Weighted Multi-view Deep Neural Networks for Weather Forecasting

Zahra Karevan, Lynn Houthuys, and Johan A. K. Suykens

KU Leuven, ESAT-STADIUS Kasteelpark Arenberg 10 B-3001 Leuven, Belgium
{zahra.karevan,lynn.houthuys,johan.suykens}@esat.kuleuven.be

Abstract. In multi-view regression the information from multiple representations of the input data is combined to improve the prediction. Inspired by the success of deep learning, this paper proposes a novel model called Weighted Multi-view Deep Neural Networks (MV-DNN) regression. The objective function used is a weighted version of the primal formulation of the existing Multi-View Least Squares Support Vector Machines method, where both the objectives from all different views, as well as the coupling term, are weighted. This work is motivated by the challenging application of weather forecasting. To predict the temperature, the weather variables from several previous days are taken into account. Each feature vector belonging to a previous day (delay) is regarded as a different view. Experimental results on the minimum and maximum temperature prediction in Brussels, reveal the merit of the weighting and show promising results when compared to existing the state-of-the-art methods in weather prediction.

Keywords: Multi-view Learning, Neural Networks, Deep Learning, Weather Forecasting

1 Introduction

Accurate weather prediction is one of the most challenging tasks in climate informatics. The prediction task is being complicated due to various environmental issues like topography of surrounding structures, the chaotic characteristics of the atmosphere, the influence of human behavior and many more. State-of-the-art methods usually apply the Numerical Weather Prediction (NWP) to get a decent prediction. Because this method is very computationally intensive [3], there is an increasing interest in data-driven methods which uses historical data to do the prediction.

Multi-view learning denotes a group of learning techniques that are applied when the data is described through multiple representations, or views. By using the information available from all views, multi-view learning aims to improve the performance over only using a single view. This could be achieved by simply concatenating the features from all views, like e.g. the work done by Zilca & Bistritz [22]. This approach is a typical example of early fusion, as the information from all views is fused early on in the training process. However, as this increases the dimensionality greatly and it ignores the statistical properties of each individual view, most state-of-the-art multi-view methods aim to jointly optimize the

objectives of each view [5,19,20]. Liu et al. [13] propose a multi-task multi-view method to predict Urban Water Quality where the spatial and temporal features are considered as two views. Houthuys et al. [10] propose a multi-view kernel-based method to predict temperature by considering the neighboring cities as different views.

Inspired by the success of deep learning in feature learning research [7,12,4], several deep multi-view methods have recently been proposed. These techniques are usually based on one of the two main approaches [18]. The first approach is based on autoencoders, like e.g. the work done by Ngiam et al. [14], the second is based on a Deep Neural Network (DNN) extension to CCA-like [9] methods, e.g. the deep CCA method proposed by Andrew et al. [2].

In this paper a novel DNN-based multi-view method is proposed and its performance is evaluated on the application of temperature prediction. The model follows the second deep multi-view approach, where the novelty lies in the weighting of the view-specific objectives and the coupling term. These weights can be tailored to the needs of each application. They can be determined through e.g. the similarity to the prediction task, some expert knowledge about the application, and so on. For our application we have chosen to weight according to the performance of each view on the validation set when used separately. Temperature prediction is a multi-variate time-series forecasting application, and hence the prediction of a certain day depends on the features of the previous days. To have a reliable prediction, one may use a large feature vector which is the concatenation of the weather features from the previous days. However, instead of simply concatenating these feature vectors, this paper regards each previous day, or *delay*, as a separate view.

We will denote matrices as bold uppercase letters and vectors as bold lowercase letters. The superscript $^{[v]}$ will denote the vth view for the multi-view method.

2 Background: Multi-View LS-SVM Regression

This section summarizes the Multi-View LS-SVM Regression (MV LS-SVM) [10] model. This model is a multi-view extension of the well known Least Squares Support Vector Machine (LS-SVM) [16], where the primal formulation contains the summation of the objective functions corresponding to each view plus a coupling term. The coupling term takes into account the correlation between the different views.

Given a number of V views and training data $\{y_k, \mathbf{x}_k^{[v]}\}_{k=1}^N$ for $v = 1, \ldots, V$, where $\mathbf{x}_k^{[v]} \in \mathbb{R}^{d^{[v]}}$ denotes the k-th input sample and $y_k \in \mathbb{R}$ the k-th target value, the primal formulation of the MV LS-SVM model is stated as follows:

$$\min_{\substack{\mathbf{w}^{[v]}, \mathbf{e}^{[v]}, \\ b^{[v]}}} \frac{1}{2} \sum_{v=1}^{V} \mathbf{w}^{[v]^T} \mathbf{w}^{[v]} + \frac{1}{2} \sum_{v=1}^{V} \gamma^{[v]} \mathbf{e}^{[v]^T} \mathbf{e}^{[v]} + \rho \sum_{v,u=1; v \neq u}^{V} \mathbf{e}^{[v]^T} \mathbf{e}^{[u]} \tag{1}$$

$$\text{s.t. } \mathbf{y} = \boldsymbol{\Phi}^{[v]} \mathbf{w}^{[v]} + b^{[v]} \mathbf{1}_N + \mathbf{e}^{[v]} \quad \text{for } v = 1, \ldots, V$$

where $\mathbf{y} = [y_1; \ldots; y_N]$, $b^{[v]}$ are bias terms, $\gamma^{[v]}$ are positive real constants and $\mathbf{e}^{[v]} \in \mathbb{R}^N$ are error variables for each view v. The term $\rho \sum_{v,u=1;v\neq u}^{V} \mathbf{e}^{[v]^T} \mathbf{e}^{[u]}$ is defined as the coupling term and the regularization parameter $\rho > 0$ as the coupling parameter.

$\boldsymbol{\Phi}^{[v]} \in \mathbb{R}^{N \times d_h^{[v]}}$ is defined as $\boldsymbol{\Phi}^{[v]} = [\varphi^{[v]}(\mathbf{x}_1^{[v]})^T; \ldots; \varphi^{[v]}(\mathbf{x}_N^{[v]})^T]$ where $\varphi^{[v]} : \mathbb{R}^{d^{[v]}} \to \mathbb{R}^{d_h^{[v]}}$ are the feature maps, related to the vth view, which map the $d^{[v]}$-dimensional input to a high dimensional feature space. Since this feature space is high dimensional, and can even be infinite dimensional, the function $\varphi^{[v]}(\cdot)$ is usually not explicitly defined. Instead, the dual model [10, Eq.(10)] is derived where the function is implicitly defined trough the use of a positive kernel function $K^{[v]} : \mathbb{R}^{d^{[v]}} \times \mathbb{R}^{d^{[v]}} \to \mathbb{R}$ where $K^{[v]}(\mathbf{x}_i^{[v]}, \mathbf{x}_j^{[v]}) = \varphi^{[v]}(\mathbf{x}_i^{[v]})^T \varphi^{[v]}(\mathbf{x}_j^{[v]})$.

However, this primal formulation can be used by a Neural Network (NN) as a loss function to optimize the parameters. Hence, an NN could be used to solve the problem in the primal, as shown by the neural networks interpretation in primal and dual by Suykens et al. [16] and as was previously done for kernel methods e.g. by Zhong & Ghosh [21] for the SVM model [17].

3 Proposed method

In this section the Weighted Multi-view DNN model (Weighted MV-DNN) is introduced. The loss function of the proposed DNN is based on the primal formulation of MV LS-SVM (Eq.(1)) where the objectives of different views are weighted, as well as the coupling term.

The loss function optimized by the Weighted MV-DNN is stated as follows:

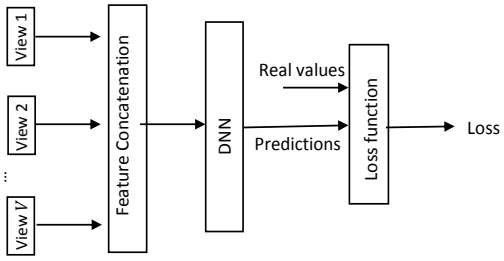
$$\min_{\mathbf{w}^{[v]}, \mathbf{e}^{[v]}} \quad \frac{1}{2} \sum_{v=1}^{V} s^{[v]} \left(\mathbf{w}^{[v]^T} \mathbf{w}^{[v]} + \gamma^{[v]} \mathbf{e}^{[v]^T} \mathbf{e}^{[v]} \right) + \sum_{v,u=1;v\neq u}^{V} \rho^{[v,u]} \sqrt{s^{[v]}} \sqrt{s^{[u]}} \ \mathbf{e}^{[v]^T} \mathbf{e}^{[u]} \tag{2}$$

where $\rho^{[v,u]}$ denotes the coupling parameter which can be different for each pairwise combination of views v and u and for which should hold that $0 \leq \rho^{[v,u]} \leq \min(\gamma^{[v]}, \gamma^{[u]})$. The reason for this upper bound on the coupling parameter is to ensure that the objective function does not converge to $-\infty$ for error variables $\mathbf{e}^{[v]}$ belonging to a certain view v, which can happen when $\rho^{[v,u]}$ is significantly larger than $\gamma^{[v]}$. Notice that $\mathbf{w}^{[v]}$ are the parameter vectors for the whole DNN.

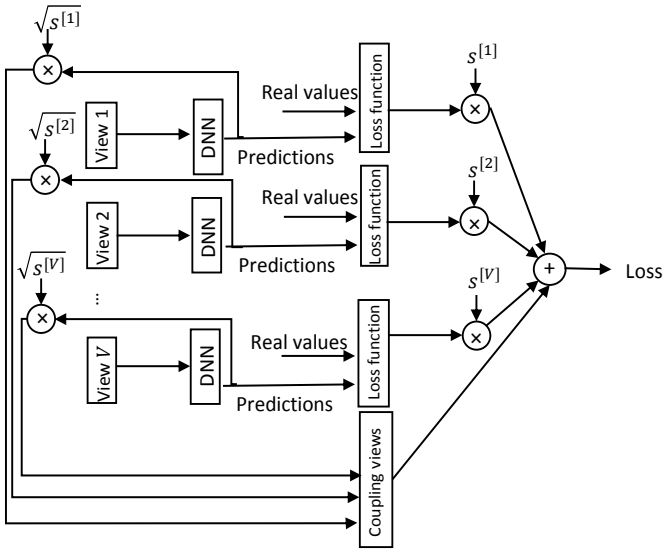
The proposed model is graphically represented in Figure 1 and compared to a graphical representation of the early fusion DNN approach, where the features from all views are simply concatenated. Notice that it is possible to use another model and learning mechanism instead of DNN like e.g. LS-SVM, Convolutional Neural Networks [11], Deep Belief Networks [8], Restricted Kernel Machines [15] and so on.

The weights $s^{[v]}$ for $v = 1, \ldots, V$ are added to control the influence of each view. These weights can be tailored to each specific application and can be manually determined by an expert, or calculated during a pre-processing step. Take for example the application of temperature prediction where the different

views represent different neighboring cities. In this example the weights could be determined by means of the similarity of different cities (views) to the target city, which could be calculated through a similarity function on the features of each city. Another example for temperature prediction is where each view represents a different weather variable (temperature, wind speed, humidity, etc.). In this example one might have some expert knowledge about which weather variables influence temperature more than others. In case this expert knowledge is not available another way to determine the weights is by looking at the performance of each individual view. In this way the views that have a weak performance can have a smaller weight than the views that perform well.



(a) Early fusion DNN



(b) Weighted Multi-view DNN

Fig. 1: General schemes of early fusion and the proposed method

The resulting regressor $\hat{y}(\cdot)$ for an unseen test point $\mathbf{x_t}$, with representations $\mathbf{x_t}^{[v]}$ for all views $v = 1, \ldots, V$, is defined as

$$\hat{y}(\mathbf{x_t}) = \frac{1}{V} \sum_{v=1}^{V} \beta^{[v]} \hat{y}^{[v]}(\mathbf{x_t}^{[v]}) \tag{3}$$

where $\hat{y}^{[v]}(\cdot)$ is the view-specific function estimation based on the obtained $\mathbf{w}^{[v]}$ and $\mathbf{e}^{[v]}$. This last prediction step thus includes another weighting, where the weighs can be equal to the weights from the training phase, i.e. $\beta^{[v]} = s^{[v]}$, or equal to 1 in order to obtain an unweighted averaged prediction.

4 Experiments

4.1 Weather data

In this paper the data are collected from the Weather Underground website [1] which is one of the popular ones in weather forecasting. The data include real measurements for weather elements such as minimum and maximum temperature, dew point and pressure from the beginning of 2007 until mid 2014 and for 5 cities including Brussels, Liege, Antwerp, Amsterdam and Eindhoven.



Fig. 2: Weather stations (from Google maps)

To assess the performance of the proposed method in different weather conditions, the experiments are conducted on two different test sets: one from mid-November 2013 until mid-December 2013 and the other one from mid-April 2014 to mid-May 2014. The prediction is done on a daily basis and for each test set, the training data includes daily weather variables of all of the five cities from the beginning of 2007 until the previous day of the test set. For each location there are 18 measured weather variables per day.

4.2 Model selection

As it was mentioned, in this study we consider each delay as a view. This was inspired by the fact that Recurrent Neural Networks (RNN) [6], which is a well-known approach in time-series prediction, splits the features based on the time delay and take into account each delay as separate input. In this study, the number of delays that has been taken into account is five; hence the number of views V is equal to five. By concatenation of the features (early fusion), the total number of features is equal to 450 (*number of views* $\times$ *number of cities* $\times$ *number of features per day per location*). Note that the number of features in each view is equal to 90 (*number of cities* $\times$ *number of features per day per location*).

To find a proper baseline, we evaluated the performance of LS-SVM, RNN and DNN with two hidden layers. For the experiments we use LSSVMlab[1] (in MATLAB) to deploy LS-SVM as a learning approach and TensorFlow[2] (in Python) to implement basic RNN [6] and DNN. The results in Tables 1 and 2 suggest that the DNN approach outperforms RNN and LS-SVM in most of the cases. Thus, we deploy DNN as our baseline approach. Note that for LS-SVM, the tuning hyperparameters are the regularization parameter and the RBF kernel bandwidth which are tuned using cross validation. In RNN and DNN, the tuning parameters are the number of neurons and the regularization parameters. To decrease the weighted MV-DNN tuning complexity, the number of neurons in each layer and the regularization parameter $\gamma^{[v]}$ for each view is tuned independently on each view. Afterwards, the views are coupled based on Eq.(2) and the $\rho^{[v,u]}$ values for each couple of views are tuned. Note that all parameters are tuned based on a validation set. The validation set for each test set is defined to include the data from last year prior to the corresponding test set. Moreover, to avoid local minima problem, we did the experiments five times. LS-SVM does not have the local minima problem; nevertheless, we did the experiments five times to tune the hyperparameters. The results are reported based on the median and the standard deviation of the Mean Absolute Error (MAE) on the test sets.

Table 1: Median MAE of the predictions in Weather LS-SVM, RNN and DNN with two hidden layers on Nov/Dec test set

Step ahead	Temp.	LS-SVM	RNN	DNN
1	Min	1.84±0.05	**1.71±0.01**	**1.71±0.02**
	Max	1.52±0.01	1.62±0.05	**1.37±0.07**
2	Min	1.84 ±0.02	1.85± 0.04	**1.81±0.04**
	Max	1.85± 0.01	**1.67± 0.1**	1.78±0.08
3	Min	2.23 ± 0.04	1.98±0.02	**1.85± 0.011**
	Max	2.03±0.007	2.06±0.2	**1.93±0.1**
4	Min	2.01±0.04	1.76±0.01	**1.70±0.01**
	Max	2.07 ± 0.009	1.99±0.03	**1.73±0.09**
5	Min	2.26±0.1	**1.84±0.008**	**1.84 ±0.08**
	Max	2.06± 0.01	2.17 ±0.3	**1.67± 0.09**
6	Min	2.18±0.05	**1.89± 0.02**	1.95± 0.02
	Max	2.02±0.01	1.93±0.3	**1.76±0.1**

In this study we define the weights of the views based on their performance on the validation set independently. Assuming $mse_{val}^{[v]}$ to be the Mean Squared Error of the view v on the validation set, the weight of this view is defined as

[1] `https://www.esat.kuleuven.be/sista/lssvmlab/`
[2] `www.tensorflow.org`

Table 2: Median MAE of the predictions in LS-SVM, RNN and DNN with two hidden layers on Apr/May test set

Step ahead	Temp.	LS-SVM	RNN	DNN
1	Min	1.63±0.007	1.67±0.04	**1.60±0.01**
	Max	**2.18±0.001**	2.25±0.07	2.37± 0.03
2	Min	**2.17 ±0.002**	2.59± 0.1	2.10±0.06
	Max	**2.40±0.008**	2.58±0.02	**2.40± 0.08**
3	Min	2.37 ± 0.003	2.24± 0.01	**2.10 ± 0.02**
	Max	2.50±0.01	**2.46±0.01**	2.54 ± 0.06
4	Min	2.59± 0.002	2.29± 0.01	**2.17±0.03**
	Max	3.00± 0.003	**2.52± 0.01**	2.67± 0.01
5	Min	2.87± 0.004	2.46±0.02	**2.33 ± 0.03**
	Max	2.94± 0.009	**2.51± 0.03**	2.70±0.06
6	Min	3.16± 0.003	2.77±0.02	**2.56± 0.06**
	Max	2.76±0.002	**2.71±0.02**	2.89±0.06

$\exp(-mse_{val}^{[v]})$. The weights are further rescaled so that $\sum_{v=1}^{V} s^{[v]} = V$. Thus, a view that performs well will have a higher corresponding weight. In Figure 3, the average weight values for different delays in one to six days ahead prediction are shown. It can be seen that for short term prediction smaller delays have higher impact on the prediction while for long term prediction the weights of the views are more similar.

In Tables 3 and 4 the performance of different multi-view methods are compared. The last two columns show the performance of the proposed method where *weighted average* refers to taking a weighted averaged prediction and *average* to taking an unweighted one (i.e. $\beta^{[v]} = s^{[v]}$ and $\beta^{[v]} = 1$, respectively, for all $v = 1, \ldots, V$ in Eq.(3)). The results yield that Weighted MV-DNN outperforms the unweighted version in most test cases. This suggests that considering weights for different delays can improve the temperature prediction performance. Unweighted MV-DNN refers to the proposed method with all weights equal to one.In Figure 4, the overall performance of Weather Underground, Early fusion DNN and MV-DNN on both test sets together are compared. The results reveal that the black-box methods are competitive with the state-of-the-art method used by Weather Underground. Moreover, it is shown that taking into account each delay as a view and deploying multi-view learning can improve the performance.

5 Conclusion

In this paper we proposed a DNN-based multi-view method which is based on the weighting of the view-specific objectives and the coupling term. These weights can be determined by different approaches. In this paper we defined each view to be the weather variables on a specific delay in the time series. The weights are determined based on the performance of each view on the validation set,

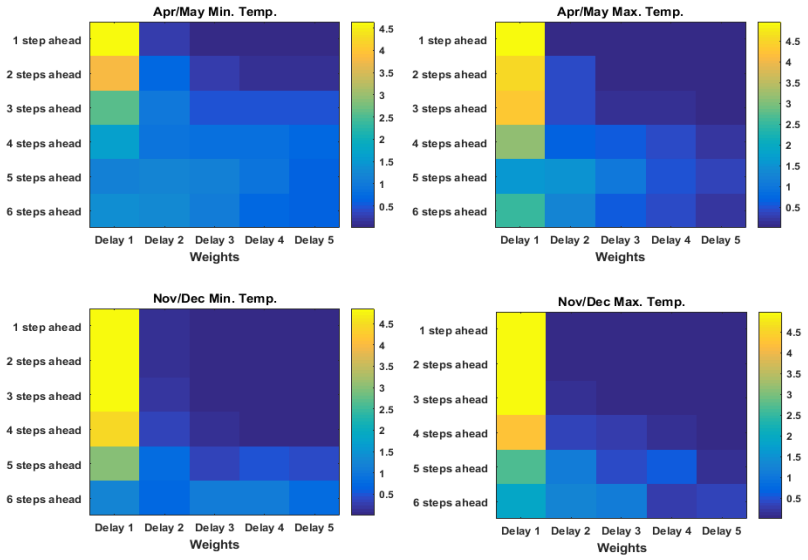


Fig. 3: Weights of different views (delays) for 1 to 6 days ahead in Nov/Dec and Apr/May data sets

Table 3: Median MAE of the predictions in Unweighted MV-DNN and Weighted MV-DNN with two hidden layers on Nov/Dec test set

Step ahead	Temp.	Unweighted MV-DNN	Weighted MV-DNN *(average)*	Weighted MV-DNN *(weighted average)*
1	Min	1.58±0.01	**1.51±0.01**	1.63 ±0.02
	Max	1.39±0.01	1.59±0.002	**1.35±0.02**
2	Min	**1.64±0.01**	1.69±0.002	1.80±0.02
	Max	**1.76±0.007**	2.00±0.01	1.89±0.02
3	Min	**1.78±0.005**	**1.78±0.003**	2.07±0.01
	Max	2.06±0.03	**1.97±0.01**	**1.97±0.006**
4	Min	**1.79±0.02**	1.84± 0.005	1.85±0.009
	Max	**1.80±0.06**	1.88±0.02	**1.80±0.03**
5	Min	1.90±0.02	**1.75±0.04**	1.84± 0.04
	Max	1.93±0.04	**1.77±0.01**	1.87±0.1
6	Min	2.18±0.01	**1.88±0.08**	1.90±0.06
	Max	1.94±0.008	**1.85±0.07**	1.90±0.05

Table 4: Median MAE of the predictions in Unweighted MV-DNN and Weighted MV-DNN with two hidden layers on Apr/May test set

Step ahead Temp.		Unweighted MV-DNN	Weighted MV-DNN *(average)*	Weighted MV-DNN *(weighted average)*
1	Min	1.89±0.002	2.01±0.0001	**1.67±0.01**
	Max	2.41±0.01	2.52±0.01	**2.11±0.009**
2	Min	2.22±0.004	2.17±0.03	**1.96±0.02**
	Max	2.53±0.02	2.49±0.03	**2.47±0.02**
3	Min	2.29±0.01	2.28±0.002	**2.06±0.01**
	Max	2.66±0.01	2.60±0.006	**2.42±0.0008**
4	Min	2.28±0.006	2.26±0.02	**2.17±0.01**
	Max	2.65±0.005	2.74±0.03	**2.56±0.01**
5	Min	**2.46±0.0006**	**2.46±0.0002**	2.49±0.004
	Max	2.78±0.01	2.83±0.03	**2.73±0.02**
6	Min	2.64±0.002	2.60±0.01	**2.59±0.002**
	Max	2.85 ±0.01	2.87±0.03	**2.75±0.03**

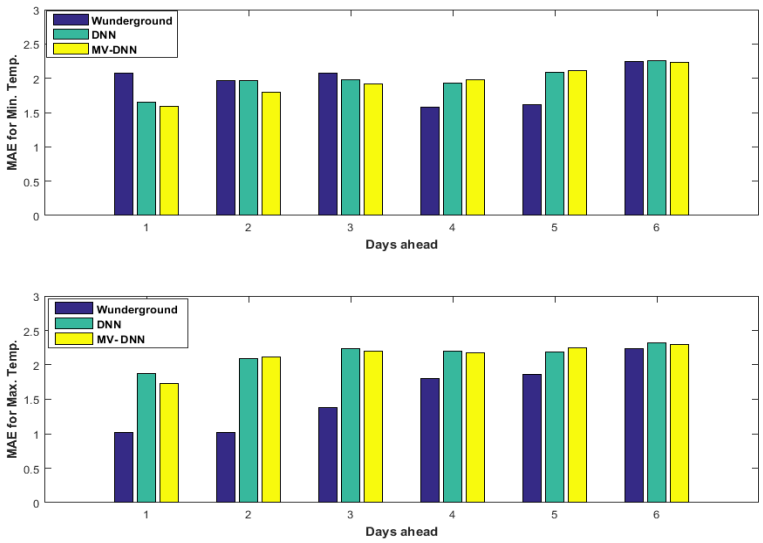


Fig. 4: Average MAE of the predictions for Weather Underground, DNN and MV-DNN on both test set

independently. The results on an application of temperature prediction show the improvement of the proposed MV-DNN method over an unweighted version as well as over the early fusion approach.

Acknowledgments. Research supported by Research Council KUL: CoE PFV/10/002 (OPTEC), PhD/Postdoc grants Flemish Government; FWO: projects: G0A4917N (Deep restricted kernel machines), G.088114N (Tensor based data similarity), ERC Advanced Grant E-DUALITY (787960).

References

1. www.wunderground.com, accessed: 2018-07-10
2. Andrew, G., Arora, R., Bilmes, J., Livescu, K.: Deep canonical correlation analysis. ICML pp. 1247–1255 (2013)
3. Bauer, P., Thorpe, A., Brunet, G.: The quiet revolution of numerical weather prediction. Nature **525**(7567), 47–55 (2015)
4. Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning **2**(1), 1–127 (Jan 2009). https://doi.org/10.1561/2200000006
5. Chaudhuri, K., Kakade, S.M., Livescu, K., Sridharan, K.: Multi-view clustering via canonical correlation analysis. ICML pp. 129–136 (2009)
6. Elman, J.L.: Finding structure in time. Cognitive science **14**(2), 179–211 (1990)
7. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science **313**, 504–507 (08 2006)
8. Hinton, G.E.: What kind of a graphical model is the brain? IJCAI pp. 1765–1775 (2005)
9. Hotelling, H.: Relations between two sets of variates. Biometrica **28**, 321–377 (1936)
10. Houthuys, L., Karevan, Z., Suykens, J.A.K.: Multi-view LS-SVM regression for black-box temperature prediction in weather forecasting. IJCNN pp. 1102–1108 (2017)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. of the IEEE **86**(11), 2278–2324 (Nov 1998). https://doi.org/10.1109/5.726791
12. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–44 (05 2015)
13. Liu, Y., Zheng, Y., Liang, Y., Liu, S., Rosenblum, D.S.: Urban water quality prediction based on multi-task multi-view learning. IJCAI pp. 2576–2582 (2016)
14. Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., Ng, A.Y.: Multimodal deep learning. ICML pp. 689–696 (2011)
15. Suykens, J.A.K.: Deep restricted kernel machines using conjugate feature duality. Neural Computation **29**(8), 2123–2163 (2017)
16. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific (2002)
17. Vapnik, V.: The nature of statistical learning theory. Springer-Verlag, New-York (1995)
18. Wang, W., Arora, R., Livescu, K., Bilmes, J.: On deep multi-view representation learning. ICML pp. 1083–1092 (2015)
19. Xu, C., Tao, D., Xu, C.: A survey on multi-view learning. eprint arXiv:1304.5634 (2013) (Apr 2013)
20. Zhao, J., Xie, X., Xu, X., Sun, S.: Multi-view Learning Overview: Recent Progress and New Challenges. Information Fusion **38**, 43–54 (2017). https://doi.org/10.1016/j.inffus.2017.02.007
21. Zhong, S., Ghosh, J.: Decision boundary focused neural network classifier. Intelligent Engineering Systems Through Articial Neural Networks (2000)
22. Zilca, R.D., Bistritz, Y.: Feature concatenation for speaker identification. EUSIPCO pp. 1–4 (Sept 2000)