ADMM Consensus for Deep LSTM Networks

Antonello Rosato, Federico Succetti, Marcello Barbirotta and Massimo Panella
Department of Information Engineering, Electronics and Telecommunications (DIET)
University of Rome “La Sapienza”, Via Eudossiana 18, 00184 Rome, Italy
Email: {antonello.rosato, massimo.panella}@uniroma1.it

Abstract—**In modern real-world applications, the need of using a decentralized data processing approach has progressively increased, facing complexity and handling issues. Pervasive data and ubiquitous computational capacity have enabled the proficient use of distributed implementation of machine learning algorithms, especially for forecasting problems. We provide in this paper a new, fully distributed prediction approach based on the Long Short-Term Memory deep neural network. When placed in a network of interconnected agents, the single predictors are able to improve the prediction accuracy by means of the Alternating Direction Method of Multipliers consensus procedure on some network parameters. Experimental tests on real-world time series prove the efficacy of the proposed approach, which regulates the information exchange in the network through high-level structures in the considered models.**

I. Introduction

In the past years, the most widespread feature in information and communication technology (ICT) applications is the pervasive demand of both data and computing power. The trends associated with this development emerged rapidly in several fields [1]–[5]. Consequently, new challenges arise regarding supervised learning, particularly concerning multiple data sources and distributed learning. Actually, in many applications it cannot be considered feasible or convenient to collect and process data at a single computing agent, mainly for privacy concerns, communication constraints, and so on. For this matter, the distributed learning paradigm has gained paramount importance in solving a plethora of problems, such as: feed-forward neural networks [6], [7]; multiple forecasting [8]; deep architectures [9]; Physical Hybrid Artificial Neural Network (PHANN) approaches [10]; more general hybrid models [11].

Despite these innovations, although there are some completely distributed learning algorithms, most of the data driven machine learning models adopted in the literature still lack of a fully decentralized the training approach, especially in deep learning, with consequent limits regarding their application to real-world decentralized contexts.

An important remark is that there are some applications for which the distributed paradigm not only solves the issues related to big data aggregation and processing, it can also improve the accuracy of the local model (i.e., a deep/shallow neural network) trained on its local data only [12], [13]. Although not the data itself, the local training agents in a connected network can exchange with their neighbors high-level information about data and the model itself, gaining advantage in the final generalization capability. A typical example where distributed learning can improve the accuracy of local models is time series forecasting. Actually, if each agent tries to share learning-related data statistics, data representation or, as in our case, the model parameters, this can help the local prediction performance by introducing global knowledge into the model.

A distributed machine learning paradigm usually relies on the same model to be deployed at the local agents. Regarding the reference model used for the single agents, it is well known that Deep Neural Networks (DNNs) are the state of the art and that, in particular, Long Short-Term Memory (LSTM) networks provide the best solution for the prediction problem [14], [15]. The use of heterogeneous models and asynchronous learning schedules are not considered in this paper, they could be investigated in future research works, as they stand in a brand new frontier.

In this paper, we propose a fully decentralized forecasting algorithm based on LSTM deep neural networks. The core concept of this new architecture is to train local models for each agent in the network, each using the local set of training data, then a global optimum is found sharing some network parameters through a distributed consensus approach. The latter is carried out making use of the Alternating Direction Method of Multipliers (ADMM) algorithm, which ensures good performance and a relatively simple implementation [16]. We tested our novel approach on real-world time series pertaining to the output power of four interconnected photovoltaic (PV) plants located in Colorado, USA. The obtained performances are compared with respect to the basic approaches considered so far: a theoretically optimal approach, although not really feasible in practice, where all data is collected and processed by a single (centralized) LSTM agent; the commonly adopted (local) approach where the LSTM of each agent is trained on its local data only, with no further consensus or data exchanges.

II. LSTM Prediction Model

LSTM networks are adopted herein as the reference forecasting model. They were firstly presented in [17] to address the issue regarding the persistence of information, introducing loops in the basic recurrent structure that, otherwise, tends to perform poorly when facing long-term dependencies in time series [18]. Regarding their performance, they have been employed with success in diverse machine learning and artificial intelligence fields, such as language modeling [19], machine translation [20], image captioning [21], hand writing generation [22], image generation [23], time series forecasting [14], and so forth.

We adopted in this work the deep model reported in Fig. 1, which is based on a standard architecture for dynamical prediction. It is composed by a first LSTM layer that implements the time series embedding so that its output, based on the vector sequence of the LSTM hidden states, will represent the evolution of the internal state of the unknown dynamical system producing the observed time series [24]; a second LSTM layer implementing the prediction model based on the said dynamical reconstruction; a fully connected dense layer for estimating the scalar value of the sample to be predicted based on the sequence of hidden states of the second LSTM layer.

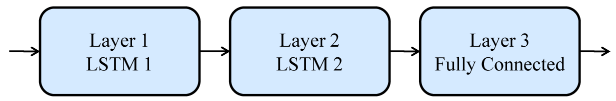


Fig. 1. Architecture of the adopted LSTM network.

In the following, we introduce a brief description of the LSTM model, in order to introduce the notation that will be used for distributed consensus as proposed successively in Sect. III.

Let $S[n]$, $n > 0$, be the scalar time series to be predicted at each time step n. The folded version of the LSTM layer is illustrated in Fig. 2; at time n, the input $\boldsymbol{x}_n \in \mathbb{R}^{N_i}$ is in general a column vector of N_i inputs, while the output is represented by the column vector $\boldsymbol{h}_n \in \mathbb{R}^{N_h}$ of 'hidden states', where each element is associated with one of the N_h hidden units of the LSTM layer. Each hidden unit is based on four well-known gate types and hence, the N_h quantities for each gate can be grouped into the following column vectors: input gate $\boldsymbol{i}_n \in \mathbb{R}^{N_h}$; forget gate $\boldsymbol{f}_n \in \mathbb{R}^{N_h}$; cell candidate $\boldsymbol{g}_n \in \mathbb{R}^{N_h}$; output gate $\boldsymbol{o}_n \in \mathbb{R}^{N_h}$:

$$\boldsymbol{i}_n = \sigma_g\left(\boldsymbol{W}_i\boldsymbol{x}_n + \boldsymbol{R}_i\boldsymbol{h}_{n-1} + \boldsymbol{b}_i\right), \tag{1}$$

$$\boldsymbol{f}_n = \sigma_g\left(\boldsymbol{W}_f\boldsymbol{x}_n + \boldsymbol{R}_f\boldsymbol{h}_{n-1} + \boldsymbol{b}_f\right), \tag{2}$$

$$\boldsymbol{g}_n = \sigma_c\left(\boldsymbol{W}_g\boldsymbol{x}_n + \boldsymbol{R}_g\boldsymbol{h}_{n-1} + \boldsymbol{b}_g\right), \tag{3}$$

$$\boldsymbol{o}_n = \sigma_g\left(\boldsymbol{W}_o\boldsymbol{x}_n + \boldsymbol{R}_o\boldsymbol{h}_{n-1} + \boldsymbol{b}_o\right). \tag{4}$$

where $\boldsymbol{W}_\gamma \in \mathbb{R}^{N_h \times N_i}$ is the matrix of input weights, $\boldsymbol{R}_\gamma \in \mathbb{R}^{N_h \times N_h}$ is the matrix of recurrent weights, while the biases are collected into the column vector $\boldsymbol{b}_\gamma \in \mathbb{R}^{N_h}$. The index γ denotes the specific gate type, with $\gamma \in \{i, f, g, o\}$. The activation functions $\sigma_g(\cdot)$ and $\sigma_c(\cdot)$ apply to the vector inputs entry-wise.

Let $\boldsymbol{c}_n$ be the column vector of 'cell states' of the LSTM layer at time n; the hidden states are obtained by:

$$\boldsymbol{c}_n = \boldsymbol{f}_n \odot \boldsymbol{c}_{n-1} + \boldsymbol{i}_n \odot \boldsymbol{g}_n\,, \tag{5}$$

and

$$\boldsymbol{h}_n = \boldsymbol{o}_n \odot \sigma_c(\boldsymbol{c}_n)\,, \tag{6}$$

where $\odot$ denotes the Hadamard product.

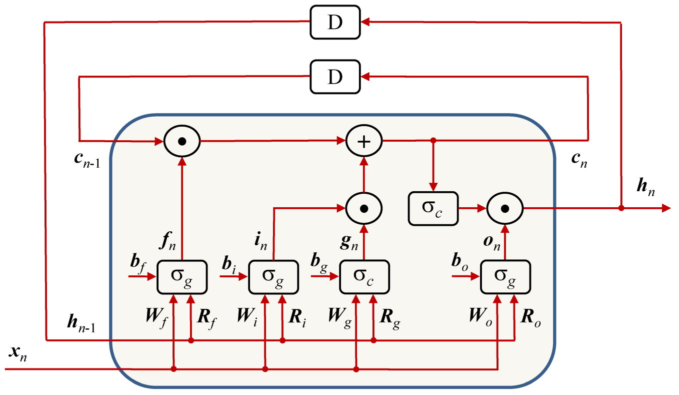


Fig. 2. Vectorized architecture of the LSTM layer at time n: each line represents a vector of multiple elements associated with N_h hidden units; blocks 'D' represent delays of the recurrent structure; at the output, the hidden states $\boldsymbol{h}_n$ will feed the next layer of the deep neural network.

At time n, the input to the first LSTM layer is the scalar sample of the sequence under analysis, i.e. $x_n^{(1)} = S[n]$, $N_i^{(1)} = 1$, where superscript '(1)' denotes the first layer while the index of the second layer will be omitted in the following in order to simplify notations. The hidden states $\boldsymbol{h}_n^{(1)}$ at the output of the first LSTM layer are used as input to the second LSTM layer, that is $\boldsymbol{x}_n = \boldsymbol{h}_n^{(1)}$. The final output of the network, which is the scalar sample y_n to be predicted, is obtained by projecting the hidden states $\boldsymbol{h}_n$ at the output of the second LSTM layer through the fully connected (dense) layer:

$$y_n = \boldsymbol{h}_n^t \boldsymbol{w}_d + b_d\,, \tag{7}$$

where $\boldsymbol{w}_d \in \mathbb{R}^{N_h}$ and b_d are the weights (column vector) and bias of the fully connected layer, respectively.

Using this forecasting architecture, the future values of the time series are predicted using the network output as $\widetilde{S}[n+q] = y_n$, where $\widetilde{S}[n+q]$ is the time series sample estimated at a prediction distance $q > 0$.

III. Definition of Distributed Network

Having defined the general structure of the LSTM-based DNN, which is used for prediction at each node, we can now formalize the distributed learning approach proposed in this paper. In the context of distributed forecasting, it is straightforward to imagine an operation mode where several agents are interconnected and operate with their own prediction system (an LSTM in this case). In our approach, the agents do not run independently nor they contribute to a global prediction, but they share some kind of information among the nodes in the network to improve the local prediction accuracy.

By discarding the centralization of all data, which results unfeasible in most operative conditions, we adopt a complete distributed approach which has been investigated in other fields, such as diffusion adaptation [25], sensor networks [26], multimedia signal processing [27], distributed optimization [16], [28], distributed databases [29], and others. The novelty lies in exchanging information about the inner parameters of the network via consensus algorithms, incorporating a global information at the local level [30].

If we consider a number L of agents in a network, each one representing a single LSTM network able to solve the prediction task on a local time series S_k, $k = 1 \dots L$, we can suppose that every kth agent sets its LSTM parameters, as defined in the previous section. Let, at a given time, the training algorithm running on each node k have determined a current estimation of the network parameters for both LSTM layers and the dense layer. It is possible to build a hidden matrix $\boldsymbol{H}_k \in \mathbb{R}^{n_p \times N_h}$ and an observation column vector $\boldsymbol{y}_k \in \mathbb{R}^{n_p}$:

$$\boldsymbol{H}_k = \begin{pmatrix} \boldsymbol{h}_{n_1}^t & 1 \\ \vdots & \vdots \\ \boldsymbol{h}_{n_p}^t & 1 \end{pmatrix} = \begin{pmatrix} h_{n_1,1} & \cdots & h_{n_1,N_h} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ h_{n_p,1} & \cdots & h_{n_p,N_h} & 1 \end{pmatrix}, \quad (8)$$

$$\boldsymbol{y}_k = \begin{pmatrix} y_{n_1} \\ \vdots \\ y_{n_p} \end{pmatrix}, \quad (9)$$

where $n_1, \dots, n_p$ are the time indexes of the p input-output samples of time series S_k in the local training set, $\boldsymbol{h}_{n_1}, \dots, \boldsymbol{h}_{n_p}$ are the related hidden states that feed the input of the dense layer (node index 'k' is omitted in these vectors for readability), $y_{n_1}, \dots, y_{n_p}$ are the consequent outputs of the dense layer (i.e., the estimated predictions).

The previous quantities, obtained at the end of training procedure, are linked through (7) that can be rewritten in this case as:

$$\boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{w}_k, \quad (10)$$

where (omitting node index 'k' also in this case for dense layer parameters $\boldsymbol{w}_d$ and b_d):

$$\boldsymbol{w}_k = \begin{pmatrix} \boldsymbol{w}_d \\ b_d \end{pmatrix} = \begin{pmatrix} w_{d,1} \\ \vdots \\ w_{d,N_h} \\ b_d \end{pmatrix}. \quad (11)$$

To regulate the exchange of information in the network and to ensure the reaching of the optimum for each parameter, a consensus strategy is put in place by means of a non-complete, connected, undirected network of interconnected agents, where a connectivity matrix $\boldsymbol{C}$ is defined. It is a square matrix of dimension L, where each element $C_{ij} \neq 0$ if and only if the two nodes i and j can exchange information. Therefore, we assume only local communication among neighboring nodes, that no pairs of nodes can exchange data between each other, and that no nodes can play a coordinating role.

IV. ADMM-based Distributed Learning of LSTM Networks

The task to be accomplished for all training agents in the network is to agree to a single parameter vector $\boldsymbol{w}^*$ of their fully connected layers, which can approximate the optimal solution represented by the following global formulation of a regularized least square problem (RLS):

$$\boldsymbol{w}^* = \underset{\boldsymbol{w} \in R^{N_h+1}}{\arg\min} \frac{1}{2} \left(\sum_{k=1}^{L} \| \boldsymbol{H}_k \boldsymbol{w} - \boldsymbol{y}_k \|_2^2 \right) + \frac{\lambda}{2} \| \boldsymbol{w} \|_2^2 . \quad (12)$$

In the proposed ADMM formulation, we consider local variables $\boldsymbol{w}_k$ for every node, forcing them to convergence to single quantity. The optimization problem can be reformulated as:

$$\boldsymbol{w}^*_{\text{ADMM}} = \underset{\boldsymbol{z}, \boldsymbol{w}_1, \dots, \boldsymbol{w}_L}{\text{minimize}} \frac{1}{2} \left(\sum_{k=1}^{L} \| \boldsymbol{H}_k \boldsymbol{w}_k - \boldsymbol{y}_k \|_2^2 \right) + \frac{\lambda}{2} \| \boldsymbol{z} \|_2^2$$
$$\text{subject to } \boldsymbol{w}_k = \boldsymbol{z} \text{ for } k = 1 \dots L, \quad (13)$$

where $\boldsymbol{H}_k$ and $\boldsymbol{y}_k$ are the hidden matrix and output vector computed for the local dataset S_k. This form is the classic RLS extended to all the L agents in the network, with $\boldsymbol{z}$ being the same as $\boldsymbol{w}_k$ to be optimized separately. Then, we construct the augmented Lagrangian as:

$$\mathcal{L} = \frac{1}{2} \left(\sum_{k=1}^{L} \| \boldsymbol{H}_k \boldsymbol{w}_k - \boldsymbol{y}_k \|_2^2 \right) + \frac{\lambda}{2} \| \boldsymbol{z} \|_2^2 + $$
$$+ \sum_{k=1}^{L} \boldsymbol{t}_k^t (\boldsymbol{w}_k - \boldsymbol{z}) + \frac{\gamma}{2} \sum_{k=1}^{L} \| \boldsymbol{w}_k - \boldsymbol{z} \|_2^2 , \quad (14)$$

where $\mathcal{L} = \mathcal{L}(\boldsymbol{z}, \boldsymbol{w}_1, \dots, \boldsymbol{w}_L, \boldsymbol{t}_1, \dots, \boldsymbol{t}_L)$, the vectors $\boldsymbol{t}_k$, $k = 1 \dots L$, are the Lagrange multipliers, $\gamma > 0$ is a penalty parameter, and the last term is introduced to ensure differentiability and convergence [16].

The ADMM proceeds iteratively, separately optimizing $\boldsymbol{w}_k$ and $\boldsymbol{z}$ at each step, updating the Lagrangian multipliers using a steepest-descent approach:

$$\boldsymbol{w}_k[m+1] = \underset{\boldsymbol{w}_k}{\arg\min}\, \mathcal{L}\big(\boldsymbol{z}[m], \boldsymbol{w}_1, \dots, \boldsymbol{w}_L, \boldsymbol{t}_1[m], \dots, \boldsymbol{t}_L[m]\big), \quad (15)$$

$$\boldsymbol{z}[m+1] = \underset{\boldsymbol{z}}{\arg\min}\, \mathcal{L}\big(\boldsymbol{z}, \boldsymbol{w}_1[m+1], \dots, \boldsymbol{w}_L[m+1], \boldsymbol{t}_1[m], \dots, \boldsymbol{t}_L[m]\big), \quad (16)$$

$$\boldsymbol{t}_k[m+1] = \boldsymbol{t}_k[m] + \gamma \left(\boldsymbol{w}_k[m+1] - \boldsymbol{z}[m+1] \right), \quad (17)$$

where m is an internal iteration index of ADMM procedure.

In this case, the updates for $\boldsymbol{w}_k[m+1]$ and $\boldsymbol{z}[m+1]$ can be computed in closed form:

$$\boldsymbol{w}_k[m+1] = \left(\boldsymbol{H}_k^t \boldsymbol{H}_k + \gamma \boldsymbol{I} \right)^{-1} \left(\boldsymbol{H}_k^t \boldsymbol{y}_k - \boldsymbol{t}_k[m] + \gamma \boldsymbol{z}[m] \right), \quad (18)$$

$$\boldsymbol{z}[m+1] = \frac{\gamma \hat{\boldsymbol{w}} + \hat{\boldsymbol{t}}}{\lambda / L + \gamma} . \quad (19)$$

The parameters $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{t}}$ composing $\boldsymbol{z}$ in (19) can be computed in a decentralized manner using the Distributed Average Consensus (DAC) procedure, which is based on the connectivity matrix $\boldsymbol{C}$ previously defined that is associated with the specific network topology [31].

ALGORITHM I
PSEUDOCODE OF THE ADMM CONSENSUS OF FULLY CONNECTED LAYERS

```
Input: Number of nodes $L$ (global), regularization factor $\lambda$
    (global), $\gamma$ (global), maximum number of iterations $M$
    (global), training set $S_k$ (local), $k = 1 \dots L$.
Output: Optimal vector $\boldsymbol{w}_k^*$.
 1: Compute $\boldsymbol{H}_k$ and $\boldsymbol{y}_k$ from $S_k$ and the training algorithm
    of the LSTM-based DNN.
 2: Initialize $\boldsymbol{t}_k[0] = \boldsymbol{0}$, $\boldsymbol{z}[0] = \boldsymbol{0}$.
 3: for $m$ from 0 to $M$ do
 4:     Compute $\boldsymbol{w}_k[m+1]$ according to (15).
 5:     Compute averages $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{t}}$ with DAC consensus with
        other nodes over the network.
 6:     Compute $\boldsymbol{z}[m+1]$ according to (16).
 7:     Update $\boldsymbol{t}_k[m]$ according to (17).
 8:     Check termination with residuals.
 9: end for
10: return $\boldsymbol{w}_k^* = \boldsymbol{z}[m+1]$, $k = 1 \dots L$.
```

The terminating residual procedure employs the computation of the 'primal residual' $\boldsymbol{r}_k[m]$ and the 'dual residual' $\boldsymbol{s}[m]$:

$$\boldsymbol{r}_k[m] = \boldsymbol{w}_k[m] - \boldsymbol{z}[m] \,, \tag{20}$$

$$\boldsymbol{s}[m] = -\gamma(\boldsymbol{z}[m] - \boldsymbol{z}[m-1]) \,. \tag{21}$$

A possible stopping criterion is that both residuals should be less (in norm) than two thresholds:

$$\|\boldsymbol{r}_k[m]\|_2 < \epsilon_{\text{primal}}^{(k)} \,, \tag{22}$$

$$\|\boldsymbol{s}[m]\|_2 < \epsilon_{\text{dual}} \,. \tag{23}$$

choosing the thresholds as in [16]:

$$\epsilon_{\text{primal}}^{(k)} = \sqrt{L}\,\epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{\|\boldsymbol{w}_k[m]\|_2, \|\boldsymbol{z}[m]\|_2\right\} \,, \tag{24}$$

$$\epsilon_{\text{dual}} = \sqrt{L}\,\epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max_k \left\{\|\boldsymbol{t}_k[m]\|_2\right\} \,, \tag{25}$$

where ϵ_{abs} and ϵ_{rel} are user-specified absolute and relative tolerances, respectively. The algorithm can also be stopped when a maximum number T of iterations is reached. The pseudocode for the described algorithm is reported in Algorithm I.

The proposed ADMM consensus strategy, based on a generalized RLS solution, relies on a model that must be linear in the parameters. Actually, this is not the case of a DNN as the one in Fig. 1, which is made of several LSTM layers. To this end, the whole training algorithm proposed in this paper is based on an alternating procedure, in which the ADAM algorithm [32] is adopted for training of every DNN. The proposed method can be applied in the same way for any other algorithm adopted for deep learning.

The whole training algorithm is based on the alternating runs of the DNN training algorithm (ADAM), for N_{loc} iterations, and the said ADMM consensus procedure (until it converges). Let t be the whole counter of ADAM iterations during training and $\boldsymbol{w}_k[t]$, $k = 1 \dots L$, be the estimated parameters of the fully connected layer at the generic iteration t. After the first run of ADAM algorithm, i.e. at $t = N_{\text{loc}}$, the ADMM procedure is launched to reach consensus on $\boldsymbol{w}_k^* = \boldsymbol{z}$ over all agents of the network, by using as initial values for the consensus procedure in Algorithm I the hidden and observation matrices obtained by the DNN training algorithm at $t = N_{\text{loc}}$. Then, the ADAM algorithm re-starts locally for N_{loc} iterations, which means that the whole algorithm will run from iteration $t = N_{\text{loc}} + 1$ up to $t = 2N_{\text{loc}}$. The initial parameters at $t = N_{\text{loc}} + 1$ in each LSTM layer are kept the same as obtained at $t = N_{\text{loc}}$, while the initial parameters used for the fully connected layer will be the $\boldsymbol{w}_k^*$ obtained at the end of the ADMM consensus step. The whole procedure iterates $N_{\max}$ times (i.e., $N_{\max}N_{\text{loc}}$ iterations of ADAM algorithm) or when convergence is reached comparing the averages of two successive ADMM steps. The details on all the implementation steps are described in the pseudocode reported in Algorithm II.

V. EXPERIMENTAL RESULTS

To assess the performance of the proposed algorithm, we applied it to a real-world case study. We considered the irradiance taken at four interconnected solar plants (nominal power of 100 kWp) located in Colorado, USA. The sequences are collected every hour and retrieved via the Measurement and Instrumentation Data Center (MIDC) database. Additional information about the geography of the considered plants can be drawn from Fig. 3; the actual coordinates can be found in Table I.

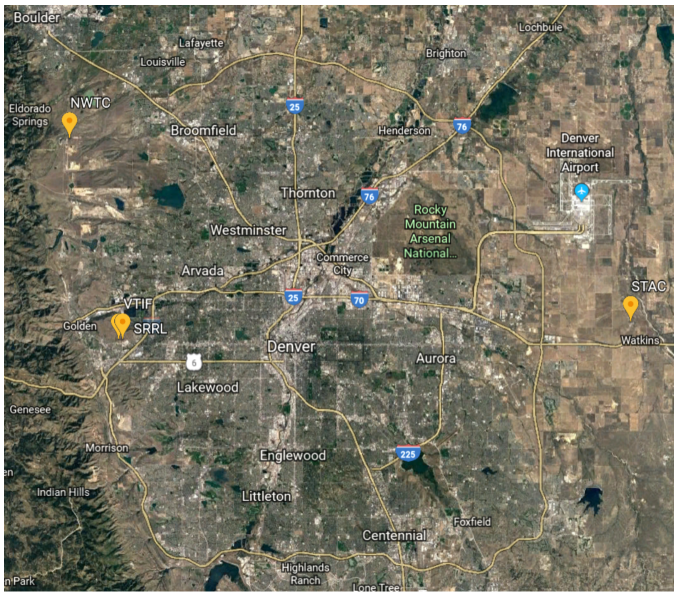


Fig. 3. Map showing the locations of the considered PV plants (courtesy of Google Maps™).

Each time series considered is the output power measured from one of the PV plants, i.e. their hourly production curves.

ALGORITHM II
PSEUDOCODE OF THE DISTRIBUTED LSTM TRAINING ALGORITHM WITH ADMM-BASED CONSENSUS

```
Input: Given a network of $L$ agents, where each agent has its own training set associated with the local time series. Given numerical hyperparameters of LSTM layers (hidden units, etc.), DNN training algorithm (learning rate, $L_2$ regularization factor, etc.), ADMM procedure (maximum iterations, $\gamma$, $\lambda$, etc.). Given $N_{\max}$, $N_{\text{loc}}$, and convergence thresholds.
    Initialize, at iteration $t = 0$, the parameters of LSTM layers and $\boldsymbol{w}_k[0]$, $k = 1 \dots L$, for fully connected layers.
1: for $r = 1$ to $N_{\max}$ do
2:     Set. The DNN training algorithm (ADAM) will run on every local DNN by using as initial parameters for LSTM layers the ones obtained at the end of iteration $t_0 = (r-1)N_{\text{loc}}$, while $\boldsymbol{w}_k[t_0]$ will be used as initial parameters for fully connected layers.
3:     Update. Run the network training algorithm in every local node for $N_{\text{loc}}$ iterations, from $t = (r-1)N_{\text{loc}} + 1$ to $t = rN_{\text{loc}}$.
4:     Consensus step. Run the ADMM procedure as in Algorithm I and compute the optimal estimation $\boldsymbol{w}_k^*[r]$. The ADMM algorithm is initialized by using the $\boldsymbol{H}_k$ and $\boldsymbol{y}_k$ obtained at the end of iteration $t = rN_{\text{loc}}$ of the DNN training algorithm.
5:     Update. Set the parameters in the fully connected layer of each agent, in order to initialize the next run of the DNN training algorithm: $\boldsymbol{w}_k[t] \leftarrow \boldsymbol{w}_k^*[r]$, $k = 1 \dots L$, where $t = rN_{\text{loc}}$ .
6:     Convergence step. Stop if any convergence criterion is satisfied between $\boldsymbol{w}_k^*[r]$ and $\boldsymbol{w}_k^*[r-1]$.
7: end for
8: return as the final DNN on each node the one associated with the parameters of the latest iteration $t = rN_{\text{loc}}$.
```

TABLE I
LIST OF PV PLANTS WITH GEOGRAPHIC COORDINATES

Plant Name	Latitude	Longitude	Elevation [m]
SRRL	$39°44'31.2''$ N	$105°10'48.0''$ W	1828
VTIF	$36°44'31.6''$ N	$105°10'32.6''$ W	1793
NWTC	$39°54'38.2''$ N	$105°14'04.9''$ W	1855
STAC	$39°45'24.7''$ N	$104°37'12.9''$ W	1674

A linear mapping is used for normalization between -1 and 1, using the physical operation parameters for the extremes of the normalization interval and hence, -1 will correspond to 0 kW and +1 to 100 kW. The problem of zero solar irradiation is handled by considering the geographical data for each plant and computing the sunrise and sunset times [33]. Thus, the predicted values are forced to zero output during the night period.

The tests are therefore carried out on a network of $L = 4$ agents, each corresponding to a single plant. The data communication network among plants is chosen randomly with a 75% degree of connectivity (i.e., each node may be linked to another one with a 0.75 probability), reflecting the realistic capabilities of the network infrastructure. The resulting graph, where each plant is a node, is therefore connected but not complete, an example is illustrated in Fig. 4.

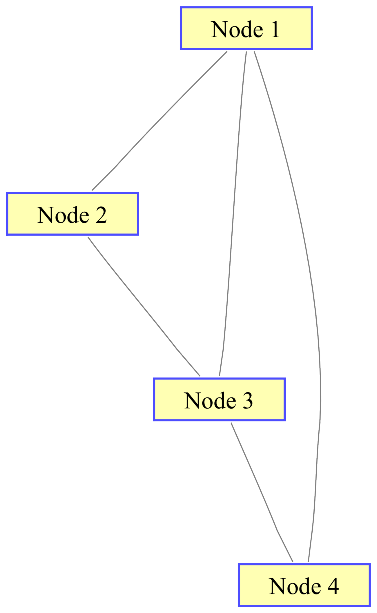


Fig. 4. An example of adopted topology with 75% degree of connectivity.

We compared three different algorithms:

- Centralized LSTM (C-LSTM): this option simulates the case where all data is gathered at a single location and the straightforward prediction of network in Fig. 1 is applied by using the ADAM training algorithm. It should be observed that the C-LSTM has only a theoretical benchmark purpose, since it is unfeasible from a practical point of view, requiring the transmission of all the data collected in the peripheral PV sites to a central location.
- Local LSTM (L-LSTM): this is the case where data is indeed distributed but there is no communication in the network so every agent trains a single DNN from its local time series. This corresponds to predictions made in each plant independently of each other.
- Distributed LSTM (D-LSTM): This is the proposed network trained by the ADMM-based distributed algorithm described in Sect. IV.

The parameters of ADAM training algorithm in C-LSTM and L-LSTM are set to the following default values: initial learning rate 0.01, with a 75% reduction every 20 iterations; maximum number of iterations 100; gradient decay factor 0.9; squared gradient decay factor 0.999. In D-LSTM, the ADAM algorithm is used with the same default parameters but $N_{\max} = 5$, $N_{\text{loc}} = 20$, and the convergence threshold is set to 0.1% or relative variation from $\boldsymbol{w}_k^*[r-1]$ and $\boldsymbol{w}_k^*[r]$.

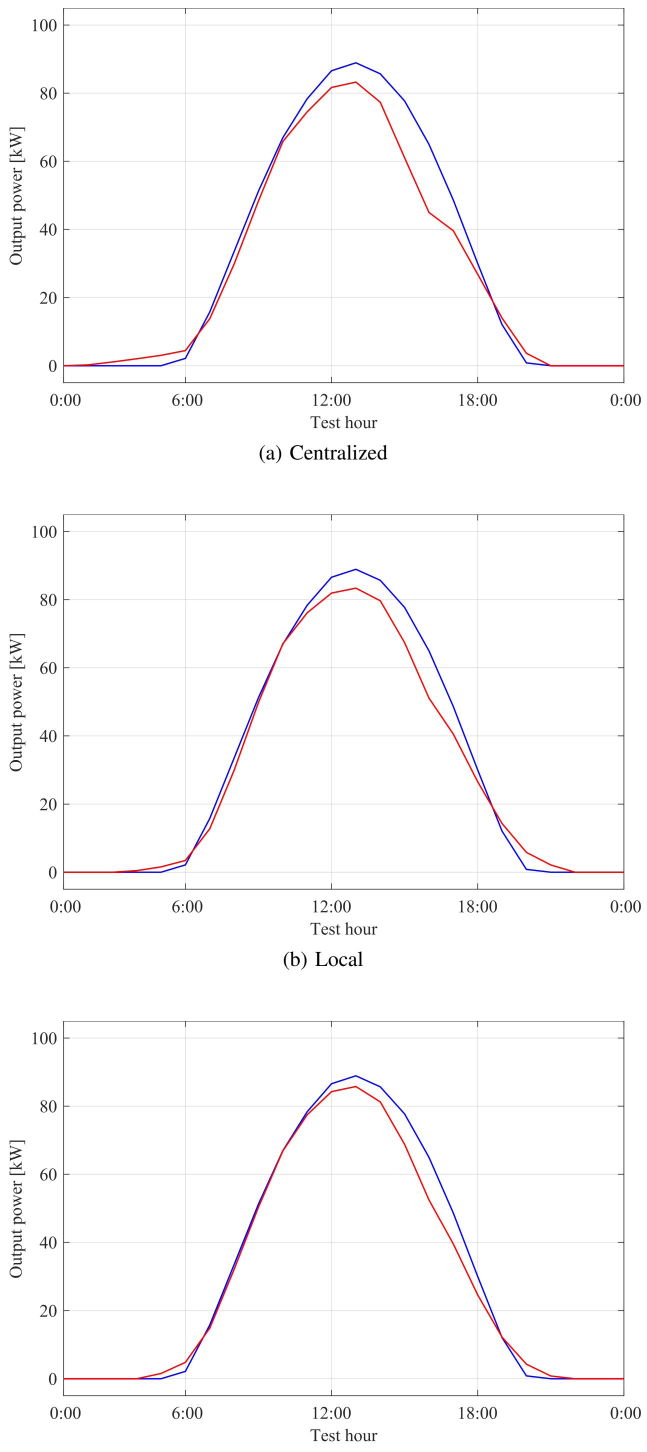
The parameters of ADMM procedure are set to $\gamma = 1$, $\epsilon_{\text{abs}} = 10^{-6}$, $\epsilon_{\text{rel}} = 10^{-6}$, maximum iterations $M = 300$. The regularization factor λ of ADMM is set equal to the L_2 regularization factor of ADAM algorithm, which is discussed in the following.

In order to avoid overfitting, a grid search procedure [34] is adopted for all algorithms by using data in the training set only (i.e., known samples of the time series only), so as to set in advance the main hyperparameters that are number of hidden units N_h in both LSTM layers and L_2 regularization factor λ in ADAM algorithm. The same values are adopted for all L nodes. We preliminary tested different values of such parameters in the following ranges: $N_h = \{10, 15, 20, 25, 30, \ldots, 100\}$ for both layers; $\lambda = 2^j$, $j = \{-12, \ldots, -1, 0, 1, \ldots, 12\}$. The final adopted values are $N_h = 50$ in the first LSTM layer, $N_h = 15$ in the second LSTM layer, $\lambda = 2^{-11}$.

We considered a training set of 30 consecutive days (i.e., one month or 720 samples) for every experiment; the training set contains the known samples that are used to forecast the future ones. The latter are associated with a test set whose length is one day (i.e., 24 samples) after the last available sample of the training set. The tests are relative to six days of 2018, which were chosen for showing a set of days with variable weather conditions. Namely, the test set starts in the mid of February, April, June, August, October, and December 2018 and it is composed by the 24 samples of the 15th day of the month. The deep network predicts one sample at time and hence, in order to use the available samples also for testing the entire day ahead, the prediction distance is set accordingly to $q = 24$.

For the numerical evaluation of performances, we used the common Root-Mean-Square error (RMSE) measure on the predicted samples of the test set. Every network is trained on a same dataset considering 10 different runs, which are carried out after a different (random) initialization of the layer parameters and of the network topology as well (D-LSTM only). On each run, once the network has been trained by using one of the considered algorithms, it will be used to forecast the time series of each plant; the total error over all plants will be also computed and reported. All RMSE values are reported in terms of mean and standard deviation over the 10 training runs. All of the experiments described in the following were carried on using Matlab™ R2019a on a machine equipped with Intel® Core™ i7-3770K 64-bit CPU at 3.50 GHz, 32 GB RAM, and NVIDA GTX 680 GPU.

The efficacy and overall soundness of the present work can be assessed by analyzing the improvement of the proposed ADMM-based distributed learning scheme with respect to the centralized and local solutions. The RMSE results for the three algorithms applied to test sets are reported in Table II. The global performance of D-LSTM is extensively better than the classical L-LSTM, with a gain in terms of reduced RMSE up to 16% in some days of the year, while this improvement achieves even 35% in some plants. It is important to outline that the performance of D-LSTM on each plant is also equivalent, and sometimes even better, to C-LSTM that is only a theoretical baseline as it cannot be realized in practice. The best accuracy is obtained in February and December, whereas models show coherent behavior for every

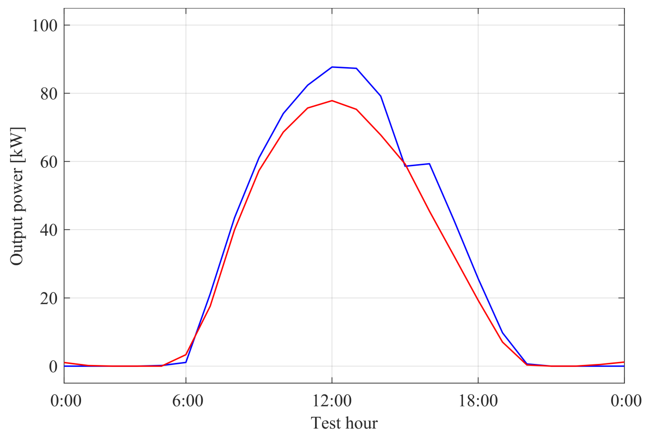


(a) Centralized

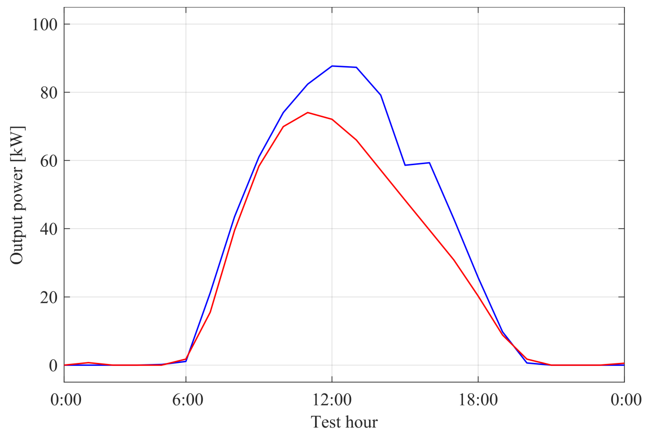
(b) Local

(c) Distributed

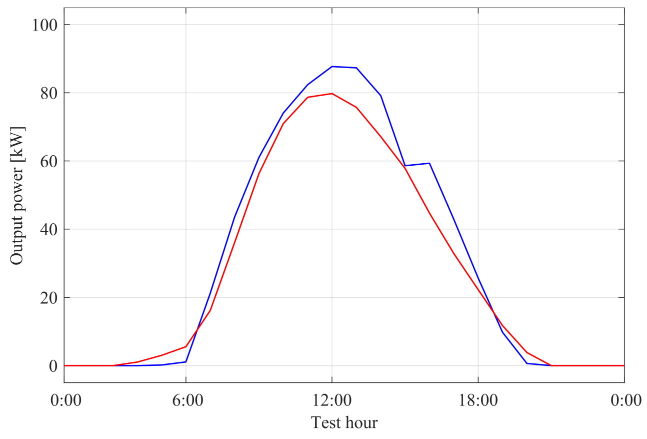
Fig. 5. Predicted (red) and real (blue) value of output power at Plant 2 in the mid of June 2018, by applying C-LSTM (a), L-LSTM (b), and D-LSTM (c).



(a) Centralized



(b) Local



(c) Distributed

Fig. 6. Predicted (red) and real (blue) value of output power at Plant 3 in the mid of June 2018, by applying C-LSTM (a), L-LSTM (b), and D-LSTM (c).

other considered test day. A visual examination of the three predictions in the month of June, which has an intermediate performance among others, is reported in Fig. 5 and Fig. 6 for Plant 2 and Plant 3, respectively, conveying the same conclusions.

VI. Conclusion

Considering the ubiquitous presence of large data and its processing, distributed approaches based on neural network processing are highly needed. In most real-world cases, the aggregation of data in a single centralized unit is not feasible for cost, computational complexity and privacy constraints. In this regard, purely distributed versions of state of the art machine learning models are hard to find. We presented here a distributed method for enforcing a global information extraction in a network of agents, relying on the LSTM for the local model and the ADMM consensus algorithm. Tests were carried out on a prediction problem with several energy-related time series, stemming from different PV plants in a network. Performance clearly show a better forecasting accuracy for the proposed algorithm, which have been compared with the local version of the LSTM and the centralized one. Future directions of this work can focus on developing a more complex framework for energy-related prediction, including prices, load and meteorological information. The whole prediction system, based on deep learning techniques, could be applied in the smart grid world, implementing a new multivariate scheme encompassing all the variables in play for a better forecasting and demand-side energy management.

References

[1] J. A. P. Lopes, N. Hatziargyriou, J. Mutale, P. Djapic, and N. Jenkins, "Integrating distributed generation into electric power systems: A review of drivers, challenges and opportunities," *Electr. Power Syst. Res.*, vol. 77, no. 9, pp. 1189–1203, 2007.

[2] A. Khamis, H. Shareef, E. Bizkevelci, and T. Khatib, "A review of islanding detection techniques for renewable distributed generation systems," *Renewable Sustainable Energy Rev.*, vol. 28, pp. 483–493, 2013.

[3] F. Chen, D. Liu, and X. Xiong, "Research on stochastic optimal operation strategy of active distribution network considering intermittent energy," *Energies*, vol. 10, no. 4, pp. 1996–1073, 2017.

[4] X. Han, S. Liao, X. Ai, W. Yao, and J. Wen, "Determining the minimal power capacity of energy storage to accommodate renewable generation," *Energies*, vol. 10, no. 4, pp. 1996–1073, 2017.

[5] A. Rosato, R. Altilio, R. Araneo, and M. Panella, "Prediction in photovoltaic power by neural network," *Energies*, vol. 10, no. 7, 2017.

[6] C. Voyant, G. Notton, S. Kalogirou, M.-L. Nivet, C. Paoli, F. Motte, and A. Fouilloy, "Machine learning methods for solar radiation forecasting: A review," *Renewable Energy*, vol. 105, pp. 569 – 582, 2017.

[7] A. K. Yadav and S. S. Chandel, "Solar radiation prediction using Artificial Neural Network techniques: A review," *Renewable Sustainable Energy Rev.*, vol. 33, pp. 772 – 781, 2014.

[8] L. Hernández, C. Baladrón, J. M. Aguiar, B. Carro, A. Sánchez-Esguevillas, and J. Lloret, "Artificial neural networks for short-term load forecasting in microgrids environment," *Energy*, vol. 75, pp. 252–264, 2014.

[9] A. Gensler, J. Henze, B. Sick, and N. Raabe, "Deep learning for solar power forecasting – An approach using AutoEncoder and LSTM neural networks," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 002 858–002 865.

[10] E. Ogliari, A. Dolara, G. Manzolini, and S. Leva, "Physical and hybrid methods comparison for the day ahead PV output power forecast," *Renewable Energy*, vol. 113, pp. 11 – 21, 2017.

[11] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F. J. M. de Pison, and F. Antonanzas-Torres, "Review of photovoltaic power forecasting," *Solar Energy*, vol. 136, pp. 78–111, 2016.

[12] S. Scardapane, R. Fierimonte, P. D. Lorenzo, M. Panella, and A. Uncini, "Distributed semi-supervised support vector machines," *Neural Networks*, vol. 80, pp. 43 – 52, 2016.

TABLE II
RMSE OF EACH PLANT AND AVERAGE RESULT FOR DIFFERENT ALGORITHMS

Month	Algorithm	Plant 1	Plant 2	Plant 3	Plant 4	Total
February	C-LSTM	6.121 ± 0.070	5.495 ± 0.079	5.984 ± 0.038	6.323 ± 0.064	5.981 ± 0.037
	L-LSTM	9.401 ± 0.086	6.484 ± 0.143	6.522 ± 0.011	8.238 ± 0.034	7.661 ± 0.031
	D-LSTM	8.308 ± 0.047	3.478 ± 0.028	6.411 ± 0.066	8.215 ± 0.073	6.603 ± 0.009
April	C-LSTM	7.964 ± 0.052	9.080 ± 0.025	10.437 ± 0.036	9.604 ± 0.029	9.271 ± 0.022
	L-LSTM	9.602 ± 0.009	5.016 ± 0.035	10.496 ± 0.079	9.573 ± 0.047	8.672 ± 0.020
	D-LSTM	8.459 ± 0.091	5.071 ± 0.006	10.067 ± 0.021	9.422 ± 0.032	8.255 ± 0.016
June	C-LSTM	7.576 ± 0.029	8.682 ± 0.007	5.860 ± 0.009	5.439 ± 0.009	6.889 ± 0.008
	L-LSTM	8.547 ± 0.051	6.087 ± 0.003	9.130 ± 0.007	7.220 ± 0.024	7.746 ± 0.005
	D-LSTM	7.947 ± 0.029	5.593 ± 0.067	8.468 ± 0.063	6.262 ± 0.017	7.068 ± 0.008
August	C-LSTM	6.480 ± 0.057	6.798 ± 0.017	9.173 ± 0.027	8.045 ± 0.059	7.624 ± 0.036
	L-LSTM	9.414 ± 0.014	7.055 ± 0.036	11.841 ± 0.056	9.741 ± 0.070	9.513 ± 0.009
	D-LSTM	9.864 ± 0.110	7.483 ± 0.055	11.967 ± 0.067	8.328 ± 0.035	9.410 ± 0.062
October	C-LSTM	7.979 ± 0.020	15.234 ± 0.077	16.140 ± 0.013	10.702 ± 0.015	12.514 ± 0.009
	L-LSTM	7.707 ± 0.020	17.423 ± 0.097	17.281 ± 0.024	11.334 ± 0.013	13.436 ± 0.023
	D-LSTM	4.821 ± 0.028	15.479 ± 0.138	16.182 ± 0.065	8.125 ± 0.084	11.152 ± 0.009
December	C-LSTM	8.115 ± 0.192	6.924 ± 0.158	4.735 ± 0.099	8.257 ± 0.156	7.008 ± 0.148
	L-LSTM	8.826 ± 0.170	7.714 ± 0.042	3.751 ± 0.042	8.027 ± 0.173	7.079 ± 0.036
	D-LSTM	7.268 ± 0.051	7.205 ± 0.053	4.023 ± 0.015	8.412 ± 0.104	6.727 ± 0.017

[13] A. Rosato, M. Panella, and R. Araneo, “A distributed algorithm for the cooperative prediction of power production in PV plants,” *IEEE Trans. Energy Convers.*, pp. 1–8, 2018.

[14] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[15] C. Persson, P. Bacher, T. Shiga, and H. Madsen, “Multi-site solar power forecasting using gradient boosted regression trees,” *Solar Energy*, vol. 150, pp. 423 – 436, 2017.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[19] M. Sundermeyer, R. Schlüter, and H. Ney, “Lstm neural networks for language modeling,” in *Thirteenth annual conference of the international speech communication association*, 2012.

[20] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[21] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[22] A. Graves, “Generating sequences with recurrent neural networks,” *arXiv preprint arXiv:1308.0850*, 2013.

[23] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, “Draw: A recurrent neural network for image generation,” *arXiv preprint arXiv:1502.04623*, 2015.

[24] H. D. I. Abarbanel, *Analysis of Observed Chaotic Data.* Springer, New York, 1996.

[25] A. H. Sayed, “Adaptive networks,” *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[26] J. B. Predd, S. R. Kulkarni, and H. V. Poor, “Distributed learning in wireless sensor networks,” *IEEE Signal Processing Magazine*, pp. 56–69, 2007.

[27] S. Scardapane, R. Fierimonte, D. Wang, M. Panella, and A. Uncini, “Distributed music classification using random vector functional-link nets,” in *Proc. of International Joint Conference on Neural Networks (IJCNN 2015)*, Killarney, Ireland, 2015, pp. 1–8.

[28] K. Slavakis, G. Giannakis, and G. Mateos, “Modeling and optimization for big data analytics: (statistical) learning tools for our era of data deluge,” *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 18–31, 2014.

[29] A. Lazarevic and Z. Obradovic, “The distributed boosting algorithm,” in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 311–316.

[30] S. Scardapane, D. Wang, M. Panella, and A. Uncini, “Distributed learning for random vector functional-link networks,” *Information Sciences*, vol. 301, pp. 271–284, 2015.

[31] L. Xiao, S. Boyd, and S.-J. Kim, “Distributed average consensus with least-mean-square deviation,” *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33 – 46, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731506001808

[32] D. P. Kingma and J. L. Ba, “ADAM: A method for stochastic optimization,” *arXiv*, 2014.

[33] J. A. Duffie and W. A. Beckman, *Solar engineering of thermal processes*, 3rd ed. New York: Wiley, 2006.

[34] P. Lerman, “Fitting segmented regression models by grid search,” *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 29, no. 1, pp. 77–84, 1980.