Temporal Convolutional Neural Networks for Solar Power Forecasting

Yang Lin
School of Computer Science
University of Sydney
Sydney, Australia
ylin4015@uni.sydney.edu.au

Irena Koprinska
School of Computer Science
University of Sydney
Sydney, Australia
irena.koprinska@sydney.edu.au

Mashud Rana
Data61
CSIRO
Sydney, Australia
mdmashud.rana@data61.csiro.au

***Abstract*—We investigate the application of Temporal Convolutional Neural Networks (TCNNs) for solar power forecasting. TCNN is a novel convolutional architecture designed for sequential modelling, which combines causal and dilated convolutions and residual connections. We compare the performance of TCNN with multi-layer feedforward neural networks, and also with recurrent networks, including the state-of-the-art LSTM and GRU recurrent networks. The evaluation is conducted on two Australian datasets containing historical solar and weather data, and weather forecast data for future days. Our results show that TCNN outperformed the other models in terms of accuracy and was able to maintain a longer effective history compared to the recurrent networks. This highlights the potential of convolutional architectures for solar power forecasting tasks.**

***Index Terms*—solar power forecasting, deep learning, temporal convolutional neural network, recurrent neural networks**

I. Introduction

Photovoltaic (PV) solar power is regarded as one of the most promising sources of renewable energy. It is clean, easily available and also cost-effective due to the recent advances in PV technology and the declining cost of solar panels.

However, the large-scale integration of solar energy into the electricity grid is challenging. The reason for this is the variable and uncertain nature of the generated solar power as it depends on solar irradiance, cloud cover and other weather factors. Specifically, unexpected changes in large, grid-connected solar power plants can destabilize the grid - conventional electricity generators need to be turned off or on to meet the downward or upward net ramping load needs. If this cannot be done, the PV power generation would need to be curtailed [1], to ensure safe and reliable operation of the power grid. However, power plants have different startup and shutdown time, e.g. conventional coal-fired generators are too slow to start and need to be scheduled 8-48 hours in advance [2]. Hence, there is a need to develop accurate PV power forecasting methods, to support scheduling and dispatch decisions, meet the minimum generation and ramping requirements, in order to facilitate reliable and efficient operation of the electricity grid.

In this paper, we consider the task of simultaneously predicting the PV power output for the next day at half-hourly intervals. Specifically, given: (1) a time series of PV power output up to day d: $PV = [PV_1, ..., PV_d]$, where PV_i is a vector of half-hourly PV power output for day i, (2) a time series of weather vectors for the same days: $W = [W_1, ..., W_d]$, where W_i is a weather vector for day i, and (3) a weather forecast vector for the next day $d+1$: WF_{d+1}, our goal is to forecast PV_{d+1}, the half-hourly PV power output for day $d+1$.

Different approaches for solar power forecasting have been proposed based on statistical and machine learning methods. The statistical approaches use classical time series forecasting methods such as exponential smoothing, autoregressive moving average and linear regression [3]–[6]. The machine learning approaches utilize a variety of algorithms, e.g. neural networks [3], [5]–[9], nearest neighbor [3], [5], [10], support vector regression [5], [6], [11]–[13] and ensembles [6], [14]–[16]. Neural network based approaches using feedforward multi-layer networks are the most popular methods and have shown good results. They are appealing as they can learn from examples, model complex nonlinear relationships and also deal with noisy data.

Another class of neural networks - Convolutional Neural Networks (CNNs) have recently gained a lot of interest, showing excellent performance in computer vision, speech and language processing tasks [17]–[19]. A few recent studies applied CNNs to time series forecasting with promising results [20]–[22]. The motivation behind applying CNNs to time series data is that they would be able to learn filters that represent repeated patterns in the series and use them to forecast future values [20]. CNNs are also able to automatically learn and extract features from the raw data without prior knowledge and feature engineering. They may also work well on noisy time series by discarding the noise at each subsequent layer, creating a hierarchy of useful features and extracting only the meaningful features [20].

In this paper, we investigate the application of Temporal Convolutional Neural Networks (TCNNs) for solar power forecasting. TCNN [23] is a novel convolutional architecture, specifically designed for sequential modelling tasks. It is informed by the best practice in convolutional networks research, combining advanced concepts such as causal convolutions, dilated convolutions and residual connections. TCNN has demonstrated excellent results on benchmark sequence tasks for processing image, language and music data. Our goal is to investigate its effectiveness for predicting solar power time series and compare its performance with multi-layer feedforward neural networks, and also Recurrent Neural Networks (RNNs) including the standard RNN, the state-of-the-art Long Short Term Memory (LSTM) and the recently proposed Gated Recurrent Units (GRU) [24].

Our contribution can be summarized as follows:

1) We investigate the application of TCNN for solar power forecasting. This is motivated by the success of TCNN for other sequence forecasting tasks.
2) We compare the performance of TCNN with multi-layer feedforward network, which is the most widely used method for solar power forecasting, and also with standard and advanced recurrent neural networks (RNN, LSTM and GRU) which have not been widely studied for solar power forecasting.
3) We propose a method for using data from multiple data sources with TCNN and the recurrent neural networks. The standard applications of these methods use univariate solar data from a single source, while we propose a representation that utilizes data from three sources: historical PV and weather data, and weather forecast data for future days.
4) We comprehensively evaluate the performance of TCNN on two Australian datasets, containing data for two years. Our results demonstrate that TCNN was the most accurate model, outperforming the feedforward and recurrent networks. This highlights its potential for solar power forecasting as a viable alternative to the widely used feedforward networks and the more complex recurrent networks.
5) We study the impact of the sequence length on the accuracy of all models and discuss the results.

II. Data

A. Data Sources and Feature Sets

We collected data from two PV plants in Australia. The two datasets are called the University of Queensland (UQ) and the Sanyo datasets and contain both PV and weather data. The data sources and extracted features for each dataset are summarized in Table I and Table II respectively.

Solar PV data. The PV data for the UQ dataset was collected from a rooftop PV plant located at the University of Queensland in Brisbane, in the state of Queensland. The Sanyo dataset was collected from the Sanyo PV plant in Alice Springs, Northern Territory. The two PV plants are situated about 2600 km apart in different climate zones. The UQ PV data was obtained from [25], and the Sanyo PV data was obtained from [26]. Both datasets contain data for two years - from 1 January 2015 to 31 December 2016 (731 days).

Weather data. We also collected corresponding weather data for the two datasets. The weather data for UQ dataset was collected from the Australian Bureau of Meteorology [27], from a weather station located close to the PV plant. The weather data for the Sanyo dataset was collected from a weather station located on the site of the Sanyo PV plant.

There are three sets of weather features - W1, W2 and WF, described in Tables I and II for the two datasets.

W1 includes the full set of collected weather features - 14 for the UQ dataset and 10 for the Sanyo dataset. The 10 features are common for both datasets; the UQ dataset contains four additional features (daily rainfall, daily sunshine hours and cloudiness at 9am and 3pm) which were not available for the Sanyo dataset.

W2 is a subset of W1 and includes only 4 features for the UQ dataset and 3 features for the Sanyo dataset. These features are frequently used in weather forecasts and available from meteorological bureaus. The weather forecast feature set WF is obtained by adding 20% Gaussian noise to the W2 data. This is done since the weather forecasts were not available retrospectively for previous years. When making predictions for the days from the test set, the WF set replaces W2 as the weather forecast for these days.

B. Data Preprocessing

The raw PV data was measured at 1-minute intervals for the UQ dataset and 5-minute intervals for the Sanyo dataset and was aggregated to 30-minute intervals by taking the average value of every 30-minute intervals.

There was a small percentage of missing values - for the UQ dataset: 0.82% in the PV power data and 0.02% in the weather data; for the Sanyo dataset: 1.98% in the PV power data and 4.85% in the weather data. These missing values were replaced using a nearest neighbour method, applied firstly to the weather data and then to the PV data. Specifically, if a day i has missing values in its weather vector W_i, we find its nearest neighbor day without missing values, day s, and replace the missing values in W_i with the corresponding values from W_s. Then, if a day i has missing values in its PV vector PV_i, we find its nearest neighbor day s, by comparing weather vectors and replace the missing values in PV_i with the corresponding values from PV_s.

Both the PV and weather data were normalised to the range [0,1].

III. Temporal Convolutional Neural Network

For TCNN, we used the generic architecture proposed by Bai et al. [23]. It builds upon recent CNN architectures for sequential data such as WaveNet [19] but is specifically designed to be simpler and to combine autoregressive prediction with a long memory.

As shown in Fig. 1, TCNN is a hierarchical architecture, consists of several convolutional hidden layers with the same size as the input layer. In our case, the input is a sequence of feature vectors corresponding to previous days, e.g. $\mathbf{x_1}, ..., \mathbf{x_d}$, and the target is the PV vector for the next day: $\hat{\mathbf{y}}_{d+1} = PV_{d+1}$.

TCNN is designed to process data from one source, element by element. Our data comes from three sources (PV solar, weather and weather forecast) and the weather features are not synchronised with the PV data - they are collected at different frequencies, e.g. there are 20 values for the PV data (every

TABLE I
UQ Dataset - Data Sources and Feature Sets

Data source	Feature set	Description
PV data	PV$\in \Re^{731\times 20}$	Daily: half-hourly solar power between 7am and 5pm.
Weather data 1	W1$\in \Re^{731\times 14}$	(1-6) Daily: min temperature, max temperature, rainfall, sunshine hours, max wind gust and average solar irradiance; (7-14) At 9am and 3pm: temperature, relative humidity, cloudiness and wind speed (cloudiness is not available in Sanyo dataset).
Weather data 2	W2$\in \Re^{731\times 4}$	Daily: min temperature, max temperature, rainfall and solar irradiance. W2 is a subset of W1.
Weather forecast data	WF$\in \Re^{731\times 4}$	Daily: min temperature, max temperature, rainfall and average solar irradiance.

TABLE II
Sanyo Dataset - Data Sources and Feature Sets

Data source	Feature set	Description
PV data	PV$\in \Re^{731\times 20}$	Daily: half-hourly solar power between 7am and 5pm.
Weather data 1	W1$\in \Re^{731\times 10}$	(1-4) Daily: min temperature, max temperature, max wind gust and average solar irradiance; (5-10) At 9am and 3pm: temperature, relative humidity and wind speed (cloudiness is not available in Sanyo dataset).
Weather data 2	W2$\in \Re^{731\times 3}$	Daily: min temperature, max temperature and solar irradiance. W2 is a subset of W1.
Weather forecast data	WF$\in \Re^{731\times 3}$	Daily: min temperature, max temperature and average solar irradiance.

30 minutes) but only 1-2 values for a weather feature, e.g. temperature at 9am and 3pm.

To facilitate the application of TCNN to data from multiple data sources, we consider each element of the sequence as a feature vector comprising the features from all sources. Specifically, we have devised the following representation: each element of the input sequence is a vector $\mathbf{x}_i = [PV_i; W1_i; WF_{i+1}]$, corresponding to day i and containing the PV features for day i, the weather features for day i and the weather forecast features for day $i+1$.

TCNN includes three main techniques: causal convolutions, dilated convolutions and residual connections.

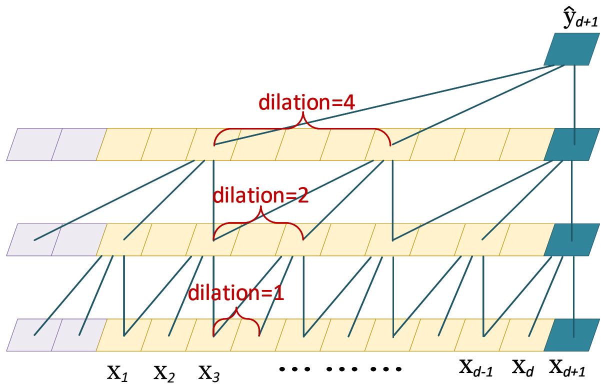


Fig. 1. TCNN architecture

Causal convolutions. The output at time t is convolved only with elements from time t or earlier time steps from the previous layer. This principle has been used in Waibel's time-delay network [28] and the WaveNet architecture [19]. For the hidden layers, zero padding of length *kernel size-1* is used to ensure that the hidden layers have the same dimensionality as the input layer to facilitate the convolutions.

Dilated convolutions. This technique was introduced in [19], [29] to enable large receptive fields, and in turn to capture a long memory, which is not possible with causal convolutions alone as they require a very deep neural network.

The dilated convolutional operator F on the sequence element s is defined as:

$$F(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d\cdot i} \tag{1}$$

where $f : \{0, \dots, k-1\} \to \mathbb{R}$ is the convolution filter, $\mathbf{x}$ is the sequential input, k is the filter size, and d is the dilation factor.

While the convolution kernel remains the same for all layers, the dilation factor increases exponentially with the depth of the network: $d_l = 2^l$, where l is the network level. For example, as illustrated in Fig. 1, d is 1 at the first layer (corresponding to regular convolutions) and then it exponentially increases at each layer, reaching 4 at the last hidden layer. This pyramidal structure and aggregation effectively increases the receptive field of TCNN without loss of input coverage, allowing for long memory.

Residual connections. Residual blocks were introduced by He et al. [30] to overcome the gradient vanishing problem in networks with many layers. The main idea is to add the input $\mathbf{x}$ to a block of stacked layers (transformation $\mathcal{F}$) to the output of this block by using shortcut connections:

$$o = \sigma(\mathbf{x} + \mathcal{F}(\mathbf{x})) \tag{2}$$

where σ is the activation function.

Fig. 2 illustrates the structure of the TCNN's residual block, consistent with [23]. There are two branches within the TCNN residual block. The first one transforms the input $\mathbf{x}$ through a series of stacked layers including two dilated causal convolution layers, while the other branch is the shortcut connection for the input $\mathbf{x}$.

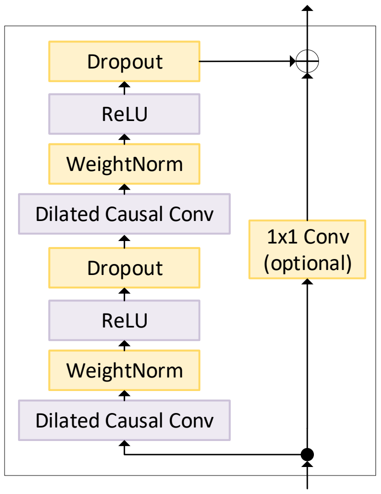


Fig. 2. Residual block in TCNN

However, the original input $\mathbf{x}$ and the output of the residual block $\mathcal{F}$ could have different widths, and the addition cannot be done. This can be rectified by using the 1×1 convolution layer on the shortcut branch to ensure the same widths.

In summary, TCNN has the following advantages compared to RNN, LSTM and GRU [23]:

1) Larger receptive field due to the dilated convolutions, allowing processing of longer input sequences. The receptive field size is flexible and can be easily adapted to different tasks by changing the number of layers and using different dilation factors.
2) More stable gradients due to the non-recurrent architecture (using a backpropagation path that is different that the sequential direction) and the use of residual blocks and shortcut connections.
3) Faster training as the input sequence can be processed at once, not sequentially as in recurrent networks, without the need to wait for the predictions of the previous timestamps to be completed before the current one. The convolutions in TCNN can be computed in parallel as each layer has the same filter.
4) Simpler and clearer architecture and operation, as it does not include gates and gating mechanisms.

IV. Methods Used for Comparison

A. Multi-layer Feedforward Neural Networks

Multi-layer feedforward neural networks (NNs) have been widely used for solar power forecasting [3], [5]–[9].

Most of these studies use the data from the previous day as input and predict the PV values for the next day simultaneously, using multiple neurons in the output layer.

In this study, we adopt a similar architecture but instead of constraining the input to a single day we allow using a sequence of previous days. The input of NN is a sequence of days $\mathbf{x}_1, ..., \mathbf{x}_d$, where each day d is represented as a vector of PV, weather and weather forecast data $\mathbf{x}_d = [PV_d; W1_d; WF_{d+1}]$, and the output of NN is the PV vector for the next day d+1: PV_{d+1}.

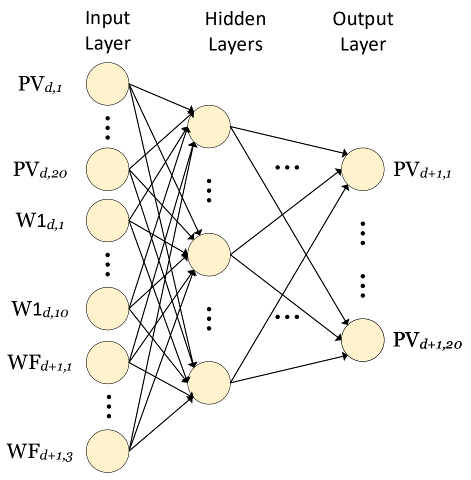


Fig. 3. NN architecture

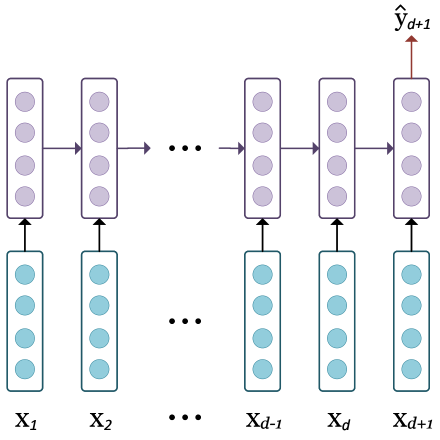


Fig. 4. RNN architecture

For example, Fig. 3 illustrates the NN structure for the Sanyo dataset with an input sequence length of 1 day. The input consists of 20 PV values for day d, 10 weather values for day d and 3 weather forecast values for day d+1, see Table II. NN has 20 output neurons corresponding to the 20 PV values for day d+1.

The best number of hidden layers and neurons in them was determined using the validation set, as discussed in Sec. V.

B. Recurrent Networks

RNNs are designed for processing sequential data. In contrast to the multi-layer feedforward networks, they have recurrent connections between the hidden neurons to keep information from previous time steps. This helps to find temporal associations between the current and previous data. RNNs can process sequences of any length.

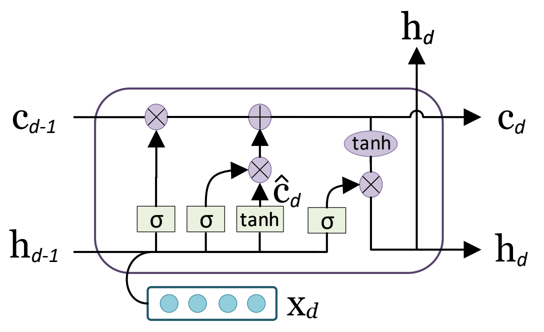


Fig. 5. LSTM cell architecture

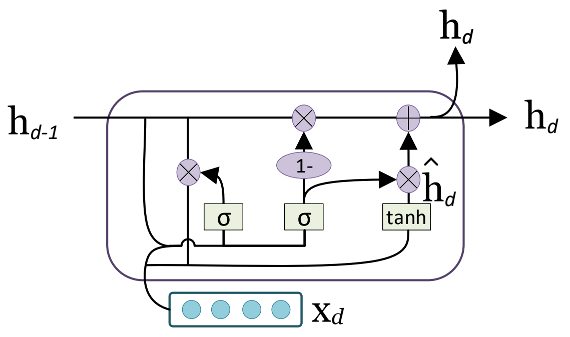


Fig. 6. GRU cell architecture

Theoretically, RNNs can capture information from many previous steps but in practice it is difficult to access this information due to the vanishing or exploding gradient problems [31], with the first one being more frequent. These gradient issues are usually more serious for RNN than for deep feedforward networks, because of the repeated multiplication of the same weight matrix during training in RNN [32]. To deal with the gradient problems, Pascanu et al. [33] proposed a modification in the standard stochastic gradient descent algorithm - the gradient clipping strategy which limits the norm of the gradient matrix.

In our implementation of RNN, we use the same input representation as in TCNN and predict all PV values for the next day simultaneously. Fig. 4 depicts the RNN structure with an input sequence of d days. The input is a sequence of feature vectors for previous days: $\mathbf{x}_1, ..., \mathbf{x}_d$ and the output is the PV vector for the next day $d+1$: $\hat{\mathbf{y}}_{d+1} = PV_{d+1}$. The feature vector $\mathbf{x_i}$ is constructed in the same way as in TCNN.

We employ the gradient clipping method to minimize the vanishing gradient problem and determine the number of hidden neurons by using the validation set as discussed in Sec. V. This applies to all prediction models, not only to RNN.

C. LSTM Networks

LSTM network [34] is an advanced version of RNN, designed to overcome the gradient problems. They have the same structure as RNN (see Fig. 4) but replace the RNN hidden layer with special units called memory cells. A memory cell has a state and three gates - input, output and forget. The memory cells store long-term information, and LSTM can erase, write and read information from the cells by controlling the input, output and forget gates.

Fig. 5 shows the architecture of a single LSTM cell at the time step for day d. This cell uses the input $\mathbf{x}_d$, the previous hidden state h_{d-1} and the past memory c_{d-1} to generate the new memory c_d which contains the information from the past sequences including $\mathbf{x}_d$. Specifically, the input gate controls which part of the new generated cell content $\hat{c}_d$ should be written to the output cell c_d; the forget gate controls which part of the past cell content c_{d-1} should be removed and the output gate controls which part of the new generated cell content $\hat{c}_d$ should be output to the new hidden state h_d.

Although the gating mechanism makes LSTM easier to preserve information from longer input sequences than RNN, the gradient still shrinks or explodes at a lower rate. The use of complex cells with gating mechanism increases the computation costs and makes LSTM difficult to interpret.

D. GRU Networks

The GRU network was proposed by Cho et al. [24] as a simpler alternative to LSTM.

As illustrated in Fig. 6, the GRU cell also employs a gating mechanism but does not have a cell state c_d as in LSTM - it uses the hidden state h_d to achieve the functionality of both the cell state c_d and hidden state h_d at the same time.

Different from LSTM, a GRU cell involves less computation and contains two gates only: the reset gate controls which part of the past hidden state is used to generate the new hidden state $\hat{h}_d$, and the update gate controls which part of the new generated hidden state $\hat{h}_d$ and past hidden state h_{d-1} is updated or preserved. Similarly to LSTM, GRU can easily retain information from the past sequences. Chung et al. [35] compared RNN, LSTM and GRU on music and speech signal processing tasks and found that LSTM and GRU performed similarly, and outperformed RNN.

E. Persistence Model

As a baseline, we developed a persistence prediction model B_{per} which considers the PV power output of day d as the forecast for day d+1.

V. Experimental Setup

All prediction models were implemented in Python 3.6 using the PyTorch 1.3.0 library.

A. Data Split

For both dataset, the PV power and corresponding weather data were split into two equal subsets: training and validation (the first year) and test (the second year). The first year data was further split into training set (70%) used for model training and validation set (30%) used for hyperparameter tuning.

TABLE III
SELECTED HYPERPARAMETERS FOR UQ DATASET

Model	Hidden layer size	Learning rate	Dropout rate	Gradient clip	Sequence length	Kernel size	Number of levels
NN	[35]	0.005	0.1	0	9	-	-
RNN	[40,30,25]	0.005	0.1	0.5	3	-	-
LSTM	[30]	0.005	0.1	0	3	-	-
GRU	[30]	0.005	0.1	0	9	-	-
TCNN	[25]	0.005	0.1	0.5	9	5	6

TABLE IV
SELECTED HYPERPARAMETERS FOR SANYO DATASET

Model	Hidden layer size	Learning rate	Dropout rate	Gradient clip	Sequence length	Kernel size	Number of levels
NN	[40,35,25]	0.005	0.1	0.5	1	-	-
RNN	[35]	0.01	0.1	0.5	1	-	-
LSTM	[40,30,25]	0.005	0.1	0	1	-	-
GRU	[40]	0.005	0.1	0.5	9	-	-
TCNN	[25]	0.01	0.1	0	9	5	6

B. Evaluation Measures

To evaluate the performance on the test set, we used the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE).

C. Tuning of Hyperparameters

The tuning of the hyperparameters was done using the validation set with grid search.

For NN, RNN, LSTM and GRU, the tunable hyperparameters and options considered were - hidden layer size: 1 layer with 25, 30, 35 and 40 neurons, 2 layers with 35 and 25, 40 and 25 neurons, 3 layers with 40, 30 and 25, 40, 35 and 25 neurons; learning rate: 0.005 and 0.01; dropout rate: 0.1 and 0.2; gradient clip norm threshold: 0 (disabled) and 0.5; sequence length: 1, 3, 6, 9, 12, 15 and 18; batch size: 64; number of epochs: 120. The activation functions were - for NN: ReLu for the hidden layers and linear for the output layer; for RNN, LSTM and GRU - tanh for the hidden layers and linear for the output layer.

For TCNN, the tunable hyperparameters and options considered were - hidden layer size: 35 and 25 neurons; convolutional kernel size: 3, 5 and 7; number of levels: 3, 4 and 6; learning rate: 0.005 and 0.01; dropout rate: 0.1 and 0.2; gradient clip norm threshold: 0 (disabled) and 0.5; sequence length: 1, 3, 6, 9, 12, 15 and 18; batch size: 64; number of epochs: 120; activation functions: ReLu for the hidden layers and linear for the output layer.

For all models, the training algorithm was the mini-batch gradient descent with Adam optimization and gradient clipping, with MAE as a loss function. The weight initialization mode was set to normal and the initial learning rates of 0.01 and 0.005 were annealing by a multiplication factor of 0.5 and 0.8, respectively, every 15 epochs.

TABLE V
ACCURACY OF ALL MODELS

	UQ dataset		Sanyo dataset	
Method	MAE (kW)	RMSE (kW)	MAE (kW)	RMSE (kW)
B_{per}	124.804	184.293	0.749	1.252
NN	75.203	101.691	0.564	0.754
RNN	74.691	102.991	0.511	0.723
LSTM	74.270	100.982	0.525	0.726
GRU	74.401	101.194	0.559	0.742
TCNN	**70.015**	**98.118**	**0.510**	**0.721**

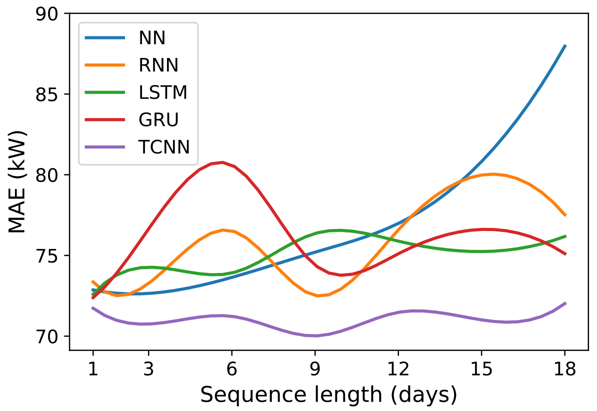
The selected best hyperparameters for all models are listed in Tables III and IV for the two datasets, and were used for the evaluation on the testing set.

VI. RESULTS AND DISCUSSION

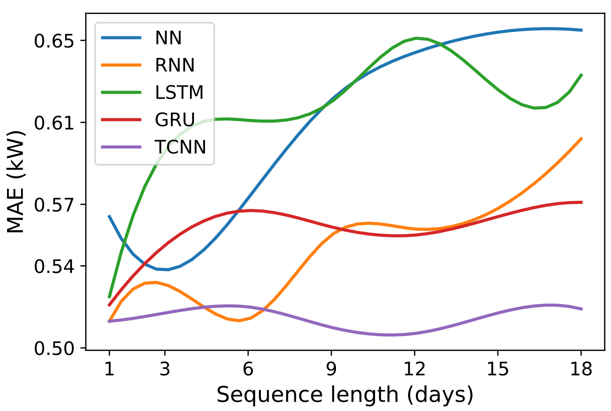
A. Overall Performance

Table V shows the MAE and RMSE results of all models for the UQ and Sanyo datasets. The main results can be summarized as follows:

- TCNN is the most accurate prediction model, and NN is the least accurate model on both datasets.
- TCNN outperforms all recurrent networks. An important advantage of TCNN is its simpler architecture and operation as it doesn't involve gating mechanisms.
- Comparing the three recurrent models: on the UQ dataset, LSTM and GRU perform similarly and better than RNN, while on the Sanyo dataset RNN is the best recurrent model followed by LSTM and GRU. Considering the results on both datasets, overall LSTM is the best recurrent network.
- On both datasets, the best TCNN structure is the same: sequence length l = 9 days, hidden layers = 6, hidden layer size = 25, convolutional kernel size = 5 and number of residual blocks = 6. This architecture may be a good



(a)



(b)

Fig. 7. Performance of all models on: (a) UQ dataset and (b) Sanyo dataset

starting point for exploring TCNNs for 1-day ahead solar power forecasting.

- Regarding the input sequence length, Table III shows that both RNN and LSTM selected the same short sequence as optimal (1 day for the UQ dataset and 3 days for the Sanyo dataset), while GRU and TCNN selected long sequences - 9 days for both datasets. The short sequence length for LSTM is consistent with the results in [22].
- Although TCNN and GRU use the same sequence length, TCNN performs much better than GRU, which shows the advantages of TCNN for effectively using long input sequences.
- All neural network models outperform the persistence model used as a baseline B_{per}.

B. Input Sequence Length and Accuracy

We also investigated the effect of the input sequence length l on the accuracy. The results are shown in Fig. 7 and can be summarized as follows:

- TCNN outperforms all other prediction models for all sequence lengths, except for one case: l = 6 on the Sanyo dataset, where RNN is the best prediction model.
- The best results are achieved by TCNN for l = 9 for the UQ dataset and l = 12 for the Sanyo dataset.
- The performance of TCNN is relatively stable for all sequence lengths, while the performance of the other prediction models varies considerably and generally decreases with increasing the sequence length.
- The NN accuracy on both datasets decreases significantly as the length of the input sequence increases. The best results are achieved for l = 1-3 for the UQ dataset l = 3 for the Sanyo dataset. This is consistent with the results of Torres et al. [8] who found that NN performs better with shorter sequences. However, we note that it may be possible to improve the results for the longer input sequences by feature selection and dimensionality reduction before the NN training.
- The trend for RNN is also a decrease in accuracy as the length of the input sequence increases, although less drastic than for NN. A possible explanation is that the gradient problems in RNN become more serious with longer sequences, due to the repeated multiplication by the same weight matrix. Hence, it is harder for RNNs to learn the recurrent matrix with longer sequence although the longer sequence could provide more information.
- LSTM also shows a decreasing accuracy on both datasets as l increase. The accuracy is more stable on the UQ dataset but highly fluctuating on the Sanyo dataset with a sharp decrease in accuracy for $l > 3$ and especially for l = 12.
- GRU shows a decreasing accuracy on the Sanyo dataset and more fluctuating performance on the UQ dataset with low accuracy for l = 3-9 but a slight improvement for $l > 12$.

Hence, the results show that in contrast to the other neural network models, TCNN is able to maintain a much longer effective history which is evident by its relatively stable and accurate performance for all sequence lengths. TCNN is able to extract informative features from long sequences effectively with the dilated causal convolutional filters. The use of residual blocks helps to preserve the input information and deal with the gradient problems.

VII. Conclusion

In this paper, we studied the application of TCNN, a novel convolutional architecture for time series data, for forecasting the PV power generation for the next day. We compared the performance of TCNN with multi-layer feedforward neural networks and three recurrent neural networks - RNN, LSTM and GRU, on two Australian solar power datasets, containing data from three sources - historical solar and weather data, and weather forecast data for future days. Our results indicate that TCNN was the most accurate model, followed by the recurrent architectures and finally the feedforward neural network. Compared to the recurrent models, TCNN offers another advantage - it is simpler as it doesn't involve gating. We also investigated the effect of the sequence length on the accuracy and found that TCNN outperforms the other models for all sequence sizes and is able to maintain a longer effective history. The best results for TCNN were achieved for sequence of length 9 and 12. Hence, we conclude that temporal convolutional architectures such as TCNN are promising methods for solar power forecasting and should be investigated further.

References

[1] C. B. Martinez-Anido, B. Botor, A. R. Florita, C. Draxl, S. Lu, H. F. Hamann, and B.-M. Hodge, "The value of day-ahead solar power forecasting improvement," *Solar Energy*, vol. 129, pp. 192 – 203, 2016.

[2] Australian Energy Market Operator. Introduction to Australia's National Electricity market. http://www.abc.net.au/mediawatch/transcripts/1234_aemo2.pdf, Last accessed on 2020-01-06.

[3] H. T. Pedro and C. F. Coimbra, "Assessment of forecasting techniques for solar power production with no exogenous inputs," *Solar Energy*, vol. 86, no. 7, pp. 2017 – 2028, 2012.

[4] H. Long, Z. Zhang, and Y. Su, "Analysis of daily solar power prediction with data-driven approaches," *Applied Energy*, vol. 126, pp. 29 – 37, 2014.

[5] Z. Wang, I. Koprinska, and M. Rana, "Solar power prediction using weather type pair patterns," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 4259–4266.

[6] M. Rana and A. Rahman, "Multiple steps ahead solar photovoltaic power forecasting based on univariate machine learning models and data re-sampling," *Sustainable Energy, Grids and Networks*, vol. 21, p. 100286, 2020.

[7] Y. Chu, B. Urquhart, S. M. Gohari, H. T. Pedro, J. Kleissl, and C. F. Coimbra, "Short-term reforecasting of power output from a 48 mwe solar PV plant," *Solar Energy*, vol. 112, pp. 68 – 77, 2015.

[8] J. Torres, A. Troncoso, I. Koprinska, Z. Wang, and F. Martínez-Álvarez, "Big data solar power forecasting based on deep learning and multiple data sources," *Expert Systems*, vol. 36, no. 4, p. e12394, 2019.

[9] Y. Lin, I. Koprinska, M. Rana, and A. Troncoso, "Pattern sequence neural network for solar power forecasting," in *International Conference on Neural Information Processing (ICONIP)*, 2019.

[10] Z. Wang and I. Koprinska, "Solar power prediction with data source weighted nearest neighbors," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2017.

[11] J. Shi, W. Lee, Y. Liu, Y. Yang, and P. Wang, "Forecasting power output of photovoltaic systems based on weather classification and support vector machines," *IEEE Transactions on Industry Applications*, vol. 48, no. 3, pp. 1064–1069, 2012.

[12] Z. Dong, D. Yang, T. Reindl, and W. M. Walsh, "A novel hybrid approach based on self-organizing maps, support vector regression and particle swarm optimization to forecast solar irradiance," *Energy*, vol. 82, pp. 570 – 577, 2015.

[13] M. Rana, I. Koprinska, and V. G. Agelidis, "2D-interval forecasts for solar power production," *Solar Energy*, vol. 122, pp. 191 – 203, 2015.

[14] M. Rana, I. Koprinska, and V. Agelidis, "Forecasting solar power generated by grid connected PV systems using ensembles of neural networks," in *International Joint Conference on Neural Networks (IJCNN)*, 2015.

[15] Z. Wang and I. Koprinska, "Solar power forecasting using dynamic meta-learning ensemble of neural networks," in *Proceedings of the International Conference on Artificial Neural Networks and Machine Learning (ICANN)*, 2018.

[16] I. Koprinska, M. Rana, and A. Rahman, "Dynamic ensemble using previous and predicted future performance for multi-step-ahead solar power forecasting," in *International Conference on Artificial Neural Networks and Machine Learning (ICANN)*, 2019, pp. 436–449.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[18] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[19] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv: 1609.03499*, 2016.

[20] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2017.

[21] M. Bińkowski, G. Marti, and P. Donnat, "Autoregressive convolutional neural networks for asynchronous time series," in *Proceedings of the Time Series Workshop at International Conference on Machine Learning (ICML)*, 2017.

[22] I. Koprinska, D. Wu, and Z. Wang, "Convolutional neural networks for energy time series forecasting," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2018.

[23] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv: 1803.01271*, 2018.

[24] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-Decoder approaches," *arXiv preprint arXiv: 1409.1259*, 2014.

[25] The University of Queensland. UQ Solar. https://solar-energy.uq.edu.au/, Last accessed on 2019-05-28.

[26] DKA Solar Centre. Sanyo, 6.3kW, HIT Hybrid Silicon, Fixed, 2010. http://dkasolarcentre.com.au/source/alice-springs/dka-m4-b-phase, Last accessed on 2019-08-18.

[27] Australian Bureau of Meteorology. Climate Data Online. http://www.bom.gov.au/climate/data/, Last accessed on 2019-05-28.

[28] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[29] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv: 1511.07122*, 2015.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv: 1512.03385*, 2015.

[31] 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.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[33] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *arXiv preprint arXiv: 1211.5063*, 2012.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[35] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv: 1412.3555*, 2014.