Deep neural networks for ultra-short-term wind forecasting

Mladen Đalto, Jadranko Matuško, Mario Vašak
University of Zagreb,
Faculty of Electrical Engineering and Computing,
Zagreb, Croatia
Email: mladen.dalto@fer.hr, jadranko.matusko@fer.hr, mario.vasak@fer.hr

***Abstract*—The aim of this paper is to present input variable selection algorithm and deep neural networks application to ultra-short-term wind prediction. Shallow and deep neural networks coupled with input variable selection algorithm are compared on the ultra-short-term wind prediction task for a set of different locations. Results show that carefully selected deep neural networks outperform shallow ones. Input variable selection use reduces the neural network complexity and simplifies deep neural network training.**

1. Introduction

Artificial neural networks with backpropagation learning algorithm are widely used in solving various classification and prediction problems. Until recently the depth of a practical neural network (NN) was mostly limited to one hidden layer whereas such NN is named shallow. Deep Neural Networks (DNN) have more than one hidden layer of activation functions. Often by using many layers of nonlinear functions we can model complicated functions more efficiently e.g. with less parameters and activation functions. Insufficiently deep architectures sometimes require larger number of computational elements to approximate the same functions [4]. Recently deep architectures have reported state-of-the-art performance across fields such as object recognition [18], speech recognition [2], natural language processing [12], physiological affect modelling [22], etc.
There is still little literature on deep neural networks and time-series prediction. Few papers on time-series prediction or classification with deep neural networks stand out. Convolutional Neural Networks (CNNs) are used for speech recognition [1], congestive heart failure detection [32], psychological models of affect [22], etc. In [32] multi-channel deep convolutional neural networks were used. This architecture extracts features by using convolutional neural networks on each input (1-D time-series) separately, and feeds them to a Multi Layer Perceptron (MLP) NN for classification.
Energy load forecasting in [8] was performed using a deep feedforward neural network, large datasets and man-made features. Best results were obtained by using 3 layers, whilst Stacked Autoencoder (SA) pretraining gave no significant performance gain.
The effect of Input Variable Selection (IVS) for DNN with applications in ultra-short-term wind forecasting is also explored. Performance is compared on various input configurations (regressors) with varying availability of measured and forecast variables. Prediction performance of DNNs is compared to simple persistent prediction on 3 hour prediction horizon. Input data in earlier mentioned state-of-the-art fields is already temporally (1-D time series) correlated, spatially correlated (images) or can be transformed to have such properties, e.g., sound, by using time-frequency analysis.
The multivariate time series data set often does not exhibit such relationships among variables as they describe different phenomena. Input Variable Selection (IVS) is used to reduce the complexity induced by high number of redundant inputs, which is a dominant contributing factor to complexity in the case of overcomplete hidden layers.
Reviews of wind speed and power prediction methods [31] [17] [27] classify prediction methods into physical (numeric weather prediction), statistical (i.e. linear models prediction, support vector machines, neural networks) and hybrid (i.e. NN-fuzzy, wavelet-NN, various prediction models ensembles) classes. Numeric Weather Prediction (NWP) is more suitable for larger horizons leaving us with the choice of statistical or hybrid methods. It has been shown, on a specific wind prediction task, that multivariate recursive neural network models outperform ARIMA models [10] and often outperform other linear models [26].
Hybrid component of our approach lies in using NWP variables as DNN inputs. This can still be regarded as a statistical approach because only the input variables set is expanded. From a machine learning perspective, shallow models such as linear models, support vector regression and neural networks with a single hidden layer [9] [19], are dominantly used in wind speed and power prediction. Support vector regression with features derived from deep autoencoders was used in [20]. Using many layers of features results in increase of performance in comparison to simple Support Vector Regression use. This motivated DNN use for ultra-short term wind prediction. Unfortunately, training deep architectures is a difficult task and classical methods that have proved efficient when applied to shallow architectures are not as easily transferable to deep architectures. Moreover, adding layers does not necessarily lead to better results [6].
In this work the prediction performance of Shallow Neural Network (SNN) is compared to DNN initialized using normalized initialization [16]. Prediction performance comparison of DNNs to shallow NNs is given on historic test data from 3 wind measurement locations in Croatia. The remainder of the work is organized as follows. Section 2 revises mathem-

The research described in this paper are part of the project "Weather Intelligence for Wind Energy" (WILL4WIND) supported by the European Union under contract IPA2007/HR/16IPO/001-040507.

atical notation of MLP. Partial Mutual Information (PMI) IVS algorithm is described in Section 3. Section 4 describes data and features used in this work. Prediction models are described in Section 5. Section 6 presents the performance increase of DNNs over SNNs with using IVS.

2. Neural network architectures

This paper focuses on MLP NN architectures. In the following work we make use of $n(\cdot)$ notation to denote dimension of some vector $\mathbf{v}$: $n(\mathbf{v})$, or $n(l)$ when denoting the dimension of a layer, where $0 \leq l \leq L$. Mathematical description of MLP NN with L layers can be formulated as follows:

$$\mathbf{y}_0 = \mathbf{x}. \tag{1}$$

$$\mathbf{x}_l = \left[\mathbf{y}_{l-1}^T, 1\right]^T, \quad 1 \leq l \leq L, \tag{2}$$

$$\mathbf{v}_l = \mathbf{W}_l \cdot \mathbf{x}_l, \quad 1 \leq l \leq L, \tag{3}$$

$$\mathbf{y}_l = \psi_l(\mathbf{v}_l), \quad 1 \leq l \leq L, \tag{4}$$

where:
$\mathbf{x} = \left[x_1, x_2, ..., x_{n(x)}\right]^T \in \mathbb{R}^{n(x)\times 1}$ - input vector;
$\mathbf{y}_0 = \left[y_{0,1}, y_{0,2}, ..., y_{0,n(0)}\right]^T \in \mathbb{R}^{n(0)\times 1}$ - layer 0 output vector;
$\mathbf{x}_l = \left[x_{l,1}, x_{l,2}, ..., x_{l,n(l-1)}, x_{l,n(l-1)+1}\right]^T \in \mathbb{R}^{(n(l-1)+1)\times 1}$ - input vector of layer l;
$\mathbf{v}_l = \left[v_{l,1}, v_{l,2}, ..., v_{l,n(l)}\right]^T \in \mathbb{R}^{n(l)\times 1}$ - synaptic operation output vector of layer l;
$\mathbf{y}_l = \left[y_{l,1}, y_{l,2}, ..., y_{l,n(l)}\right]^T \in \mathbb{R}^{n(l)\times 1}$ - output vector of layer l;
$\mathbf{W}_l = \left[\mathbf{W}_l' \; \mathbf{b}_l\right] \in \mathbb{R}^{n(l)\times(n(l-1)+1)}$ - synaptic weight coefficients matrix of layer l;
$\boldsymbol{\psi}_l(\mathbf{v}_l) = \left[\psi_{l,1}(v_{l,1}), \psi_{l,2}(v_{l,2}), ..., \psi_{l,n(l)}(v_{l,n(l)})\right]^T \in \mathbb{R}^{n(l)\times 1}$ - element-wise activation functions vector of layer l (we use $\psi_l = \psi_{l,1} = \cdots = \psi_{l,n(l)}$).

All elements of matrices $\mathbf{W}_l$ are concatenated into one vector of parameters:

$$\boldsymbol{\Theta} = \left[vec(\mathbf{W}_1)^T \quad \cdots \quad vec(\mathbf{W}_L)^T\right]^T \in \mathbb{R}^{n(\Theta)\times 1}, \tag{5}$$

where

$$n(\Theta) = \sum_{l=1}^{L} n(l) \cdot [n(l-1) + 1] \tag{6}$$

is the total number of MLP NN parameters, and $vec(\cdot)$ is a linear transformation which converts the matrix into a column vector.
Neural networks are deep if $L > 2$.

3. Input Variable Selection

IVS is very important in identification of the optimal structure for statistical models. The task of choosing input variables is common in the development of all statistical models and greatly depends on discovering connections between available input-output data for detection of suitable inputs (so-called predictors) for output prediction. In the case of NN, as well as some other similar data-based models, there is no assumption on a specific input-output mapping structure, so a general structure is assumed. In that case several problems occur:

- large number of input variables,
- correlation amongst available variables,
- uninformative variables.

Many IVS procedures do not give good results because linear relationships are assumed or because of redundancy in the input candidate set.

A. Partial mutual information

Recently Mutual information (MI) is being used as a more suitable measure for IVS. Mutual information is defined in [13] as:

$$I(X,Y) = \int_Y \int_X p(x,y) \log\left(\frac{p(x,y)}{p(x)\,p(y)}\right) dx\,dy, \tag{7}$$

where $p(x,y)$ represents joint probability distribution and $p(x)$ and $p(y)$ marginal probability distributions of X and Y. In practical context probability distributions in (7) are rarely known. MI advantage is in the fact that it does not make assumptions about the structure of the analysed variables and is invariant to transformations and noise [23]. Simple application of MI cannot detect redundancy between candidate input variables and lacks a stopping criterion. Partial Mutual Information (PMI) is a nonlinear IVS algorithm that can completely or partially overcome these problems. PMI between the output Y and candidate C_j with preselected predictors $\mathbf{S}$ is defined as:

$$\begin{aligned} PMI(Y, C_j|\mathbf{S}) =& H(Y,\mathbf{S}) + H(\mathbf{S}, C_j) \\ & - H(\mathbf{S}) - H(Y,\mathbf{S},C_j), \end{aligned} \tag{8}$$

where Y, C_j are random scalars, $\mathbf{S}$ a random vector and $H(\cdot)$ represents Shannon entropy:

$$H(X) = -\sum_i p(x_i) \log p(x_i). \tag{9}$$

A detailed formulation of entropy for multiple arguments is provided in [15]. In a special case, when no predictors are preselected, PMI becomes $PMI(Y, C_j|\emptyset) = I(Y, C_j)$. Two variants of PMI were implemented and tested: one based on Kernel Density Estimation (KDE) from [23] and another based on k-nearest neighbours from [15]. In further work KDE based PMI is used.
Consider a system with output Y, and with S and C_j as predictor candidates. If $I(Y;S) \geq I(Y;C_j)\,\forall j$, predictor S is selected as a first candidate because it introduces the most information required to predict Y. Now we perform a second step of the algorithm with preselected S. The information contribution of any candidate C_j is represented by $PMI(Y;C_j|S)$.

B. IVS and MLP NN complexity

We use the number of parameters of a NN to describe its complexity. This is not a fully informative measure but it is widely used to describe complexity of a structure and it is sufficient for this work. This subsection deals with the effect IVS has on DNN complexity. Practical recommendation [5] is to use DNN layers of same size ($n(1) = \cdots = n(L-1)$) so we consider such a setting. Taking layer constraints and number of MLP NN parameters given by (6) we get:

$$n(\Theta) = (L-2)(n(l)+1)n(l)+[n(0) + n(L) + 1]\, n(l)+n(L). \tag{10}$$

The following complexity comparison is performed under a reasonable assumption of choosing the size of the first layer as $n(1) = \alpha n(0)$ where $\alpha > 1$ and constant for two different input sizes $n(0)$.

In the first case, using all available input variables $n_1(0)$ is not practical, so IVS is performed to obtain a reduced input regressor $n_2(0) = \beta n_1(0)$, where $0 < \beta < 1$. We compare the complexity of using all available inputs to complexity of using a subset of variables obtained by IVS. Complexity ratio $\frac{n_2(\Theta)}{n_1(\Theta)}$, derived from (10), dominantly depends on the square of input size reduction β^2.

This analysis should be taken with some caution as it is valid under given assumptions and constraints, but still greatly depends on the problem and data for which it is used.

4. Data and features

Three years (2010-2012) of measurement data were obtained for wind on 10 m relative height above the ground in main meteorological stations of Split, Šibenik and Knin in Croatia. The used data is owned by Croatian Meteorological and Hydrological Service - DHMZ which is a public institution. The data consists of physical quantities measured with sampling time of $T_s = 10$ min. They represent either 10-min mean, maximum or 10-min interval terminal values of various quantities (wind temperature, pressure etc.); times of maximum, minimum within T_s and other artificiality derived features (e.g. turbulence, gustiness). Measured data is described in more detail in [3]. Additionally we use numeric weather prediction products of ALADIN model [29] for the nearest geographic point to each of the measuring locations. ALADIN inputs represent hourly past values of wind speed, direction, temperature, atmospheric pressure etc for the same interval (2010-2012). Forecast data was persistently interpolated to obtain 10 min samples. All data were normalized to $[-1, 1]$.

The use of supervised learning requires a proper series of cause-consequence data pairs. Data pairs need to be formed from historic data which contain considerable amount of missing values and other faulty measurements. These were labeled as such, and removed from the set. Original measurements of outputs Y represent wind speed and direction. Because of seemingly abrupt changes in wind direction measurements $\theta(i)$ when crossing 360° or 0° we use orthogonal wind components given by:

$$v_x(i) = ||v(i)||\cdot \cos\left(\frac{\pi\theta(i)}{180}\right), \tag{11}$$

$$v_y(i) = ||v(i)||\cdot \sin\left(\frac{\pi\theta(i)}{180}\right). \tag{12}$$

By doing so smooth functions are used for mapping direction and wind speed change.

We define the orthogonal wind components regressor:

$$\begin{aligned}\mathbf{Y}_{t,d} = [&v_x(t), v_x(t-1), \cdots, v_x(t-d),\\ &v_y(t), x_y(t-1), \cdots, v_y(t-d)]^T,\end{aligned} \tag{13}$$

where d denotes the number of delayed samples. Datasets contain concatenated measured and forecast variables $\mathbf{X}_t$ with orthogonal components of the future wind as prediction targets $\mathbf{Y}_{t+T,T-1}$. Data is grouped in cause-consequance pairs:

$$\mathcal{D} = \{(\mathbf{X}_i^T, \mathbf{Y}_i^T) : \mathbf{X}_i \in \mathbb{R}^{n(\mathbf{X})\times 1}, \mathbf{Y}_i \in \mathbb{R}^{n(L)\times 1}\}_{i=1}^N \tag{14}$$

where N denotes the number of samples.

A. *Dataset for NN without IVS*

A simple way to choose NN inputs is to use as much variables as available. The initial regressor consists of past input variables and past outputs:

$$\mathbf{X}_t = \left[\mathbf{U}_{t,d}^T \mathbf{Y}_{t,d}^T\right]^T, \tag{15}$$

where $\mathbf{U}_{t,d}$ denotes the regressor of non-wind variables:

$$\begin{aligned}\mathbf{U}_{t,d} = [&x_1(t), x_1(t-1), \cdots, x_1(t-d),\\ &x_2(t), x_2(t-1), \cdots, x_2(t-d), \cdots,\\ &x_k(t), x_k(t-1), \cdots, x_k(t-d)]^T\end{aligned} \tag{16}$$

where d denotes the number of delayed samples and k is the number of variables before regressor formation. Regressor was formed to use past 3 hours of data, which corresponds to $d = 17$. Without using IVS we select the initial regressor $\mathbf{S}_t = \mathbf{X}_t$ as the prediction model input.

B. *Datasets for NN with IVS*

Input regression vector $\mathbf{X}_t$ increases the number of inputs compared to k initial multivariate input sources. This is necessary in order to capture dependencies between past and future events. Inputs formed in such a way may include unnecessary past variables. For this reason IVS is used to select a proper subset of input regressor vector elements.

The predictor candidates $\mathbf{C}_t$ consist of variables in the regressor $\mathbf{X}_t$, $\mathbf{C}_t = \mathbf{X}_t$. Choice between predictor candidates $\mathbf{C}_t$ is made with PMI IVS algorithm described in Subsection 3.1. If we define the NN to have $n(L) = 2T$ outputs, it is necessary to use PMI algorithm for every output $\mathbf{y}_{L,j}$ and candidates in $\mathbf{C}_t$ to obtain selected candidates $\mathbf{S}_{t,j}$ for $j = 1, ..., n(L)$. After obtaining $\mathbf{S}_{t,j}$ unique variables are used to form the final vector of predictors $\mathbf{S}_t$ for the use with the NNs. With the use of PMI a subset of neural network inputs was obtained with statistics in Table I.

Location	N	$\lvert\mathbf{C}\rvert = k\cdot 18$	$\lvert\mathbf{S}\rvert$	Reduction
Split	100022	$1512 = 84\cdot 18$	125	91.73%
Šibenik	9905	$1530 = 85\cdot 18$	235	84.64%
Knin	5206	$1584 = 88\cdot 18$	207	86.93%

Table I: Dataset statistics.

PMI IVS drastically reduces the number of inputs compared to the use of all available inputs. It consequently reduces the number of NN parameters as well.

5. Ultra-short term wind prediction

In this work we use a single prediction model to predict orthogonal wind components on a 3 hour horizon with 10 min sampling time. This problem definition gives us the prediction horizon of $T = 18$ samples for 2 components. Mathematical models used for prediction have the following form:

$$\hat{\mathbf{Y}}_{t+T,T-1} = f(\mathbf{S}_t, \Theta) \in \mathbb{R}^{2T\times 1}. \quad (17)$$

A. Persistent prediction

Persistence or persistent estimator is the simplest forecast model and in this work serves for NN forecast evaluation exclusively [28]. It assumes that wind speed and direction will remain the same for the prediction horizon T. It is defined as follows:

$$\left[\hat{\mathbf{Y}}_{t+T,T-1}\right]_j = \begin{cases} v_x(t), & j = 1,...,T \\ v_y(t), & j = T+1,...,2T. \end{cases} \quad (18)$$

B. Neural networks

A simple feedforward MLP architecture is used for predictions. Using a purely feedforward network we ignore some of the temporal structure in time series data and gain in training and use simplicity [25].
NN prediction uses the prediction model (17) where for $f(\cdot)$ we use an MLP NN with parameters Θ. For wind prediction $L = 4$ layers were used with only hyperbolic tangent functions $\psi_l(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ as activation functions. Neural network has $n(4) = 2T = 36$ outputs which represent 3 hour horizon wind predictions with 10 min resolution, for North and East direction.
Datasets for 3 locations of N samples were divided by uniform sampling into subsets with training samples N_t (70%), validation (15%) and testing (15%). Neural network parameters $\mathbf{\Theta}$ are optimized to minimize the mean squared error with L2 regularization:

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} \frac{1}{N_t} \sum_{i=1}^{N_t} \|\mathbf{Y}_i - f(\mathbf{S}_i, \Theta)\|_2^2 + \lambda \sum_{j=1}^{n(\mathbf{\Theta})} \theta_j^2, \quad (19)$$

where λ represents the regularization parameter. A Scaled Conjugate Gradient algorithm [24] for fast supervised learning is used to minimize Mean Squared Error (MSE) learning criteria with regularization. Early stopping and $L2$ regularization were used to ensure generalisation.
DNNs were initialized using normalized initialization [16], which considerably contributes to finding good solutions, especially when used with hyperbolic tangent activation functions.

6. Results

This section presents NN prediction performance obtained with the data, input configurations and prediction models described in former sections. Mean Absolute Error (MAE) is used for results comparison. Motivated by deep learning literature it was our hope that unsupervised pre-training of Stacked Denoising Autoencoders (SDA) [30] and marginalized Stacked Denoising Autoencoder (mSDA) [11] could help improve the DNN prediction results. Results obtained with SDA pre-training were almost the same as backpropagation supervised training results, with added computational complexity. This is attributed to using sufficient labelled data for training. Unsupervised pre-training has been shown to help in problems where large amount of unlabeled data is available, which is not the case with our data. However, results with mSDA initialization were discouraging. The difference between standard MLP NN and mSDA is in the pre-training phase and mSDA layer size constraint ($n(1) = \cdots = n(L-1) = n(0)$), otherwise it has an equivalent structure as MLP NN. Layer size constraint was a limiting factor in mSDA use. Using outputs of intermediate layers (generated with SDA or mSDA) as NN inputs was avoided because of the large size of the initial regressor.

A. Model complexity considerations

Figure 1 shows MAE of NNs trained on data for 3 locations and varying NN layer number and sizes. Hidden layers with 300 neurons are added to form a deeper structure for every new training. Every point represents a MAE of a NN structure (obtained with one training) and the underlying number of parameters. Because of random parameter initialization results can vary. DNNs trained on data from 3 locations show im-

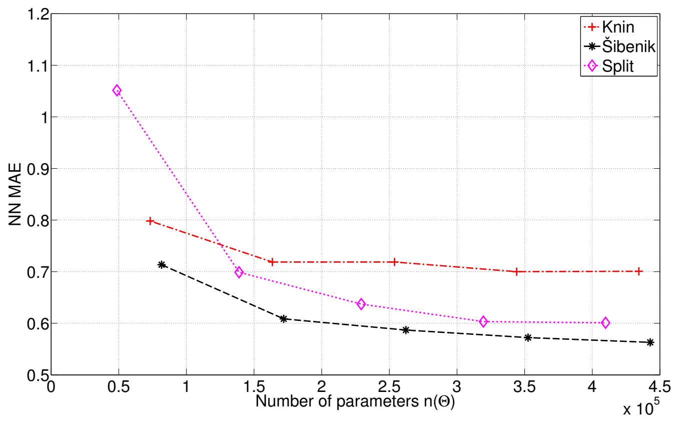

Figure 1: NN topology selection.

provements with increased depth and number of parameters. For locations with smaller datasets (Šibenik and Knin) DNNs ($n(1) = 170, n(2) = n(3) = 120$), with smaller number of parameters, show better performance compared to shallow NN and DNNs of varying depth and with larger number of neurons. Table II shows structures chosen for further work. Depth was limited to $L = 4$ because of increase in complexity which considerably slows down learning for the largest dataset.

Location	Structure n(1)-n(2)-n(3)	MAE
Split	300-300-300	0.55
Šibenik	170-120-120	0.52
Knin	170-120-120	0.52

Table II: DNN structures used in further work.

B. IVS results

Without using IVS ($\mathbf{S}_t = \mathbf{C}_t$) the initial regressor contains variables and their past values that might change very slowly. These values then become redundant i.e. they provide very small information contribution to the supervised optimization task at hand. Redundant variables increase the number of local minima when training NN. On the other hand irrelevant variables add noise which makes training more difficult. Because of these problems and increased model complexity without using IVS, PMI algorithm is used to obtain NN inputs [7][21]. Their use for prediction is compared to the initial regressor inputs for Šibenik and Knin locations. Deep neural networks with structures in Table II are trained with $\mathbf{S}_t$ inputs obtained by PMI and the initial regressor. Results are shown in Figure 2.

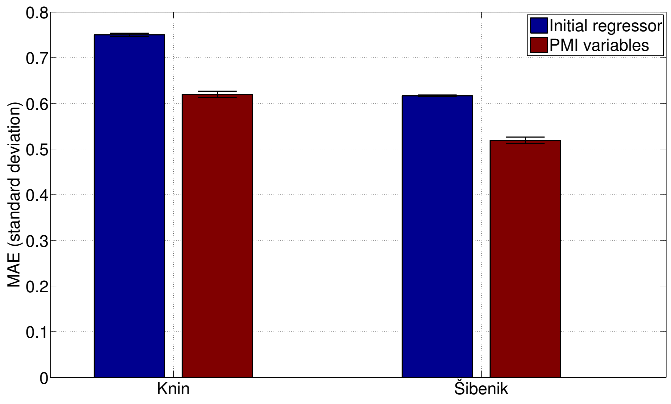

Figure 2: DNN prediction MAE comparison of IVS $\mathbf{S}_t$ and $\mathbf{X}_t$ inputs.

C. DNN and SNN comparison

Comparison of SNNs and DNNs is performed by training each model 5 times. Figure 3 shows a MAE comparison with standard deviation on the same training, validation and test data.

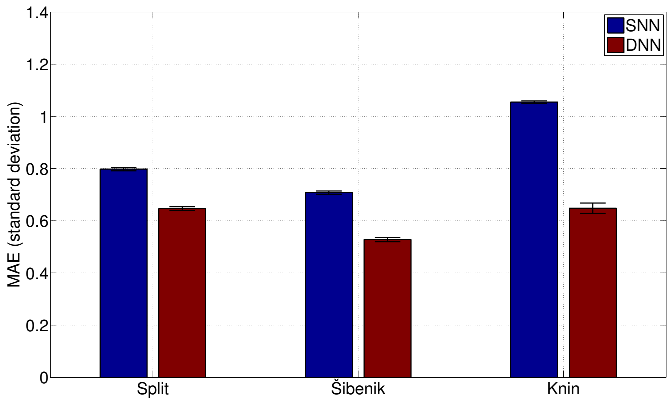

Figure 3: SNN and DNN MAE comparison for 3 locations.

The hidden layer size $n(1)$ of SNN was determined to match the number of parameters $n(\Theta)$ of structures in Table II using the following equation:

$$n(1) = \left\lceil \frac{n(\Theta) - n(2)}{n(0) + n(2) + 1} \right\rceil. \quad (20)$$

Table III shows the hidden layer sizes obtained by using (20) and MAE of SNNs and DNNs.

Location	n(1)	SNN MAE	DNN MAE
Split	1415	1.05	0.65
Šibenik	293	0.71	0.53
Knin	326	0.78	0.65

Table III: SNN structures and MAE comparison.

In both DNNS and SNNs considered hyperbolic tangent function is chosen as activation function in all layers, while inputs $\mathbf{S}_t$ are obtained by IVS. Results clearly favour DNNs over shallow NNs with approximately the same number of parameters. This result indicates that DNNs are more efficient structures for wind prediction on data for our 3 locations.

D. Input configuration performance tests

Figure 4 shows test results for DNNs obtained by a single training with inputs selected with PMI algorithm on various input configurations. A decrease in MAE is evident as we add more variables (input candidates). Largest error is obtained when using only forecast variables to predict measured wind, except for Šibenik location which seems to have more benefit from using forecast variables than a regressor of past measured wind variables. Using other measured data besides wind (pressure, relative humidity, etc.) further decreases the MAE. Finally, a combination of measured and forecast variables gives the lowest MAE.

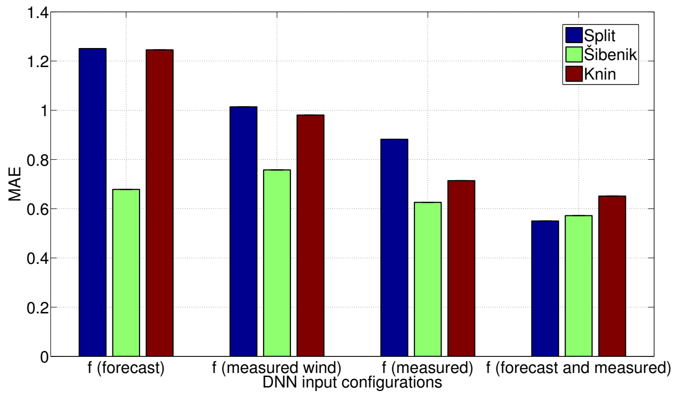

Figure 4: Various input configurations comparison for 3 locations.

E. DNN vs. persistance on the prediction horizon

This subsection compares the results of DNN and persistant prediction on the entire prediction horizon. Mean absolute error with standard deviation of prediction error on the test set is presented in Figure 5. Only the upper errorbar is used for clarity. Test set was not used during training in any way. Because of space limitations results are shown only for a single location (Split), for which the largest test set was used. Prediction MAE for Knin and Šibenik locations is similar. Results clearly show the advantages of using DNN instead of persistent forecast, even on short horizons, where persistent estimation is hard to beat [28][14].

7. Conclusion

Although the increase in the number of informative variables for neural network predicition of ultra-short term wind

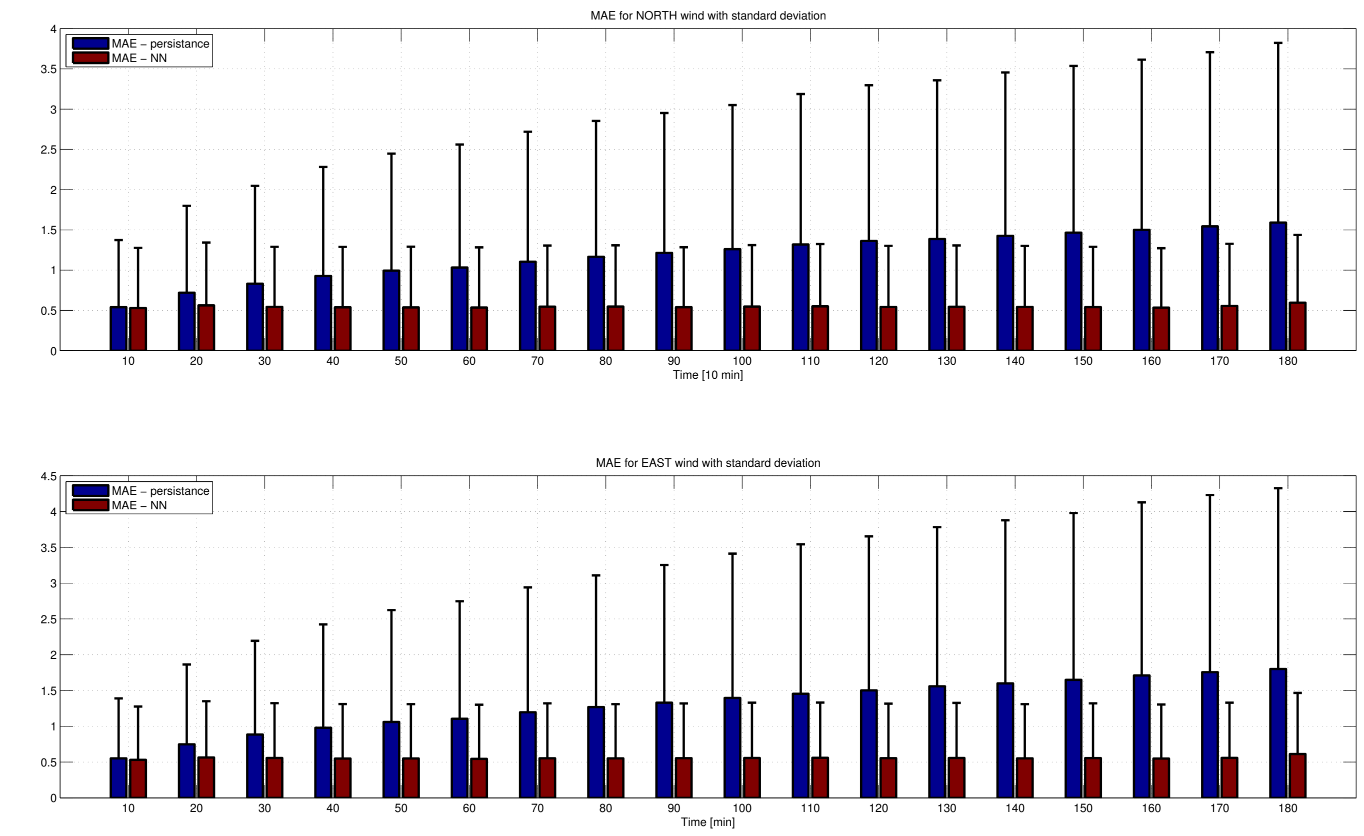

Figure 5: DNN MAE with standard deviation on 3 hour horizon for Split location.

improves prediction performance, it also increases the prediction model complexity. Classic neural networks prediction approach with all available variables has limited prediction performance because of redundant and irrelevant input variables and less efficient structure. PMI based IVS algorithm is used to reduce input size and number of NN parameters. Many recent papers on deep neural networks and training techniques provide evidence of advantages over classic, shallow neural networks. Deep neural networks benefit from input variable selection to reduce complexity and make learning computationally possible in reasonable time. In this paper additional evidence to support DNNs use is provided for ultra-short-term wind forecast. DNN results obtained in this work rely on the simple normalized initialization and represent a baseline for future work on more advanced architectures. Increased performance of DNNs compared to SNNs with approximately the same number of parameters and for 3 different forecasting locations indicate increased efficiency of deep architectures for ultra-short-term wind prediction.

References

[1] O. Abdel-Hamid, L. Deng, and D. Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Proceedings of Interspeech*, pages 3366–3370, 2013.

[2] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, and G. Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Proceedings of Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4277–4280. IEEE, 2012.

[3] M. Đalto, M. Vašak, M. Baotić, J. Matuško, and K. Horvath. Neural-network-based ultra-short-term wind forecasting. *European Wind Energy Association 2014 Annual Event (EWEA 2014)*, 2014.

[4] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[5] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

[6] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[7] G. J. Bowden, G. C. Dandy, and H. R. Maier. Input determination for neural network models in water resources applications. Part 1—background and methodology. *Journal of Hydrology*, 301(1):75–92, 2005.

[8] E. Busseti, I. Osband, and S. Wong. Deep learning for time series modeling. Technical report, Stanford University, 2012.

[9] E. Cadenas and W. Rivera. Short term wind speed forecasting in La Venta, Oaxaca, México, using artificial neural networks. *Renewable Energy*, 34(1):274–278, 2009.

[10] Q. Cao, B. T. Ewing, and M. A. Thompson. Forecasting wind speed with recurrent neural networks. *European Journal of Operational Research*, 221(1):148–154, 2012.

[11] M. Chen, Z. Xu, F. Sha, and K. Q. Weinberger. Marginalized denoising autoencoders for domain adaptation.

In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 767–774, 2012.

[12] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[13] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[14] I. Espino and M. Hernández. Nowcasting of wind speed using support vector regression. Experiments with time series from Gran Canaria. In *Proceedings of International Conference on Renewable Energies and Power Quality*, 2011.

[15] S. Frenzel and B. Pompe. Partial mutual information for coupling analysis of multivariate time series. *Physical review letters*, 99(20), 2007.

[16] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[17] J. Jung and R. P. Broadwater. Current status and future advances for wind speed and power forecasting. *Renewable and Sustainable Energy Reviews*, 31:762–777, 2014.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in neural information processing systems*, pages 1097–1105, 2012.

[19] G. Li and J. Shi. On comparing three artificial neural networks for wind speed forecasting. *Applied Energy*, 87(7):2313–2320, 2010.

[20] J. N. K. Liu, Y. Hu, J. J. You, and P. W. Chan. Deep neural network based feature representation for weather forecasting. In *Proceedings of The 2014 Internatonal Conference on Artificial Intelligence*, pages 261–267, 2014.

[21] I. Luna, S. Soares, and R. Ballini. Partial mutual information criterion for modelling time series via neural networks. In *Proceedings of The 11th Information Processing and Management of Uncertainty International Conference*, volume 1, pages 2012–2019, 2006.

[22] H. P. Martinez, Y. Bengio, and G. N. Yannakakis. Learning deep physiological models of affect. *Computational Intelligence Magazine, IEEE*, 8(2):20–33, 2013.

[23] R. May, G. Dandy, and H. Maier. Review of input variable selection methods for artificial neural networks. *Artificial neural networks—methodological advances and biomedical applications*, pages 19–44, 2011.

[24] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6:525–533, 1993.

[25] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *Journal of Machine Learning Research WCP 28 (3) :1310-1318*, 2013.

[26] A Sfetsos. A comparison of various forecasting techniques applied to mean hourly wind speed time series. *Renewable Energy*, 21(1):23–35, 2000.

[27] S. S. Soman, H. Zareipour, O. Malik, and P. Mandal. A review of wind power and wind speed forecasting methods with different time horizons. In *Proceedings of North American Power Symposium (NAPS), 2010*, pages 1–8. IEEE, 2010.

[28] S. S. Soman, H. Zareipour, O. Malik, and P. Mandal. A review of wind power and wind speed forecasting methods with different time horizons. In *Proceedings of North American Power Symposium (NAPS), 2010*, pages 1–8. IEEE, 2010.

[29] M. Tudor, S. Ivatek-Šahdan, A. Stanešić, K. Horvath, and A. Bajić. Forecasting weather in Croatia using ALADIN numerical weather prediction model. *Climate Change and Regional/Local Responses*, pages 59–88, 2013.

[30] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.

[31] Y. K. Wu and J. S. Hong. A literature review of wind forecasting technology in the world. In *Proceedings of Power Tech, 2007 IEEE Lausanne*, pages 504–509. IEEE, 2007.

[32] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao. Time series classification using multi-channels deep convolutional neural networks. In *Web-Age Information Management*, pages 298–310. Springer, 2014.