

# Using Echo State Networks for Anomaly Detection in Underground Coal Mines

Oliver Obst      X. Rosalind Wang      Mikhail Prokopenko  
CSIRO Information and Communication Technology Centre  
Locked Bag 17  
North Ryde, NSW 1670, Australia  
{Oliver.Obst | Rosalind.Wang | Mikhail.Prokopenko}@csiro.au

## Abstract

*We investigate the problem of identifying anomalies in monitoring critical gas concentrations using a sensor network in an underground coal mine. In this domain, one of the main problems is a provision of mine specific anomaly detection, with cyclical (moving) instead of flatline (static) alarm threshold levels. An additional practical difficulty in modelling a specific mine is the lack of fully labelled data of normal and abnormal situations. We present an approach addressing these difficulties based on echo state networks learning mine specific anomalies when only normal data is available. Echo state networks utilize incremental updates driven by new sensor readings, thus enabling a detection of anomalies at any time during the sensor network operation. We evaluate this approach against a benchmark – Bayesian network based anomaly detection, and observe that the quality of the overall predictions is comparable to the benchmark. However, the echo state networks maintain the same level of predictive accuracy for data from multiple sources. Therefore, the ability of echo state networks to model dynamical systems make this approach more suitable for anomaly detection and predictions in sensor networks.*

## 1. Introduction

We investigate the problem of identifying anomalies in monitoring critical gas concentrations using a sensor network in an underground coal mine. Since the 1980s, electronic gas monitoring sensor networks have been introduced in the underground coal mining industry. One of the main problems in this domain until today is a provision of mine specific anomaly detection. The ventilation monitoring systems have high false alarm rates, a potentially costly problem to mining operations, as alarms indicate potentially explosive gas concentrations and mines have to be evacuated in case of an alarm. Systems currently in use are based on flat line thresholds, and the natural, periodic variation

in gas concentrations increase the number of false alarms. Furthermore, an intrinsic weakness of the current systems is their ignorance of spatial relations between data gathered at different sensor network nodes. These spatial relationships between data could identify anomalies missed by individual sensors. Conversely, the spatial relationships could explain away the anomalies identified by the individual gas sensors, thus avoiding false alarms. This weakness to handle complex relations lies mainly in the way data from sensor nodes is handled.

Existing systems integrate and interpret incoming data in accordance with a manually determined set of rules, produce a risk profile, and autonomously initiate a response to a breach of these rules. A problem with this approach is that no clear-cut definition of abnormal situations with respect to the concentrations of different gases exist, so that it is difficult to produce a good set of rules. The underground coal mining industry has been struggling with the issues of mine-based moving threshold levels for critical gases since the introduction of electronic gas monitoring systems. No satisfactory, scientifically validated methodology is currently in use, providing a mine with its own specific moving threshold levels. Best guess estimates, universal rules-of-thumb and experience-based trigger points are the industry norm [21].

In this paper we investigate a method for identifying anomalies in sensor data that supports mine-specific anomaly detection with no previous definition of abnormal situations. We regard the gas concentration data as a dynamic system, where future values depend on current and past values. Methods like feedforward neural networks, employed for processing time series data however implicitly assume a functional dependency between their current input and the output. To consider a history of inputs, previous values have to be buffered outside the mechanism or explicitly coded into it. Other approaches, such as recurrent neural networks, are able to compute outputs as a function over the input history, but have traditionally only rarely been used in practice due to long training times and intricate set up. Fur-

thermore, many approaches require all input data of a time step to be present before it is possible to compute output values, which is in conflict with sensor network methodology where new data can arrive at any time.

To address these practical and methodological questions, we present an approach for anomaly detection in sensor networks based on echo state networks [7]. Echo state networks (ESNs) are a particular type of recurrent neural networks and have the ability to model dynamic systems. Unlike common recurrent neural networks, they do not suffer from the problem of slow convergence in training.

We show that our approach using ESN leads to results equivalent to an approach using Bayesian networks with an explicit encoding of the history of inputs. Our anomaly detection works with real-world data, and requires no complicated parameter tuning. These features are important in practice when anomaly detection is performed on sensor nodes in a large network.

The main contributions of this paper are:

- We present an approach to discover anomalies in sensor network data using echo state networks (Section 3). Our approach supports an automatically learned, mine-specific anomaly detection of gas concentrations.
- We evaluate our approach in the domain introduced above – gas monitoring in underground coal mines – against a benchmark using Bayesian networks with an explicit encoding of the input history (Section 4). Our domain is a real-world problem, an existing coal mine sensor network with 27000 sensors distributed over many nodes and several kilometers. Our data includes gas concentrations of four different gases from a single location with more than 68000 sensor readings each.
- We provide results of extensive experiments (Section 5) using echo state networks for anomaly detection under conditions with lower amounts of memory, computation and training data used. These results have practical implications for the use of echo state networks in sensor networks.

Sections 6 and 7 contain a discussions of our results and related work, respectively. Section 8 concludes the paper.

## 2. Identifying anomalies as novelty detection

Anomaly detection in environmental monitoring is one of the important applications of sensor networks. There are three fundamental approaches to creating a model of normality and abnormality, and to detecting anomalies in data [6]: (a) detecting anomalies with no prior knowledge of the data (by e.g. unsupervised clustering), (b) detecting anomalies when a full model of the data can be built (supervised

classification), and (c) detecting anomalies when only normal (or only abnormal) data is available to create a model. To follow an approach of type (b), a fully labeled set of data would be required, which is, unfortunately, not always available in our coal mining domain.

### 2.1. Anomalies in coal mine sensor data

The data we have available support an approach of type (c) above, where we use data to create a model of normality and regard deviations from the model as abnormal. Approaches of this type are regularly called novelty detection methods in the literature [6, 20].

In practice, situations with ventilation data identified as abnormal can be anything from situations with simple explanations and no special action required (e.g. a door unintentionally left open, machinery moving in the mine), changes in the mine topology, or also by gas evaporations or leakages, which could require the evacuation of the mine. Obviously, the site personnel is interested in reducing the number of false alarms caused by the flat-line thresholds currently in use.

In our particular case, each node in the sensor network monitors several different kinds of gases in order to ensure safety and productivity in a coal mine. The data used in this paper is real-world data gathered from a deployed coal mine sensor network. In this network, each sensor node measures gas concentrations at a 30 seconds interval of the four different gases methane ( $\text{CH}_4$ ), carbon dioxide ( $\text{CO}_2$ ), carbon monoxide ( $\text{CO}$ ) and oxygen ( $\text{O}_2$ ). The sensor nodes in the coal mine pass on all data to a central node, so that the problem of a distributed computation is currently not pressing for us.

In general, the problem is to detect an abnormal event distributed over different sensors, without a priori knowledge of what exactly “abnormal” means, and an approach to automatically learn mine-specific thresholds certainly simplifies the current way of anomaly detection. For the time being we regard anomalies in the coal mine sensor network as irregular patterns in multiple time series, i.e. a combination of  $\text{CH}_4 - \text{CO}_2 - \text{CO} - \text{O}_2$ .

## 3. Anomaly detection using ESN

The data in our domain typically exhibit both cyclical and chaotic dynamics. We use the 68194 readings for each sensor type at one node, collected during a period of slightly over 3 weeks when no abnormality was identified by the system in use.

With only normal data available, the basic idea of our approach is as follows: using the available data, we create a dynamical model of the gas concentration. During runtime, anomalies are computed as deviations of the incoming data

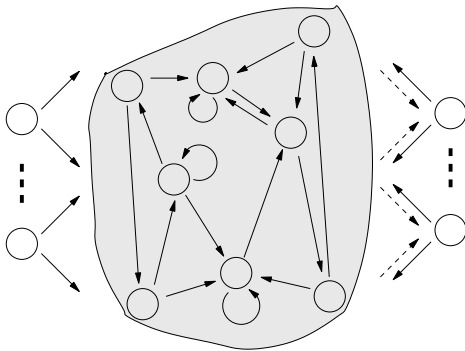
from the predictions of the model. The initial threshold for deviations has to be set by the site personnel; once statistics of false and correct alarms have been collected, a threshold for a specified false alarm rate  $\lambda$  can be set using the Neyman-Pearson test:

$$\lambda = \frac{p(\text{false alarm})}{p(\text{correct alarm})} \quad (1)$$

### 3.1. Modeling gas concentrations using echo state networks

A possible approach predicting gas concentrations is recurrent neural networks, as these are able to model dynamic non-linear systems. Traditionally, however, recurrent neural networks have only rarely been used in practice, because of the slow convergence on training methods even for a small number of neurons.

ESN provide a specific architecture and a training procedure for recurrent neural networks that aims to solve the problem of slow convergence [7]. The basic idea is to internally use a random recurrent neural network as a dynamic reservoir (DR) of neurons. Input and output units are connected to the DR using random connections. To train the network, a training signal is fed into input units, while the teacher signal is written to the output units. The weights of the output units are computed as the linear regression weights of the teacher signal on the states of the DR. That means, instead of changing all the weights of the network, only weights of the output units are updated during training (see also Figure 1).



**Figure 1. The basic ESN architecture. The dashed connections to the output units are the only connections changed during training.**

**Brief formal description** More formally, an ESN consists of  $K$  input units,  $N$  internal units and  $L$  output units. Then, activations of input, internal, and output

units at time step  $t$  are  $\mathbf{u}(t) = \langle u_1(t), \dots, u_K(t) \rangle$ ,  $\mathbf{x}(t) = \langle x_1(t), \dots, x_N(t) \rangle$ , and  $\mathbf{y}(t) = \langle y_1(t), \dots, y_L(t) \rangle$ , respectively. Connection weights between units are kept in four connection matrices. There are  $K \times N$  weights in the input weight matrix  $\mathbf{W}^{in} = (w_{ij}^{in})$ ,  $N \times N$  weights in the internal weight matrix  $\mathbf{W} = (w_{ij})$ ,  $L \times (K + N + L)$  weights in the output weight matrix (i.e. direct connections from input to output units are possible), and  $L \times N$  weights in a matrix  $\mathbf{W}^{back} = (w_{ij}^{back})$  for connections projecting back from the output to internal units.

The activation of internal units is calculated as

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{W}^{in}\mathbf{u}(t+1) + \mathbf{W}\mathbf{x}(t) + \mathbf{W}^{back}\mathbf{y}(t)), \quad (2)$$

with  $\mathbf{f} = \langle f_1, \dots, f_N \rangle$  the output functions of the internal units – a sigmoid function for the experiments in this paper. Similarly, the output is computed as

$$\mathbf{y}(t+1) = \mathbf{f}^{out}(\mathbf{W}^{out}(\mathbf{u}(t+1), \mathbf{x}(t+1), \mathbf{y}(t))), \quad (3)$$

with  $\mathbf{f}^{out} = \langle f_1^{out}, \dots, f_L^{out} \rangle$  the output functions of the output units and  $(\mathbf{u}(t+1), \mathbf{x}(t+1), \mathbf{y}(t))$  the concatenation of input, internal, and previous output activation vector [7].

**Motivation** ESN share the idea of using randomly connected internal units as DR together with the related Liquid State Machines [13]. Therefore, both approaches are also called “reservoir computing methods”. Using the DR, input is non-linearly projected into an high-dimensional state space. Connected to the DR, linear output units can be trained and used to provide outputs at any time (see also [13]). Because the random DR is not task-specific, the same ESN can be used to solve different tasks in parallel: the only connections that have to be trained are the ones for the task-specific output units, and there could be several of them connected to the same DR.

According to our knowledge, ESN have previously not been applied in sensor network settings yet. However, an approach for anomaly detection using ESN seems promising, because (a) ESN are capable to mimic dynamic systems, (b) have shown to predict chaotic time series better than any previous technique [10], and (c) ESN could help to reduce synchronization issues, because they can provide outputs at any time. In addition to solving a practical problem, our investigation is intended to advance the applicability of ESN to sensor networks in general.

With the different outputs of the sensors, different approaches to prediction are possible: (a) predict the concentration of a gas using previous concentrations of that gas, and (b) predict the concentration of a gas using previous readings of all available sensors at the same location.

### 3.2. A benchmark using Bayesian networks

Bayesian networks [18] are an alternative approach to detecting anomalies in sensor network data. To compare the performance of our approach, we train a Bayesian network to model and predict the same time series.

In a Bayesian network, each random variable is independent of its non-descendants in the graph given the state of its parents. This independence can be exploited to reduce the number of parameters needed to characterize the network. Thus it is possible to efficiently compute posterior probabilities given some evidence or observations. One set of probability parameters are encoded for each variable, in the form of the local conditional distribution given the variables parent. Using the independence statements encoded in the network, the joint probability distribution is uniquely determined by these local conditional distributions [3, 11]. Inference in a Bayesian network can be used to directly perform anomaly detection by computing likelihoods. Here, however, we use the inference to do predictions of sensor readings for comparison purposes.

We use capital letters such as  $X, Y$  for names of random variables, and lower cases  $x, y$  for values taken by these variables. A set of variables such as  $\{X_1, X_2, X_3\}$  are written as  $X$ , likewise, a set of values such as  $\{x_1, x_2, x_3\}$  are written as  $x$ . Thus,  $x$  are values taken by  $X$ . Let  $P(\mathbf{U})$  be a joint probability distribution over  $\mathbf{U} = \{X_1, \dots, X_k\}$ , where  $X_i$  is a random variable expressed by a node of the network. A Bayesian network for  $\mathbf{U}$  is a pair  $B = \langle G, \Theta \rangle$ . The first component,  $G$ , represents the graph structure of the network.  $G$  is an acyclic directed graph whose nodes correspond to the random variables  $X_1, \dots, X_k$ , and whose edges represent direct dependencies between the variables. The second component,  $\Theta$ , represents the set of conditional probabilities that quantify the nodes of the network. It contains a set of parameters  $\theta_{X_i|\Pi_{X_i}} = P_B(X_i|\Pi_{X_i})$  for each node  $X_i$ , where  $\Pi_{X_i}$  denotes the set of parents of  $X_i$  in  $G$ . A Bayesian network  $B$  defines a unique joint probability distribution over  $\mathbf{U}$  given by

$$P_B(\mathbf{U}) = \prod_{i=1}^k P_B(X_i|\Pi_{X_i}) = \prod_{i=1}^k \theta_{X_i|\Pi_{X_i}} \quad (4)$$

In Bayesian networks in general, the learning process is to estimate the parameter set  $\Theta$  as well as to find the structure of the network,  $G$ . The objective in the learning is to find a  $B = \langle G, \Theta \rangle$  that “best describe” the probability distribution over the training data [19].

### 3.3. Overview of our approach

Section 4 contains the details of our approach:

- In Section 4.1 we describe the details of training the ESN to detect anomalies within the coal mine sensor network data. The task of the ESN in this approach is prediction rather than the entire anomaly detection, and an anomaly is detected when there is a larger than specified deviation from the prediction.
- For a benchmark, we set up a Bayesian network for predictions, using the same data sets (Section 4.2).
- Going from this benchmark, we change the setup and the training of the ESN to evaluate scalability of our approach. Dependent on available hardware and training data, the evaluations described in Sections 4.3 and 4.4 are relevant for different sensor network nodes.

## 4. Details

### 4.1. Predictions based on single sensors

Predicting gas concentrations based on readings of only that type of gas involves using a single ESN for each type of gas. We aim to use the default configuration of the ESN Matlab toolbox [9] without major changes: the ESN has 30 randomly connected internal units, with a connectivity of 10% and a spectral radius of 0.5 for the connection matrix. The weights of the input units are random in the range  $[-0.1, 0.1]$ . The units we are using are standard additive-sigmoid neurons. To train the ESN, we split the data into a training set and a test set (50% each). The ESN is trained by “teacher forcing”, i.e. the input units are driven by the training data, while the output unit is set to the desired output. During this process, the network state is stored for each of the steps. From this data, we compute new output weights (for more details on this procedure, we refer to [8]) and the ESN can be used. In case of a one-step prediction of a single gas concentration, we use two input units: one input unit is just driven by a sequence of ones in order to avoid symmetries, and the other unit is driven by the actual gas concentration at time  $t$ . The output unit is teacher-forced by the concentration of the same gas at time  $t + 1$ .

To evaluate the performance of the trained ESN, we calculate the normalized root mean square error (NRMSE) of the  $n$  predictions  $p$  of the ESN against the test data  $z$  by

$$\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^n (z(i) - p(i))^2}{n \text{var}(z)}}, \quad (5)$$

where  $\text{var}(z)$  is the variance of the test data.

In both the training and the exploitation of the ESN, it is a common procedure to not use the first 100 values of a sequence to “wash out” the effects of the initial network state (see [8], the actual number depends on the network size).

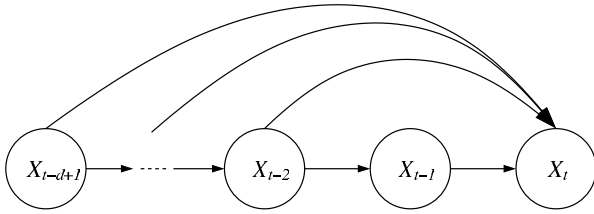
All NRMSE of ESN reported in Section 5 are actually average errors of 10 experiments using the same setup, as single results may slightly vary due to the random connections weights of the DR.

## 4.2. Benchmark using a Bayesian network

For a comparison of the prediction capabilities of the ESN, we construct a Bayesian network with explicit access to the history of inputs. Let a single variable time series be  $X = \{x_1, x_2, \dots, x_N\}$ , where  $N$  is the total number of data points in the series. We can embed this data in a  $d$ -dimensional phase space as the following [17]:

$$\mathbf{y}_t = (x_t, x_{t-1}, \dots, x_{t-d+1}), \quad (6)$$

where  $d$  is the embedding dimension, and  $t = d, d + 1, \dots, N$ .



**Figure 2. The model used for learning and predicting data in embedded phase space. It is a  $(d - 1)$ -th order Markov model presented in Bayesian network representation.**

Figure 2 shows the model used to learn this data. The network is constructed from the underlying dependencies in a time series, that is the data at time  $t$  is dependent on the data at time  $t - 1, \dots, t - d + 1$ . The joint distribution of the model is

$$P(\mathbf{U}) = P(X_t | X_{t-1}, \dots, X_{t-d+1}) \times \prod_{k=1}^{d-2} P(X_{t-k} | X_{t-(k+1)}), \quad (7)$$

where  $\mathbf{U} = \{X_t, X_{t-1}, \dots, X_{t-d+1}\}$ . All the nodes are modeled as one dimensional Gaussian. For example, a Bayesian network model of Fig. 2 with  $d = 3$  has the dependencies as  $X_{t-2} \rightarrow X_{t-1} \rightarrow X_t$  as well as  $X_{t-2} \rightarrow X_t$ . The joint distribution of the model is  $P(\mathbf{U}) = P(X_t | X_{t-1}, X_{t-2})P(X_{t-1} | X_{t-2})P(X_{t-2})$ , where each  $P(\cdot)$  is a Gaussian or a conditional Gaussian distribution.

Since we provide the structure of the Bayesian network, only the parameter set  $\Theta$  needs to be learned. The Maximum Likelihood (ML) algorithm [14, 16] is thus used to

estimate  $\Theta$ . In the ML estimator, the likelihood function,  $p(x|\theta)$ , is treated as a function of  $\theta$  for fixed  $x$ , where  $x_j^t$  is the  $j$ -th data sample for the node  $X_t$  in the Bayesian network. This likelihood function can be used to evaluate the choices of  $\theta$ . The ML estimator choose the value of  $\theta$  that maximizes the probability of the data  $\mathbf{x}$ :

$$\hat{\theta}_{ML} = \arg \max_{\theta} p(\mathbf{x}|\theta). \quad (8)$$

To compare the quality of the predictions computed by the ESN, we use the learned Bayesian network to do prediction as well. In a Bayesian network, the probability distribution of a child node can be computed given the values of its parents. In our case, this is the prediction of values in  $X_i$  in Figure 2 given the values of  $\{X_{t-1}, \dots, X_{t-d+1}\}$ .

**Benchmark data sets** We create four different training/test sets with four different gases each from the basic time series data to compare ESN and the Bayesian network benchmark: set 1 containing a 50/50 split of raw sensor data, set 2 containing a 50/50 split of smoothed sensor data, set 3 containing a 50/50 split of raw sensor data with 5% white noise added, and set 4 containing a 10/90 split of smoothed sensor data.

## 4.3. Predictions based on multiple sensors

Instead of using a separate ESN for each of the sensor inputs, it is possible to connect all available sensors at the same location to input units of the same ESN and compute predictions of single gases by using all of the available sensor information. We use the same ESN setup as above, but instead of two input units we use six: again, the first unit is driven by a sequence of constant values, and the other five units get the sensor readings for  $\text{CH}_4$ ,  $\text{CO}_2$ ,  $\text{CO}$ ,  $\text{O}_2$ , and the atmospheric pressure. The output units attached to this ESN are trained to predict these sensor readings using the combined data sets from the single input prediction benchmarks. For each required prediction, a different output unit can be attached and trained to the same DR of the ESN in parallel. Using just one neural network instead of four in a sensor network node reduces the amount of memory and computation used, e.g. for moderately equipped sensor network nodes. A further application of using several inputs and outputs to the same ESN would be to create a distributed model of neighbor sensor nodes: if, in each sensor node, the same ‘‘random’’ reservoir is created, neighbors just need to communicate the output weight matrix to transmit the model. The local ESN could then be trained to perform predictions on readings of its own and neighbor sensors.

**Multiple sensors data sets** We use the four different training/test sets with four different gases from above to

compare the approach of using single ESN for each gas against the approach of using just one ESN for all gases.

#### 4.4. Trade-offs between ESN parameters

The final two evaluations of ESN using underground coal mine sensor network data involved changing two of the basic variables to neural network training in each case: (1) the amount of training/test data used, and the number of internal units of the ESN, and (2) the amount of noise added to the data, and the number of internal units of the ESN, respectively.

For (1), we vary the amount of training/test data used from the 50/50 default in steps of 5 percentage points down to 5/95, while we vary the number of internal units from 30 down to 3 in steps of 3 units. For (2), we artificially add white noise to the data, in 10 steps from 1% to 10% of the maximum value of the training/test data, while we vary the number of internal units again.

**Trade-offs data sets** We use the smoothed data training/test sets with four different gases to compute trade-offs of the respective parameters (data with added white noise if applicable).

### 5. Results

For each gas in each data set, the prediction errors (NRMSEs) of the respective approach have been computed. As explained above, in case of ESN predictions this involved computing the average NRMSE of 10 freshly created ESN for each experiment. For some of the experiments we present plots of the predictions or prediction errors, additionally.

#### 5.1. Predictions based on single sensors

The ESN performed equally well to the Bayesian network in all of the prediction tasks. Table 1 shows the NRMSEs of runs of both the ESN and the Bayesian network for the four benchmark data sets. None of the tests shows a significant difference between the two techniques. Figure 3 shows plots of the test data vs. predictions of the ESN and absolute errors for the smoothed data set.

#### 5.2. Predictions based on multiple sensors

Results from the last subsection show that the anomaly detection approach using ESN predictions gives results equivalent to using BN predictions. From here, we change the setup of the ESN and compare the results against the ESN benchmark from above. We use one ESN to predict all

**Table 1. Comparison of prediction errors (NRMSE) between ESN and Bayesian network.**

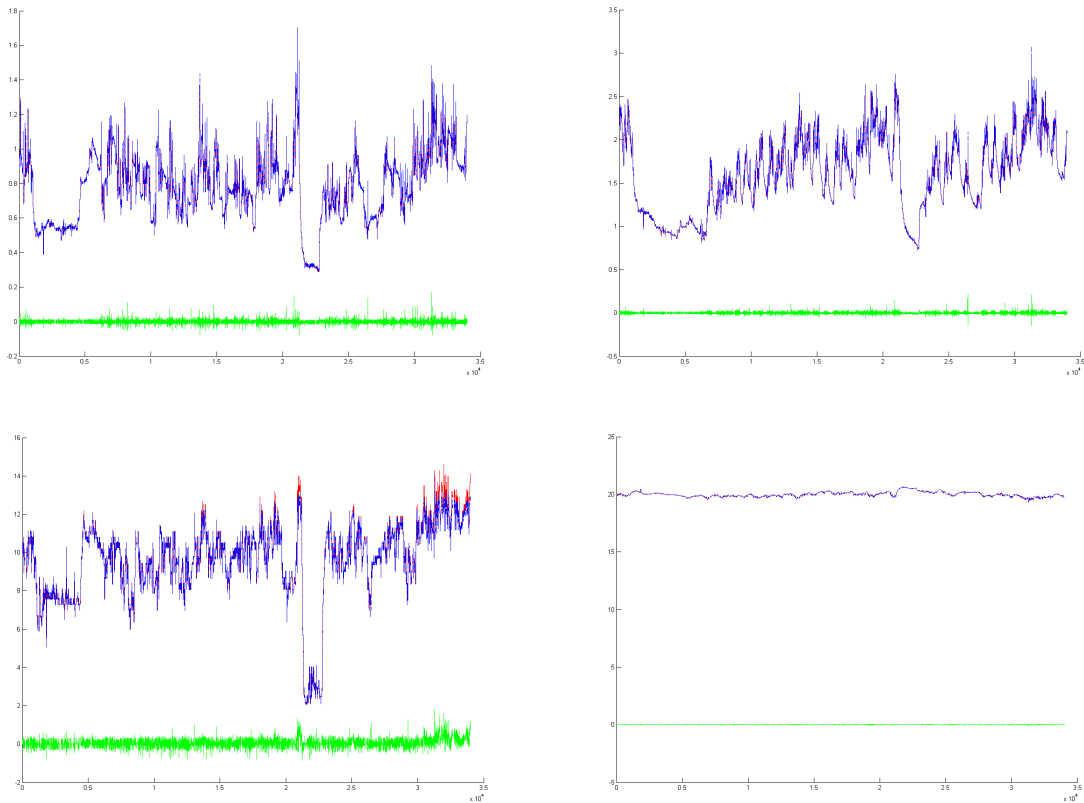
Data set		NRMSE <sub>BN</sub>	NRMSE <sub>ESN</sub>	Diff.
Smoothed	CH <sub>4</sub>	0.0404	0.0403	0.0001
	CO <sub>2</sub>	0.0210	0.0198	0.0012
	CO	0.0468	0.0511	-0.0043
	O <sub>2</sub>	0.0302	0.0302	0.0
Raw	CH <sub>4</sub>	0.6894	0.0678	0.6216
	CO <sub>2</sub>	0.0253	0.0248	0.0005
	CO	0.0891	0.0957	-0.0066
	O <sub>2</sub>	0.1214	0.1237	-0.0023
Raw + 5% noise	CH <sub>4</sub>	0.2598	0.2579	0.0019
	CO <sub>2</sub>	0.2110	0.2108	0.0002
	CO	0.2399	0.2422	-0.0023
	O <sub>2</sub>	0.9927	0.9941	-0.0014
Smoothed 10/90	CH <sub>4</sub>	0.0374	0.0404	-0.003
	CO <sub>2</sub>	0.029	0.0448	-0.0158
	CO	0.0520	0.15	-0.098
	O <sub>2</sub>	0.0348	0.0429	-0.0081

four gases instead of a separate ESN for each of the gases. Table 2 gives the complete results, Figure 4 shows graphs for CH<sub>4</sub> and CO<sub>2</sub> predictions. In most cases, predictions involving just one ESN totally instead of one ESN for each gas have an equivalent prediction quality, with two exceptions: when all available data is fed into the ESN, the prediction error of the raw CO series increases by a factor of 3. In the case of using only 10% of the data for training, the CO series prediction causes almost only half the error when using all available sensor inputs.

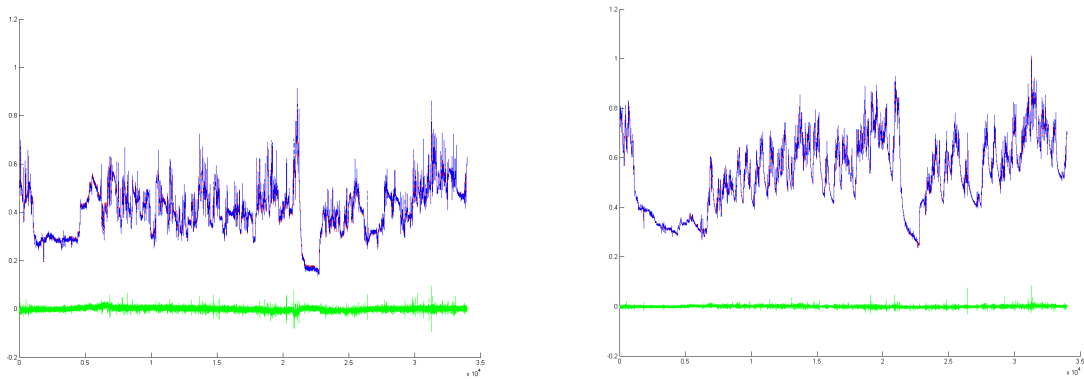
#### 5.3. Trade-offs between ESN parameters

To further explore ESN-based anomaly detection, we trade off the number of neurons in the dynamic reservoir and the amount of training data (see Table 3 and the plots in Figure 5). The prediction quality for higher amounts of training data does not vary significantly with the numbers of internal units. For small amounts of training data, the best prediction performance is achieved with a smaller number of internal units. The prediction quality does not vary significantly with the number of internal units.

To verify this behavior with a time series where a higher number of internal units is required, we ran an experiment with an artificial time series created by the function  $f(t) = \sin(\frac{2\pi t^2}{N})$ , with  $t = 0, \dots, N$ . We have chosen  $N = 68193$  to create a series of same length as our test data and added 5% white noise (results in Table 4). As for our sensor data, for any fixed number of units, the results get better with an increasing amount of training data. For low amounts of training data the results get worse by using



**Figure 3. Results of training of predicting sensor readings. The red line shows the teacher signal, the blue line the prediction, and the green line at the bottom of each graph the difference between both. Top row: gas concentrations of  $\text{CH}_4$ ,  $\text{CO}_2$ . Bottom row:  $\text{CO}$  and  $\text{O}_2$  .**



**Figure 4. Prediction and difference to sensor readings of  $\text{CH}_4$  (left) and  $\text{CO}_2$  (right) based on previous values of all gases and the atmospheric pressure.**

**Table 2. Comparison of prediction errors (NRMSE) between separate ESN for each gas (ESN<sub>1</sub>) and one ESN for all gases (ESN<sub>4</sub>).**

Data set		NRMSE <sub>ESN<sub>1</sub></sub>	NRMSE <sub>ESN<sub>4</sub></sub>	Diff.
Smoothed	CH <sub>4</sub>	0.0403	0.0437	-0.0034
	CO <sub>2</sub>	0.0198	0.0247	-0.0049
	CO	0.0511	0.0489	0.0022
	O <sub>2</sub>	0.0302	0.0294	0.0008
Raw	CH <sub>4</sub>	0.0678	0.0722	-0.0044
	CO <sub>2</sub>	0.0248	0.0273	-0.0025
	CO	0.0957	0.0895	0.0062
	O <sub>2</sub>	0.1237	0.3624	-0.2387
Raw + 5% noise	CH <sub>4</sub>	0.2579	0.2634	-0.0055
	CO <sub>2</sub>	0.2108	0.2136	-0.0028
	CO	0.2422	0.254	-0.0118
	O <sub>2</sub>	0.9941	0.9935	0.0006
Smoothed 10/90	CH <sub>4</sub>	0.0404	0.0599	-0.0195
	CO <sub>2</sub>	0.0448	0.0356	0.0092
	CO	0.15	0.0826	0.0674
	O <sub>2</sub>	0.0429	0.0458	-0.0029

more units. When “enough” training data is used, adding more units improves the results.

When varying the noise for different numbers of internal units, higher noise leads to bigger errors. For a fixed level of noise, the number of units does not play a significant role for the prediction error (Table 5 and Figure 6). The same (high) amount of training data has been used in this experiment.

## 6. Discussion

We have shown that an approach for anomaly detection based on ESN can achieve results of equal quality comparing to an approach using Bayesian networks with an explicit encoding of past inputs. For the tests involving ESN, we used the Matlab toolbox for ESNs almost “as is”, with no parameter tuning and the training function and the number of input/output units changed for our time series. We have also run experiments (not previously mentioned in this paper) with a higher number of internal units with no significant improvements of prediction performance using the same data. The ESN Matlab *toolbox* learned and predicted the time series significantly faster than the open source Bayesian network Matlab *toolbox* [15] we used. However, neither of the toolboxes is optimized for performance, and our own implementations show that computation times of the Bayesian network can be drastically reduced by omitting code not used in our application. Because of this, we do not compare execution times of the two approaches here: the results would not be meaningful for sensor network applications, because practical implementations would be tailored to sensor nodes and a specific purpose anyway. To

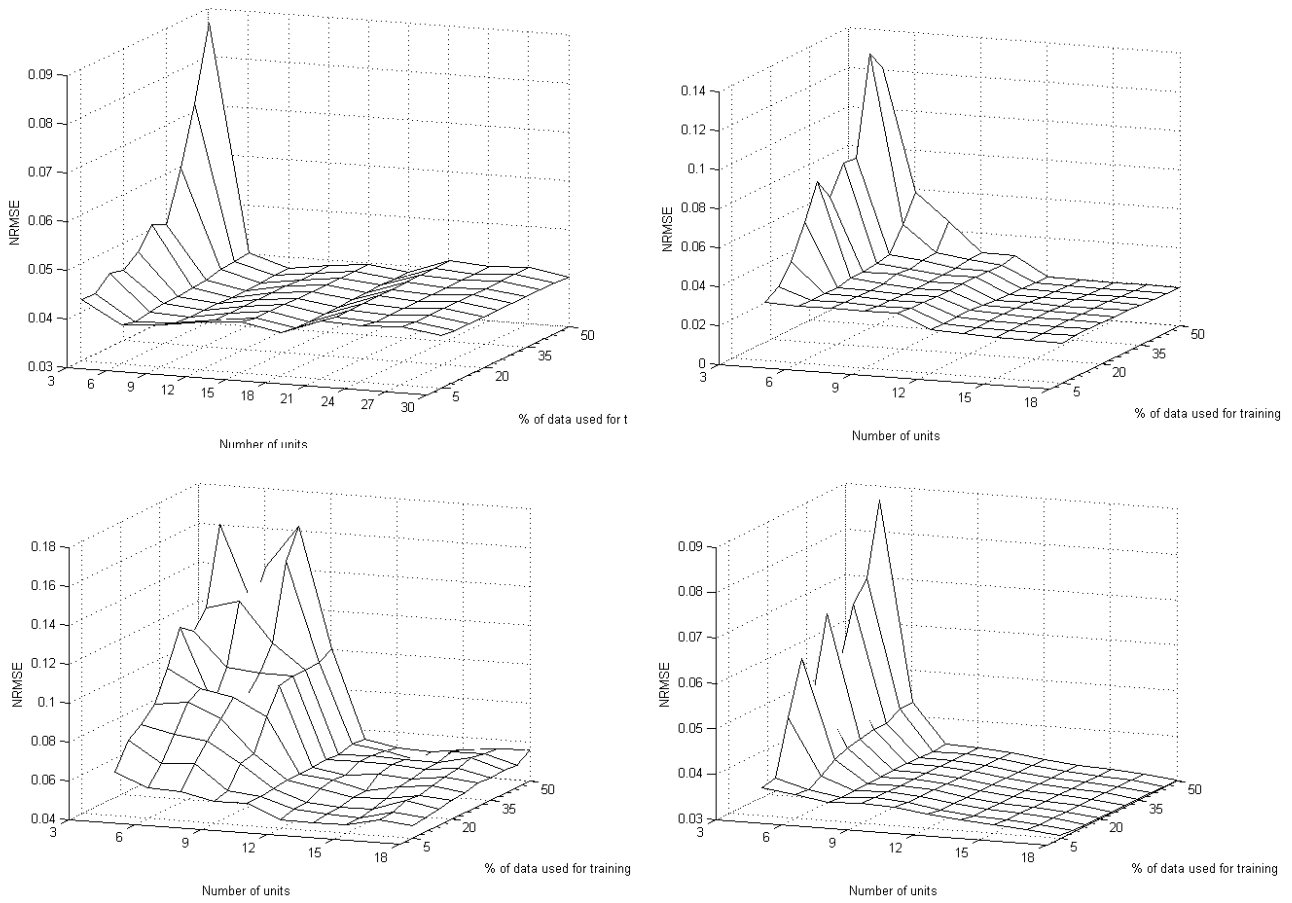
nevertheless provide a rough idea on the time used with both approaches we can say that training and predicting our time series consisting of over 68000 data points (split 50/50) takes less than half a minute on a 2 GHz Intel Core 2 Duo, 2 GB of RAM using ESN, and roughly half an hour using Bayesian networks.

The results of the ESN predictions using all available sensor data as input seem surprising at first: even though *more* of the available information is actually used, the prediction quality decreases in one case, and remains equivalent in almost all others. Only in one case, the additional data increases the quality of the predictions. A likely explanation for the decrease is that the single gases do not provide much information for concentrations of other gases (also due to the fact that natural air consists of more gases than the measured four). The learning task for the ESN gets actually harder because it has to filter the right data “streams” among several possible ones. In most cases, the prediction quality remains the same, so that using one ESN for all sensors on a node appears viable.

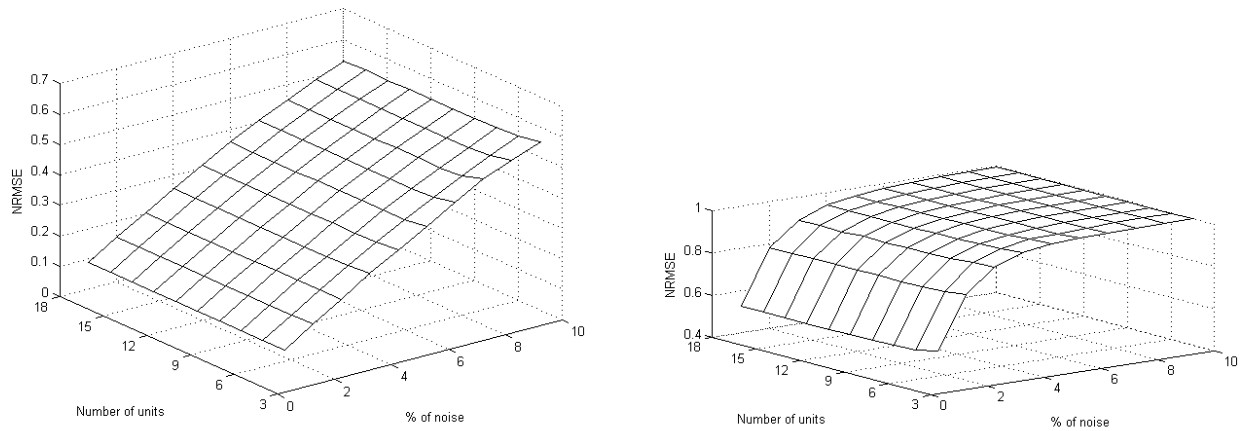
Similarly surprising is the result of the experiments computing relations between the number of internal units and the training data: starting from the standard setup, a decrease of training data decreases the predictive performance at the low end as expected. However, a simultaneous decrease of internal network units helps to maintain the error close to the original error. This is surprising because ESN are based on the idea of using a *large* pool of units — for our training data however, a low amount of internal units in conjunction with less training data works almost equally well. The reason for the good results with only few internal units may lie within the data we used (e.g. due to a lower than expected dimensionality of the data). Providing an explanation for the increase of the error by increasing the number of units when using a low amount of training data is a subject of future work. For practical purposes, this means that a low amount of available training data could be compensated by using a smaller ESN. Using an artificial time series, we have also shown that, for different types of data, the number of internal units has to be bigger than just a few nodes, and actually plays an important role for the prediction quality. The number of connection weights updated during training grows linearly with the number of internal units.

## 7. Related work

Anomaly detection has different applications in sensor networks. Examples are varying sample and communication rates, detection of unusual behavior of environment or sensor network hardware, or compression of data. In some cases, anomalies are taken as a sign for a possible intrusion into the network. Varying communication rates can help saving energy, with the basic idea that normal, i.e. pre-



**Figure 5. NRMSEs for varying numbers of neural units and varying training/test data sizes. Top row: prediction errors for CH<sub>4</sub> and CO<sub>2</sub>. Bottom row: CO and O<sub>2</sub>.**



**Figure 6. NRMSEs for varying numbers of neural units with varying noise levels. Left: CH<sub>4</sub> Right: O<sub>2</sub>**

**Table 3. NRMSEs for ESN using a varying number of units in the dynamic reservoir with a varying training/test data size. Each NRMSE is an average of 10 trials.**

	3 units			15 units			30 units		
	5%	25%	50%	5%	25%	50%	5%	25%	50%
CH <sub>4</sub>	0.0425	0.0404	0.0406	0.0468	0.0397	0.0403	0.0868	0.0396	0.0402
CO <sub>2</sub>	0.0299	0.0295	0.0207	0.0789	0.0280	0.0211	0.1208	0.0298	0.0197
CO	0.0624	0.0516	0.0462	0.1027	0.0535	0.0506	0.1499	0.0554	0.0562
O <sub>2</sub>	0.0362	0.0340	0.0306	0.0526	0.0336	0.0302	0.0871	0.0336	0.0302

**Table 4. NRMSEs for ESN with an artificially created time series using a varying number of units in the dynamic reservoir with a varying training/test data size.**

	3 units			30 units			300 units		
	5%	25%	50%	5%	25%	50%	5%	25%	50%
$\sin(2\pi t^2/68193) + 5\%$ noise	2.1529	1.010	1.0091	7.9781	0.7173	0.7041	61.7641	0.2176	0.1972

dictable data is not communicated and only abnormal (read “interesting”) data consumes energy by being transmitted.

In [20], the authors perform anomaly detection using a modified time-based Multi-Layer Perceptron (MTBMLP). These neural networks consist of multiple time-based multi-layer perceptrons (MLPs) connected to a single MLP. Time-based MLPs are feedforward neural networks with multiple layers, where inputs are time-delayed values. They are used as predictors for functions  $y = f(x)$ . The MTBMLP is used for prediction similarly to our approach, with the differences that (1) the predictions are dependent on the values of the previous step only, and (2) learning is also used during runtime to adapt to long-time changes of the environment. As mentioned in the introduction, (1) is a problem inherent to feedforward neural networks. Point (2), the online-learning of changes in the long-time behavior of time-series is an interesting feature of the approach, but it depends on the application if it is a desired one. There is no principal reason preventing online-learning with our approach.

A distributed approach to model sensor network data based on kernel linear regression is presented in [4]. Nodes in the sensor network fit a global function to local measurements. Communication in the sensor networks is reduced by transmitting (constraints on) the model parameters, rather than data. In our approach, the dynamic reservoir of local ESN can be regarded as a kernel, and together with the learned output weights describes a non-linear, dynamic model of the data of the respective node. This model can be transferred to neighbour nodes in a compact way by using the same “random” reservoir on each node and just transmitting the output weights, as already mentioned above. With the functional descriptions of models in [4], it is however easier to compute different single points of a time series.

The idea to see intrusion into a computer system as an anomaly was described already 20 years ago [1]; likewise, there are also approaches as the one in [22] applying the basic idea on a sensor network. The work in [2] can be described as a special type of anomaly and intrusion detection for wireless sensor networks in adversarial environments. Here, anomalies in localization information is assumed to be caused by intruders, and the developed scheme helps to detect anomalies in the localization of single nodes.

Causes of anomalies can be detected by Kim and colleagues [12]. They use Bayesian networks to detect anomalies in a monitored office environment, and they use them to also model causes of these anomalies. In their approach, it is possible to trace and visualize sensor relations, a feature that is supported by bayesian, but not by neural networks.

In [5], the authors use dynamic Bayesian networks to detect anomalies in environmental sensor network data for quality assurance and control. Two different strategies for detecting anomalies are developed in this paper, with one of them also able to indicate likely causes of faulty measurements.

## 8. Conclusions and further work

We presented a systematic evaluation of an anomaly detection technique for sensor networks based on ESN. Our approach enables an automatically learned, mine-specific anomaly detection of gas concentrations. While solving a real-world problem, we compared our approach against a benchmark with an explicit encoding of the input history. We provided results of extensive experiments using echo state networks for anomaly detection under conditions where lower amounts of memory, computations and train-

**Table 5. NRMSEs for ESN using a varying number of units with a varying amount of noise.**

	3 units			15 units			30 units		
	1%	5%	10%	1%	5%	10%	1%	5%	10%
CH <sub>4</sub>	0.0899	0.3188	0.5530	0.0877	0.3110	0.5327	0.0869	0.3095	0.5302
CO <sub>2</sub>	0.0597	0.2287	0.4198	0.0593	0.2213	0.3992	0.0591	0.2203	0.3969
CO	0.0718	0.2409	0.4291	0.0767	0.2313	0.4104	0.0777	0.2314	0.4078
O <sub>2</sub>	0.5620	0.9833	0.9979	0.5292	0.9715	0.9969	0.5249	0.9663	0.9964

ing data are used. These results have practical implications for the use of echo state networks in sensor networks.

Concluding, we believe ESN can help to solve the problem of high false alarm rates in sensor networks for underground coal mines. For future work, we plan to expand on the idea of using ESN distributively in clusters of sensor nodes, using data from different domains, as well as doing longer term predictions for anomaly detection with entirely unlabeled data.

## References

- [1] D. E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232, 1987.
- [2] W. Du, L. Fang, and P. Ning. Lad: Localization anomaly detection for wireless sensor networks. In *Proceedings of The 19th International Parallel and Distributed Processing Symposium*, Denver, Colorado, USA, Apr 2005.
- [3] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2–3):131–163, 1997.
- [4] C. Guestrin, P. Bodi, R. Thibau, M. Paski, and S. Madde. Distributed regression: an efficient framework for modeling sensor network data. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 1–10, New York, NY, USA, 2004. ACM.
- [5] D. J. Hill, B. Minsker, and E. Amir. Real-time bayesian anomaly detection for environmental sensor data. In *32nd Congress of the International Association of Hydraulic Engineering and Research (IAHR 2007)*, 2007.
- [6] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.
- [7] H. Jaeger. The “echo state” approach to analysing and training recurrent neural networks. GMD Report 148, GMD – German National Research Institute for Computer Science, 2001.
- [8] H. Jaeger. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. GMD Report 159, Fraunhofer Institute AIS, 2002.
- [9] H. Jaeger. Matlab toolbox for ESNs. [http://www.faculty.iu-bremen.de/hjaeger/esn\\_research.html](http://www.faculty.iu-bremen.de/hjaeger/esn_research.html), 2008. Last checked: 18.02.2008.
- [10] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80, 2004.
- [11] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.
- [12] S. Y. Kim, M. Imada, and M. Ohta. Detecting anomalous events in ubiquitous sensor environments using bayesian networks and nonparametric regression. In *AINA '07: Proceedings of the 21st International Conference on Advanced Networking and Applications*, pages 236–243, Washington, DC, USA, 2007. IEEE Computer Society.
- [13] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.
- [14] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.
- [15] K. Murphy. Bayes net toolbox for matlab. <http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html>, 2007. Last checked: 18.02.2008.
- [16] I. J. Myung. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47(1):90–100, 2003.
- [17] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. Geometry from a time series. *Phys. Rev. Lett.*, 45(9):712–716, Sep 1980.
- [18] J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society*, pages 329–334, Irvine, Aug 1985. University of California.
- [19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [20] G. Von Pless, T. A. Karim, and L. Reznik. Time-based multi-layer perceptron for novelty detection in sensor networks. *Machine Learning and Applications, 2004. Proceedings. 2004 International Conference on*, pages 156–163, 2004.
- [21] P. Wang, X. R. Wang, Y. Guo, V. Gerasimov, M. Prokopenko, O. Fillon, K. Haustein, and G. Rowan. Anomaly detection in coal-mining sensor data. Technical report, CSIRO, 2007.
- [22] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 275–283, New York, NY, USA, 2000. ACM.