AVIATION
ISSN: 1648-7788 / eISSN: 1822-4180
2018 Volume 22: 6–12
https://doi.org/10.3846/aviation.2018.2048

FORECASTING AIRCRAFT MILES FLOWN TIME SERIES USING A DEEP LEARNING-BASED HYBRID APPROACH

Victor SINEGLAZOV[1], Olena CHUMACHENKO[2], Vladyslav GORBATIUK[3, *]

[1]*Aviation Computer-Integrated Complexes Department, Educational and Research Institute of Aeronavigation, National Aviation University, Kosmonavta Komarova 1, 03058, Kyiv, Ukraine*
[2, 3]*Technical Cybernetics Department, Faculty of Informatics and Computer Science, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Peremogy Ave 37, 03056, Kyiv, Ukraine*

Received 09 September 2016; accepted 10 May 2018

Abstract. Neural network-based methods such as deep neural networks show great efficiency for a wide range of applications. In this paper, a deep learning-based hybrid approach to forecast the yearly revenue passenger kilometers time series of Australia's major domestic airlines is proposed. The essence of the approach is to use a resilient error backpropagation algorithm with dropout for "tuning" the polynomial neural network, obtained as a result of a multi-layered GMDH algorithm. The article compares the performance of the suggested algorithm on the time series with other popular forecasting methods: deep belief network, multi-layered GMDH algorithm, Box-Jenkins method and the ANFIS model. The minimum reached MAE of the proposed algorithm was approximately 25% lower than the minimum MAE of the next best method – GMDH, thus indicating that the practical application of the algorithm can give good results compared with other well-known methods.

Keywords: forecasting, neural networks, time series, deep learning, hybrid algorithm, group method of data handling.

Introduction

Predicting market demand for air transportation is of great significance for airlines, as well as for investors, since the accuracy of such a prediction has a big impact on investment efficiency (Blinova, 2007). Therefore, airline transportation demand metrics, like Revenue Passenger Kilometers (RPK), are one of the key factors that are considered when preparing an airline's annual operating plan, performing fleet planning and developing the route network (Ba-Fail, Abed, & Jasimuddin, 2000; Doganis, 2009). Besides, examining and estimating an airline's transportation demand may likewise help an airline mitigate its risk through an objective assessment of the demand side of the airline business (Abed, Ba-Fail, & Jasimuddin, 2001; Ba-Fail et al., 2000).

In recent years, very good results for a wide range of applications have been shown by neural network-based methods such as deep neural networks, including the following applications: image recognition (Girshick, Donahue, Darrell, & Malik, 2013), speech recognition (Bahdanau, Chorowski, Serdyuk, Brakel, & Bengio, 2016), machine translation (Bahdanau, Cho, & Bengio, 2014), reinforcement learning (Mnih et al., 2013) and many others. In essence, many of these applications require some kind of function approximation to be performed in order to build a corresponding model, which should indicate the general suitability of deep learning-based methods as a function approximation mechanism. We propose a deep learning-based hybrid approach to forecast the yearly revenue passenger kilometers (RPK) time series of Australia's major domestic airlines, which are publicly available (Australian Domestic Airline Activity-time series, n.d.).

1. Formulation of the problem

Suppose that a sequence of values $\vec{x} = [x_1,...,x_n]^T$ was observed at successive moments of time $\vec{t} = [t_1,...,t_n]^T$ with a given constant period $t_i - t_{i-1} = T, i = 2,...,n$. Then the forecasting problem (Figure 1) that is considered in this paper can be formulated as: by using the available data (Figure 1, A), construct a model of the process being forecast, which takes k successive values $[x_{i-k+1},...,x_i]^T$ as inputs and gives the prediction for value x_{i+h} at some future point in time t_{i+h} (Figure 1, B).

*Corresponding author. E-mail: *vladislav.horbatiuk@gmail.com*

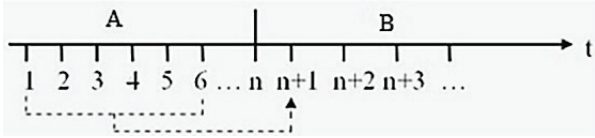

Figure 1. Graphical representation of the forecasting problem: A – known values, B – future values

2. Review of existing methods

Currently, main approaches to forecasting problems can be divided into 2 groups:

- "Classical" methods like linear regression, or Box-Jenkins method (Box & Jenkin, 1970), etc.
- Methods based on artificial intelligence – group method of data handling (Stepashko, 1988) methods family, artificial neural networks-based methods, genetic algorithms, fuzzy logic-based methods and different hybrid algorithms, like adaptive network-based fuzzy inference system.

Let us shortly review the main methods from both groups.

The *Box-Jenkins method* is an ARIMA (Asteriou & Hall, 2011) model-based method that uses an iterative three-stage modelling approach, as described below:

1. Model identification and model selection: checking the stationarity of variables, checking the seasonality in the dependent series, plotting the autocorrelation and partial autocorrelation functions of the dependent time series to decide which autoregressive or moving average component should be used in the model;
2. Parameter estimation using computation algorithms to arrive at coefficients that best fit the selected ARIMA model. The most common methods use maximum likelihood estimation or non-linear least-squares estimation;
3. Model checking by testing, to determine whether the estimated model conforms to the specifications of a stationary univariate process. In particular, the residuals should be independent of each other and constant in mean and variance over time. Plotting the mean and variance of residuals over time and performing a Ljung–Box test or plotting autocorrelation and partial autocorrelation of the residuals are helpful to identify misspecification. If the estimation is inadequate, we have to return to step one and attempt to build a better model.

Group method of data handling (GMDH) is a set of forecasting algorithms that are based on a selection of the best models from the set of trained simple models and the subsequent construction of more complex models using the selected ones. The accuracy of the forecast is improved with the increase in the complexity of the models. The selection criterion is based on the performance of the models on the validation set, while the model parameters are determined using the training set. The simplest models, also called the basis functions, are usually of the following form:

$$F(x_1,...,x_n) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j + \dots .$$

Nevertheless, it is possible to use any other type of basis functions, including harmonic series, exponential series, etc.

GMDH-like algorithms show good forecasting accuracy for real-life processes, mainly due to their use of an external criterion (i.e. the models are selected using data that were not used for their training).

Artificial neural networks (Rosenblatt, 1958) are a system of connected and interacting artificial neurons – simplified mathematical models of biological neural cells. ANN cannot be programmed in the usual sense of the word: they are trained. During the training, the neural network is able to detect the complex relationship between the input and output data and to perform a generalization. The ability of neural networks to carry out the prediction follows directly from their ability to generalize and find hidden relationships between input and output data. After training, the network is able to predict the future value of a specific sequence based on a number of previous values and/or any current factors.

Nowadays deep learning and deep neural networks are the most promising direction in the study of ANN. This trend began to develop in connection with the problem of effective learning of neural networks with a large number of layers (it is worth noting that the recurrent networks can also be represented as feedforward networks with a very large number of layers): the standard backpropagation algorithm gave poor results for these networks – the gradient of the error either faded on the backward pass by layers, leading to the weights of the first layers almost being not trained at all, or grew indefinitely, leading to a divergence of the learning process. In (Hinton, 1989), a solution to this problem was proposed for the first time: the network was constructed layer by layer, and the initial weights between the two layers were found using a restricted Boltzmann machine algorithm. Afterwards, the resulting network was additionally tuned using a standard backpropagation algorithm, since the initial weights already provided a "reasonable" network behaviour, the problem with fading or over-increasing gradients disappeared. Thus, we can distinguish two main stages of deep neural networks' training:

1. "Preliminary" deep network training (pre-training), the essence of which is to add new layers one by one and during which the weights between two layers are trained separately – most often using the restricted Boltzmann machine training algorithm.
2. "Tuning" of the obtained network structure using the error back propagation algorithm (or some modification of it), sometimes with the use of regularization methods (currently, a dropout algorithm (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012) is the most popular regularization method for the training of deep networks).

In the area of time series forecasting, deep belief networks (DBN) (Lee, Grosse, Ranganath, & Ng, 2009) are showing the most promising results. A deep belief network (DBN) is a deep neural network composed of multiple layers of hidden units, with connections between the layers but not between units within each layer. When trained on a set of examples without supervision, a DBN can learn to probabilistically reconstruct its inputs. The layers then act as feature detectors. After this learning step, a DBN can be further trained with supervision to perform classification or regression. A few articles describing the application of DBNs to time series forecasting are (Kuremoto, Kimura, Kobayashi, & Obayashi, 2014), (Qiu, Zhang, Ren, Suganthan, & Amaratunga, 2014), (Chao, Shen, & Zhao, 2011), and others.

Adaptive network-based fuzzy inference system (Jang, 1993) is a hybrid algorithm that combines both ANN and fuzzy inference systems (Fang, 2012; Liu, C., Liu, X., Huang, Zhao, 2008) and aims at utilizing the advantages of both approaches by using prior information in a form of fuzzy rules and capturing hidden dependencies by performing parameter "learning" (Jang, Sun, & Mizutani, 1997; Xiao et al., 2014; Yetilmezsoy, Fingas, & Fieldhouse, 2011). In other words, during learning, the ANFIS tries to find optimal parameters of some fuzzy inference system. "Optimal" here is defined by some error criterion that usually measures how accurate a given model is on a test data, hence, the parameters that minimize a given error criterion are optimal (Goyal, Bharti, Quilty, Adamowski, & Pandey, 2014). Usually, in the ANFIS, membership function parameters and fuzzy rule parameters are optimized during learning.

3. Literature review and algorithm description

In (Schmidhuber, 2015), it is mentioned that despite the relative newness of deep learning methods, GMDH was the first method allowing to effectively train polynomial deep neural networks. Indeed, a multilayered GMDH algorithm builds a deep polynomial neural network, see (Kondo, 1998; Srinivasan, 2008) for an example. The network weights are trained in a "greedy" manner by tuning the weights of each new layer of neurons separately, with all previous weights being fixed. This results in a very computationally efficient procedure for training deep networks. However, since only a portion of the entire network's weights is trained at each iteration, the training may stop without finding globally optimal weights. Actually, it may stop without even finding locally optimal weights, since an appropriate error function is never minimized explicitly.

Later, another method for training multilayered networks was introduced – well known nowadays as backpropagation (LeCun, Bottou, Orr, & Müller, 1998). Backpropagation allowed calculating the true gradient of error function with respect to every weight of a multilayer neural network, thus one could use any suitable first-order gradient descent method to perform weights optimization and find local optima. Theoretically, this should "solve" the problem of training multilayer neural networks, but, in practice, results were not very good for deep networks, see (Bengio, De Mori, Flammia, & Kompe, 1992), (Bengio, Simard, & Frasconi, 1994) for an example. The core issue with training deep networks using backpropagation was soon found: vanishing/exploding gradients problem (Hochreiter, Bengio, Frasconi, & Schmidhuber, 2001). Essentially, according to the backpropagation method, an error function's gradient for the weights in "early" layers (layers, that are closer to inputs) is a sum of the products of terms involving an error function's gradients for the weights in all following layers. When there are many layers that is an intrinsically unstable situation – if many gradients in deeper layers are smaller than 1 in an absolute value, their product will be close to 0, thus leading to a "vanishing" gradient for the early layers; and vice versa – if many gradients in deeper layers are bigger than 1 in an absolute value, their product will increase exponentially, leading to "exploding" gradients in early layers.

One approach to these issues was first suggested in (Hinton, 2009). The main idea of the approach is to first perform "pre-training" of each layer of the network by separately using the Restricted Boltzmann Machines (RBM) training algorithm. Then, the obtained network weights are used as initial weights in the backpropagation algorithm. Though not very well-founded theoretically, in practice, such "pre-training" does indeed help in reducing the vanishing/exploding gradient problem effect. In (Erhan et al., 2010), it is suggested that pre-training overcomes the challenges of deep learning by introducing a useful prior to the fine-tuning training procedure.

Another problem common for all "complex" forecasting models, including deep neural networks, is overfitting –complex models can potentially "memorize" training sets instead of finding actual dependencies, and perform badly on new data it has not seen before. Different techniques have been suggested to overcome this issue: regularization (Krogh & Hertz, 1992), early training stopping (Prechelt, 1998), validation set checking, etc. In (Hinton et al., 2012), a new "dropout" approach tailored specifically to deep neural networks was suggested. According to the approach, one should randomly perform temporary deletion of the network's neurons during training iterations. This can be viewed as a crude approximation to performing Bayesian model averaging (BMA) (Hoeting, Madigan, Raftery, & Volinsky, 1998), which is another way of dealing with overfitting. The problem with BMA is that is does not scale well to very large network sizes, so it cannot be applied to deep neural networks practically, while dropout scales extremely well.

The essence of the proposed approach is to view a multi-layered GMDH algorithm as a pre-training stage of fitting a deep polynomial neural network and then to apply a resilient error backpropagation algorithm (rprop) (Riedmiller & Braun, 1992) with dropout to perform fine-tuning.

The proposed algorithm for constructing and training deep polynomial neural networks consists of the following stages.

1. Transforming original time series $\{x_n\}$:
   - calculating the finite difference time series: $d_i = x_{i+1} - x_i$;
   - normalizing difference time series to zero mean and unit standard deviation: $d_i = \frac{d_i - \mu}{\sigma}$;
   - preparing the input samples matrix X and the outputs vector y using the time series embedding method with some embedding dimension k:

$$X = \begin{bmatrix} d_1 & \dots & d_k \\ \vdots & \ddots & \vdots \\ d_{N-k} & \cdots & d_{N-1} \end{bmatrix}, \vec{y} = \begin{Bmatrix} d_{k+1} \\ \vdots \\ d_N \end{Bmatrix}.$$

2. Pre-training stage using a multi-layered GMDH algorithm to obtain the original structure and weights of the polynomial neural network:
   - the entire sample set is **randomly** divided into training and validation sets; usually 70% of all samples go into the training set and 30% of the remaining samples go into the validation set, but one can choose another ratio – the training set size to validation set size ratio is one of the hyperparameters of this approach;
   - C_k^2 models of the form:

$$f(x_i, x_j) = a_{i1}x_i + a_{j1}x_j + a_{ij}x_i x_j + a_{i2}x_i^2 + a_{j2}x_j^2,$$

   - are trained using a linear regression on the training set;
   - for each model f, its error is calculated on the validation set:

$$E(f) = \sum_{(\vec{x},y)\in(X_v,\vec{y}_v)} (f(\vec{x}) - y)^2;$$

   - s_l models with the lowest error are selected, where l is the layer number in a multi-layered GMDH algorithm (for the simplicity of implementation, $s_l = k$ is often used);
   - outputs of these models for each sample of the training set form a new matrix of inputs $X^{(1)}$ for the next layer of models:

$$X^{(1)} = \begin{bmatrix} f_1(\vec{d}_1) & \cdots & f_{s_l}(\vec{d}_1) \\ \vdots & \ddots & \vdots \\ f_1(\vec{d}_{N-k}) & \cdots & f_{s_l}(\vec{d}_{N-k}) \end{bmatrix},$$

   - where $\vec{d}_p = \{d_p, d_{p+1}, \dots, d_{p+k-1}\}^T$, and $f_m(\vec{d}_p)$ means that only the elements under the indices i and j which correspond to the model f_m will be taken from vector $\vec{d}_p$;
   - if the minimum error on a validation set of all models in the current layer is less than the minimum error of all models from the previous layer (or if it is the first layer), a transition to the following layer is performed (starting from stage 2.2), otherwise the algorithm stops and the best model of the previous series is selected as the "final" one; the criterion for the GMDH algorithm stop can be stated as:

$$\begin{cases} l > 1, \\ \min\limits_{f\in F_l} E(f) \geq \min\limits_{f\in F_{l-1}} E(f), \end{cases}$$

   - where l is the number of the current layer; F_l – set of all models of the current layer. As the output of this stage, the polynomial neural network with a structure shown in Figure 2 below is obtained:

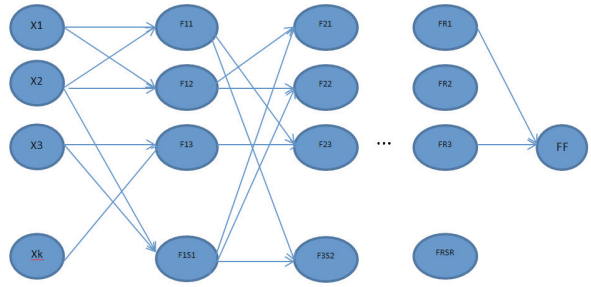

Figure 2. Polynomial neural network

3. Tuning stage, where the weights of the resulting network are trained with the use of the resilient error backpropagation algorithm:
   - for each pair <input vector, output value> from the training set, two so-called passes are performed:
1. a "straight pass", where the vector of input values is given to the first layer of the network, and outputs of each polynomial "neuron" are calculated until the very last "output" neuron;
2. a "backward pass" where error derivative functions of each weight are calculated using the following formulas:

$$\frac{\partial f_{l,m}}{\partial \vec{a}} = \left\{x_i, x_j, x_i x_j, x_i^2, x_j^2\right\}^T,$$

$$\frac{\partial f_{l+1,h}}{\partial f_{l,m}} = \begin{cases} a_{i1} + a_{ij}x_j + 2a_{i2}x_i, \text{if } f_{l,m} \text{ is used as input } x_i \text{ for } f_{l+1,h} \\ a_{j1} + a_{ij}x_i + 2a_{j2}x_j, \text{if } f_{l,m} \text{ is used as input } x_j \text{ for } f_{l+1,h} \\ 0, \text{if } f_{l,m} \text{ is not used as input for } f_{l+1,h} \end{cases};$$

   - according to the dropout regularization procedure, in the beginning of each iteration on a pair <input vector, output value> from the training set, one should generate a random number between 0 and 1 for every neuron in the network (including fictitious neurons that represent the network's inputs) and, with some probability, "delete" this neuron from a network during this iteration. "Deleting" a neuron here means setting all its input and output weights to 0. In the original paper, the authors suggest using a probability of deletion $p = 0.2$ for fictitious input neurons, and a probability of deletion $p = 0.5$ for hidden neurons;
   - all calculated derivatives are summed over all examples;

– each weight is updated by the following rule:

$$\Delta_{ij}^{(t)} = \begin{cases} \alpha^{+}\Delta_{ij}^{(t-1)}, \dfrac{\partial E^{(t)}}{\partial w_{ij}} * \dfrac{\partial E^{(t-1)}}{\partial w_{ij}} > 0, \\ \alpha^{-}\Delta_{ij}^{(t-1)}, \dfrac{\partial E^{(t)}}{\partial w_{ij}} * \dfrac{\partial E^{(t-1)}}{\partial w_{ij}} < 0 \end{cases},$$

$$w_{ij}^{(t)} = w_{ij}^{(t-1)} - sign\left(\frac{\partial E^{(t)}}{\partial w_{ij}}\right) * \Delta_{ij}^{(t)}.$$

If the sign of the error function's derivative by the weight coincides with the sign of the derivative on the previous iteration, the correction value for this weight will be multiplied by some factor $\alpha^{+} > 1$. Otherwise, the correction value is multiplied by a factor $\alpha^{-} < 1$. On the very first iteration some constant correction value is used $\Delta_{ij}^{(0)} = c$. The following values of the constants are recommended in the original paper: $\alpha^{+} = 1.2$, $\alpha^{-} = 0.5$, $c = 0.1$ (these parameters also belong to the hyperparameters vector, meaning that, to achieve the best results for a particular problem, these parameters should also be adjusted);

– after the updates of all weights, a network's error on the training set is calculated, and, if the error is less than on the previous iteration, the training continues, if the opposite is true, the training stops and weights "rollback" to their values on the previous iteration.

– according to the dropout regularization procedure, after the training stops, outgoing weights of all neurons are multiplied by value 1-*p*, where *p* is the "deletion" probability that was used for this neuron during training.

4. As a result, we simply have a polynomial neural network, so forecasting on new data is performed as usual: the input vector $\vec{x}$ is sent on the first network layer, and, after that, the outputs of all neurons are calculated layer by layer, up to the last layer with one neuron, the output of which will be the forecast itself.

4. Comparison on test data sets

Publicly available time series of Australia's major domestic airlines yearly revenue passenger kilometers (RPK) for the years 1944–2012 were used to test the performance of the suggested algorithm compared to:

– the Box-Jenkins method;
– the DBN network;
– the multilayered GMDH algorithm;
– the ANFIS method.

The MATLAB software package was used to train forecasting models using the corresponding methods (a new function was written to implement the proposed algorithm). In order to perform a more comprehensive comparison, multiple possible values for the number of previous time series values to be used as inputs were tested, namely all values from 2 to 10 inclusively. All models were trained to give a year ahead prediction. All methods were used with their default hyperparameters values. 80% of an entire examples set were **randomly** selected as a training set, the remaining 20% were used to test the forecasting model's performance. Comparisons of the results are presented in the following tables.

Table 1. Forecasting errors obtained by Box-Jenkins models

Number of previous time series values used as inputs	2	3	4	5	6	7	8	9	10
MAE	0.1614	0.2444	0.1437	0.1697	0.1814	0.1466	0.202	0.1554	0.3299
MAPE	1.3065	1.1357	0.812	2.2625	0.9043	1.6064	1.7148	2.5098	1.3746
MSE	0.1018	0.2174	0.0622	0.1131	0.1074	0.0758	0.0959	0.0481	0.2063

Table 2. Forecasting errors obtained by DBN models

Number of previous time series values used as inputs	2	3	4	5	6	7	8	9	10
MAE	0.1672	0.2562	0.1492	0.1735	0.1773	0.1482	0.1958	0.1617	0.3419
MAPE	1.3139	1.2164	0.8119	2.0429	0.8815	1.5303	1.6578	2.6484	1.4589
MSE	0.1052	0.2306	0.0674	0.1071	0.1012	0.0756	0.0904	0.0486	0.2181

Table 3. Forecasting errors obtained by multilayered GMDH models

Number of previous time series values used as inputs	2	3	4	5	6	7	8	9	10
MAE	0.088	0.3146	0.1021	0.1214	0.1506	0.1479	0.1491	0.3302	0.317
MAPE	1.3717	2.5654	1.0731	2.1795	1.4383	1.947	0.9468	5.2686	1.5316
MSE	0.0184	0.386	0.0212	0.0332	0.0547	0.0882	0.0888	0.3199	0.2225

Table 4. Forecasting errors obtained by ANFIS models

Number of previous time series values used as inputs	2	3	4	5	6	7	8	9	10
MAE	0.1517	0.2236	0.2441	0.2759	0.2150	0.3424	0.2027	0.163	0.3196
MAPE	1.6127	2.4546	2.6067	2.6413	2.5064	2.5098	1.6554	2.6853	1.2624
MSE	0.0818	0.2131	0.2130	0.1970	0.1963	0.4117	0.0977	0.0532	0.1945

Table 5. Forecasting errors obtained by models built using proposed algorithm

Number of previous time series values used as inputs	2	3	4	5	6	7	8	9	10
MAE	0.0656	0.3253	0.0948	0.1007	0.1251	0.158	0.1247	0.3063	0.2692
MAPE	1.231	1.6255	1.0662	0.9084	1.3409	1.0142	0.9153	1.9276	1.4185
MSE	0.0102	0.4127	0.0183	0.0228	0.0378	0.1006	0.0622	0.2753	0.1605

Table 6. MAE of the tested methods, aggregated over all number of inputs used

Aggregation method	Mean	Min	Max
Box-Jenkins method	0.1927	0.1437	0.3299
DBN	0.1968	0.1482	0.3419
GMDH	0.1912	0.088	0.3302
ANFIS	0.2376	0.1517	0.3424
Suggested algorithm	0.1744	0.065	0.3253

As seen from the results of the comparison, the suggested algorithm's average, min and max MAE is the smallest of all the compared methods.

The Figure 3 below shows the forecast on an entire data set of the best accuracy model trained by using the suggested algorithm.

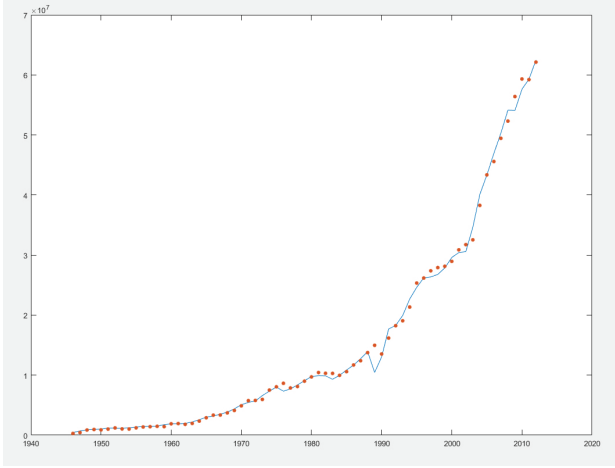

Figure 3. Forecast of the best model trained by the suggested algorithm vs the original data. Continuous line – original data, dots – forecast

Conclusions

The paper has proposed a new time series forecasting method based on GMDH and deep learning approaches. The proposed algorithm was used to predict Australia's major domestic airlines yearly revenue passenger kilometers, together with other forecasting methods, namely: GMDH, ANFIS, DBN and Box-Jenkins. As seen from the tables comparing the results, the average MAE error of the proposed method is approximately 10% smaller than the average MAE of next best method – GMDH, and the best reached MAE is approximately 25% smaller than the corresponding best MAE of the GMDH. This indicates that the practical application of the method can give good results compared to other well-known methods. The originality of this paper is the combination of the GMDH to perform a network's pre-training and gradient descent together with dropout to perform fine-tuning.

Disclosure statement

No financial, professional, or personal interests from other parties were reported by the authors.

References

Abed, S. Y., Ba-Fail, A. O., & Jasimuddin, S. M. (2001). An econometric analysis of international air travel demand in Saudi Arabia. *Journal of Air Transport Management*, *7*(3), 143-148. https://doi.org/10.1016/S0969-6997(00)00043-0

Asteriou, D., & Hall, S. G. (2011). ARIMA Models and the Box–Jenkins methodology. In *Applied Econometrics* (2nd ed., pp. 265-286). Palgrave MacMillan.

Australian Domestic Airline Activity-time series. (n.d.). Retrieved from https://bitre.gov.au/publications/ongoing/files/domestic_airline_activity_Domestic_Annual_Summary_1944_2012-13.xls

Ba-Fail, A. O., Abed, S. Y., & Jasimuddin, S. M. (2000). The determinants of domestic air travel demand in the Kingdom of Saudi Arabia. *Journal of Air Transportation World Wide*, *5*(2), 72-86.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate*. arXiv preprint arXiv:1409.0473.

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., & Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai (pp. 4945-4949). https://doi.org/10.1109/ICASSP.2016.7472618

Bengio, Y., De Mori, R., Flammia, G., & Kompe, R. (1992). Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transactions on Neural Networks*, *3*(2), 252-259. https://doi.org/10.1109/72.125866

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157-166. https://doi.org/10.1109/72.279181

Blinova, T. O. (2007). Analysis of possibility of using neural network to forecast passenger traffic flows in Russia. *Aviation 11*(1), 28-34.

Box, G., & Jenkins, G. (1970). *Time series analysis: Forecasting and control* (pp. 211-216). San Francisco: Holden-Day.

Chao, J., Shen, F., & Zhao, J. (2011, July). Forecasting exchange rate with deep belief networks. In *The 2011 International Joint Conference on Neural Networks (IJCNN)* (pp. 1259-1266). IEEE. https://doi.org/10.1109/IJCNN.2011.6033368

Doganis, R. (2009). *Flying off course: airline economics and marketing* (4th ed.). Abingdon: Routledge.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning?. *Journal of Machine Learning Research*, *11*(Feb), 625-660.

Fang, H. (2012). Adaptive neuro fuzzy inference system in the application of the financial crisis. *International Journal of Innovation, Management and Technology*, *3*(3), 250-254.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2014.81

Goyal, M. K., Bharti, B., Quilty, J., Adamowski, J., & Pandey, A. (2014). Modelling of daily pan evaporation in subtropical climates using ANN, LS-SVR, Fuzzy Logic, and ANFIS. *Expert Systems with Applications*, *41*(11), 5267-5276. https://doi.org/10.1016/j.eswa.2014.02.047

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). *Improving neural networks by preventing co-adaptation of feature detectors*. WS13/14 Machine Learning Seminar, (pp. 1-18). Retrieved from https://arxiv.org/pdf/1207.0580.pdf

Hinton, G. E. (1989). Deterministic Boltzmann learning performs steepest descent in weight-space. *Neural Computation*, *1*(1), 143-150. https://doi.org/10.1162/neco.1989.1.1.143

Hinton, G. E. (2009). Deep belief networks. *Scholarpedia*, *4*(5), 5947. https://doi.org/10.4249/scholarpedia.5947

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. Retreived from http://www.bioinf.jku.at/publications/older/ch7.pdf

Hoeting, J. A., Madigan, D., Raftery, A. E., & Volinsky, C. T. (1998, May). Bayesian model averaging. In *Proceedings of the AAAI Workshop on Integrating Multiple Learned Models* (Vol. 335, pp. 77-83).

Jang, J. S. R. 1993. ANFIS-adaptive-network-based fuzzy inference system. *IEEE Transactions Systems, Man and Cybernetics*, *23*(3), 665-685. https://doi.org/10.1109/21.256541

Jang, J. S. R., Sun, C. T., & Mizutani, E. (1997). *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence* (1st ed.). New Jersey: Prentice Hall.

Kondo, T. (1998, July). GMDH neural network algorithm using the heuristic self-organization method and its application to the pattern identification problem. In *SICE'98. Proceedings of the 37th SICE Annual Conference*. International Session Papers (pp. 1143-1148). IEEE. https://doi.org/10.1109/SICE.1998.742993

Krogh, A., & Hertz, J. A. (1992). A simple weight decay can improve generalization. In M. I. Jordan, Y. LeCun & S. A. Solla. *Advances in neural information processing systems* (pp. 950-957). The MIT Press.

Kuremoto, T., Kimura, S., Kobayashi, K., & Obayashi, M. (2014). Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neurocomputing*, *137*, 47-56. https://doi.org/10.1016/j.neucom.2013.03.047

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (1998). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9-50). Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-49430-8_2

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009, June). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 609-616). ACM. https://doi.org/10.1145/1553374.1553453

Liu, C., Liu, X., Huang, H., & Zhao, L. (2008). Low circle fatigue life model based on ANFIS. In D. S. Huang, D. C. Wunsch II, D. S. Levine, et al. (Eds.). *Advanced intelligent computing theories and applications: With aspects of contemporary intelligent computing techniques* (pp. 139-144). Berlin: Springer Verlag. https://doi.org/10.1007/978-3-540-85930-7_19

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing atari with deep reinforcement learning*. arXiv preprint arXiv:1312.5602.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, *11*(4), 761-767. https://doi.org/10.1016/S0893-6080(98)00010-0

Qiu, X., Zhang, L., Ren, Y., Suganthan, P. N., & Amaratunga, G. (2014, December). Ensemble deep learning for regression and time series forecasting. In *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)* (pp. 1-6). IEEE. https://doi.org/10.1109/CIEL.2014.7015739

Riedmiller, M., & Braun, H. (1992). Rprop – A fast adaptive learning algorithm. In *Proceedings of the International Symposium on Computer and Information Science VII* (pp. 57-64).

Rosenblatt, F. (1958). The perceptron: A probalistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386-408. https://doi.org/10.1037/h0042519

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61,* 85-117. https://doi.org/10.1016/j.neunet.2014.09.003

Srinivasan, D. (2008). Energy demand prediction using GMDH networks. *Neurocomputing*, *72*(1-3), 625-629. https://doi.org/10.1016/j.neucom.2008.08.006

Stepashko, V. S. (1988). GMDH Algorithms as basis of modeling process automation after experimental data. *Soviet Journal of Automation and Information Sciences*, *21*(4), 43-53.

Xiao, Y., Liu, J. J., Hu, Y., Wang, Y., Lai, K. K., & Wang, S. (2014). A neuro-fuzzy combination model based on singular spectrum analysis for air transport demand forecasting. *Journal of Air Transport Management*, *39*, 1-11. https://doi.org/10.1016/j.jairtraman.2014.03.004

Yetilmezsoy, K., Fingas, M., & Fieldhouse, B. (2011). An adaptive neuro-fuzzy approach for modelling of water-in-oil emulsion formation. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, *389*(1-3), 50-62. https://doi.org/10.1016/j.colsurfa.2011.08.051