National Meteorological Administration and collected on the Central Transylvania region. The obtained results provided an empirical evidence that in both normal and severe weather conditions the values for a radar product at a given moment in a certain location are predictable from the values of the neighboring locations from previous time moments. This evidence is essential for further using convolutional neural network models for automatically extracting from radar data features which would be relevant for predicting the radar products' values at a certain time moment based on their historical values. A data cleaning step was introduced for correcting the erroneous input radar and its impact on increasing the predictive performance of the *NowDeepN* model was highlighted. In addition, the relevance of the features considered in the supervised learning task was empirically proven. More specifically, the experiments shown that the radar products' values from the neighboring area of a certain geographical location l at time $t-1$ are useful for predicting the radar products' values on location l at time t.

The experimental results highlighted that our *NowDeepN* model has a good performance particularly for high values of the reflectivity threshold, which indicate its ability to detect moderate and heavy precipitation and medium and large hail. While from a meteorological point of view, the performance of *NowDeepN* for predicting radar reflectivity values one time step ahead is satisfactory, in order to assess its performance compared to techniques currently employed in weather nowcasting, further development of the model for multiple time steps (e.g., 5 or 10 time steps, covering 30 or 60 min) is needed.

The experimental evaluation of *NowDeepN* will be further extended by enlarging the data set used for training the model. As future plans we aim to investigate convolutional neural network models [53] as well as supervised classifiers based on *relational association rule mining* [54,55] for detecting relationships between the meteorological products' values which may distinguish between normal and severe meteorological phenomena. In addition, we will analyze the possibility to extend the features used in the learning process, by combining radar data with other features (e.g., geographic and antropic features).

Author Contributions: Conceptualization, G.C. and A.M.; methodology, G.C. and A.M.; software, G.C. and A.M.; validation, G.C., A.M. and E.M.; formal analysis, G.C. and A.M.; investigation, G.C.,

A.M. and E.M.; resources, G.C., A.M. and E.M.; data curation, G.C., A.M. and E.M.; writing–original draft preparation, G.C.; writing–review and editing, G.C., A.M. and E.M.; visualization, G.C., A.M. and E.M.; funding acquisition, G.C., A.M. and E.M. All authors have read and agreed to the published version of the manuscript.

Funding: The research leading to these results has received funding from the NO Grants 2014–2021, under Project contract no. 26/2020.

Acknowledgments: The research leading to these results has received funding from the NO Grants 2014–2021, under Project contract no. 26/2020. The authors acknowledge the assistance received from the National Meteorological Administration from Romania, for providing the meteorological data sets used in the experiments. The authors also thank the editor and the reviewers for their useful suggestions and comments that helped to improve the paper and the presentation.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Franch, G.; Nerini, D.; Pendesini, M.; Coviello, L.; Jurman, G.; Furlanello, C. Precipitation Nowcasting with Orographic Enhanced Stacked Generalization: Improving Deep Learning Predictions on Extreme Events. *Atmosphere* **2020**. *11*, 267.
2. Chen, L.; Cao, Y.; Ma, L.; Zhang, J. A Deep Learning-Based Methodology for Precipitation Nowcasting With Radar. *Earth Space Sci.* **2020**, *7*, e2019EA000812.
3. Swedish Meteorological and Hydrological Institute. Cooperation Is a Must for Adaptation to and Mitigation of Climate Change. 2018. Available online: https://www.smhi.se/en/news-archive/ (accessed on 10 July 2018).
4. WMO—World Meteorological Organisation. Weather Climate Water, 2018. Available online: https://www.wmo.int (accessed on 10 July 2018).
5. Kimura, R. Numerical weather prediction. *J. Wind. Eng. Ind. Aerodyn.* **2002**, *90*, 1403–1414.
6. NOAA's Radar Operations Center. NEXRAD Technical Information, 2018. Available online: https://www.roc.noaa.gov/WSR88D/Engineering/NEXRADTechInfo.aspx (accessed on 10 July 2018).
7. Hering, A.M.; Morel, C.; Galli, G.; Sénési, S.; Ambrosetti, P.; Boscacci, M. Nowcasting thunderstorms in the Alpine region using a radar based adaptive thresholding scheme. In *Proceedings of ERAD*; Copernicus GmbH: Visby, Sweden, 2014; pp. 206–211.
8. James, P.; Reichert, B.; Heizenreder, D. NowCastMIX—optimized automatic warnings from continuously monitored nowcasting systems based on fuzzy-logic evaluations of storm attributes. In *European Conference on Severe Storms*; American Meteorological Society: Wiener Neustadt, Austria, 2015; pp. 1413-1433.
9. Merlet, N.; Cau, P.; Jauffret, C.; Maynard, K.; Sanchez, I.; Brousseau, P. AROME-NWC Overview, Results, Evolution and Perspectives. *Eur. Forecast.* **2017**, *22*, 40–43.
10. Dixon, M.; Wiener, G. TITAN: Thunderstorm Identification, Tracking, Analysis, and Nowcasting—A radar-based methodology. *J. Atmos. Ocean. Technol.* **1993**, *10*, 785–797.
11. Johnson, J.T.; MacKeen, P.L.; Witt, A.; Mitchell, E.D.W.; Stumpf, G.J.; Eilts, M.D.; Thomas, K.W. The Storm Cell Identification and Tracking Algorithm: An Enhanced WSR-88D Algorithm. *Weather. Forecast.* **1998**, *13*, 263–276.
12. Jung, S.H.; Lee, G. Radar-based cell tracking with fuzzy logic approach. *Meteorol. Appl.* **2015**, *22*, 716–730.
13. Auger, L.; Dupont, O.; Hagelin, S.; Brousseau, P.; Brovelli, P. AROME—NWC: A new nowcasting tool based on an operational mesoscale forecasting system. *Q. J. R. Meteorol. Soc.* **2015**, *141*, 1603–1611.
14. Haiden, T.; Kann, A.; Wittmann, C.; Pistotnik, G.; Bica, B.; Gruber, C. The Integrated Nowcasting through Comprehensive Analysis (INCA) System and Its Validation over the Eastern Alpine Region. *Weather. Forecast.* **2011**, *26*, 166–183.
15. IBM. Deep Thunder, 2014. Available online: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepthunder/ (accessed on 15 September 2019).
16. Panasonic. Panasonic Global 4D Weather, 2016. Available online: https://www-media.panasonic.aero/2016/01/010616-PWS4DGlobal.pdf (accessed on 15 September 2019).
17. Climacell. The ClimaCell Engine, 2019. Available online: https://www.climacell.co/ (accessed on 15 September 2019).
18. TempoQuest. TempoQuest, 2015. Available online: http://tempoquest.com/ (accessed on 15 September 2019).
19. Mitchell, T.M. *Machine Learning*, 1st ed.; McGraw-Hill, Inc.: New York, NY, USA, 1997.
20. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
21. Srivastava, N.; Hinton, G.E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
22. Gogas, P.; Papadimitriou, T.; Agrapetidou, A. Forecasting bank failures and stress testing: A machine learning approach. *Int. J. Forecast.* **2018**, *34*, 440–455.
23. Tai, C.C.; Lin, H.W.; Chie, B.T.; Tung, C.Y. Predicting the failures of prediction markets: A procedure of decision making using classification models. *Int. J. Forecast.* **2019**, *35*, 297–312.

24. Maymin, P.Z. Wage against the machine: A generalized deep-learning market test of dataset value. *Int. J. Forecast.* **2019**, *35*, 776–782.
25. Nowlan, S.J.; Hinton, G.E. Simplifying Neural Networks by Soft Weight-Sharing. *Neural Comput.* **1992**, *4*, 473–493.
26. Le, Q. Building high-level features using large scale unsupervised learning. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 8595–8598.
27. Deng, J.; Zhang, Z.; Marchi, E.; Schuller, B. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In Proceedings of the Humaine Association Conference on Affective Computing and Intelligent Interaction, Geneva, Switzerland, 2–5 September 2013; pp. 511–516.
28. Teletin, M.; Czibula, G.; Bocicor, M.I.; Albert, S.; Pandini, A. Deep Autoencoders for Additional Insight into Protein Dynamics. In *International Conference on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 79–89.
29. Teletin, M.; Czibula, G.; Codre, C. AutoSimP: An Approach for Predicting Proteins' Structural Similarities Using an Ensemble of Deep Autoencoders. In *International Conference on Knowledge Science, Engineering and Management*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 49–54.
30. Jolliffe, I.T.; Cadima, J. Principal component analysis: a review and recent developments. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **2016**, *374*, 20150202.
31. Han, L.; Sun, J.; Zhang, W.; Xiu, Y.; Feng, H.; Lin, Y. A machine learning nowcasting method based on real-time reanalysis data. *J. Geophys. Res. Atmos.* **2017**, *122*, 4038–4051.
32. Beusch, L.; Clementi, L.; Foresti, L.; Hamann, U.; Hering, A.M.; Leonarduzzi, E.; Nerini, D.; Nisi, L.; Sassi, M.; Germann, U. Towards thunderstorm nowcasting by applying machine learning to a multi-sensor observation and NWP model database. In Proceedings of the European Conference on Severe Storms 2017, Pula, Croatia, 18–22 September 2017; Volume 136.
33. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.k.; Woo, W.c. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems—Volume 1*; MIT Press: Cambridge, MA, USA, 2015; pp. 802–810.
34. Kim, S.; Hong, S.; Joh, M.; Song, S. DeepRain: ConvLSTM Network for Precipitation Prediction using Multichannel Radar Data. *arXiv* **2017**, arXiv:1711.02316.
35. Heye, A.; Venkatesan, K.; Cain, J. Precipitation Nowcasting: Leveraging Deep Convolutional Recurrent Neural Networks. In *Proceedings of the Cray User Group (CUG)*; American Meteorological Society: Austin, Texas, 2017; pp. 1–8.
36. Shi, X.; Gao, Z.; Lausen, L.; Wang, H.; Yeung, D.; Wong, W.; Woo, W. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5622–5632.
37. Narejo, S.; Pasero, E. Meteonowcasting using Deep Learning Architecture. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 16–23.
38. Sprenger, M.; Schemm, S.; Oechslin, R.; Jenkner, J. Nowcasting Foehn Wind Events Using the AdaBoost Machine Learning Algorithm. *Weather Forecast.* **2017**, *32*, 1079–1099.
39. Ji, Y. Short-term Precipitation Prediction Based on a Neural Network. In Proceedings of the 3rd International Conference on Artificial Intelligence and Industrial Engineering, AIIE 2017, Shanghai, China, 26–27 September 2017; pp. 246–251.
40. Tran, Q.K.; Song, S.k. Computer Vision in Precipitation Nowcasting: Applying Image Quality Assessment Metrics for Training Deep Neural Networks. *Atmosphere* **2019**, *10*, 244.
41. Han, L.; Sun, J.; Zhang, W. Convolutional Neural Network for Convective Storm Nowcasting Using 3D Doppler Weather Radar Data. *IEEE Trans. Geosci. Remote. Sens.* **2020**, *58*, 1487–1495.
42. Yan, Q.; Ji, F.; Miao, K.C.; Wu, Q.; Xia, Y.; Li, T. Convolutional Residual-Attention: A Deep Learning Approach for Precipitation Nowcasting. *Adv. Meteorol.* **2020**, *2020*, 1–12.
43. Dietterich, T.G. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Mach. Learn.* **2000**, *40*, 139–157.
44. Mass, C. Nowcasting: The Promise of New Technologies of Communication, Modeling, and Observation. *Bull. Am. Meteorol. Soc.* **2012**, *93*, 797–809.
45. Czibula, G.; Mihai, A.; Mihuleţ, E.; Teodorovici, D. Using self-organizing maps for unsupervised analysis of radar data for nowcasting purposes. *Procedia Comput. Sci.* **2019**, *159*, 48–57.
46. Agarap, A.F. Deep Learning using Rectified Linear Units (ReLU). *arXiv* **2018**, arXiv:1803.08375.
47. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
48. Hyndman, R.J.; Koehler, A.B. Another look at measures of forecast accuracy. *Int. J. Forecast.* **2006**, *22*, 679–688.
49. Brown, L.; Cat, T.; DasGupta, A. Interval Estimation for a proportion. *Stat. Sci.* **2001**, *16*, 101–133.
50. Keras. The Python Deep Learning Library, 2018. Available online: https://keras.io/ (accessed on 15 December 2020).
51. A. Mihai. NowDeepN Software, 2020. Available online: https://github.com/mihaiandrei1294/NowDeepN (accessed on 15 December 2020).
52. Scikit-learn. Machine Learning in Python, 2018. Available online: http://scikit-learn.org/stable/ (accessed on 15 December 2020).
53. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: an overview and application in radiology. *Insights Into Imaging* **2018**, *9*, 611–629.

54. Miholca, D.L.; Czibula, G.; Czibula, I.G. A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. *Inf. Sci.* **2018**, *441*, 152–170.
55. Crivei, L.M. Incremental relational association rule mining of educational data sets. *Stud. Univ. Babes-Bolyai Ser. Inform.* **2018**, *63*, 102–117.