Hindawi
Complexity
Volume 2018, Article ID 9327536, 10 pages
https://doi.org/10.1155/2018/9327536

Research Article

Using Deep Learning to Predict Complex Systems: A Case Study in Wind Farm Generation

J. M. Torres and R. M. Aguilar

Department of Computer and Systems Engineering, Universidad de La Laguna, La Laguna 38200 Tenerife, Spain

Correspondence should be addressed to R. M. Aguilar; raguilar@ull.edu.es

Received 30 November 2017; Accepted 25 February 2018; Published 3 April 2018

Academic Editor: José Manuel Andújar

Making every component of an electrical system work in unison is being made more challenging by the increasing number of renewable energies used, the electrical output of which is difficult to determine beforehand. In Spain, the daily electricity market opens with a 12-hour lead time, where the supply and demand expected for the following 24 hours are presented. When estimating the generation, energy sources like nuclear are highly stable, while peaking power plants can be run as necessary. Renewable energies, however, which should eventually replace peakers insofar as possible, are reliant on meteorological conditions. In this paper we propose using different deep-learning techniques and architectures to solve the problem of predicting wind generation in order to participate in the daily market, by making predictions 12 and 36 hours in advance. We develop and compare various estimators based on feedforward, convolutional, and recurrent neural networks. These estimators were trained and validated with data from a wind farm located on the island of Tenerife. We show that the best candidates for each type are more precise than the reference estimator and the polynomial regression currently used at the wind farm. We also conduct a sensitivity analysis to determine which estimator type is most robust to perturbations. An analysis of our findings shows that the most accurate and robust estimators are those based on feedforward neural networks with a SELU activation function and convolutional neural networks.

1. Introduction

A region's electricity grid consists of a series of components that have to work together to achieve a balance between generation and demand, while at the same time ensuring the security of the electricity supply and providing a certain level of quality and service. The structure of the power system may be divided into four key activities: generation, transport, distribution, and marketing. The start of the electricity supply process takes place at power plants, where the electricity is generated. Depending on the type of facility, different types of primary energy sources are used to drive a turbine or motor, thus converting the primary energy into mechanical energy. The turbine is connected to a generator, which turns the mechanical energy into electrical energy. The process of supplying electricity continues via the transport network, which links the various production plants to consumption centres. This process takes place at high voltages to lower the currents and thus the losses. The distribution process comes next, in which the electricity is relayed from substations to the transport network for the various consumption points. These substations reduce the voltage from that of the transport network to values that are suitable for use by consumers. The electricity supply process concludes with the marketing activity, in which the electricity is sold to consumers based on their contracted power.

In Spain, Law 54/1997 went into effect in 1998. This law is notable because, as in the rest of Europe, it deregulated the generation and marketing activities, while continuing to regulate the transport and distribution aspects. Ever since, two primary operators have been charged with managing the technical and economic aspects of Spain's electricity market. One is Red Eléctrica de España (REE) and the other is Operador del Mercado Ibérico de Energía (OMIE), which are required to coordinate their efforts. The latter is charged with handling the bids for selling and buying energy. With this power system model, the price of energy became defined by the matching processes that started to be used in the various market sessions: daily, intraday, and ancillary services.

Table 1: Hours of operation for intraday market sessions.

	1st session	2nd session	3rd session	4th session	5th session	6th session
Session opens	17:00	21:00	01:00	04:00	08:00	12:00
Session closes	18:45	21:45	01:45	04:45	08:45	12:45
Matching	19:30	22:30	02:30	05:30	09:30	13:30
Reception of scheduling disaggregations	19:50	22:50	02:50	05:50	09:50	13:50
Release PHF	20:45	23:45	03:45	06:45	10:45	14:45
Scheduling horizon	27 h	24 h	20 h	17 h	13 h	9 h
Time periods	22–24	1–24	5–24	8–24	12–24	16–24

The daily session, or daily market, takes place at 12:00, during which the bids for the 24 hours following the session close are placed. This is the main market and therefore the session in which much of the energy is negotiated.

The intraday markets are convened over the course of the previous day and the delivery day. Once the new offers are matched, they are added to the daily schedule to yield what is known as the final hourly schedule (PHF in Spanish). Obviously, less energy is traded in these markets since their time horizons are gradually reduced. They are designed to accommodate potential changes to trading forecasts. Table 1 shows the different time spans for the six intraday sessions, which are only open to those buyers or sellers that have taken part in the daily sessions.

Finally, the ancillary services are used when needed to resolve the imbalance between demand and generation, regulate the frequency/power, and control the voltage in the transport network. Their purpose, then, is to guarantee the balance, security, quality, and reliability of the electrical system.

For each hour, the producers and consumers that want to produce or consume electricity must place a bid in the various markets depending on their needs. Hourly in each session, the bids received are arranged from the highest to the lowest sale price and the highest to the lowest purchase price, with the lowest price being 0 and the highest being 180.3 €/MWh. Graphically, the result would be two aggregate curves, where the x-axis is the energy and the y-axis is the price. The matching method is "marginalist," meaning that the matching price for that hour and session is set at the intersection of the two aggregate curves. Any units remaining below and above that value will be sold and bought, respectively, at that price. In other words, all of the power contracted will be sold at that price.

To illustrate the matching process, Figure 1 shows an example for a case with five power plants and six large consumers placing a bid in the market for hour H.

Since most of the demand is not manageable, it offers to buy at the maximum of 180.3 €/MWh. But it is worth asking what criteria producers use to craft their sale offers to cover this demand. Nuclear and renewable plants tend to sell at 0 €/MWh to ensure that all of the energy they produce is consumed. This is due to their technical limitations, such as the inability to halt production in the case of nuclear and the inability to store primary energy in the case of renewables. The difference between the total system demand and the

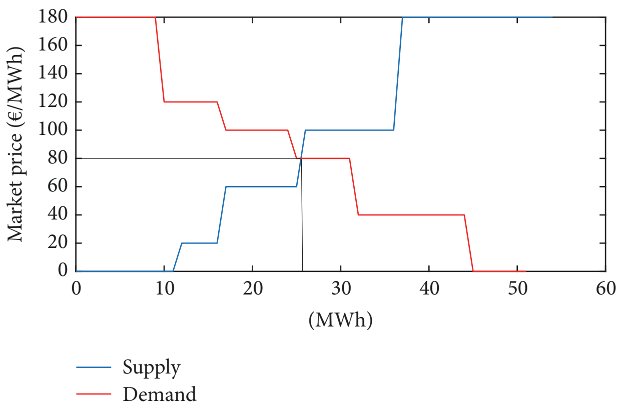

Figure 1: Aggregated bid-offer curves and market splitting.

energy produced by the above technologies and hydropower (in which the water flow can be regulated by using reservoirs) is known as the thermal gap. This difference is the energy demand that must be met using thermal technologies (such as gas and coal), the variable costs of which are higher than for renewables and nuclear. Therefore, the competition in the electricity market lies between the thermal generating plants, as it is on these plants that the intersection between the demand and supply curves during the market matching process, and therefore the final sale price, depends.

As Figure 1 shows, a lower demand entails lower prices by requiring fewer plants to be in operation and excluding the most expensive plants from the matching process. An increase in renewable production can result in a sharp drop in the matching price, leading to cases in which all of the demand is covered by the production priced at 0 €/MWh, as has already happened on numerous occasions. As the proportion of renewables in the energy mix grows, the average price in the electricity market drops. Specifically, the average energy price in Spain's electricity market in 2016 (48.4 €/MWh) fell by 23% with respect to 2015 to the lowest price since 2010. This was due primarily to the large response by hydro and wind power to cover the demand in the first few months of the year.

Reducing the use of thermal technologies requires incorporating more wind power into the energy mix. To do so, producers have to know how many units to supply to the daily market. A wind farm's generation must be forecast one day in

Table 2: Features in data examples.

Description	Units	Used as
Timestamp	IS0 8601	Input
Wind speed	m/s	Input
Wind direction	(°)	Input
Relative humidity	(1)	Input
Temperature	°C	Input
Atmospheric pressure	Pa.	Input
Issues	-	Input
Power generated	KWh.	Output
Power forecast	KWh.	-

advance, and, if necessary, these supply units can be corrected in the six intraday market sessions with time horizons of 27, 24, 20, 17, 13, and 9 hours. Unfortunately, the main drawback of wind energy is how difficult it is to make accurate forecasts due to this energy source's heavy reliance on atmospheric conditions, and how difficult, in turn, these are to predict.

In this paper we present a series of regression estimators that rely on deep-learning techniques to predict the generation of a wind farm based on an estimate of atmospheric conditions. These predictions are intended for the daily market, meaning they must offer sufficiently accurate results 12 to 36 hours in advance.

2. Materials and Methods

Below we describe the estimators implemented, the data used, and the procedures employed to train, evaluate, and compare the estimators.

2.1. Data Sets. The Institute of Technology and Renewable Energies (ITER) is the agency that runs the largest wind farm on the island of Tenerife. It also provided us with the data for this study. The ITER runs the MADE farm, which has a rated power of 4800 kW, supplied by eight MADE AE-46 generators. A weather forecast for the following 48 hours in one-hour periods is generated twice a day. Once a day the wind speed for the following 12 to 36 hours is forecast and a polynomial regression is carried out that is used to estimate the generation for each hour of said interval. This estimate is sent to the OMIE to be used in the daily market.

The ITER gave us a data set with an hourly sample with the results of the Numerical Weather Prediction (NWP) for different meteorological variables, the generation forecast made by the ITER using a polynomial regression, and the actual wind generation measured and a free-text field containing problems involving the operation of the generators. Table 2 summarises the features in each sample of the data set provided by the ITER and shows which were used as inputs and outputs for the estimators.

To prepare the data, we used certain feature engineering techniques. The timestamp was broken down into day of year and time of day, each of which was represented using the pair of values for their sine and cosine in an effort to capture their periodic nature and their effects on the local daily wind cycles. The wind direction was also encoded using this method for the same reason. The following shows an example of this using the wind direction:

$$h \longrightarrow \left(\sin\frac{h}{360}, \cos\frac{h}{360}\right). \tag{1}$$

The text for the issues was manually converted into the fraction of generators not in service at a given time, since some of the samples for the total farm output, which were used to adjust the predictor's output, were obtained when some of the generators were out of service. When the validation and test sets were configured, however, only those samples taken when all of the generators were in service during the period measured were used.

All of the inputs were normalised by min-max scaling between 0 and 1, the goal being to achieve maximum efficiency during training. For the training, we had data sampled each hour from January 2014 to April 2016, which were randomly divided into three sets: 60% comprised the data training set, 20% the validation set, and the remaining 20% the test data. The data were stored in TensorFlow TFRecords files for efficiency purposes for use with the TensorFlow framework [1], which was the tool used to develop, train, and evaluate the various predictors.

2.2. Feedforward Neural Networks. The first architecture evaluated was the Feedforward Neural Network (FNN). In a FNN [2] every neuron in one layer receives as its input all of the outputs from the neurons in the previous layer. The output a_j^l of the jth neuron in the lth layer can be expressed as indicated in the following:

$$a_j^l = \sigma^l\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right), \tag{2}$$

where w_{jk}^l is used to denote the weight for the output of the kth neuron in the $(l-1)$th layer as the input to the jth neuron in the lth layer, b_j^l is the bias of the jth neuron in the lth layer, and σ^l is the activation function of the neurons in the lth layer. The most common activation function is the sigmoid function, which is expressed as shown in the following:

$$\sigma^l(x) = \frac{e^x}{e^x + 1}. \tag{3}$$

In a regression problem like the one at hand, the sigmoid activation function is used in every layer except the last one, the output layer, which uses the identity function $\sigma^l(x) = x$ as the activation function.

The learning was done using standard deep learning techniques, like minibatch gradient descent [3] and Adagrad (for adaptive gradient algorithm) optimiser [4]. The latter allows different step sizes for different features, so it does not require a learning rate to be specified for it. The adjusted model was validated every 5000 trained batches. The stopping criterion employed was for the evaluation's Mean Square Error (MSE) not to decrease during three consecutive iterations. But in order to compare the accuracy of different predictors, we use the Mean Absolute Error (MAE) [5] and the Mean Absolute Scaled Error (MASE) [6]. The MAE and MASE are common measures of forecast error in time series analysis.

To avoid overadjusting the models when training them, the cost function includes a component to $L2$-norm to regularise all the weight w^l_{jk} and the bias b^l_j of the entire model. In some specific cases we used dropout [7] to check its effects when attempting to further generalise the trained models, but it did not help to improve the results. When dropout is used during training, only a selection of neurons chosen with probability P_{keep} can be activated. The following shows the generalisation of the output expression for neuron a^l_j when dropout is used:

$$a^l_j = \begin{cases} \dfrac{\sigma^l\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right)}{P_{\text{keep}}}, & \text{if } p \le P_{\text{keep}} \\ 0, & \text{otherwise} \end{cases} \qquad \text{where } p \sim U(0,1). \tag{4}$$

2.2.1. ReLU Activation Function. When training using minibatch gradient descent, the backward propagation undergoes a phenomenon called vanishing gradient [8], which considerably hampers the training of networks with a large number of layers. In these cases the ReLU activation function is very practical because it does not suffer from the vanishing gradient problem. Moreover, in regression problems it has the advantage of not being limited to outputs between 0 and 1, thus favouring the dispersity of the solution in the hidden neurons. The following shows the expression for the activation function for the lth layer of a neural network with N layers.

$$\sigma^l_{\text{relu}}(x) = \begin{cases} \max(0, x), & \text{if } l \ne N \\ x, & \text{if } l = N, \end{cases} \tag{5}$$

where x is used to indicate the input to the activation function, that is, the weighted sum of the inputs to the neuron, as shown in (2). The FNNs with the ReLU activation function that we trained use this function for the neurons in every layer except the output layer.

2.2.2. SELU Activation Function. Even using the ReLU activation functions, truly deep FNNs are difficult to train, which hampers their ability to handle high-level abstract relationships in the input samples. The Scaled Exponential Linear Unit (SELU) activation function induces self-normalising properties that make the neuron activations automatically converge toward an average of 0 and a variance of 1 [9]. This property propagates throughout the network even in the presence of noise and perturbations. This allows training networks with more layers and the use of strong regularisation and it makes the training more robust.

The following shows the expression for this type of activation function:

$$\sigma^l_{\text{selu}}(x) = \lambda \begin{cases} x, & \text{if } x > 0 \\ \alpha e^x - \alpha, & \text{if } x \le 0, \end{cases} \tag{6}$$

where x is used to indicate the input to the activation function. Klambauer et al. [9] justify why α and λ must have the values shown in

$$\begin{aligned} \alpha &= 1.6732632423543772848170429916717 \\ \lambda &= 1.0507009873554804934193349852946 \end{aligned} \tag{7}$$

in order to ensure that the neuron activations converge automatically toward an average of 0 and a variance of 1.

2.3. Convolutional Neural Networks. It is possible for the atmospheric conditions in previous hours to contain information that can be used to improve the forecast at any given time. This information was introduced into predictors based on FNNs to prepare samples that contained the weather forecast features for every hour t and for the $N-1$ previous hours, with the input layer for said models being suitably expanded.

In order to check the results when the neural network is forced to exploit the time-local correlation between the features by forcing a connection pattern between adjacent neurons in each layer, we implemented some predictors using models based on Convolutional Neural Networks (CNNs). CNNs are biologically inspired variants of FNNs used primarily in computer vision problems [10], although their ability to exploit spatially local correlation in images can also be used in time-series forecasting.

In these models, the output of each neuron a^l_j is not generated based on the output of every neuron in the previous stage, as shown in (2); rather, it is generated from a subset of spatially adjacent neurons. So to improve the learning efficiency, every neuron in the same layer shares the same weight and bias, meaning the layer can be expressed in terms of a filter that is convoluted with the output of the previous layer. The following shows the output a^{lk}_j of the jth neuron of the lth convolutional layer:

$$a^{lk}_j = \sigma^{lk}\left(\left(W^{lk} * a^{l-1}\right)_j + b^{lk}\right), \tag{8}$$

where $(W^{lk} * a^{l-1})_j$ is the jth element resulting from the convolution of the filter defined by W^{lk} with the output of the previous layer a^{l-1}, σ^{lk} is the activation function for the

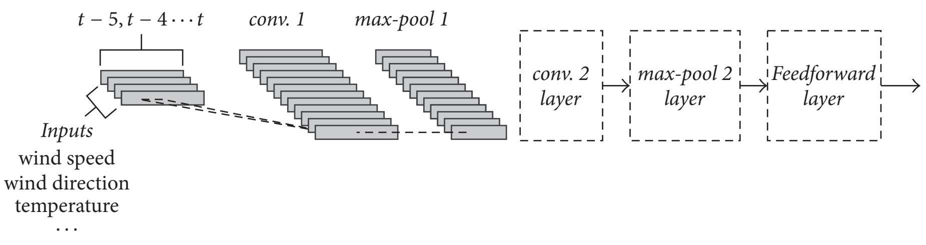

Figure 2: General diagram of the CNNs used.

convolutional layer, and $k \in [0 \cdots K]$ indicates that it is the output of the kth channel of the layer. In each convolutional layer, different outputs can be applied to the output of the previous layer to generate different representations or channels, thus yielding a fuller representation of the data. In order to apply a CNN to the problem of time-series forecasting, we arranged the samples in such a way that the time series of each characteristic is an input channel to the network and thus to the first convolution later.

Figure 2 shows a general diagram of the CNN developed in this paper to forecast time series. Behind each convolutional layer with a ReLU activation function is a max-pooling layer, which partitions the input into a set of nonoverlapping ranges and, for each range, outputs the maximum value. Behind several convolutional exchange layers and max-pooling layers there is a feedforward layer (as described in (2)) to yield the output of the entire network.

2.4. Recurrent Neural Networks. As we have shown, FNNs and CNNs can use historical data series but they lack the memory to store information over the long term. They also cannot use the information contained in the output of the network at previous instants. Recurrent Neural Networks (RNNs) [11] solve this problem by making the output y_t at timestamp t depend on previous computations through a hidden state s_t that acts as a memory for the network, as shown in the following:

$$\begin{aligned} s_t &= \sigma\left(W x_t + U s_{t-1}\right) \\ y_t &= f\left(s_t\right), \end{aligned} \tag{9}$$

where x_t is the input to the network, $\sigma(x)$ is the state activation function, W and U are the weights by which the inputs and the state for the previous instant are multiplied, respectively, to generate the new state s_t, and $f(x)$ is the function that generates the network's output based on the state. On occasion this function $f(x)$ will be the identity function $f(x) = x$, but it can also be a feedforward layer like the one described in (2).

Figure 3 shows the RNN we used, unfolded into a full network. By unfolded we simply mean that we write out the network for the complete sequence of inputs and take the output at t as the network's prediction for fitting. Instead of the basic RNN cell explained previously, we used two, more advanced cells: long short-term memory (LSTM) [11] and Gated Recurrent Unit (GRU) [12] cells.

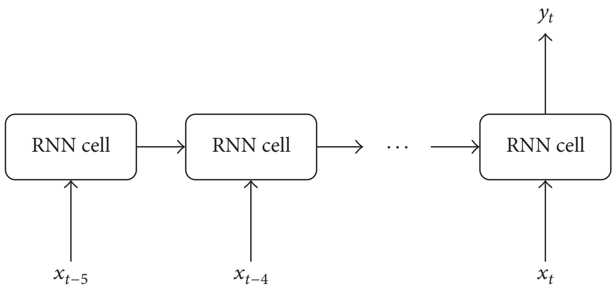

Figure 3: RNN unfolded into a full network.

LSTM recurrent neural networks are capable of learning and remembering over long input sequences and tend to work very well for time-series forecasting problems [13]. As (10) well shows, the output y_t depends on the state s_t of the LSTM cell through the activation function $\sigma_y(x)$ (which is generally $\tanh(x)$). The output gate o_t controls the extent to which the state s_t is used to compute the output y_t by means of the Hadamard product ($\circ$):

$$\begin{aligned} f_t &= \sigma_g\left(W_f x_t + U_f y_{t-1} + b_f\right), \\ i_t &= \sigma_g\left(W_i x_t + U_i y_{t-1} + b_i\right), \\ o_t &= \sigma_g\left(W_o x_t + U_o y_{t-1} + b_o\right), \\ \widetilde{s}_t &= \sigma_s\left(W_s x_t + U_s y_{t-1} + b_s\right), \\ s_t &= f_t \circ s_{t-1} + i_t \circ \widetilde{s}_t, \\ y_t &= o_t \circ \sigma_y\left(s_t\right). \end{aligned} \tag{10}$$

The state s_t depends on the state of the previous instant s_{t-1} and on the candidate for the new value of the state $\widetilde{s}_t$. The input gate i_t controls the extent to which $\widetilde{s}_t$ flows into the memory and the forget gate f_t controls the extent to which s_{t-1} remains in memory. The o_t, f_t, and i_t gates and the candidate for the new value of the state of the cell $\widetilde{s}_t$ can be interpreted as the outputs of conventional artificial neurons whose inputs are the input to cell x_t at t and the output of cell y_{t-1} at $t-1$. The activation function for the gates σ_g is the sigmoid function, while for σ_s it is $\tanh(x)$.

GRU recurrent neural networks use a simpler cell without a forget gate and with fewer parameters, meaning they can generally be trained with fewer samples. Chung et al. [14] shows experimentally its superiority over LSTM for simple networks, but cannot conclude that GRU is better in different cases. The following shows the equations that govern the behaviour of these cells:

$$\begin{aligned}
z_t &= \sigma_g\left(W_z x_t + U_z y_{t-1} + b_z\right),\\
r_t &= \sigma_g\left(W_r x_t + U_r y_{t-1} + b_r\right),\\
\tilde{y}_t &= \sigma_y\left(W_y x_t + U_y\left(r_t \circ y_{t-1}\right) + b_y\right),\\
y_t &= z_t \circ y_{t-1} + \left(1 - z_t\right) \circ \tilde{y}_t,
\end{aligned} \tag{11}$$

where $\tilde{y}_t$ is the cell's output candidate, z_t is the update gate, which controls the extent to which y_{t-1} or $\tilde{y}_t$ is used to compute the output, and r_t is the reset gate, which controls the extent to which y_{t-1} flows into the cell's output candidate $\tilde{y}_t$. In GRU the activation functions $\sigma_g(x)$ and $\sigma_y(x)$ are the sigmoid function and $\tanh(x)$, respectively.

2.5. Sensitivity to Disturbances. An important aspect in this paper is to analyse the behaviour of our models in the presence of input disturbances. In the following equation we show the expression for the input x_j assuming that it undergoes a small incremental change Δx_j:

$$x_j = x_j^* + \Delta x_j, \tag{12}$$

where x_j^* is the jth value of the input sample without disturbance. It is important to note that the input x_j to the model is the jth input to the first layer. Similarly, y is the output of the last layer.

A perturbation Δx_j in the input x_j induces a disturbance in the output y of the neural network. When there is no perturbation in any of the model's inputs, the output of the neural network is y^*. In order to determine if the model is robust against perturbations in the jth input, the sensitivity s_j has to be calculated [15]. We show its expression in the following:

$$s_j = \frac{\Delta y / y^*}{\Delta x_j / x_j^*} = \frac{\Delta y / y^*}{\delta_j}, \tag{13}$$

where Δy is the corresponding change in the value of the output variable y and $\delta_j = \Delta x_j / x_j^*$ is the input perturbation ratio.

If the sensitivity s_j is lower than 1.0, it means that the network attenuates the input disturbances, whereas if it is equal to 1.0, it means that the network neither attenuates nor amplifies disturbances.

3. Results and Discussion

As noted earlier, the wind farm currently uses a polynomial regression to predict the farm's generation, as required to participate in the daily market. To give an idea of its accuracy, Table 3 shows the MAE and MASE for some estimators using the historical data available. The MASE indicates the absolute error relative to the error in the one-hour naive forecast reference estimator. Therefore, a MASE greater than 1.0 indicates the predictor works worse than the reference estimator, while a MASE lower than 1.0 indicates that it works better.

Table 3: Accuracy of current estimators.

Estimator	MAE (kWh)	MASE
Naive 1 h	261	1.00
Naive 24 h–48 h	780	2.99
Polynomial	552	2.11

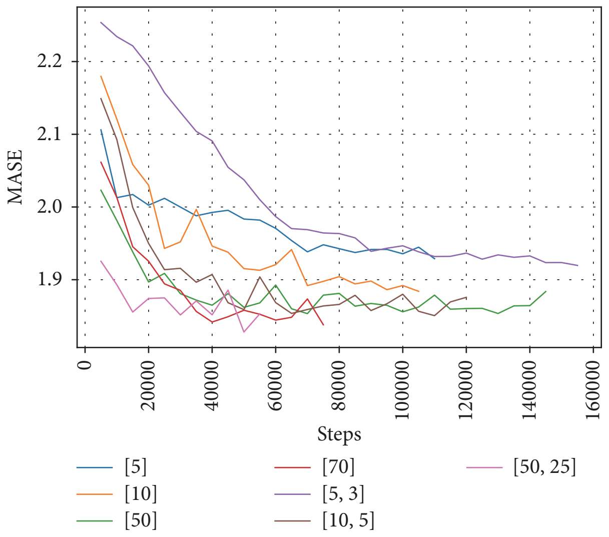

Figure 4: Trend in MASE for different FNNs with ReLU.

As Table 3 shows, the polynomial estimator is a little over two times worse than the one-hour naive forecast reference estimator; however, this naive estimator would never be able to be used because the prediction has to be made and sent to the grid operator 24 hours in advance.

To obtain a more realistic comparison, it was compared with another naive estimator that uses the actual generation measured 24 or 48 hours earlier for its prediction at a given time. This naive estimator could be used at the farm, though Table 3 shows that the polynomial regression is considerably better than this second naive estimator.

3.1. Feedforward Neural Networks. Figures 4, 5, and 6 show how the MASE evolved for the validation data set over the course of the iterations for networks with different numbers of hidden layers, neurons and ReLU, sigmoid, and SELU activation functions, respectively.

In every case the final result is similar and slightly better than for the polynomial regression, whose MASE is 2.11. Figure 4 shows that a ReLU activation function yields good results with around 20 neurons between all the hidden layers. If this size is increased, the number of overfitting cases rises gradually and the MASE is not reduced by either adding more layers or making the layers larger.

Figure 5 shows the results for FNNs with a sigmoid activation function. In this case we clearly see that two hidden layers yield better results than one, but after that no further improvements are obtained by expanding the network. The

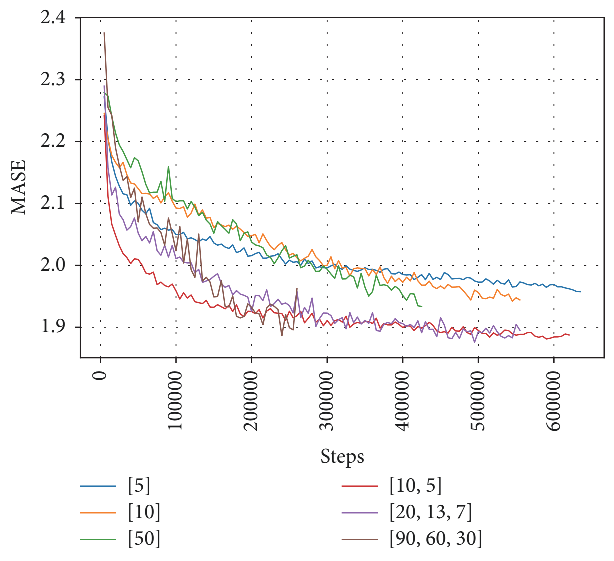

FIGURE 5: Trend in MASE for different FNNs with a sigmoid activation function.

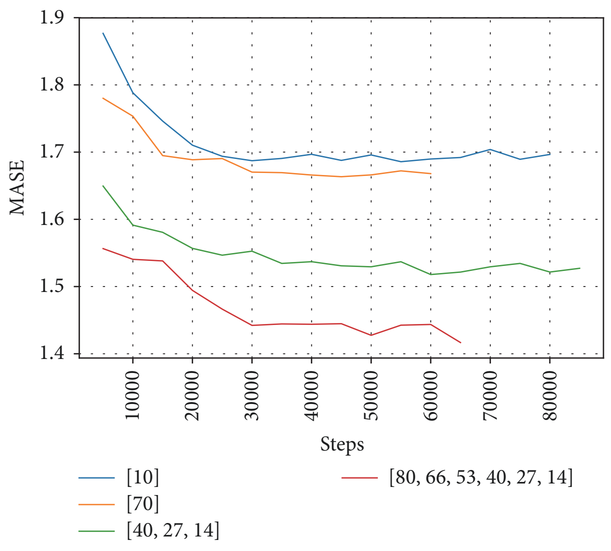

FIGURE 7: Trend in MASE for different FNNs with ReLU activation function and 6 h historical input data.

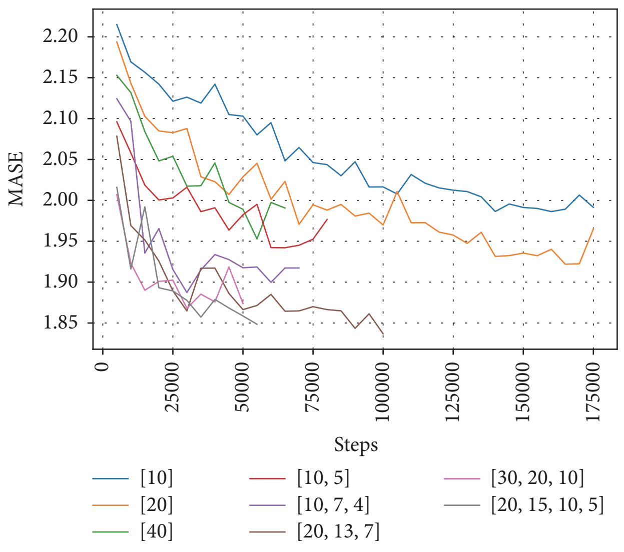

FIGURE 6: Trend in MASE for different FNNs with SELU activation function.

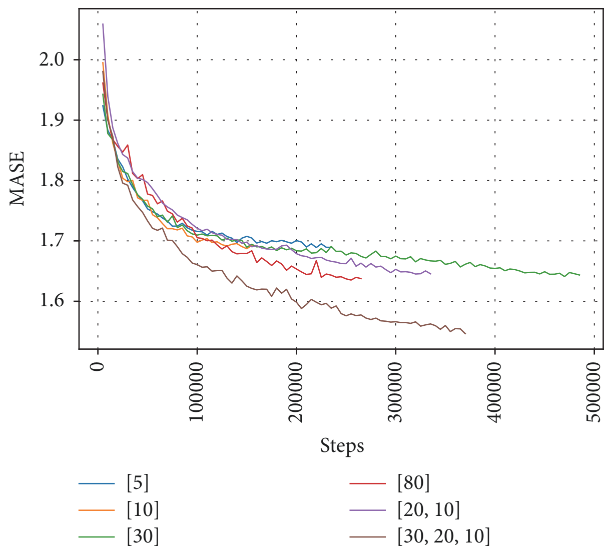

FIGURE 8: Trend in MASE for different FNNs with sigmoid activation function and 6 h historical input data.

FNNs with a sigmoid activation function need three or four times more steps than a ReLU network to converge. In fact, of all the network types studied, they required the highest number of steps to converge. This problem grows with the number of layers due to the vanishing gradient problem.

Figure 6 shows the same curves but for FNNs with a SELU activation function. The improvement resulting from increasing the size of the first layer has a bound that can be overcome by increasing the number of layers. In general, it does not yield better results than the ReLU activation function for our problem, but this is to be expected since the benefits of this function are evident when used in problems that require a large number of layers Klambauer et al. [9].

We then trained estimators similar to the above, but with samples that contained the weather forecast features for every hour t and for the previous 5 hours. As before, Figures 7, 8, and 9 show how the MASE evolved for this new data set. The final result is similar in every case, but better than for the polynomial regression and for the previous FNNs. This shows that these estimators are capable of making good use of time information.

Figures 8 and 9 also exhibit behaviour similar to Figures 5 and 6 respectively, converging to solutions with a smaller MASE. It is interesting to note that although the networks with a SELU activation function behave similarly to those with the ReLU function, the former exhibit fewer overfitting problems as the size of the network grows. In other words,

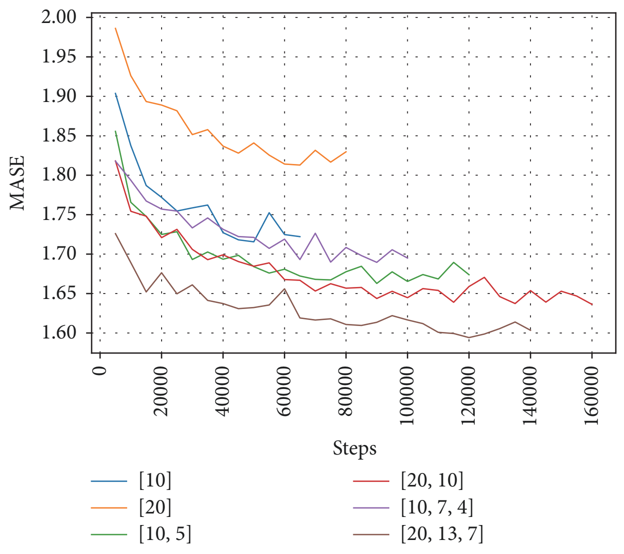

FIGURE 9: Trend in MASE for different FNNs with SELU activation function and 6 h historical input data.

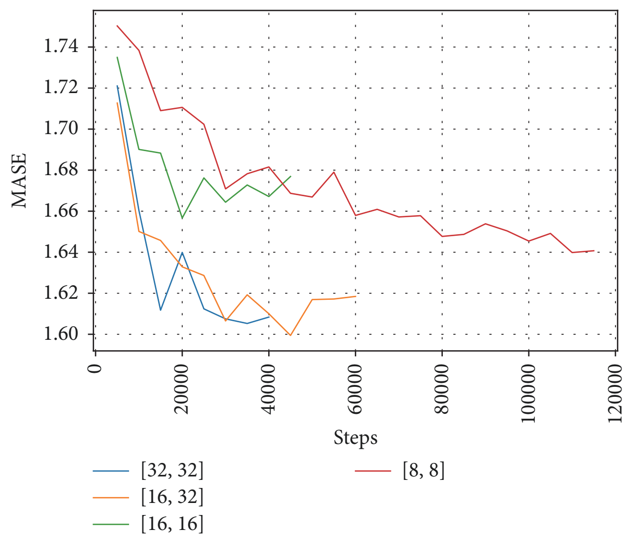

FIGURE 10: Trend in MASE for different CNNs.

with the SELU activation function, increasing the number of layers does not improve the results but the estimators are trained equally well.

3.2. Convolutional Neural Networks. We trained an CNN-based estimator using the architecture shown in Figure 2 in order to compare its performance with that of previous FNNs and to try to improve the results obtained by the latter. The size of the convolution filters was set at five, the ReLU activation function was selected for the convolutional layers, as is usual for these networks, and the size of the max-pooling window was set at three. Figure 10 shows how the MASE evolved for the validation data set over the course of the iterations for networks with different numbers of channels at the output of each convolution: between 32 and 8. The MASE for networks with at least 32 to 16 channels is very similar to that obtained for FNNs, but the CNNs converge much faster, in approximately half the time. The advantage is that the training for this type of network can be speeded up considerably by using graphics processing units (GPU), although this possibility was not explored for this paper.

Figure 11 uses an image with 256 grey levels to show a representation of the coefficients of the filters trained for the first convolutional layer of the CNN with eight channels. Each filter has one row per input characteristic and shows how the filter uses said characteristic to contribute to the layer's output.

3.3. Recurrent Neural Networks. A similar procedure was used to train the RNNs with LSTM and GRU cells of various sizes for the output y_t. Figures 12 and 13 show the trend in the MASE when training the RNN LSTM and GRU, respectively. Both RNN types exhibit an error that is very slightly larger than that of the FNNs and CNNs. The number of steps needed to complete the training is also very similar, though in reality each step consumes much more time. Although not shown in the graph, the amount of time needed by the RNNs was approximately five times that needed by the equivalent FNNs.

It should be noted that no large differences are evident between the RNN LSTM and the RNN GRU. It is also surprising how well both types of RNNs work, even with size 1 cells, which store considerably less information in their status than larger cells.

In light of these results, the problem of predicting wind generation for the daily market is better resolved by providing as input a time series with the forecast for previous hours and using estimators based on FNNs with a ReLU or SELU activation function, or CNNs. Specifically, the latter can be trained with a lower number of iterations and, presumably, in less time with the use of GPUs. As concerns ReLU FNNs and SELU FNNs, it is easier to avoid overfitting in the latter, though, if trained correctly, both are equally accurate.

3.4. Sensitivity to Disturbances. Finally, we selected the best models of each type in order to analyse their behaviour in response to disturbances in the input. In every case, we used models trained with 6 h time series. As we discussed in Section 2.5, we ran the models with and without perturbations in the inputs in order to calculate their sensitivity. The perturbations were applied into the matching inputs for the forecasts of wind velocity and direction for the time when the generation is to be predicted, because these are the inputs with the greatest influence on the model's output.

In this paper the input perturbation ratio and the sensitivity values obtained are shown in percentages. Sensitivities below 100% indicate that the estimators are capable of attenuating the perturbations, while sensitivities in excess of 100% indicate that they are amplified.

Figure 14 compares the average sensitivity of the best estimators for various network classes and sizes for different perturbation levels in wind speed. The figure shows that all of the estimators attenuate the input perturbations, reducing their influence on the output. In every case, the best

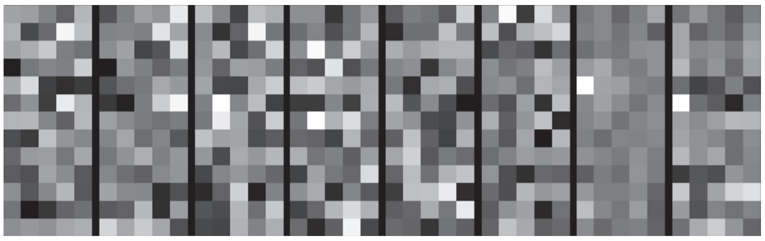

Figure 11: Coefficients of the filters in the first convolutional layer with 8 channels.

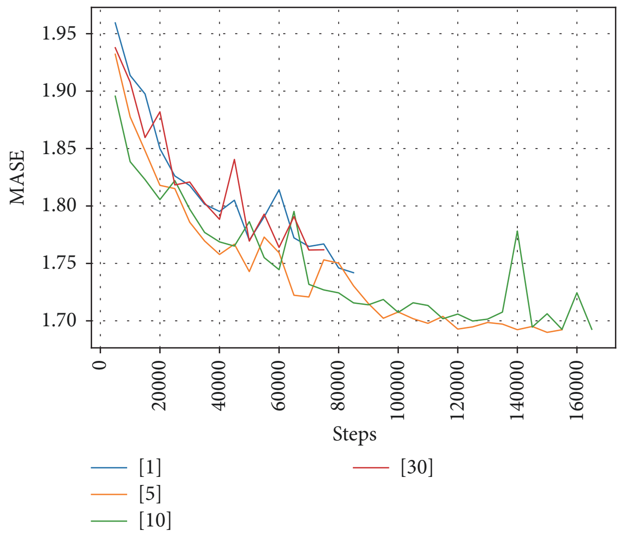

Figure 12: Trend in MASE for different LSTM RNNs.

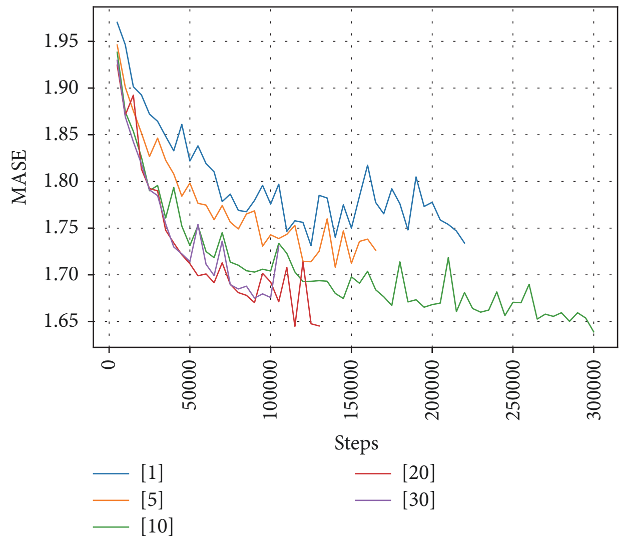

Figure 13: Trend in MASE for different GRU RNNs.

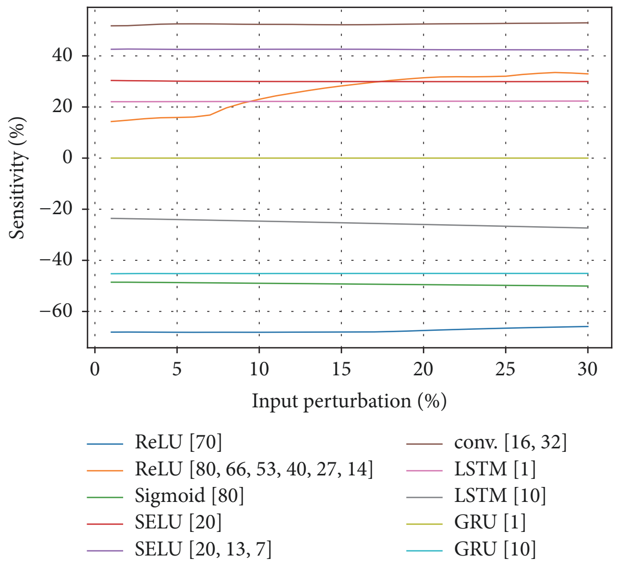

Figure 14: Trend in sensitivity for different disturbances in the wind speed.

performance is exhibited by the estimators based on RNNs, followed by SELU FNNs, sigmoid FNNs, and CNNs.

Figure 15 shows the average sensitivity of the same models for different perturbation ratio values in the wind direction input. In this case the results are much more dissimilar, undoubtedly because the relationship between wind direction and power output is highly nonlinear and much more complicated to model adequately during training. As Figure 15 shows, only the estimators based on SELU FNNs, CNNs, and RNNs cells of size 1 are capable of attenuating this type of perturbations.

In light of these results and considering those obtained previously involving the accuracy of the different estimator types evaluated, the problem of predicting wind power generation for the daily market is better resolved by using estimators based on SELU FNNs or CNNs.

4. Conclusions

In this paper we considered the problem of predicting wind power generation in order to take part in the daily market that regulates the supply and demand in the Spanish electric system. We used deep-learning techniques to develop different predictors based on neural networks that were trained using data provided by the MADE wind farm, operated by the ITER on the island of Tenerife.

The predictors evaluated are based on feedforward neural networks of varying sizes and with different activation functions, convolutional neural networks, and recurrent neural

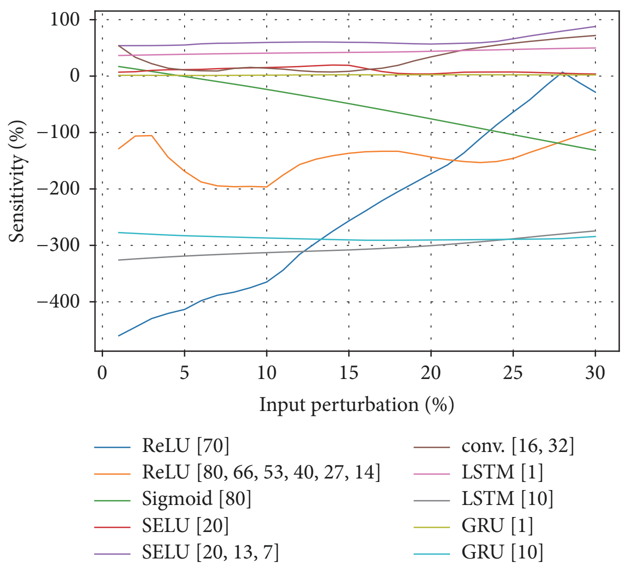

FIGURE 15: Trend in sensitivity for different disturbances in wind direction.

networks. The conditions were the same as those employed for the farm with the polynomial model now in use, namely, relying on the weather forecast at least 24 hours in advance to output a predicted generation for the farm. The methodology was checked by training and validating the model with samples taken every hour during the past three years.

The results were adequate, yielding better results than the 1-hour reference naive forecast estimator and the polynomial model used at the wind farm. Specifically, the use of time series for the input samples proved to be the best way to minimise the error. Moreover, of the different types of neural networks evaluated, the CNNs and FNNs with the ReLU or SELU activation function were shown to be the most accurate, although the differences between the best candidates from the various network types were not significant. The traditional sigmoid FNNs are on a par with the other types trained, though they converge much more slowly during training.

Finally, we conducted a sensitivity analysis of the models, which revealed that trained neural networks are able to attenuate some input disturbances. For disturbances in the wind speed input, the best candidates from every network type were able to attenuate the disturbances, though this is much more difficult to achieve with perturbations in the wind direction input, which even caused some network types to amplify the perturbations. In this case, the CNN, SELU FNNs, and the various RNN types exhibit the best performance. Taking all the results into consideration, the best neural network estimators, from the standpoint of offering the lowest absolute error and being the least sensitive to perturbations, are those based on SELU FNNs and CNNs.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors wish to thank the Institute of Technology and Renewable Energies (ITER), and especially Guillermo Galván García, Head of the Wind Department, and Jonatán Felipe García, Manager of Engineering Department R&D Projects, for their collaboration.

References

[1] M. Abadi, A. Agarwal, P. Barham et al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015.

[2] H. Yañez-Badillo, R. Tapia-Olvera, O. Aguilar-Mejía, and F. Beltran-Carbajal, "On Line Adaptive Neurocontroller for Regulating Angular Position and Trajectory of Quadrotor System," *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, vol. 14, no. 2, pp. 141–151, 2017.

[3] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, pp. 1139–1147, PMLR, 2013.

[4] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research (JMLR)*, vol. 12, pp. 2121–2159, 2011.

[5] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005.

[6] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[8] X. Glorot, X. Glorot, and Y. Bengio, in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10)*, Society for Artificial Intelligence and Statistics, 2010.

[9] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," 2017, http://arxiv.org/abs/1706.02515.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] K. Cho, B. van Merrienboer, C. Gulcehre et al., "Learning phrase representations using rnn encoder-decoder for statistical machine translation," http://arxiv.org/abs/1406.1078.

[13] D. L. Marino, K. Amarasinghe, and M. Manic, "Simultaneous generation-classification using LSTM," in *Proceedings of the IEEE Symposium Series on Computational Intelligence, SSCI '16*, Greece, December 2016.

[14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, http://arxiv.org/abs/1412.3555.

[15] J. Quan, X. Feng, G. Su, and G. Chen, *Chinese Journal of Rock Mechanics and Engineering*, vol. 26, article 2654, 2007 (Chinese).