University of Reading

A substitution of the general partial differential equation with extended polynomial networks

Conference or Workshop Item

Accepted Version

Zjavka, L., Snásel, V., Ojha, V. ORCID: https://orcid.org/0000-0002-9256-1192 and Pedrycz, W. (2016) A substitution of the general partial differential equation with extended polynomial networks. In: International Joint Conference on Neural Networks (IJCNN), 25-29 Jul 2016, Vancouver, Canada, pp. 4819-4826. doi: https://doi.org/10.1109/IJCNN.2016.7727833

A Substitution of the General Partial Differential Equation with Extended Polynomial Networks

Ladislav Zjavka, Václav Snášel, Varun Kumar Ojha
Department of Computer Science and IT4Innovations
Faculty of Electrical Engineering and Computer Science
VŠB-Technical University of Ostrava
Ostrava, Czech Republic

Witold Pedrycz
Department of Electrical & Computer Engineering
University of Alberta
Edmonton, Canada

***Abstract* — General partial differential equations, which can describe any complex functions, may be solved by means of the dimensional similarity analysis to model polynomial data relations on the basis of discrete observations. Designed new differential polynomial networks define and substitute for a selective form of the general partial differential equation using fraction derivative units to model an unknown system or pattern. Convergent series of relative derivative substitution terms, produced in all network layers, describe partial derivative changes of some combinations of input variables to generalize elementary polynomial data relations. The general differential equation is decomposed into polynomial network backward structure, which defines simple and composite sum derivative terms in respect of previous layers variables. The proposed method enables to form more complex and varied derivative selective series models than standard soft-computing techniques allow. The sigmoidal function, commonly employed as an activation function in artificial neurons, may improve the polynomial and substituting derivative term abilities to approximate complicated periodic multi-variable or time-series functions in a system model.**

Index Terms* — *partial differential equation substitution; differential polynomial network; substitution derivative sum term; multi-variable function approximation

I. Introduction

Conventional artificial neural networks (ANN) are not able to generalize input pattern elementary data relations, using only weighted sums of inputs, which describe overall similarity relationships of new presented test input patterns with the trained ones. The ANN generalization from the training data, based on the absolute interval values, may be difficult or problematic if the model has not been trained with inputs or outputs around the range covered by testing data [1]. Polynomial neural networks (PNN) decompose the Kolmogorov-Gabor polynomial (1), which can express the general connections between input and output variables of a system, into many simpler relationships, each described by low-order multi-variable polynomial processing functions (2) of single neurons.

$$Y = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ijk} x_i x_j x_k + \dots \quad (1)$$

n – number of variables $A(a_1, a_2, \dots, a_n), \dots$ *- vectors of parameters* $X(x_1, x_2, \dots, x_n)$ *- vector of input variables*

Group Method of Data Handling (GMDH) created by a Ukrainian scientist A. Ivakhnenko in 1968 [2], forms the PNN in successive steps, adding the next new layers, calculating polynomial parameters of the last layer and selecting its best neurons (nodes), while an improvement is attainable. The PNN can model highly non-linear systems as the neurons polynomial degree doubles in each following hidden layer [3].

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \quad (2)$$

Partial differential equations can model a variety of systems, which are not possible to describe unambiguously by means of unique explicit functions and may be solved e.g. by means of evolutionary strategies [4], genetic programing techniques [5] [6] or ANN [7]. Differential polynomial neural network (D-PNN) is a new neural network type, designed by L. Zjavka [8], which extends the complete PNN structure to define and solve the general partial differential equation (PDE). It produces convergent series of relative polynomial derivative terms, which can substitute for the selected PDE terms to model a searched function on account of data samples. The D-PNN operating principles differ from that of the GMDH, based on the Taylor-series expansions, however it decomposes the general PDE analogous to the PNN does the general connection polynomial (1). In contrast with the ANN each D-PNN neuron (i.e. substitution derivative term in this concept), regardless of its layer, can be directly involved (selected) in the total network output sum (PDE solution).

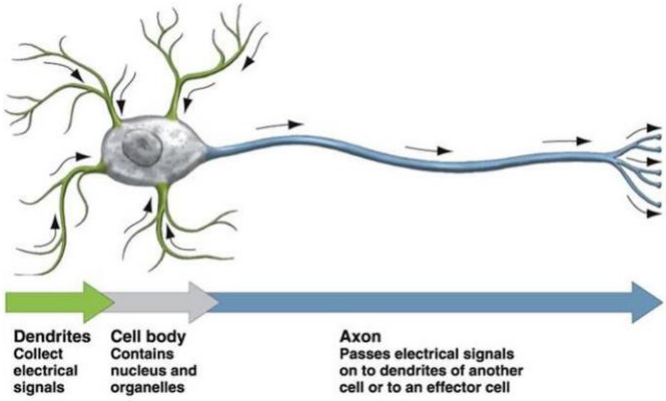

Fig. 1. Biological neural cells can remind a multinomial form.

Biological neural cells seem to apply a similar principle. The dendrites collect signals incoming from other neurons but unlike the ANN functionality the signals can interact already in single dendrites (input lines) (Fig.1.). Multi-variable polynomials might model this framework by means of the products of some input variables. The weighted combinations are summed in the cell of the body and then transformed through time-delayed dynamic periodic activation function (the activated neural cell generates series of time-delayed output pulses in response to its input signals). The period of this activation function depends on values of input variables combinations and seems to represent a derivative operator of a single PDE substitution term. According to these assumptions brain applies combined techniques of relative data processing to compose and substitute for systems of differential equations, forming time-dependent relative pulse models, very efficient for a large scale variability, variable-differences and adaptability of varied (shape) input pattern forms.

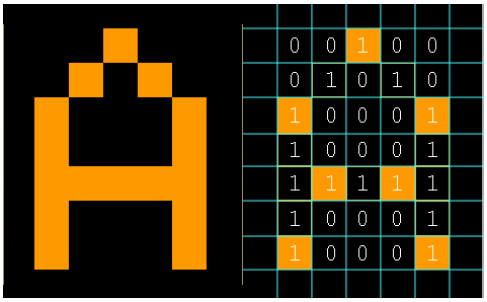

Fig. 2. Low-level properties - line terminations.

The D-PNN may also decompose and generalize a (visual) input pattern into some characteristic elements relations of a model function, which can identify all its manifold forms (analogous to the ANN function approximation and pattern classification). The D-PNN can correctly recognize any untrained variable-shape pattern forms, which keep the trained data relations, regardless of the size and position in the input matrix [8]. Many biological and psychological studies suggest the brain applies just relative units of input variables, contrary e.g. to the absolute input signal processing of a CCD camera [9], and a reductive decomposition of complete input patterns into some major characteristic elements - low-level properties (Fig.2.). Line terminations are by far the most important features for the correct human letter identification [10]. Other features as intersections, curvatures or slants are little considered [11]. Generalized relations of the fragment positions could define a type model for all alterative visual pattern forms [8]. The PNN application in the field of differential equation solutions is a novelty however the experimental results indicate the method is efficient, using only a few substitution derivative terms [12], and can model physical or natural dynamic processes or systems that are too uncertain or complex to be easy described unambiguously by means of standard composite computational techniques.

II. Partial Differential Equation Substitution

D-PNN forms and solves the general partial differential equation (3), with a sum combination (4) of selected substitution fractional multi-variable polynomial derivative terms (9). The unknown function u is possible to calculate from the PDE (3), which involves also its simple form, as the sum of the rest of its partial function derivative terms (4).

$$a + bu + \sum_{i=1}^{n} c_i \frac{\partial u}{\partial x_i} + \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \ldots = 0 \qquad (3)$$

$u = f(x_1, x_2, \ldots, x_n)$ - searched function of n-input variables
$a, B(b_1, b_2, \ldots, b_n), C(c_{11}, c_{12}, \ldots)$ - parameters

In the case of the pattern recognition (described by the modelled function of a PDE solution), the simple function u term (without derivatives) must be added to the derivative fraction sum (in each block) to keep the complete PDE substitution and produce a coequal output identification to all presented input patterns of the same class (shape-form) [8].

$$u = \sum_{k=1}^{\infty} u_k \qquad (4)$$

If each variable x_i of a function u (4) is independent of other variables, then the u function is separable and might be approximated by its partial sum u_k functions (4), i.e. its derivative terms, formed in respect of 1 or more variables. The searched function u may be expressed in the form of sum series (4), consisting of convergent series arising from the competent partial derivative terms (5) of 2 input variables.

$$\left(\sum \frac{\partial u_k}{\partial x_1}, \sum \frac{\partial u_k}{\partial x_2}, \sum \frac{\partial^2 u_k}{\partial x_1^2}, \sum \frac{\partial^2 u_k}{\partial x_1 \partial x_2}, \sum \frac{\partial^2 u_k}{\partial x_2^2} \right) \qquad (5)$$

The function partial derivatives can be expressed in a form of the product of 2 functions, where g means one (or several) - variable function of x_i only and h is any function of all input vector variables $\boldsymbol{x}(x_1, x_2, \ldots, x_n)$ (6).

$$\frac{\partial u(x_1, x_2, \ldots, x_n)}{\partial x_i} = g(x_i) \cdot h(x_1, x_2, \ldots, x_n) \qquad (6)$$

The similarity theory is based on the hypothesis functional relationships exist among the non-dimensional parameters, which can describe a physical system. The Buckingham π theorem removes extraneous information from a problem by forming dimensionless groups and is the fundamental of dimensional analysis. It states if the eq. (7) is the only relationship among the q_i's and if it holds for any arbitrary choice of the units in which $q_1, q_2, \ldots, q_n$ are measured, then (7) can be written in the form using $\pi_1, \pi_2, \ldots, \pi_m$ as independent dimensionless products of the q_i's (8). If k is the minimal number of principal quantities necessary to express the dimensions of the $\boldsymbol{q}$'s, then $m = n - k$ [13].

$$\phi(q_1, q_2, \ldots, q_n) = 0 \qquad (7)$$

$$\phi(\pi_1, \pi_2, \ldots, \pi_m) = 0 \qquad (8)$$

where $\pi_1, \pi_2, \ldots, \pi_m$ are independent dimensionless products of the q's.

If a differential equation form is unknown, the dimensional analysis can search for a non-dimensional set of units from variables using matrix methods of linear algebra. The searched function must be invariant to a change of model units for each i^{th} dimensional variable $X_i' = \alpha_1^{D_{1i}} \alpha_2^{D_{2i}} ... \alpha_j^{Dji} X_i$. If a physical model with *i*-dimensional variables is assumed, the invariance for all possible α changes of *j*-units may be written in the following forms (9) (10).

$$F(X_1, X_2, ..., X_i) = \\ = F\left(\alpha_1^{D_{1i}} \alpha_2^{D_{2i}} ... \alpha_j^{Dj1} X_1, \alpha_1^{D_{1i}} \alpha_2^{D_{2i}} ... \alpha_j^{Dj2} X_2, ..., \alpha_1^{D_{1i}} \alpha_2^{D_{2i}} ... \alpha_j^{Dji} X_i\right) \quad (9)$$

$$\frac{\partial F}{\partial \alpha_j} = \frac{\partial F}{\partial X_1} \cdot \frac{\partial X_1}{\partial \alpha_j} + \frac{\partial F}{\partial X_2} \cdot \frac{\partial X_2}{\partial \alpha_j} + ... + \frac{\partial F}{\partial X_i} \cdot \frac{\partial X_i}{\partial \alpha_j} = 0 \quad (10)$$

α_j - unit scale change D_{ji} - dimensionality of the i^{th} variable
j - number of fundamental units

The designed PDE substitution terms (11) are formed according to the adapted method of integral analogues, which is a part of the similarity dimensional analysis. It replaces mathematical operators and symbols of a PDE by the ratio of corresponding values. Derivatives are replaced by the integral analogues, i.e. derivative operators are removed and along with all operators replaced by analogue or proportion signs in equations to form dimensionless units (groups) of variables [14]. According to the above-mentioned concept definitions the relative polynomial fractions (11) describe partial derivative relations of *n*-input variables through the PDE terms (3).

$$\frac{\partial^m f(x_1, ..., x_n)}{\partial x_1 \partial x_2 ... \partial x_m} = \\ = w_i \frac{\left(a_0 + a_1 x_1 + a_2 x_2 + ... + a_n x_n + a_{n+1} x_1 x_2 + ...\right)^{m/n}}{b_0 + b_1 x_1 + ...} \quad (11)$$

n - combination degree of a complete polynomial of n-variables
m - combination degree of a derivative polynomial denominator

The complete polynomials of *n*-input variables (2), substitute for the PDE term numerators (3) and define the partial u_k functions (4) from the sum series (5) in the complete searched function *u* solution. The denominator (11) is a derivative part, which gives a partial dependent derivative change of some polynomial combinations of variables. It arose from a competent partial derivation of the complete *n*-variable input polynomial. The root functions of D-PNN neurons, i.e. substitution PDE terms (11), reduce the numerator combination degree in order to form dimensionless fractions.

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = \frac{\left(1 + x_1 + x_2 + x_1 x_2\right)^{1/2}}{1 + x_1} \quad (12)$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = \frac{\left(1 + x_1 + x_2 + x_1 x_2\right)^{1/2}}{1 + x_2} \quad (13)$$

The simplest linear form of 2-variable polynomial fraction terms, represented by 1st derivative substitutions only, is considered to prove the validity. All parameters a_i, b_i can be set simplified to *1* (12) (13). If $\partial f/\partial x_1$ is integrated with respect to x_1 and $\partial f/\partial x_2$ with respect to x_2, the equations (14) (15) should give the same result: $f(x_1, x_2)$ as $\partial f/\partial x_1$ and $\partial f/\partial x_2$ are partial derivatives of the same function *f* (16) (17). Partial derivatives of the functions *f* in eq. (11) are valid if denominator exponents equal *1*, only the complete polynomial numerator may apply a root *m/n* quotient to balance the combination degree.

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = \frac{((1 + x_1)(1 + x_2))^{1/2}}{1 + x_1} = \frac{(1 + x_2)^{1/2}}{(1 + x_1)^{1/2}} \quad (14)$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = \frac{((1 + x_1)(1 + x_2))^{1/2}}{1 + x_2} = \frac{(1 + x_1)^{1/2}}{(1 + x_2)^{1/2}} \quad (15)$$

$$g = \int \frac{\partial f}{\partial x_1} dx_1 = \frac{2(1 + x_1)\left((1 + x_1)(1 + x_2)\right)^{1/2}}{1 + x_1} = f(x_1, x_2) \quad (16)$$

$$h = \int \frac{\partial f}{\partial x_2} dx_2 = \frac{2(1 + x_2)\left((1 + x_1)(1 + x_2)\right)^{1/2}}{1 + x_2} = f(x_1, x_2) \quad (17)$$

Blocks of the D-PNN group neurons with the same inputs (Fig.3.), one for each fractional polynomial derivative combination of the sum PDE terms (3). Each block contains a single output polynomial without derivative part, which enters the next hidden layer, i.e. the D-PNN block skeleton is formed by the GMDH PNN. Neurons don't affect the block output but can be directly involved in the total network sum output of a PDE solution (4). Each block has *1* and neuron *2* vectors of adjustable parameters ***a*** and ***a, b*** respectively. All neuron and block polynomial outputs cannot become negative values.

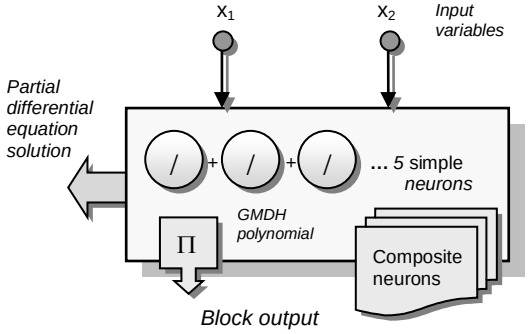

Fig. 3. A block of derivative neurons

The simple non-linear 2-variable GMDH polynomial (2) in the block neurons, which form the PDE substitution terms (11), is applied in all following experiments. 2 or more variable single combination blocks (Fig.3.) can approximate any multi-variable function; published experiments of n>*2*, using only 1-block solutions, are only demonstration cases [8]. The probability of neurons activations P_A is set around the value *0.5* (may be adapted in respect of the application), so a block largely produces about half of all the possible simple neurons to form an optimal PDE solution.

$$F\left(x_1, x_2, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \frac{\partial^2 u}{\partial x_2^2}\right) = 0 \quad (18)$$

where $F(x_1, x_2, u, p, q, r, s, t)$ is a function of 8 variables

While using 2 input variables an equivalent 2nd order PDE (3) may be expressed in the form (18), which derivative variables of the PDE terms, correspond exactly to all the GMDH polynomial (2) variables. The 2-variable block neurons form and substitute for all the relevant partial derivative terms, so each block includes *5* simple neurons formed with respect to 2 single linear x_1, x_2 (19), 2 squared x_1^2, x_2^2 (20) and 1 combination x_1x_2 (21) derivative variables of the 2nd order PDE substitution (18) in a searched 2-variable *u* function model.

$$y_1 = \frac{\partial f(x_1, x_2)}{\partial x_1} = w_1 \frac{\left(\frac{1}{6} \cdot (a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2)\right)^{\frac{1}{2}}}{\frac{1}{2} \cdot (b_0 + b_1x_1)} \quad (19)$$

$$y_3 = \frac{\partial^2 f(x_1, x_2)}{\partial x_2^2} = w_3 \frac{3 \cdot (a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2)}{6 \cdot (b_0 + b_1x_2 + b_2x_2^2)} \quad (20)$$

$$y_5 = \frac{\partial^2 f(x_1, x_2)}{\partial x_1 \partial x_2} = w_5 \frac{4 \cdot (a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2)}{6 \cdot (b_0 + b_1x_{11} + b_2x_{12} + b_3x_{11}x_{12})} \quad (21)$$

The D-PNN total output *Y* is the arithmetic mean of all the active neuron outputs (22) so as to prevent the varying number of neurons (of a sum combination) from influencing the total network output sum value.

$$Y = \frac{\sum_{i=1}^{k} y_i}{k} \qquad k = \text{number of active neurons (PDE terms)} \quad (22)$$

The input variables (of a data set) are normalized to the range *<0.5, 1.5>*. All the neuron and block output polynomials are divided by the number of the items (members) they include (19), which is especially useful for an applications of the block outputs in next hidden layers and the parameters adjustment. This way all the neurons (PDE terms) and blocks (the complete network) produce its outputs around the value *1.0*, mostly in the interval *<0.9, 1.1>*, which may be adapted (optimized). The internal total network output must be scaled (denormalized) to retrieve desired range values of the output function.

III. Multi-Layer Backward D-PNN

Multi-layer networks form composite functions (24). The blocks preceding layers create internal functions (23), which substitute for the next hidden layer input variables of neuron and block polynomials to produce external functions (24). Composite PDE terms, i.e. composite function derivatives with respect to the variables of previous layers blocks, are calculated according to the partial derivation rules (25)(26).

$$y_i = \varphi_i(X) == \varphi_i(x_1, x_2, ..., x_n) \qquad i=1, \dots, m \quad (23)$$

$$F(x_1, x_2, \dots, x_n) = f(y_1, y_2, \dots, y_m) = f(\phi_1(X), \phi_2(X), ..., \phi_m(X)) \quad (24)$$

$$\left.\begin{aligned} \frac{\partial F}{\partial x_1} &= \frac{\partial f}{\partial y_1} \cdot \frac{\partial \varphi_1}{\partial x_1} + \frac{\partial f}{\partial y_2} \cdot \frac{\partial \varphi_2}{\partial x_1} + ... + \frac{\partial f}{\partial y_m} \cdot \frac{\partial \varphi_m}{\partial x_1} \\ \frac{\partial F}{\partial x_2} &= \frac{\partial f}{\partial y_1} \cdot \frac{\partial \varphi_1}{\partial x_2} + \frac{\partial f}{\partial y_2} \cdot \frac{\partial \varphi_2}{\partial x_2} + ... + \frac{\partial f}{\partial y_m} \cdot \frac{\partial \varphi_m}{\partial x_2} \\ &\dots\dots\dots \\ \frac{\partial F}{\partial x_n} &= \frac{\partial f}{\partial y_1} \cdot \frac{\partial \varphi_1}{\partial x_n} + \frac{\partial f}{\partial y_2} \cdot \frac{\partial \varphi_2}{\partial x_n} + ... + \frac{\partial f}{\partial y_m} \cdot \frac{\partial \varphi_m}{\partial x_n} \end{aligned}\right\} \quad (25)$$

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^{m} \frac{\partial f(y_1, y_2, ..., y_m)}{\partial y_i} \cdot \frac{\partial \phi_i(X)}{\partial x_k} \qquad k=1, \dots, n \quad (26)$$

The blocks of the 2nd and following hidden layers form additionally composite terms (CT), i.e. neurons, which substitute for composite function derivatives with respect to the output and input variables of back connected previous layers blocks (Fig.4.). For example the 1st block of the last 3rd hidden layer forms 5 simple neurons, i.e. basic terms (19) (20) (21) of the general PDE (3) solution using its own 2 input variables only (Fig.5.). Additionally it creates 10 CT (a double of the neurons) with respect to the previous 2nd layer 2 blocks derivative input variables using composite function derivatives products with respect to reverse outputs of the 2 back-connected blocks (27). As the couples of variables of the internal functions $\phi_1(x_1, x_2)$ and $\phi_2(x_3, x_4)$ can differ from each other (Fig.5.), their partial derivations are calculated separately in respect of each individual block variables, so each sum (26) consists of only 1 term (single neuron). The 20 CT, formed with respect to the 1st layer 3 blocks input variables, are created analogously (28). The back-calculation of the composite function derivatives is well done by a recursive algorithm in the network tree-like structure (Fig.5.).

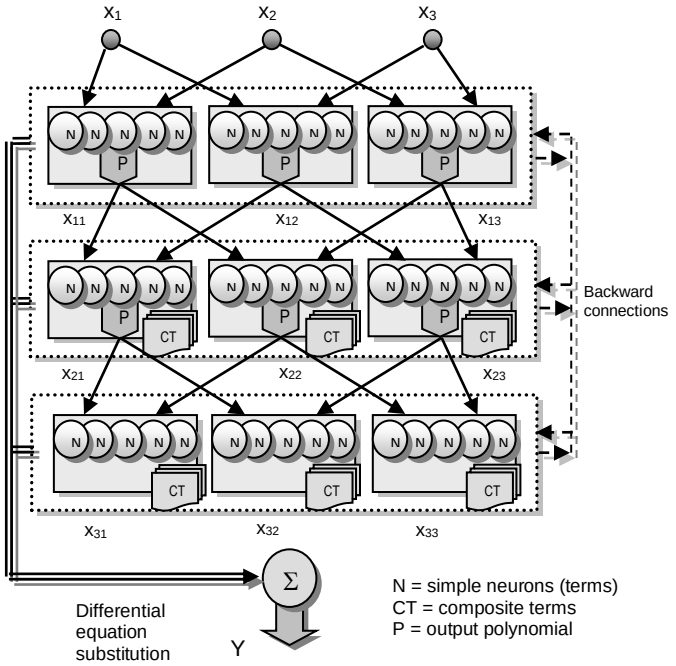

Fig. 4. 3-variable multi-layer D-PNN with 2-variable combination blocks

The number of block neurons, which include composite function derivatives, doubles each previous back-connected layer, so the probability activation P_A of CT, which derivatives are formed with respect to the previous layers block input variables must halve together with the increasing number of hidden layers they comprise (Fig.5.). All the blocks, regardless of the network layer and position form equivalent neurons, which sum (total network output) substitutes for the general PDE (3) and have the same initial probabilities of neurons activations P_A that may be optimized [15].

$$y_2 = \frac{\partial^2 f(x_{21}, x_{22})}{\partial x_{11}^2} = w_2 \cdot \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2)^{1/2}}{x_{22}} \cdot \frac{x_{21}}{b_0 + b_1 x_{11} + b_2 x_{11}^2} \quad (27)$$

$$y_3 = \frac{\partial f(x_{21}, x_{22})}{\partial x_2} = w_3 \cdot \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2)^{1/2}}{x_{22}} \cdot \frac{(x_{21})^{1/2}}{x_{12}} \cdot \frac{x_{11}}{b_0 + b_1 x_2} \quad (28)$$

Only some of all the potential neurons (substitution terms) may be included in the PDE sum composition, even though they have an adjustable term weight w_i. The selection of a fit neuron combination is the principal part of the D-PNN model composition and it may apply the Particle Swarm Optimization (PSO) to adjust the neuron activation probabilities P_A in each block (Fig.5.), binary PSO to combine particle binary vectors in new generation solutions or Simulated Annealing (SA), able to solve large combinatorial optimization problems with random solutions in the initial composing phase. The standard PSO applied the self-cognition and social coefficients (c_1, c_2 = *1.5*), inertia weight (ω = *0.5*) to form 20 individual solutions with the velocity limit <*-1.5, 1.5*>. The binary PSO uses binary operators and the corresponding coefficients settings in the standard PSO (velocity) equations to form new individuals [15]. Parameters of polynomials and PDE term weights are represented by real numbers, randomly initialized from the interval <*0.5, 1.5*> and adjusted by means of the gradient steepest descent method [17] combined with a difference evolution algorithm (EA) [16], performed simultaneously with the best-fit neuron combination search [18].

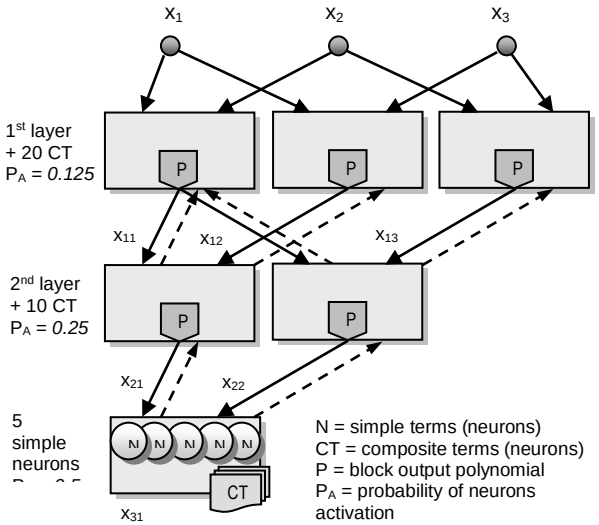

Fig. 5. D-PNN 3rd layer 1st block backward connections, applied for the composite substitution terms formation.

The root mean squared error (RMSE) (29) was applied for the parameter optimization and PDE term selection. The D-PNN can be trained with only a small set of input-output data samples, analogous to the GMDH algorithm [2].

$$E = \sqrt{\frac{\sum_{i=1}^{M} \left(y^d - y_i\right)^2}{M}} \rightarrow \min \quad (29)$$

The D-PNN (also GMDH) approximation ability of complicated periodic functions is possible to improve by means of a sigmoidal transformation (sig) of the squared power items together with their parameters in both the neuron and block output polynomials (30).

$$y = (a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + sig(a_4 x_i^2) + sig(a_5 x_j^2)) / 6 \quad (30)$$

IV. EXPERIMENTAL STUDIES

Neural networks can learn function ***f(x)*** relations of 3 random input ***x*** variables, for a defined range of the values. ANN models were compared, since it is not necessary to keep the order of multi-variable function training samples. Following experiments apply training data sets, which include 50 data samples x_1, x_2, x_3 -> $f(x_1, x_2, x_3)$, randomly generated by a benchmark. *Friedman's* benchmark approximation models were first trained within a reduced function output range <*0, 10*> values for the input vector ***x*** interval <*0, 1*> variables. After the models were tested with 3 random input variables from $\boldsymbol{x} \in$ <*0, 1*> again to estimate the gradually increasing true function ***f(x)*** values on a complete (extended) test interval <*0, 15*>. The graph cannot display all the 3D-benchmark 3 input parameters ***x***, so it features only the lowest approximation RMSE of the output function successive course values (Fig.6.). The accuracy of both models is co-equal on the training interval <*0, 10*> function values however the ANN approximation ability falls more rapidly outside of this range, while the D-PNN alternate errors grow just slowly (Fig.6.).

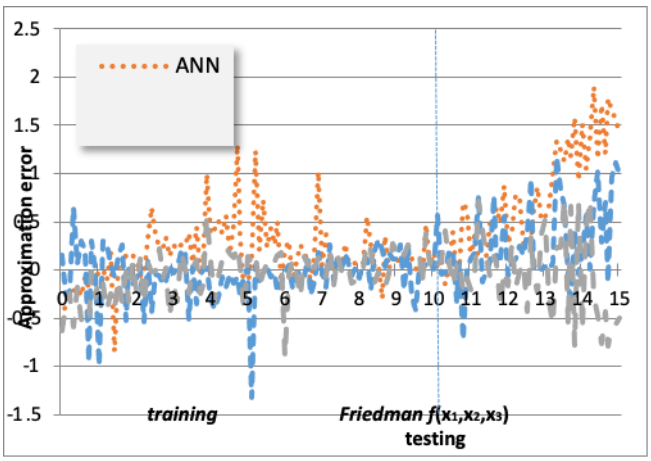

Fig. 6. 3-D *Friedman's* benchmark approximations minimal testing RMSE: ANN=0.61, D-PNN=0.36 (PSO) and D-PNN=0.32 (B-PSO).

D-PNN models are more succesful in the approximation of polynomial benchmarks and if the models are not trained and tested within the same ranges of input or output function values (Fig.6.), which holds for all the tested benchmarks, as the D-PNN applies relative data [15]. Training and testing intervals of all following experiments coincide. The complete

test RMSEs of the benchmark 4D-graph hyper-surface approximation, were calculated with the gradually increasing 3 input parameters in a step 0.05 for $\boldsymbol{x} \in \langle 0, 1 \rangle$ (in a matrix 20x20x20) from exact function $f(\boldsymbol{x})$ values compared with the models output estimations $f'(\boldsymbol{x})$. The characteristics - Average training, Minimal testing, Average testing total RMSE and Standard deviation (Tab.1.) were calculated from 25 successful model experiments.

TABLE I. *FRIEDMAN'S* BENCHMARK APPROXIMATIONS

Method	Complete matrix RMSE			
	$Test_{Min}$	$Test_{Aver}$	$Train_{Aver}$	St.dev. σ
ANN (1-layer)	0.30	0.39	0.08	0.078
D-PNN	0.30	0.43	0.13	0.109

The ANN and D-PNN using the sigmoidal polynomial transformations (30) (D-PNNs) approximated *Schwefel's* periodic benchmark for input variables $\boldsymbol{x} \in \langle 0, 100 \rangle$ in the function output range $f(\boldsymbol{x}) \in \langle 1040, 1420 \rangle$. The models, trained with 50 random samples on the complete function output interval, were tested with gradually increasing values of input variables $\boldsymbol{x}$ in a step 5 (20x20x20 matrix) to calculate the complete surface total RMSE. Tab.2. shows both models results in 25 successful runs.

TABLE II. *SCHWEFEL'S* BENCHMARK APPROXIMATIONS

Method	Complete matrix RMSE			
	$Test_{Min}$	$Test_{Aver}$	$Train_{Aver}$	St.dev. σ
ANN (1-layer)	8.7	13.6	4.8	2.42
D-PNNs	5.9	11.2	3.5	2.08

The ANN applied 1-hidden layer with around 50-60 neurons, for this and following *cosine-mixture* benchmark approximations (Tab.3). Both the models were tested with the total RMSE again for the complete matrix values of input variables $\boldsymbol{x} \in \langle 0, 1 \rangle$ on the training function interval $f(\boldsymbol{x}) \in \langle 0.25, 8.5 \rangle$.

TABLE III. *COSINE-MIXTURE* BENCHMARK APPROXIMATIONS

Method	Complete matrix RMSE			
	$Test_{Min}$	$Test_{Aver}$	$Train_{Aver}$	St.dev. σ
ANN (1-layer)	0.18	0.24	0.07	0.039
D-PNNs	0.10	0.19	0.04	0.052

V. REAL SYSTEM MODELS

Real multi-variable functions may be represented by local relative humidity observations, related to 3 weather input variables: the wind speed, temperature and sea level pressure. The models can roughly estimate the time and amount of precipitation and indicate also the cloudiness progress, according to the current state 3 input variables. The relative humidity values increase at night hours (along with a temperature decrease), a straight or sudden grow can indicate precipitations (Fig.9.), a slight or gradual changes, in the slope curve, feature a variable cloudiness (Fig.8.) [19].

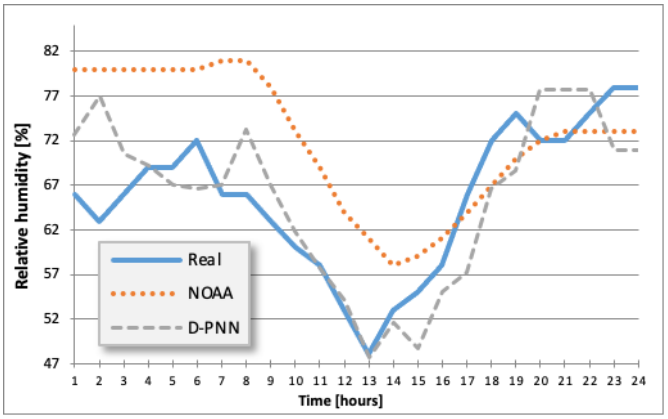

Fig. 7. Helena, 19.12.2013: RMSE - *NOAA = 10.0, D-PNN = 5.54.*

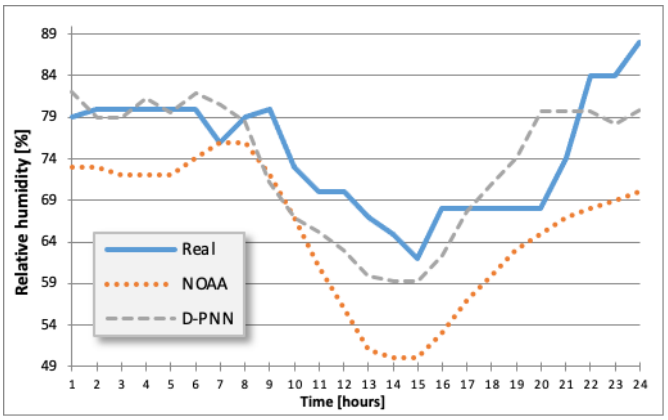

Fig. 8. Helena, 20.12.2013: RMSE - *NOAA = 10.45, D-PNN = 5.34.*

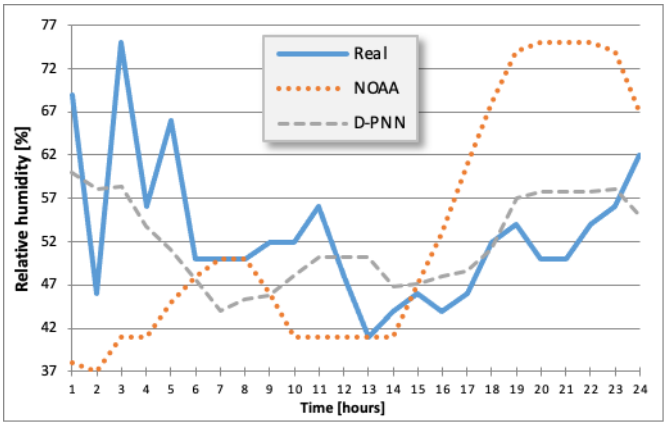

Fig. 9. Helena, 12.1.2014: RMSE - *NOAA = 16.22, D-PNN = 7.07.*

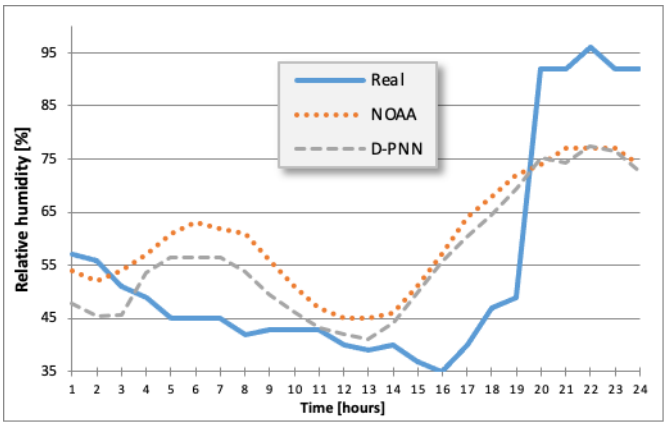

Fig. 10. Helena, 13.1.2014: RMSE - *NOAA = 14.94, D-PNN = 13.14*.

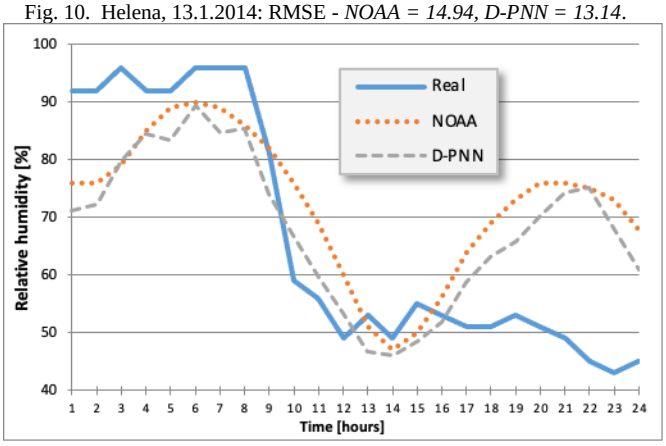

Fig. 11. Helena, 14.1.2014: RMSE - *NOAA = 16.13, D-PNN = 14.27*.

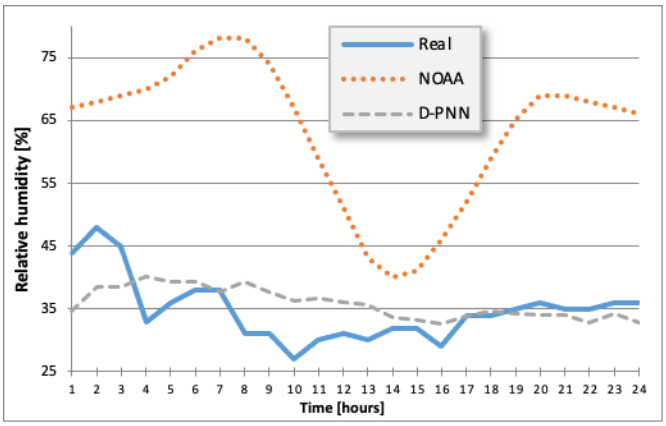

Fig. 12. Helena, 15.1.2014: RMSE - *NOAA = 30.11, D-PNN = 5.10*.

The D-PNN, trained for local actual weather relevant data relations of several last days (2-6), can revise a meso-scale numerical weather prediction (NWP) model prognosis in the cases of settled weather periods. The correction model applies corresponding NWP model outputs to revise one target 24-hour original prognosis. The National Oceanic and Atmospheric Administration (NOAA) provides forecasts [A] and current daily observations [B] for a selected locality [C] and also complete free data archives [D]. NOAA forecasts enter the D-PNN model, locally trained with the same data types of observations (Helena, Montana), to produce output hourly revisions of the relative humidity 24-hour forecast at the same time points (Fig.7. - Fig.12.). The presented D-PNN correction models were tested (compared with real values) on the complete 24-hour forecasting interval, which is naturally not possible in real-time. The trained models might be tested with the last trained day forecasts but this naturally reduces the prediction accuracy. The optimal number of training days (model initialization time) is another parameter necessary to determinate. The applied relative humidity models do not allow for any time-series but only multi-variable function relations.

VI. Pattern Identification Models

The D-PNN can generalize input pattern forms by means of its model function, which represents the trained feature relations, to recognize the correct class. Each block must add its polynomial output (2) to the substitution fraction series sum (if selected any) in order to allow the network to form the complete PDE (3) and produce a coequal output to the same kind (class) of patterns. Average identification results for the Breast Cancer Wisconsin (Original) Data Set [E] from the UCI archives are presented in the Tab.4., which compare several published methods using the Artificial meta-plasticity neural network (AM-NN) [20], Entropy based neural network (EB-NN) [21], Discrete particle swarm optimization (D-PSO) [22], Least square support vector machine (LS-SVM) [23], Association rule neural network (AR-NN) [24] and Genetic algorithm rotation forest (GA-RF) [25]. Conventional validation (complete data are portioned into one training and test set) and multi-fold cross-validation (CV) techniques were performed to compare the accuracy of the chosen methods. In *k*-fold CV whole data are randomly divided to *k*-mutually exclusive and approximately equal size subsets. The classification algorithm is trained and tested *k*-times. In each case, one of the folds is taken as test data and the remaining folds are added to form training data. Thus *k*-different test results exist for each training-test configuration. The average of these results gives the test accuracy of the algorithm.

TABLE IV. *Breast Cancer Identification*

Method	Accuracy [%]	Train./Test
AM-NN	*99.26*	*60-40%*
D-PNNs	*98.9*	*70-30%*
EB-NN	*98.83*	*10-fold CV*
D-PSO	*98.71*	*2/3 - 1/3*
LS-SVM	*98.53*	*10-fold CV*
AR-NN	*97.4*	*3-fold CV*
GA-RF	*96.78*	*10-fold CV*

The D-PNN gets with the best testing accuracy 99.5% (trained on the first 70% data) and 98.5% (trained with 50%). The experiments were done with the standard D-PNN (for pattern identification), which was not specially adapted for the disease recognition as the compared methods usually do. A feature and block selection algorithm (see the Discussion) might improve its performance. The data rows with missing attribute values were removed from the original data set according to the published compared results.

VII. Discussion

The D-PNN forms and solves the general PDE, which model solutions enables to form its own independent weather forecasts based on time-series observations only. However the next hour forecasts for each grid point (in a selected area), which enter the model calculations for the next step ahead predictions would be extremely time-consuming. If the number of input variables increases then the number of the D-PNN 2-combination couples grows exponentially in each next hidden layer (the previous identification models apply 9 inputs). Thus the D-PNN (also PNN) with more than 3 input variables must face to the *"combinatorial explosion"* (analogous to the GMDH) and select from the best blocks in each hidden layer [8] along with the overall neuron (substitution PDE term) selection process.

VIII. Conclusions

The D-PNN combines the multi-layer network composite function structures with mathematical techniques of PDE substitutions. It may implement the PDE terms using other substitution methods, e.g. Fourier series. The presented models define and solve the general PDE with a sum combination of selected simple and composite substitution derivative terms, produced in all layers of the complete GMDH PNN. The D-PNN function approximation and pattern recognition models are based on the polynomial derivative generalization of elementary data relations. The D-PNN (also PNN) model complexity is proportional to the increasing number of input variables, as additional hidden layers of blocks can define all the potential combination PDE terms. This is contrary to conventional ANN 1 or 2-hidden layer flat structures, which are not able to form more complex and versatile models of dynamic systems (with more variables). The D-PNN can model complex dynamic systems that a PDE can preferably describe and which exact representation is unknown. The D-PNN is preferable to approximate polynomial-like functions however the sigmoidal transformation of the polynomial squared items improves its ability to model complicated periodic functions.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project „IT4Innovations excellence in science - LQ1602" and is partially supported by Grant of SGS No. SP2016/ 146, VŠB - Technical University of Ostrava, Czech Republic.

References

[1] C. Giles, "Noisy time series prediction using recurrent neural networks and grammatical inference," *Machine Learning*, vol. 44, p. 161–183, 2001.

[2] N. Y. Nikolaev and H. Iba, *Adaptive Learning of Polynomial Networks*. Genetic and evolutionary computation, New York: Springer, 2006.

[3] L. Menezes and N. Nikolaev, "Forecasting with genetically programmed polynomial neural networks," *International Jounal of Forecasting*, pp. 249–265, 2006.

[4] J. Chaquet and E. Carmona, "Solving differential equations with fourier series and evolution strategies," *Applied Soft Computing*, vol. 12, p. 3051–3062, September 2012.

[5] H. Iba, "Inference of differential equation models by genetic programming," *Information Sciences*, vol. 178, p. 4453–4468, December 2008.

[6] H. Cao, L. Kang, Y. Chen, and J. Yu, "Evolutionary modeling of systems of ordinary differential equations with genetic programming," *Genetic Programming and Evolvable Machines*, vol. 1, pp. 309–337, October 2000.

[7] I. Tsoulos, D. Gavrilis, and E. Glavas, "Solving differential equations with constructed neural networks," *Neurocomputing*, vol. 72, pp. 2385–2391, 2009.

[8] L. Zjavka, "Recognition of generalized patterns by a differential polynomial neural network," *Engineering, Technology & Applied Science Research*, vol. 2, no. 1, pp. 167–172, 2012.

[9] M. Druckmüller, "Phase correlation method for the alignment of total solar eclipse images," *Astrophysical Journal*, vol. 706, no. 2, p. 1605–1608, 2009.

[10] D. Fiset, C. Blais, C. Ethier-Majcher, M. Arguin, D. Bub, and F. Gosselin, "Features for identification of uppercase and lowercase letters," *Psychological science*, vol. 19, no. 11, 2008.

[11] V. Willenbockel, J. Sadr, D. Fiset, G. Horne, F. Gosselin, and J. Tanaka, "Controlling low-level image properties: The shine toolbox," *Behavior Research Methods*, vol. 42, pp. 671–684, 2010.

[12] L. Zjavka and A. Abraham, "Failure and power utilization system models of differential equations by polynomial neural networks," in *Proceedings of the 13th international conference on Hybrid Intelligence Systems, Hammamet, Tunisia, December 4-6* (A. Abraham, ed.), 2013.

[13] D. Randall, *Dimensional Analysis, Scale Analysis, and Similarity Theories*. September 2012.

[14] K. Chan and W. Y. Chau, "Mathematical theory of reduction of physical parameters and similarity analysis," *International Journal of Theoretical Physics*, vol. 18, pp. 835–844, November 1979.

[15] L. Zjavka and W. Pedrycz, "Constructing general partial differential equations using polynomial and neural network," *Neural Networks*, vol. 73, p. 58–69, 2016.

[16] S. Das, A. Abraham, and A. Konar, "Particle swarm optimization and differential evolution algorithms," in *Studies in Computational Intelligence*, vol. 116, (Berlin), pp. 1–38, Springer-Verlag, 2008.

[17] N. Y. Nikolaev and H. Iba, "Polynomial harmonic GMDH learning networks for time series modeling," *Neural Networks*, vol. 16, p. 1527–1540, 2003.

[18] L. Zjavka and V. Snášel, "Constructing ordinary sum differential equations using polynomial networks," *Information Sciences*, vol. 281, pp. 462–477, 2014.

[19] L. Zjavka, "Numerical weather prediction revisions using the locally trained differential polynomial network," *Expert Systems With Applications*, vol. 44, p. 265–274, 2016.

[20] A. Marcano-Cedeno, J. Quintanilla-Dominguez, and D. Andina, "Wbcd breast cancer database classification applying artificial metaplasticity neural network," *Expert Systems with Applications*, vol. 38, p. 9573–9579, 2011.

[21] M.-L. Huang, Y.-H. Hung, and W.-Y. Chen, "Neural network classifier with entropy based feature selection on breast cancer diagnosis," *Journal of Medical Systems*, vol. 34, p. 865–873, 2010.

[22] W.-C. Yeh, W.-W. Chang, and Y. Y. Chung, "A new hybrid approach for mining breast cancer pattern using discrete particle swarm optimization and statistical method," *Expert Systems with Applications*, vol. 36, p. 8204–8211, 2009.

[23] K. Polat and S. Gunes, "Breast cancer diagnosis using least square support vector machine," *Digital Signal Processing*, vol. 17, p. 694–701, 2007.

[24] M. Karabatak and M. C. Ince, "An expert system for detection of breast cancer based on association rules and neural network," *Expert Systems with Applications*, vol. 36, p. 3465–3469, 2009.

[25] E. Alickovic and A. Subasi, "Breast cancer diagnosis using ga feature selection and rotation forest," *Neural Computing & Applications*, p. 1–11, 2016.

[A] National Oceanic and Atmospheric Administration (NOAA) forecasts http://forecast.weather.gov/MapClick.php?lat=46.5927&lon=-112.0361&unit=0&lg=english&FcstType=digital

[B] NOAA local observations www.wrh.noaa.gov/mesowest/getobext.php?wfo=tfx&sid=KHLN&num=168&raw=0&dbn=m&banner=header

[C] NOAA Montana station locations www.ndsu.edu/fileadmin/ndsco/normals/documents/7100/MTnorm.pdf

[D] National Climatic Data Center (NCDC) historical data archives http://cdo.ncdc.noaa.gov/qclcd_ascii/

[E] Breast Cancer Wisconsin (Original) Data Set http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29