Wind power prediction using mixture density recurrent neural networks

Conference Paper · April 2010

CITATIONS
9

READS
1,018

3 authors, including:

Martin D. Felder
Zentrum für Sonnenenergie und Wasserstoff-Forschung Baden-Württemberg
24 PUBLICATIONS **302** CITATIONS
SEE PROFILE

Anton Kaifel
Zentrum für Sonnenenergie und Wasserstoff-Forschung Baden-Württemberg
29 PUBLICATIONS **270** CITATIONS
SEE PROFILE

WIND POWER PREDICTION USING MIXTURE DENSITY RECURRENT NEURAL NETWORKS

Martin Felder[1], Anton Kaifel[1] and Alex Graves[2]

[1] *Zentrum für Sonnenenergie- und Wasserstoff-Forschung (ZSW), 70565 Stuttgart, Germany, Email: martin.felder@zsw-bw.de*

[2] *Technische Universität München (TUM), 85748 Garching b. M., Germany*

ABSTRACT

Machine learning techniques have proven effective at forecasting the power output of wind turbine generators. However predictions typically use a single input vector of NWP forecasts, disregarding the potentially informative history of previous inputs. Moreover prediction uncertainty is often provided only when NWP ensembles are available. We address these shortcomings by using mixture density recurrent neural networks to forecast a time-dependent probability distribution over power outputs. Using historical wind farm data, we demonstrate the viability of our approach for power prediction up to 48 h, and provide a comparison with multilayer perceptrons and baseline predictors.

Key words: wind energy prediction; recurrent neural networks.

1. INTRODUCTION

The forecasting of wind power from wind turbine generators (WTGs) and its upscaling to wind farms and regions has drawn considerable attention in recent years, not only in classical engineering sciences, but also in the fields of data mining and machine learning. This is due to the discovery that a collection of generic pattern recognition techniques, when applied to numerical weather prediction (NWP) forecasts, can compete with or even outperform the best physical models available for the task. The subsequent development of combined forecast systems using both physical models and machine learning was a logical conclusion. For those systems, it was shown that performance is usually directly related to the diversity of a) NWP models and b) forecasting techniques [7]. While standard techniques like multi-layer perceptron (MLP) type neural networks, support vector machines and a variety of time series analysis methods routinely make it into such systems, research on recurrent neural networks (RNNs) has been relatively sparse [1, 9], despite being particularly suitable for the joint analysis of multiple time series.

One reason for this may be that being dynamical systems, RNNs are inherently more difficult to control than stationary, feed-forward methods [2]. On the other hand, their behaviour being radically different from both static methods and traditional time series analysis makes them orthogonal to existing combinations of methods. In this paper, we strive to introduce the very successful class of long short-term memory RNNs [5] into wind power applications.

Another issue we aim to address is the estimation of uncertainties for our predictions. Traditionally, neural networks produce spot forecasts, and merely provide a global RMSE value calculated from a suitable validation data set. This may be acceptable for statistical method comparisons, but doesn't help the decision maker when looking at a particular forecasted power time series. Recently, a lot of work has been done on using ensembles of NWP forecasts to define a possibly multi-modal power probability density function (PDF) by means of combining power predictions from the ensemble. For an overview, we refer to several papers in the forecasting session of these proceedings. However, using ensembles leads to a major in-

crease in data processing logistics and possibly cost. Our approach utilizes a mixture density output layer instead, as explained in more detail below. With this, the RNN learns from the training data set the reliability of its predictions together with the most likely power value.

2. METHODS

2.1. Recurrent Neural Networks

Neural networks (NNs) are said to be recurrent if they contain feedback connections from neurons to themselves. The NNs' behaviour thus becomes time dependent, and under some preconditions they have been theoretically proven to be able to approximate mappings between time series to arbitrary accuracy. Here, we map a series of NWP forecasts onto a series of WTG power outputs, using an three layer RNN with feedback connections between the cells in the hidden layer.

We employ Long Short-Term Memory (LSTM) RNNs that have been shown to be superior to generic RNNs and Echo-State RNN in many experiments [5, 6, 4, 12]. The hidden layer of these networks consists of composite cells (Figure 1) designed to preserve the networks state and error information over many time steps.

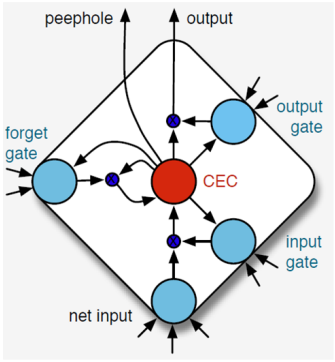

Figure 1. Long Short-Term Memory memory cell schematic. CEC = constant error carousel.

2.2. Mixture density models

A quite general way of modelling continuous target distributions with RNNs is to use the network outputs to represent the parameters of a mixture of paramterized PDF functions for every target variable, resulting in a *mixture density recurrent neural network* [13]. In contrast to conventional RNNs, this is a heteroskedastic model, meaning it allows the error variance to depend on the input.

A mixture density output layer hence consists on one neuron per parameter of the PDFs used, plus one neuron per mixing coefficient (weight) of the PDF function. Figure 2 shows the situation for 1-dimensional outputs using three Gaussian PDFs to represent the target probabilities.

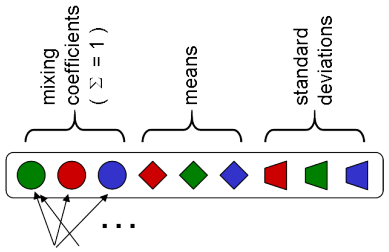

Figure 2. Example mixture density output layer using three Gaussian PDFs as components.

We will briefly derive the required equations simplified to a 1-dimensional output, which in our case will be the predicted wind power z. First we assume that $\mathbf{x}$ and z are sequences of length T, and that the $z(t)$, are conditionally independent given the K-dimensional input vector $\mathbf{x}$:

$$p(z|\mathbf{x}) = \prod_{t=1}^{T} p(z(t)|\mathbf{x}) \tag{1}$$

For any mixture distribution $p(z(t)|\mathbf{x})$ with M components we can write

$$p(z(t)|\mathbf{x}) = \sum_{m=1}^{M} p(m|\mathbf{x}, t)p(z(t)|\mathbf{x}, m) \tag{2}$$

where the probabilities $p(m|\mathbf{x}, t)$ of the mixture components given $\mathbf{x}$ and t (but prior to z) are usually referred to as the *mixing coefficients*. In the particular case where the components are 1-dimensional Gaussians, we have

$$p(z(t)|\mathbf{x}, m) = \mathcal{N}(z(t)|\mu_m(\mathbf{x}, t), \sigma_m^2(\mathbf{x}, t)) \tag{3}$$

where the scalars $\mu_m(\mathbf{x}, t)$ and $\sigma_m^2(\mathbf{x}, t)$ are respectively the means and variances of the m^{th}

Gaussian component and $\mathcal{N}(a|\mu,\sigma^2)$ is defined as the probability density at point a of Gaussian distribution with mean μ and covariance σ:

$$\mathcal{N}(a|\mu,\sigma^2) \stackrel{\text{def}}{=} (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right) \quad (4)$$

In what follows we show how the mixing coefficients, means and variances can all be modelled by a recurrent network with a suitable choice of output activation functions, following the derivation in [2].

A total of $3M$ output units is required to represent the complete distribution: M *coefficient* units whose inputs are denoted $a(t,\pi_m)$ and whose activations are denoted $y(t,\pi_m)$, where $y(t,\pi_m) = p(m|\mathbf{x},t)$, M *mean* units with inputs $a(t,\mu_m)$ and activations $y(t,\mu_m) = \mu_m(\mathbf{x},t)$, and M *standard deviation* units with inputs $a(t,\sigma_m)$ and activations $y(t,\sigma_m) = \sigma_m(\mathbf{x},t)$.

Since the mixing coefficients define a discrete probability distribution over the components they must satisfy

$$\sum_{m=1}^{M} p(m|\mathbf{x},t) = 1, \qquad 0 \leq p(m|\mathbf{x},t) \leq 1 \quad (5)$$

A suitable activation function for modelling such distributions is the softmax function [2], which we now apply to the coefficient units:

$$y(t,\pi_m) = \frac{\exp\left(a(t,\pi_m)\right)}{\sum_{m'=1}^{M} \exp\left(a(t,\pi_{m'})\right)} \quad (6)$$

The standard deviations must satisfy $\sigma_m(\mathbf{x},t) \geq 0$, and the corresponding outputs can therefore be scaled by the exponential function.

$$y(t,\sigma_m) = \exp\left(a(t,\sigma_m)\right) \quad (7)$$

One reason to favour the exponential function over other positive-valued functions is that an activation of $a(t,\sigma_m) = -\infty$ is required to create a pathological zero-variance distribution.

The means on the other hand are unconstrained and can be represented by the identity function:

$$y(t,\mu_m) = a(t,\mu_m) \quad (8)$$

Combining (1), (2) and (3) into a suitable network error function, we get

$$E(\mathbf{x},z) = -\sum_{t=1}^{T} \ln\left(\sum_{m=1}^{M} p(m|\mathbf{x},t) \mathcal{N}(z(t)|\mu_m(\mathbf{x},t),\sigma_m^2(\mathbf{x},t))\right) \quad (9)$$

Expanding out the Gaussian terms, dropping additive constants and substituting in the network outputs we get

$$E(\mathbf{x},z) = -\sum_{t=1}^{T} \ln\left(\sum_{m=1}^{M} \frac{y(t,\pi_m)}{y(t,\sigma_m)} \exp\left(-\frac{(y(t,\mu_m)-z(t))^2}{2y^2(t,\sigma_m)}\right)\right) \quad (10)$$

As usual the next step is to determine the derivatives of $E(\mathbf{x},z)$ with respect to the output activations. Before we begin however, it is convenient to introduce the posterior probabilities $p(m|\mathbf{x},t,z(t))$, often referred to as *responsibilities*, resulting from the mixing component prior $p(m|\mathbf{x},t)$ after $z(t)$ is presented. For the sake of brevity we will make the identification $\gamma_m \stackrel{\text{def}}{=} p(m|\mathbf{x},t,z(t))$. The rules of probability tell us that

$$\gamma_m = \frac{p(m|\mathbf{x},t)p(z(t)|\mathbf{x},m)}{p(z(t)|\mathbf{x})} \quad (11)$$

Differentiating (10) with respect to each of the output units and rearranging in terms of γ_m gives

$$\frac{\partial E(\mathbf{x},z)}{\partial y(t,\pi_m)} = -\frac{\gamma_m}{y(t,\pi_m)} \quad (12)$$

$$\frac{\partial E(\mathbf{x},z)}{\partial y(t,\mu_m)} = \gamma_m\left(\frac{\mu_m(\mathbf{x},t)-z(t)}{y^2(t,\sigma_m)}\right) \quad (13)$$

$$\frac{\partial E(\mathbf{x},z)}{\partial y(t,\sigma_m)} = \frac{\gamma_m}{y(t,\sigma_m)}\left(1-\frac{(\mu_m(\mathbf{x},t)-z(t))^2}{y^2(t,\sigma_m)}\right) \quad (14)$$

It can now be shown with some more rearranging that

$$\frac{\partial E(\mathbf{x},z)}{\partial a(t,\pi_m)} = y(t,\pi_m) - \gamma_m. \quad (15)$$

For the outputs representing the means the identity function was used and it follows that

$$\frac{\partial E(\mathbf{x},z)}{\partial a(t,\mu_m)} = \gamma_m\left(\frac{z(t)-y(t,\mu_m)}{y^2(t,\sigma_m)}\right), \quad (16)$$

while from (7) we have

$$\frac{\partial y(t,\sigma_m)}{\partial a(t,\sigma_m)} = y(t,\sigma_m), \quad (17)$$

and therefore

$$\frac{\partial E(\mathbf{x},z)}{\partial a(t,\sigma_m)} = \gamma_m\left(1-\frac{(z(t)-y(t,\mu_m))^2}{y^2(t,\sigma_m)}\right) \quad (18)$$

These derivatives can now be fed into the usual error backpropagation type neural network training algorithmns found in any introductory textbook on the subject.

In the wind power prediction context, let us remark that while NWP wind prediction errors are generally assumed to be Gaussian, they are transformed by local effects and the power curve. So far we do not take special account of this, meaning we have to interpret probablilitiy mass at negative powers as belonging to power zero, and similarly at the upper end of the possible power output range.

3. DATA

3.1. NWP predictions

We use historical open access data from the GFS-4 model [3] to provide our method with training and validation data. The model version used has a horizontal resolution of 0.5° and provides analysis fields at 00z, 06z, 12z, and 18z. Forecasts from these analyses are calculated up to a 104 h horizon every 3 h. Most of the fields have been interpolated to 26 standard pressure levels.

Diverse forecasted variables from the four grid points closest to the target wind farm were selected as predictors, since at the given NWP model resolution there is no reason to assume additional information about the wind field onsite can be gained from grid points further away. This has also been shown statistically for a similar set-up by Kusiak et al. [8]. After some sensitivity studies we also found that using merely the 10 m winds provided by the model, together with the last known power reading were sufficient.

3.2. Wind farm data

A historical data set of 10-minute average wind power readings, spanning July 2007 to December 2008, has kindly been made available to us by the company Natenco. The data were recorded at a wind farm in northern Germany, which was only anonymously identified by its post code, 17495. This corresponds to an approximate location of (51.5°N, 13.5°E). The wind park consists of 16 WTGs with nameplate rated power (NRP) of 1.5 MW each. These raw data were pre-processed by removing outliers and negative values, and averaging the remainder over all WTGs in 1 h intervals. These hourly power values directly serve as our target quantity, since other groups have shown that with neural network methods, there does not seem to be a particular benefit associated with trying to predict wind speed first, and subsequently convert it to power readings.

4. MODEL SETUP

The recurrent neural network used for generating wind power predictions was set up to receive heterogeneous 72 h sequences of data in the following way, similar to [1], as shown in Figure 3: At each time T for which we want to produce 48 h forecasts, we start by feeding it the last 24 h of hourly averaged power measurements, i.e. for $T-24$ h, $T-23$ h, ..., T. In parallel, selected variables from the four grid points of the corresponding GFS-4 analysis are input. The analyses are available only every 6 h, therefore the same values are reiterated for six successive time steps. To simplify input data assembly, we restrict ourselves to T also being an analysis time step, although this restriction could be relaxed in a later operational setting. For instance, the network receives the data from the analysis at time $T-6$ h during time steps $T-6$ h, $T-5$ h, ..., $T-1$ h. This initial 24 h period serves as a settling phase, initializing the internal activations of the RNN. Operation of the model is shown schematically in Figure 4.

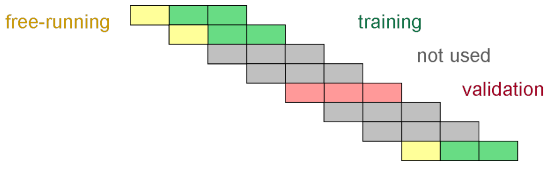

Figure 3. Presentation of training and validation sequences to the RNN. Each box corresponds to 24 h.

The RNN is trained using backpropagation through time with the RProp algorithm [11]. For the evaluation and comparison of model performance, we follow the recommendations given by Madsen et al. [10]. Apart from the usual per-

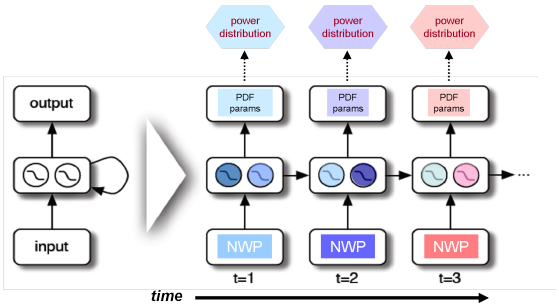

Figure 4. Left: Schematic of the RNN employed. Only our hidden layer employs recurrence. On the right, the network is "unrolled" in time, showing the data flow.

sistence forecast

$$P(T + \delta t) = P(T) \tag{19}$$

they describe an improved reference model

$$P(T + \Delta t) = \alpha_{\Delta t} P(T) + (1 - \alpha_{\Delta t}) \bar{P}(T), \tag{20}$$

which takes into account the correlation coefficient $\alpha_{\Delta t}$ between the power output at times T and $T+\Delta t$ and the average power measured up to time T, $\bar{P}(T)$. The correlation coefficients are calculated from the entire training data. Therefore this model already uses a lot more information than simple persistence, although in a straightforward way.

As a third reference, we trained a standard MLP network on the same input parameters as the RNN uses in the forecasting phase. The special way in which the training and test data for the RNN are staggered leads to a sizeable loss of training data, a disadvantage the MLP does not have. Therefore we used all available data here, split randomly into 80% training and 20% test data. The MLP is also trained using RProp.

5. FIRST RESULTS

After the first iterations of improving our training data and finding the best metaparameters for the new method, we have already achieved prediction accuracy on par with a feed-forward (MLP) network. To make this a fair comparison, we took great care to optimize the MLP. Several hundred experiments were run, varying the selection of input data, the training algorithm and the hidden layer size. The system seems very robust with regard to most combinations, with the nRMSE staying within 1–2% of the best combination found. Figure 5 shows a comparison plot. For the sake of correctness, we should note that due to the different nature of the two methods, we could not use exactly the same validation data to calculate the numbers.

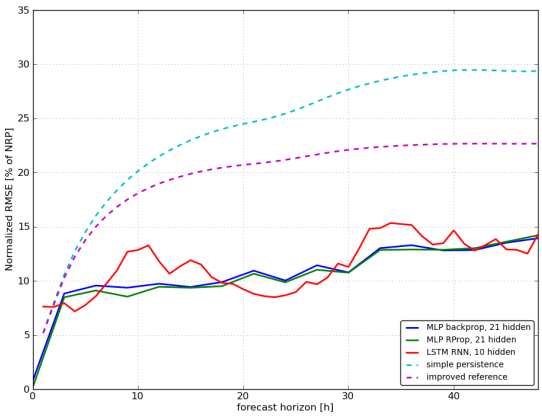

Figure 5. Normalized RMSE comparison for feed-forward networks (MLP) and LSTM RNNs with persistence. The curves were averaged over 50 experiments per model setup.

Interestingly, the RNN seems to make more use of the correlation between the last known power measurement and the values at about the same time of day 24 and 48 hours later, leading to improved performance at the corresponding horizons. We are currently investigating this effect for further exploitation.

When looking at some examples of predicted sequences shown in Figure 6, we note that in many cases the mixture PDF resembles a single Gaussian, as is expected considering the wind errors of NWP analysis are also mostly Gaussian distributed. At the boundaries of the valid power range, secondary peaks appear. These are probably due to machinery related effects, reflecting a more sophisticated implicit power curve model than just using a smooth functional relationship between wind and power.

Note also the smooth change in the PDFs over time, which is often lost when training several MLPs to perform the same task (not shown).

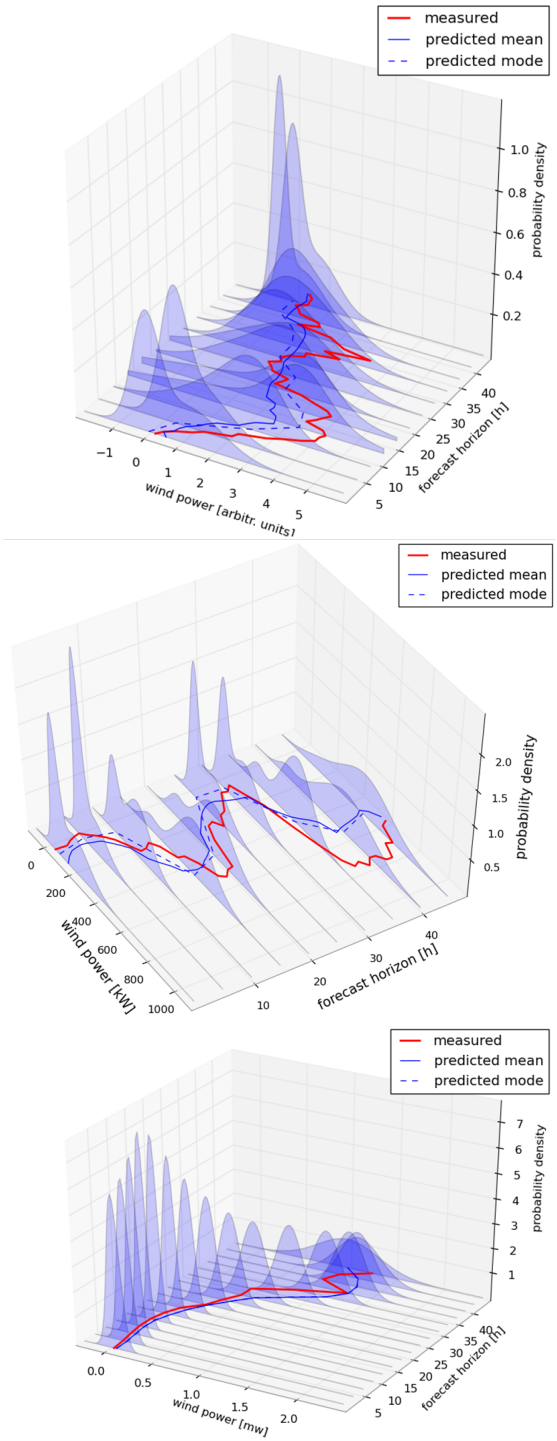

Figure 6. Examples of hourly RNN wind power prediction sequences from the validation set. For these plots, the PDF was modelled by mixing two Gaussians.

6. CONCLUSION AND OUTLOOK

We have shown a simple RNN system that not only exhibits novel behaviour not seen in the reference models, but also constitutes a powerful prediction method of its own right. While these promising results already hint at the great potential of the model, optimization work is expected to continue for some time, especially regarding the way of presentation and the length of the input sequences, of which there are many more possible combinations than for a static model.

Our data base of wind power measurements is growing right now, such that we will be able to apply this technique on additional wind farms and standalone WPGs. This is important especially when faced with a dynamical system as an RNN, in order to better understand its performance and find the right tuning parameters. In addition, we have recently acquired historical forecast data from the COSMO-EU model featuring a 7 km grid spacing. With the introduction of longer sequences and more detailed forecast data we think LTSM can fully play out its strength especially for short term forecasts (nowcasting), where NWP data play a minor role.

REFERENCES

[1] T.G. Barbounis, J.B. Theocharis, M.C. Alexiadis, and P.S. Dokopoulos. Long-term wind speed and power forecasting using local recurrent neural network models. *Energy Conversion, IEEE Transactions on*, 21(1):273–284, 2006. ISSN 0885-8969. doi: 10.1109/TEC.2005.847954.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Berlin, New York, 2006.

[3] Global Climate and Environmental Modeling Center Weather Modeling Branch. The GFS atmospheric model. Technical Report Office Note 442, NCEP, NOAA, Environmental Modelling Center, Camp Springs, MD, USA, 2003.

[4] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18:602–610, 2005.

[5] S. Hochreiter and J. Schmidhuber. Long Short-Term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[6] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

[7] Jeremie Juban, Lionel Fugon, and George Kariniotakis. Uncertainty estimation of wind power forecasts – comparison of probabilistic modelling approaches. 2008.

[8] Andrew Kusiak, Haiyang Zheng, and Zhe Song. Wind farm power prediction: a data-mining approach. *Wind Energy*, 12(3): 275–293, 2009. doi: 10.1002/we.295.

[9] Ma Lei, Luan Shiyan, Jiang Chuanwen, Liu Hongling, and Zhang Yan. A review on the forecasting of wind speed and generated power. *Renewable and Sustainable Energy Reviews*, 13(4):915–920, May 2009. ISSN 1364-0321. doi: 10.1016/j.rser.2008.02.002.

[10] Henrik Madsen, Pierre Pinson, George Kariniotakis, Henrik Aa. Nielsen, and Torben S. Nielsen. Standardizing the performance evaluation of ShortTerm wind power prediction models. *Wind Engineering*, 29:475–489, December 2005. doi: 10.1260/030952405776234599.

[11] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proc. of the ICNN*, pages 586–591, San Francisco, 1993. IEEE.

[12] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez. Training recurrent networks by EVOLINO. *Neural Computation*, 19(3): 757–779, 2007.

[13] Mike Schuster. *On supervised learning from sequential data with applications for speech recognition*. PhD thesis, Nara Institute of Science and Technology, 1999.