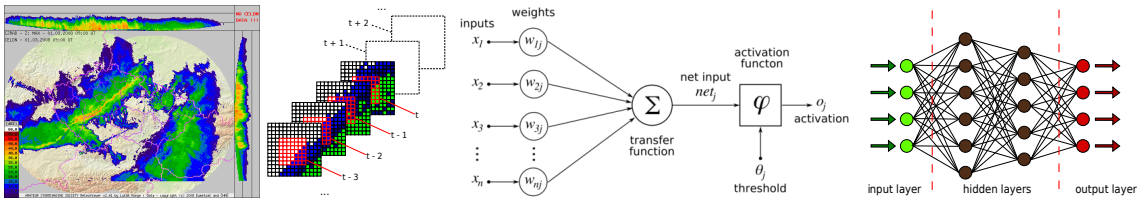
Excel@FIT 2015
http://excel.fit.vutbr.cz

Rain prediction from meteoradar images

Michael Vlček



Abstract

This paper presents a software solution for making a short-term rain prediction. It focuses on predicting precipitation solely on the soil of Czech Republic, although it is possible to apply to other areas, if proper modifications are made. This solution uses meteoradar images from Czech radars and a specific machine learning algorithm called *Artificial neural network*. The received meteoradar images are further processed and fed to the neural network, which makes an informed prediction of upcoming precipitation for a specific location anywhere in Czech Republic. The project is in an early experimenting phase, and as of now it is delivering a short-term prediction with success rates up to 73% for the earliest of predictions (10 minutes). The prediction success and length has to rise yet. This solution provides a robust way to predict rain on a national scale. It can be further improved to be used on mobile devices using utilities like GPS.

Keywords: rain prediction — neural network — image processing

Supplementary Material: *N/A*

*xvlcek21@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Artificial neural networks are being increasingly used to solve a wide range of problems in many areas of interest worldwide, like character recognition, stock market prediction, traffic monitoring, medicine and many other areas, which can be accounted to their excellent ability of finding trends and patterns in large quantities of data. This property can valuable in meteorology as well, as it can provide the approximation of future development of various meteorological parameters. Targeting the precipitation specifically, they can be used to anticipate the upcoming fronts by analysis of recent history from images in a similar way as their biological equivalent (human brain) would perform it.

The goal of this work is to provide a precise prediction of rain in the diverse topographic relief in Czech Republic using an artificial neural network. This method represents one of many ways to perform a short-term rain forecast, and even though it has been used already before and it is not a standard way, the potential rises with rising computational power of present computers, because the training is a very demanding process in this regard. The most important task in this work is to create a reliable model by using a sizeable database of radar images for neural network training as well as implementation and choosing a correct configuration of the neural network. This results in a properly trained neural network, which will then be tested for prediction success. Considering that this work also serves as an evaluation of the methodology, proper testing and experimenting with different configurations is a vital part. The evaluation will be done by comparing the outputs of trained neural network with expected outputs (images, that were not seen by neural network before). This process should result

in finding an ideal configuration for the final form of neural network, which has the most success.

There already exist some systems performing rain predictions. Commercial software providing this kind of service is distributed as a black-box. Its specifics describing the means by which the predictions are performed are not publicly available, and as such, no information about implementation is provided. Thus, the comparison of models can be based only on the results. Many rain prediction systems are focusing on long-term predictions, with less emphasis on spatial precipitation accuracy, but they also often apply a different model for short-term predictions, which can use *image processing* (cloud motion detection, etc.), *measured data analysis* (wind speed, radar image reflectivity,...), or even neural networks. These systems are often developed and maintained by national weather institutions and can cover a much wider range of functions. The most significant is the *Meteor - Aladin* system, which gives a forecast of various weather elements (temperature, precipitation, wind, pressure, humidity). This work provides an independent and transparent solution specifically for the short-term rain prediction. Also, a technical report was made by Lukáš Putna in 2009 on a similar topic, and some main differences are discussed in the section 3.2.

Now, the process of creating the model for this specific task will be briefly described. First, considerable amount of work is needed to process and group the images to serve as an input for the neural network. Then, a precise and possibly universal implementation of neural network training process must be done, which will be used to teach the neural network. The experiments with different neural network configurations follow, looking for the highest forecast success rate. Subsequently, the means to using a trained artificial neural network have to be implemented. Once the ideal configuration is found, it can be used for any following predictions.

The project is in the experimenting phase at the moment, testing many possible neural network configurations and training set collections. So far, the earliest predictions have up to 73% success in rain prediction. This number is expected to rise throughout the experimenting phase. Longer prediction results will be added once the earliest prediction success rates stabilize.

2. Artificial neural network

The basic idea behind this machine learning algorithm is to create a computational model, which behaves in a similar way to the human brain. To understand how it works, the basic elements of artificial neural networks have to be described.

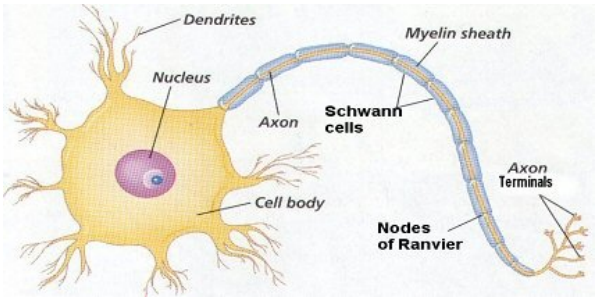


Figure 1. Brain neuron. [1]

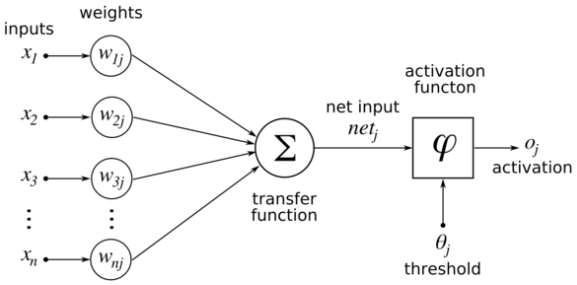


Figure 2. Artificial neuron.

2.1 Neuron

Both biological and artificial neural network consist of elements called *neurons*, which are capable of receiving, carrying, processing and sending information. The structure of a brain neuron is illustrated in figure 1. The neuron receives signals from its many *dendrites*. Each of these carry a signal from one neuron to the other, and every one of the signals has different weight, as it goes through to the target neuron. All the weighted signals received are then accumulated in the neuron body (*soma*), and if the total energy passes a certain threshold value, the neuron sends its own impulse through the output channel called the *axon*, which is connected to the dendrites of other neurons. Figure 2 shows the structure of *artificial neuron*. We can see, that the structures of a brain neuron and an artificial neuron are analogous. Inputs and weights represent the dendrites carrying the weighted signals, sum function and threshold value represent the neural cell body and the output can be viewed as an artificial axon. Output of the artificial neuron y_j can be expressed as follows [5]:

$$y = \varphi\left(\sum_{k=0}^{n} x_k w_k + \theta\right) \tag{1}$$

where:

- φ is the transformation function,
- $\sum_{k=0}^{n} x_{j_k} w_{j_k}$ is the aggregation function,
- x_k is the k-th value from input value vector,
- w_k is the k-th value from weight value vector,

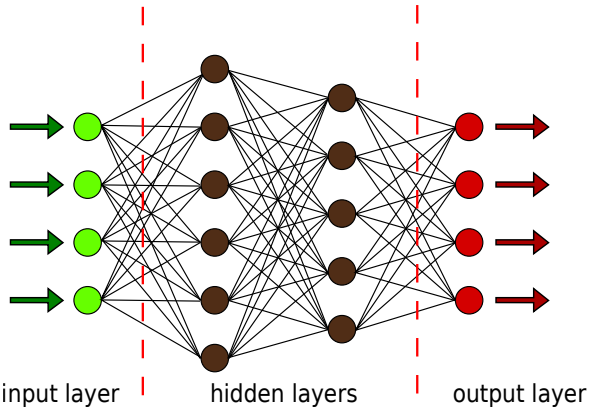


Figure 3. Layered neural network topology example.

- θ is the threshold function,
- n is the value count in input value vector.

2.2 Neural network topology and training

Connecting two or more neurons together forms a neural network. The topologies of bigger networks may vary, but the most widely used topology is *layered*. Neurons are formed into layers, where first layer serves as the *input layer*, last one serves as the *output layer* and every layer in between them (*hidden layer*) connects two surrounding layers together. The example of this topology is illustrated in figure 3.

Now we have a neural network, but like it is with human brain, in order to carry out a given task it must learn how to do it. And as there are various methods to teach people, there are also many *learning algorithms* for teaching neural networks. The process of neural network learning (or, in other words, approximating the function required for the specific task) consists of modifying the weights at each input of every neuron in a way defined by a learning algorithm [3]. In this project, the *feed-forward back-propagation supervised learning algorithm* is used, other algorithms will not be mentioned here. With supervised learning algorithms, the expected result is required before the training process in addition to the input value vector. These two elements together form a *training set*. When this training set is given to the neural network, the input value vector is "fed" to the neural network at the input layer, and the information goes (only) forward through the network (hence the feed-forward algorithm). The output is collected from the output layer and used to calculate the weight modifications and consequently update all the weights. For example, one of the most widely used error calculation functions is the *cross entropy error* function:

$$E(\vec{y},\vec{d}) = -\sum_{k=0}^{n} (ln(d_k) - y_k) \qquad (2)$$

where:

- d_k is the k-th value from the expected output vector and
- y_k is the k-th value from the neural network output vector.

The calculated error is then propagated back from the output layer to the input layer through the neural network by the back-propagation algorithm. Technically speaking, back-propagation calculates the gradient of the error of the network regarding the network's modifiable weights [4]. After this is done and the weights are modified accordingly, the next training set is used. All the training sets can be utilized multiple times, depending on how well is the neural network learning. Each training *epoch* includes going through all the training sets and cross-validation sets. This process continues as long as there is certain progress in learning. When it finishes, the final weight values are saved. The weight values are not modified any more, and provided that training process went well, the neural network can be used for the task it was trained to perform. The training itself can be viewed as a precomputation process, because it only runs once before the actual usage. It is also vital to mention, that even though the training is a very demanding task in terms of computation due to the excessively high training set count needed, the hardware and time requirements to use a properly trained neural network are significantly lower.

3. Data preparation for training sets

This section depicts the procedure of processing the radar images, so they can be used for training and evaluation. The images were taken by *CHMI* (*Czech Hydrometeorological Institute*), and they use the data from both Czech meteoradars. The example of such image is in figure 4. Also, the definition of training set form will be described, as the image processing aims to transform image data into training sets.

3.1 Image processing - Feature extraction

Image data processing begins with cropping the images, to get rid of the irrelevant data. Next step is specifying coordinates used for the creation of training sets. Each pixel in the cropped image can be used, as the training sets are going to be scrambled before usage, therefore the neural network will see no correlation between consecutive training sets given to it. When the size of input vector area per image is chosen (as written in 3.2), the cropped image is decoded (as they are provided in a *PNG palette-based* format)

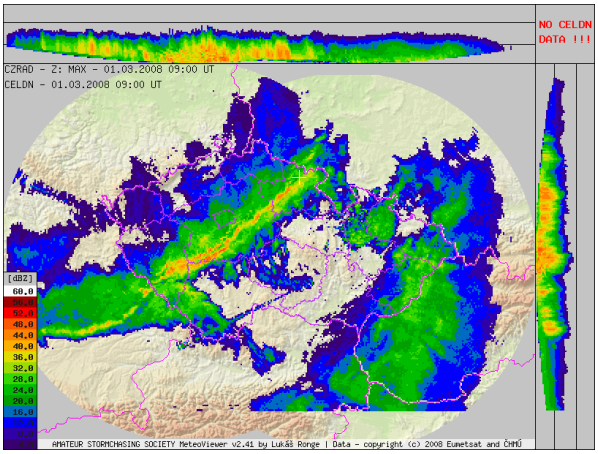


Figure 4. Radar image from 1. 3. 2008, 10:00 CET

and areas of every pixel are extracted from each image. Next, these areas from consecutive images (in time) are bound together by the coordinates they are located in, forming *sequences* (for example, all the areas with center in pixel at coordinates "160,190" and time 00:00, 00:10, 00:20, and so on form a sequence, as shown in figure 5). From these sequences we can create training sets, depending on the input vector and expected output vector sizes.

3.2 Training set form

The goal is to predict precipitation for a specific place in Czech republic. This place is represented by a pixel in a radar image. Thus, we expect the neural network to output the information about possible precipitation in the short future. This prediction is based on the previous front development, which can be extracted from images from recent history. The input vectors are constructed from the area of the target pixel from several images in the recent history and the expected result are only individual pixels taken from the images after these in time, which were not seen by the neural network. Together, these make an uniform pattern for making training sets. There is still a lot of room for different configurations of training sets. We can for example use different sizes of areas surrounding the target pixel. The larger the area, the bigger the neural network has to be, training will be longer and there has to be a higher count of training sets to train the neural net properly. On the other hand, larger area gives us more information to make a longer prediction.

To extend the amount of information in the used areas, a *compression algorithm* is used. The *DCT-II algorithm (discrete cosine transform - type II)* helps compress information from a large area into a much smaller space (the *energy compaction* property), and even though it is a lossy transformation, it is still necessary to make a longer prediction through a wider area

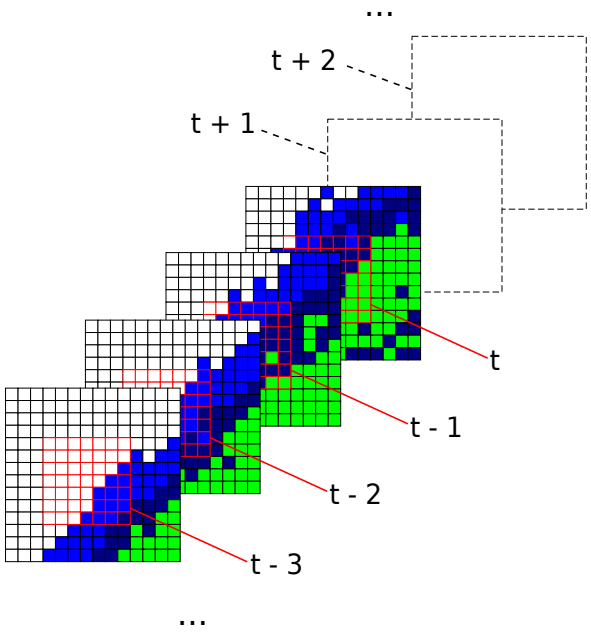


Figure 5. This image illustrates, how individual image parts are put together into an input value vector. The reflectivity values from selected areas in each image are extracted and concatenated together with other images.

context while maintaining a sustainable neural network size. This method is a *Fourier-related transform* but opposed to *DFT* transform it does not work with imaginary numbers. Also, the *KL* (*Karhunen-Loève*) transform provides very similar results but due to its computational complexity and image dependency, the DCT-II is considered a much better solution. This transformation has another valuable property, and it is, in addition to the energy compaction, the main difference between this paper and the paper written by Lukáš Putna [2]. It converts the discrete reflection values from images (which represent the amount of precipitation in specific areas) to real number coefficients, smoothing the transition between reflection levels in original image and therefore providing a more acceptable input for the neural network.

Another variable in creating training sets is the number of images used for a training set (both for the input vector part and, to extend the prediction possibilities, the expected result part). The input vector size is being chosen experimentally. The expected result vector is still kept small (up to 20 minutes), to calibrate earliest predictions first.

4. Experimenting with different configurations

At the time of writing this paper, the work is in the early experimentation stage. The neural net is trained on the images from the faculty database (these images

were taken by CHMI, as written in section 3), covering 7 full days with rain activity in images taken every 10 minutes. The configuration used for the training sets and neural network was following:

- Input vector contains 5 consecutive image parts,
- every part covers $7 * 7$ pixel area,
- expected output vector contains a 10 and 20 minute prediction,
- this prediction is incorporated into 1 of 4 classes, depending on the rain intensity of the prediction,
- cross-validation sets cover about 30% of all training sets, the rest is used for training,
- neural network has two hidden layers with 50 or 200 neurons each,
- it uses *softmax* function to classify neural network output into 4 classes.

So far, the model has thus been tested as a classifier. With 4 classes of rain intensity, it has been successful in making a precise prediction in approximately 73% of all cases for earliest term prediction (10 minutes). This number is expected to rise after proper experimental evaluation of different neural net and training set configurations.

Table 1. Experiment results

Configuration		Prediction success	
Area size	Hidden layer sizes	10 min	20 min
$7 * 7$	$40, 40$	55%	40%
$7 * 7$	$200, 200$	71%	65%
$7 * 7$	$500, 500$	73%	68%

5. Conclusions

Even from the early experiments, it is clear that the computational model of neural network is fit to be used on rain prediction problem. Its specifics and configuration are vital in the success rate of the forecast.

Neural network already in these early stages exhibits 73% success rate in earliest image-based predictions. As the experiments progress, the success rate is expected to rise and longer predictions will be made.

This work is being developed to provide an independent, robust and transparent solution for short-term precipitation prediction. Its final prediction capabilities are yet to be found in the next stages.

As the work shows a promising growth in successful predictions as experimenting continues, it could be deployed in the future into mobile application, using the GPS and image feed to make immediate rain predictions and using the smartphone notification system to warn of incoming fronts.

References

[1] BrainU, University of Minnesota Department of Neuroscience and Department of Curriculum and Instruction, *About neurons [online]*, http://brainu.org/files/tn_about_neurons.pdf, 2000-2011 [cit. 2015-01-17].

[2] Lukáš Putna, *Rain prediction using meteo-radar*, Tech. report, Brno University of Technology, Faculty of Information Technology, 2009.

[3] Frank C. Keil Robert A. Wilson, *The MIT encyclopedia of the cognitive sciences*, The MIT Press, 2001.

[4] Pal J. Werbos, *The roots of backpropagation: From ordered derivatives to neural networks and political forecasting*, John Wiley & Sons, Inc., New York, 1994.

[5] M. Šnorek, *Neuronové sítě a neuropočítače*, first ed., Prague, Czech Technical University in Prague, 1996.