Temporal Convolutional Attention Neural Networks for Time Series 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*—Temporal Convolutional Neural Networks (TCNNs) have been applied for various sequence modelling tasks including time series forecasting. However, TCNNs may require many convolutional layers if the input sequence is long and are not able to provide interpretable results. In this paper, we present TCAN, a novel deep learning approach that employs attention mechanism with temporal convolutions for probabilistic forecasting, and demonstrate its performance in a case study for solar power forecasting. TCAN uses the hierarchical convolutional structure of TCNN to extract temporal dependencies and then uses sparse attention to focus on the important timesteps. The sparse attention layer of TCAN enables an extended receptive field without requiring a deeper architecture and allows for interpretability of the forecasting results. An evaluation using three large solar power data sets demonstrates that TCAN outperforms several state-of-the-art deep learning forecasting models including TCNN in terms of accuracy. TCAN requires less number of convolutional layers than TCNN for an extended receptive field, is faster to train and is able to visualize the most important timesteps for the prediction.**

***Index Terms*—time series forecasting, deep learning, temporal convolutional neural network, sparse attention**

I. Introduction

Time series forecasting is an essential task in many areas, e.g. in industry - predicting electricity demand, in finance - predicting stock prices and exchange rate, in retail - predicting sales, supply and demand, in health - predicting immune response, disease progression and hospital length of stay.

Statistical methods such as linear regression, ARIMA and exponential smoothing [1] are well-established and widely used by industry forecasters. However, they require domain knowledge for model selection, fit each time series independently and are not able to infer shared patterns from related time series [2], [3].

On the other hand, deep learning methods have been increasingly applied for time series forecasting, showing very promising results. They are able to learn from raw data with less domain knowledge and feature engineering, and can extract complex patterns, including shared patterns, from related time series. For example, Salinas et al. [4] proposed DeepAR, a probabilistic forecasting model based on Long Short Term Memory (LSTM) networks. The Transformer architecture [3] is a new sequence model that uses only attention mechanism for data processing, without any recurrent or convolutional layers, and can access any part of the historical sequence regardless of temporal distance. Li et al. [5] proposed the LogSparse Transformer, which solves the locality-agnostics and memory bottleneck problems of the Transformer.

Another prominent class of deep learning methods, convolutional neural networks, have also been applied for time series forecasting. They are attractive due to their ability to represent repeated patterns in the time series via convolutional filters and extract useful features from raw data without prior knowledge or feature engineering [6]–[8].

Temporal Convolutional Neural Networks (TCNNs) [9] are specifically designed for sequence modeling tasks and have been applied for different types of data - image, language and music [9], solar power forecasting [8], retail, electricity and traffic [10]. The probabilistic TCNNs were proposed in [10].

TCNN is a hierarchical architecture consisting of several convolutional layers. It uses casual convolutions, dilated convolutions and residual connections to enable a larger receptive field, reduce the unstable gradient problem and boost training speed [9]. However, when the input sequence is long, TCNN may need many temporal convolutional layers in order to have a sufficiently large receptive field that covers the input sequence. In addition, TCNN is a black-box architecture, not able to provide interpretable results. Many applications of time series forecasting involve critical decisions; providing explanations improves the confidence of decision makers and is hence highly desirable. In this paper, we present a new approach to address these issues.

The contributions of our work are as follows:

1) We propose Temporal Convolutional Attention Neural Network (TCAN), which employs convolutional architecture and attention mechanism. TCAN learns the temporal dependency via hierarchical convolutional architecture and generates forecasting results with a sparse attention layer. The use of sparse attention layer enables TCAN to access all historical input steps regardless of the sequence length, to focus on the most important timesteps from the input sequence and to provide visualization of the results for interpretability.
2) We evaluate TCAN for time series forecasting on three real-world solar power data sets (two sets contain multivariate series and one set contains related series). The

results show that TCAN outperforms the state-of-the-art deep learning models DeepAR, LogSparse Transformer, N-BEATS and TCNN, and a persistence baseline. The attention mappings visualize the important timesteps for interpretability of the results and demonstrate that accessing longer previous history is important. Our results show that TCAN with the sparse attention layer requires less number of convolutional layers than TCNN for an extended receptive field and performs better than TCNN in terms of accuracy and training speed.

II. Case Study: Solar Power Forecasting

As a case study we consider solar power forecasting: given an input sequence of previous PV solar power generation data (e.g. hourly or half-hourly), predict the solar power for the next time step.

Solar power forecasting is needed for optimal scheduling of generators and for the integration of solar into the electricity grid. The penetration of solar energy into the electricity grid is rapidly increasing but since solar has a variable and intermittent nature, there is a need for accurate forecasting in order to to ensure stability and efficient operation of the electricity grid.

A. Data

We use three publicly available data sets: Sanyo [11], Hanergy [12] and Solar [13].

Sanyo and Hanergy contain solar power generation data from two PV plants in Australia - from 1/2011 to 12/2016 (6 years) for Hanergy and and 1/2011 to 12/2017 (7 years) for Sanyo. Only the data between 7am and 5pm was considered and it was aggregated at half-hourly intervals. For both datasets, weather and weather forecast data was also collected (see [14] for more details) and used as covariates.

Solar contains solar power data from 137 PV plants in Alabama, USA, from 01/2006 to 08/2006. The Solar data is aggregated into 1-hour intervals.

Following [5], [14], calendar features were also added according to the granularity of the datasets: Sanyo and Hanergy use *month, hour-of-the-day and minute-of-the-hour*, and Solar uses *month, hour-of-the-day and age*.

All data was normalized to have zero mean and unit variance.

B. Problem Statement

Given is:

1) a set of N univariate time series $\{\mathbf{Y}_{i,1:T_l}\}_{i=1}^{N}$ (PV solar in our case study, N=1 for Sanyo and Hanergy and N=137 for Solar), where $\mathbf{Y}_{i,1:T_l} \triangleq [\mathrm{y}_{i,1}, \mathrm{y}_{i,2}, ..., \mathrm{y}_{i,T_l}]$, T_l is the input sequence length, and $\mathrm{y}_{i,t} \in \Re$ is the ith PV power generated at time t;
2) a set of associated time-based multi-dimensional covariate vectors $\{\mathbf{X}_{i,1:T_l+T_h}\}_{i=1}^{N}$, where T_h is the length of forecasting horizon. For our case study, the covariates for the Sanyo and Hanergy datasets include: weather $\{\mathbf{W1}_{i,1:T_l}\}_{i=1}^{N}$, weather forecasts $\{\mathbf{WF}_{i,T_l+1:T_l+T_h}\}_{i=1}^{N}$ and calendar features $\{\mathbf{Z}_{i,1:T_l+T_h}\}_{i=1}^{N}$, while the covariates for the Solar dataset include calendar features only.

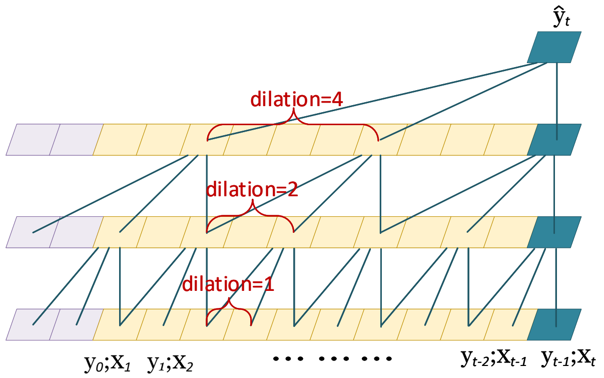


Fig. 1. TCNN architecture

Task: Predict $\{\mathbf{Y}_{i,T_l+1:T_l+T_h}\}_{i=1}^{N}$, the PV power for the next T_h time steps after T_l.

The input of TCAN at step t is the concatenation of $\mathbf{y}_{i,t-1}$ and $\mathbf{x}_{i,t}$. TCAN produces the probability distribution of future values, given the past history:

$$p\left(\mathbf{Y}_{i,T_l+1:T_l+T_h} \mid \mathbf{Y}_{i,1:T_l}, \mathbf{X}_{i,1:T_l+T_h}; \Phi\right) = \prod_{t=T_l+1}^{T_l+T_h} p\left(\mathbf{y}_{i,t} \mid \mathbf{Y}_{i,1:t-1}, \mathbf{X}_{i,1:t}; \Phi\right) \quad (1)$$

where Φ denotes the parameters of TCAN.

Note that the subscript i is omitted in the rest of the paper for simplicity.

III. Background

A. Temporal Convolutional Neural Network

Bai et al. [9] proposed a generic architecture of TCNN, which is informed by CNN architectures for sequential data such as WaveNet [15] but is specifically designed to be simpler and to combine autoregressive prediction with a long memory. TCNN is a hierarchical architecture, consisting of several convolutional hidden layers with the same size as the input layer, as shown in Fig. 1. It is designed to process data element by element.

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

Causal convolutions. The output at time t is convolved only with elements from time t or earlier time steps from the previous layer. This concept has been used in Waibel's time-delay network [16] and the WaveNet architecture [15]. Zero padding is used in hidden layers 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 [15], [17] to enable large receptive fields, and consequently to capture a long memory, which is not possible with causal convolutions alone as they require a very deep NN.

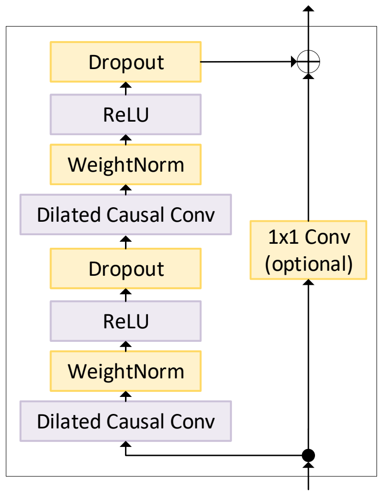


Fig. 2. TCNN's residual block

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

$$F(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{2}$$

where $f : \{0, \dots, k-1\} \rightarrow \mathbb{R}$ is the convolution filter, x is the sequential input (concatenation of the solar time series and covariates for our case study), k is the filter size, and d is the dilation factor.

The convolution kernel remains the same for all layers but the dilation factor increases exponentially with the depth of the network: $d_l = 2^l$, where l is the network level. For example, as shown in Fig. 1, d_1 is 1 at the first layer (corresponding to regular convolutions) and then increases at each layer, reaching 4 at the last hidden layer. This pyramidal structure and aggregation mechanism effectively increases the receptive field of TCNN, allowing to cover a long input sequence.

Residual connections. Residual blocks [18] help to overcome the gradient vanishing problem in networks with many layers. The main idea is to add the input x to a block of stacked layers (a series of transformation $\mathcal{F}$) to the output of this block by using shortcut connections:

$$o = \sigma(x + \mathcal{F}(x)) \tag{3}$$

where σ is the activation function.

Fig. 2 illustrates a residual block of TCNN [9]. There are two branches - the first one transforms the input x through a series of stacked layers including two dilated causal convolution layers, while the second one is the shortcut connection for the input x. 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 this work, we implemented an autoregressive TCNN [9]. Given an input sequence with pre-defined size (input window), it predicts the next value, then adds the predicted value to the input window and shifts the input window with one step, makes the next prediction and so on until all values from the forecasting horizon are predicted.

In summary, compared to recurrent architectures such as RNN, LSTM and GRU, TCNNs have the advantage of larger receptive field size, more stable gradients and parallelism [9]. However, although the use of dilated convolutions enables larger receptive fields, this also requires more temporal convolutional layers to cover long input sequences, which makes the architecture more complex and slower to train, and may also lead to overfitting. In this paper we propose TCAN, which allows to access all input timesteps without increasing the number of temporal convolutional layers, and preserves the other advantages of TCNN.

B. Attention Mechanism

The attention mechanism [19] was initially proposed for sequence-to-sequence (seq2seq) tasks in natural language processing but since then has been successfully used in other domains as well. It is applied in an encoder-decoder framework and allows to automatically identify the parts of the encoder input sequence that are important for the decoder outputs.

In the seq2seq framework, the encoder and decoder take the sequential steps as input and generate the hidden state $h_t \in \Re^{1 \times d_{hid}}$ at each step, where d_{hid} is the hidden layer size. Soft attention takes as input the encoder hidden states $h_{1:T_l}$ and the decoder hidden state h_t at time step t, and generates a context vector c_i by calculating their dot product:

$$c_i = h_{1:T_l} \cdot h_t^T \tag{4}$$

Then, the context vector is normalised by the softmax function to produce attention weights. The weight a_i of each encoder hidden state h_i is given by:

$$a_i = \frac{\exp(c_i)}{\sum_{j=1}^{T_l} \exp(c_i)} \tag{5}$$

The weight represents how important the encoder step i is to the decoder output at step t.

Finally, the attention layer output is computed by the dot product between the attention weights $a_{1:T_l}$ and the encoder hidden states $h_{1:T_l}$. The weighted output is concatenated with the decoder hidden state to generate the decoder output.

Intuitively, the attention mechanism helps the decoder to pay attention to the parts of the historical sequence steps that are important regardless of the length of input steps and thus overcomes the limitation of seq2seq framework to encode a whole sequence into a single fixed-size vector. The use of attention helps the seq2seq model to have better performance when processing longer sequences.

However, attention using softmax always results in positive attention weights for all timesteps, including the irrelevant steps which can be harmful to long sequences [20], [21]. To overcome this issue, recent studies have proposed sparse attention mechanisms to learn sparse attention mappings [5], [20]–[22]. The sparse approach shows advantages over dense attention on multiple sequential modelling tasks in term of accuracy and also interpretability of the results as it tends to generate less scattered attention maps [5].

IV. Temporal Convolutional Attention Neural Networks

A. Motivation and Novelty

TCAN aims to:

1) Enable large receptive fields without increasing the number of convolutional layers.
2) Focus on the timesteps from the input sequence that are important for prediction and to ignore the irrelevant ones to improve accuracy.
3) Provide visualization of the most relevant timesteps to facilitate understanding and interpretability of the results.

Although TCNN [8]–[10] uses exponentially dilated convolution to extend the receptive field, if the input sequence is long, it may need many convolutional layers which increases the complexity and training time.

Given a TCNN with n_L temporal convolutional layers, convolutional filters of size k and dilation factor $d_l = 2^l$, the effective history of layer l ($\{l \in \mathbb{Z} : 0 \leq l \leq n_L\}$) that is used by TCNN to generate predictions is $(k-1) \times d_l$ [9].

The number of historical input steps that TCNN could use to make predictions is the sum of the effective histories of all convolutional layers: $\sum_{l=0}^{n_L-1} (k-1) \times d_l$.

It can be shown that to have a receptive field covering an input sequence with length T_l, TCNN needs at least $n_L = \lceil log_2(\frac{T_l}{k-1} + 1) \rceil$ convolutional layers:

$$\begin{aligned}
\sum_{l=0}^{n_L-1} (k-1) \times d_l &\geq T_l \\
\sum_{l=0}^{n_L-1} (k-1) \times 2^l &\geq T_l \\
\sum_{l=0}^{n_L-1} 2^l &\geq \frac{T_l}{k-1} \\
2^{n_L} - 1 &\geq \frac{T_l}{k-1} \\
2^{n_L} &\geq \frac{T_l}{k-1} + 1 \\
n_L &\geq log_2(\frac{T_l}{k-1} + 1)
\end{aligned} \tag{6}$$

In our case study for solar power forecasting, the length of the input sequence (previous history) is 20 or 24 steps per day, and we use a small kernel with a filter size of $k = 3$ to focus on the local content [9]. This means that a TCNN with at least 4 convolutional layers ($n_L = \lceil log_2(\frac{24}{3-1} + 1) \rceil = 4$) is needed to take into account all input steps. When the size of the input sequence increases (e.g. due to a different data granularity or the need to use more previous timesteps), more convolutional layers will be needed. This increases the complexity of the architecture, may lead to overfitting and also increases the training time.

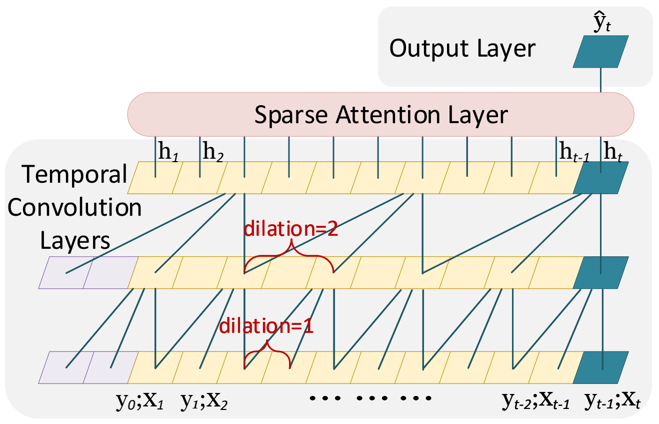


Fig. 3. TCAN architecture

Below we present our proposed approach, TCAN, which uses attention mechanism to enable an extended receptive field without adding more temporal convolutional layers.

The attention mechanism also allows TCNN to focus on the important input timesteps and ignore the irrelevant ones when making predictions which may improve accuracy. In addition, the attention layer is used to provide visualization of the most relevant timesteps for the prediction of each instance. This enables interpretability and justification of the results which is important for real-world time series forecasting applications.

B. Model Architecture

As illustrated in Fig. 3, TCAN consists of three parts: 1) *temporal convolution layers*, 2) *sparse attention layer* and 3) *output layer*.

TCAN employs hierarchical convolutional architecture to encode the input sequence and extract temporal pattern as latent variables. The latent variables are then used by an attention mechanism to learn the most relevant features (previous timesteps) and generate the final prediction. The latent variables encode information from the whole input window, enabling large receptive fields without adding more convolutional layers. In addition, the attention mechanism allows to visualise the most important timestamps for each instance to facilitate understanding and interpretability of the results.

TCAN uses the benefits of sparse attention mechanism to: 1) access all previous timesteps without the requirement of a deep architecture and 2) focus on the important input timesteps and ignore the irrelevant ones when making predictions, 3) provide visualization of the most relevant timesteps for the prediction.

Temporal Convolution Layers. Firstly, TCAN extracts the temporal latent factors $h_{t-T_l:t}$ via the multiple dilated temporal convolutional layers (TC) from the historical data within the input window T_l ($y_{t-T_l:t}, x_{t-T_l:t}$) as:

$$h_{t-T_l:t} = \text{TC}(y_{t-T_l:t}, x_{t-T_l:t}) \tag{7}$$

where the extracted latent factors encode all information of the input sequence within the rolling window.

Sparse Attention Layer. The sparse attention layer takes the temporal latent factors ($h_{t-T_l:t}$) as input and generates the attention vector ($\tilde{h}_t$) that is used to make the prediction. Standard attention scores used in Transformer and RNN architectures are computed by the softmax function. However, softmax never assigns a probability of zero to any previous timesteps, so it never fully rules out the unimportant parts of the input sequence [21]. In sequence modelling tasks, including solar power forecasting, the future timestep is typically strongly related to a few historical timesteps and it is desirable to increase the focus on them. For example, the solar power at step t is more related to the solar power at the previous hours on the same day and the same time on the previous day (step $t - T_l$) rather than the other timesteps. This is supported by Fig. 4 which shows the partial aurocorrelation plot of the solar series for all data sets for 30 lags; we can see two strong linear dependencies: the first is at lag 1 and the second is at lag 20 for Sanyo and Hanergy and lag 24 for Solar.

Recent studies have developed variations such as sparse attention which increase the focus on the most relevant input timesteps. Specifically, we applied α-entmax attention [21], defined as:

$$\begin{aligned} &a_{t-T_l:t-1} \\ =&\alpha - \text{entmax}(h_{t-T_l:t-1} \cdot h_t^T) \\ =&\,\text{ReLU}((\alpha - 1) \times (h_{t-T_l:t-1} \cdot h_t^T) - \tau\mathbf{1}))^{1/\alpha - 1} \end{aligned} \tag{8}$$

where τ is the Lagrange multiplier, $\mathbf{1}$ is the all-one vector and α is the hyperparameter. α-entmax maps latent variables into sparse attention scores $a_{t-T_l:t-1} \in \Re^{T_l \times 1}$ base on the dot product similarity between the historical steps ($h_{t-T_l:t-1} \in \Re^{T_l \times d_{hid}}$) and current step ($h_t \in \Re^{1 \times d_{hid}}$).

Note that α-entmax is equivalent to using softmax when $\alpha = 1$ and sparsemax when $\alpha = 2$ [20]. In TCAN we set α to 1.5 for a balance between softmax and sparsemax as in [21].

Then we employ concatenation to combine the information from the attention context vector c_t and target hidden state h_t to produce the attention vector $\tilde{h}_t$. The context vector is the dot product between the attention score and hidden states of the historical steps and could be considered as the weighted sum of hidden states h_t:

$$\tilde{h}_t = [c_t \circ h_t] = [(a_{t-T_L:t-1}^T \cdot h_{t-T_L:t-1}) \circ h_t] \tag{9}$$

Output Layer. The output layer uses the attention vectors to make the final predictions. In this work, we consider data is distributed in Gaussian distribution, which is commonly used in real-world time series modelling [4]. We transfer the attention vector as the forecasting results including the mean and variance of the distribution, as illustrated in Eq. (10) and (11). In Eq. (11), the softplus function guarantees that the variance is always positive.

$$\hat{y}_t = \text{linear}(\tilde{h}_t)) \tag{10}$$

$$\begin{aligned} \sigma_t^2 =& \,\text{softplus}(\text{linear}(\tilde{h}_t)) \\ =& \log(1 + \exp(\text{linear}(\tilde{h}_t))) \end{aligned} \tag{11}$$

Eq. (10) and (11) form the Gaussian distribution $\mathcal{N}(\hat{y}_t,\, \sigma_t^2)$, and predictions could be sampled from the distribution. The ρ-quantile output are generated via the inverse cumulative probability distribution: $\hat{y}_t = F_t^{-1}(\rho)$.

Loss Function. Finally, the parameters of TCAN are optimised by minimising the loss function shown in Eq. (12) below, where $\hat{y}_t$ is the point forecast. To provide both accurate point and probabilistic forecasts, it combines the Mean Absolute Error (MAE) and the Negative Log-Likelihood (NLL) using the regularization parameter a. Higher a increases the weight of the probabilistic forecast; we used $a = 0.5$.

$$\begin{aligned} &L(\hat{y}_{T_l+1:T}, \sigma^2_{I_{T_l+1:T}}, y_{T_l+1:T}, a) \\ &= a \times \text{NLL}(\hat{y}_{T_l+1:T}, \sigma^2_{I_{T_l+1:T}}, y_{T_l+1:T}) \\ &\quad + \text{MAE}(\hat{y}_{T_l+1:T}, y_{T_l+1:T}) \\ &= -\frac{a}{2T_h} \times \Big(T_h \log(2\pi) + \sum_{t=T_l+1}^{T} \log\left|\sigma^2_{I_t}\right| \\ &\quad + \sum_{t=T_l+1}^{T} (y_t - \hat{y}_t)^2 \sigma_{I_t}^{-2}\Big) + \frac{1}{T_h}\sum_{t=T_l+1}^{T} |y_t - \hat{y}_t| \end{aligned} \tag{12}$$

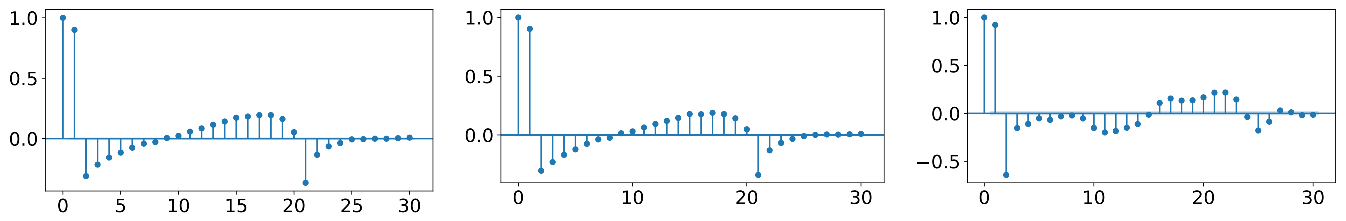
In summary, the information flow in TCAN includes several temporal convolution layers, a sparse attention layer and an output layer; the network is trained end-to-end optimizing the loss function from Eq. (12).

V. Experimental Setup

A. Methods Used for Comparison

We compare the performance of TCAN with five state-of-the-art deep learning models (DeepAR, N-BEATS-G, N-BEATS-I, LogSparse Transformer and TCNN) and a persistence model.

- DeepAR [4] is a widely used sequence-to-sequence probabilistic forecasting model.
- N-BEATS [2] is based on backward and forward residual links and stacks of fully connected layers. N-BEATS-G provides generic forecasting results, while N-BEATS-I provides interpretable results by decomposing the time series into trend and seasonality. We introduced covariates to N-BEATS at the input of each block to facilitate multivariate series forecasting.
- LogSparse Transformer [5] is a recently proposed variation of the Transformer architecture for time series forecasting. It is denoted as "LS Transformer" in Table II.
- TCNN [8]–[10] is a novel convolutional architecture and has been successfully applied to solar power forecasting [8]. As we consider probabilistic forecasts, we selected the autoregressive probabilistic TCNN from [10] for the comparison and used a fixed length input sequence. TCNN-3 and TCNN-4 correspond to TCNN with 3 and 4 temporal convolutional layers respectively.
- Persistence is a typical baseline in forecasting. It considers the time series of the previous day as the prediction for the next day.



(a) (b) (c)

Fig. 4. Partial autocorrelation for (a) Sanyo, (b) Hanergy and (c) Solar.

B. Data Split and Hyperparameter Tuning

All models were implemented with PyTorch 1.6 on Tesla P100 16GB GPU under Linux environment. The deep learning models were optimized by mini-batch gradient descent with the Adam optimizer and a maximum number of epochs 200. We used Bayesian optimization for hyperparameter search with a maximum number of iterations of 20. The models used for comparison were tuned based on the recommendations of the authors in the papers. The hyperparameters which have obtained a minimum loss on the validation set were selected and used to evaluate the performance on the test set.

Following the experimental setup in [14] and [5], we used the following training, validation and test split: for Sanyo and Hanergy - the data from the last year as test set, the second last year as validation set for early stopping and the remaining data (5 years for Sanyo and 4 years for Hanergy) as training set; for Solar - the last week data as test set (from 25/08/2006), the week before as validation set. For all data sets, the data preceding the validation set is split in the same way into three subsets and the corresponding validation set is used to select the best hyperparameters.

For TCAN, the learning rate λ is fixed to 0.005 for all data sets, the batch size n_{batch} is 256 for Sanyo set and 512 for Hanergy and Solar sets, the regularization parameter a and the entmax attention α are set to 0.5 and 1.5 respectively, the dropout rate δ is chosen from $\{0,\ 0.1,\ 0.2\}$ and the kernel size d_k from $\{3,\ 4\}$; the convolutional layer size d_{hid} and the number of convolutional layers are chosen from the descent-sorted permutations of $\{20,\ 16,\ 12,\ 8,\ 6,\ 4\}$ and $\{2,\ 3\}$.

For TCNN, all settings are the same as for TCAN except that we consider both TCNN with 3 and 4 convolutional layers to allow for comparable receptive fields with TCAN. The selected best hyperparameters for TCAN and TCNNs with 3 (TCNN-3) and 4 (TCNN-4) layers are listed in Table I and used for the evaluation on the test set.

C. Evaluation Measures

Following [4], [23], we report the standard ρ0.5 and ρ0.9-quantile losses. Note that ρ0.5 is equivalent to the Mean Absolute Percentage Error (MAPE) [24]. Given the ground truth y and ρ-quantile of the predicted distribution $\hat{y}$, the ρ-quantile loss is given by:

$$\mathrm{QL}_\rho(y,\hat{y}) = \frac{2 \times \sum_t P_\rho\left(y_t,\hat{y}_t\right)}{\sum_t |y_t|}, \qquad P_\rho(y,\hat{y}) = \begin{cases} \rho(y-\hat{y}) & \text{if } y > \hat{y} \\ (1-\rho)(\hat{y}-y) & \text{otherwise} \end{cases} \tag{13}$$

TABLE I
HYPERPARAMETERS FOR TCAN AND TCNN

		δ	d_{hid}	d_k
TCNN-3:	Sanyo	0.1	[20,16,8]	3
	Hanergy	0.1	[16,12,6]	3
	Solar	0.1	[16,12,8]	3
TCNN-4:	Sanyo	0.2	[20,16,12,8]	3
	Hanergy	0.1	[12,8,6,4]	3
	Solar	0.1	[16,12,8,6]	3
TCAN:	Sanyo	0.1	[12,6]	2
	Hanergy	0.1	[12,8,4]	3
	Solar	0.1	[12,6,4]	3

TABLE II
ACCURACY RESULTS - ρ0.5/ρ0.9-LOSS. $\diamond$ DENOTES RESULTS FROM [5].

	Sanyo	Hanery	Solar
Persistence	0.154/-	0.242/-	0.256/-
DeepAR	0.070/**0.031**	0.092/0.045	0.222$^\diamond$/0.093$^\diamond$
LS Transformer	0.067/0.036	0.124/0.066	0.210$^\diamond$/0.082$^\diamond$
N-BEATS-I	0.091/-	0.154/-	0.215/-
N-BEATS-G	0.077/-	0.132/-	0.212/-
TCNN-3	0.066/0.032	0.088/0.045	0.230/0.088
TCNN-4	0.069/**0.031**	0.078/0.041	0.222/**0.080**
TCAN	**0.062/0.031**	**0.068/0.035**	**0.209**/0.081

VI. RESULTS AND DISCUSSION

The ρ0.5 and ρ0.9-losses are shown in Table II. Since N-BEATS and Persistence do not produce probabilistic forecasts, only the ρ0.5-loss (equivalent to MAE) is reported for them.

Overall, the most accurate model is TCAN - it outperforms all other methods for both point (ρ0.5) and probabilistic (ρ0.9) forecasts on all datasets except for one case - ρ0.9 for Solar, where it is ranked second after TCNN-4. For the point forecasts, the second-best performing model is LogSparse Transformer, followed by TCNN-4 and TCNN-3 (equal rank), then DeepAR, N-BEATS-G, N-BEATS-I and finally Persistence. For the probabilistic forecasts, TCAN and TCNN-4 are equally first, followed by TCNN-3, DeepAR and LogSparse Transformer. Hence, overall TCNN-4 is the second-best performing model.

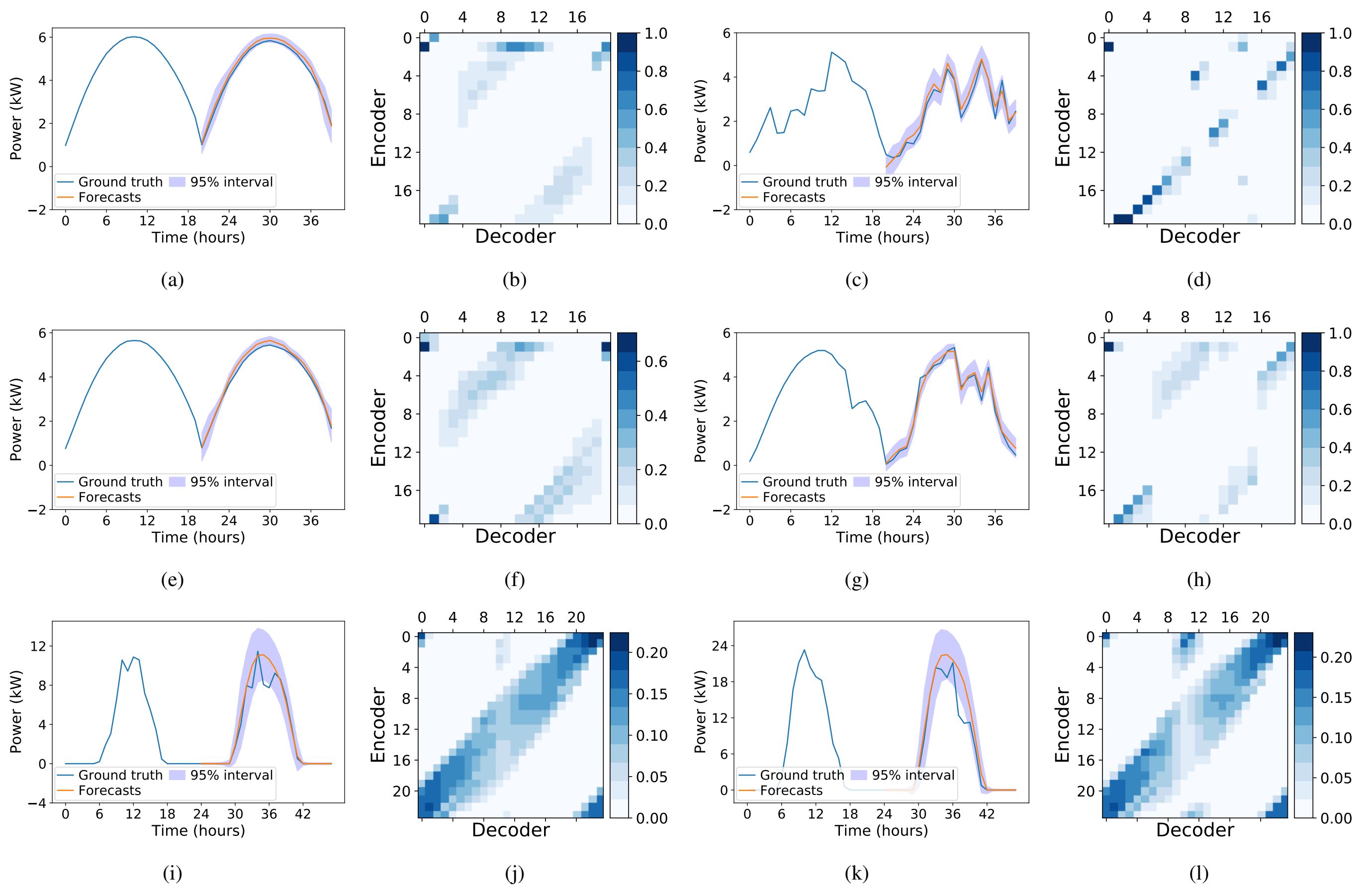


Fig. 5. TCAN case study: (1) actual vs predicted values and (2) attention map for two samples from each dataset. First, second and third rows correspond to the Sanyo, Hanergy and Solar datasets respectively.

By comparing the two TCNN models, we can see that TCNN-4 outperformed TCNN-3 for both point and probabilistic forecasts on all datasets except for Sanyo for $\rho 0.5$. While the receptive field of TCNN-4 is able to cover all values from the input sequence (20 for Sanyo and Hanergy and 24 for Solar), the receptive field of TCNN-3 covers only 14 steps. This shows that enabling a larger receptive field in TCNN-4 was beneficial.

TCAN is more accurate than both TCNN-4 and TCNN-3 which shows the effectiveness of the sparse attention mechanism for improving accuracy and extending the receptive field without adding more convolutional layers. The receptive fields of both TCAN and TCNN-4 can cover all values from the input sequence but TCAN uses a smaller number of convolutional layers than TCNN-4.

Fig. 5 illustrates TCAN's forecasting results for two consecutive days from the test set of each dataset - (i) actual vs predicted values for each day and (ii) the corresponding sparse attention map. The attention map shows the pair attention scores which represent the importance of the previous time series steps (y-axis) in predicting the future steps (x-axis). The plots show that TCAN is able to model the solar power series accurately, and the attention maps show that: 1) the dependency between future and past steps is sparse and 2) accessing longer previous history is important. For example, all maps show high attention scores for some early time steps. Fig. 5 (d) shows an extreme case - the first future prediction is determined by the second input step only.

The attention-map visualization is useful to understand the importance of the input features for each instance and can be used for explanation and justification of decisions to increase trust in the system in practical applications.

We also compare the training speed of TCAN and TCNN-4; both models can cover all input steps via their receptive fields. Both are trained on the same device, and the average elapsed time per batch and standard deviation are reported in Fig. 6. TCAN is considerably faster than TCNN-4 on all datasets because it has fewer temporal convolutional layers and trainable parameters.

Overall, the superior performance of TCAN indicates its effectiveness for capturing sparse dependency between future and past steps and enabling an extended receptive field without a deep architecture. TCAN also provides explainable results by showing which timesteps are most important for the prediction.

VII. Conclusion

In this work, we present TCAN, a new approach for time series forecasting. TCAN employs multiple temporal

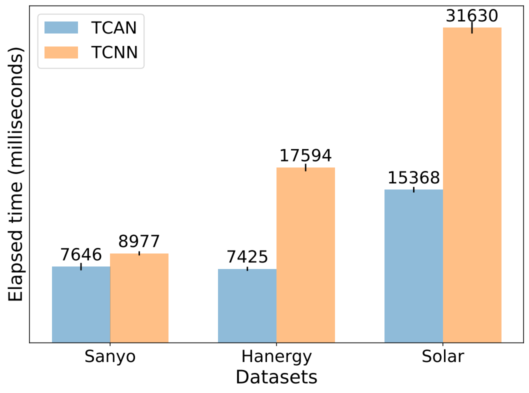


Fig. 6. Comparison of training time of TCAN and TCNN

convolutional layers to learn temporal patterns and a sparse attention layer to enable an extended receptive field without adding more layers. The sparse attention layer uses the latent factors generated by the temporal convolutional layers and identifies the important time steps to produce the attention vectors for the output layer and compute the final forecasts. The performance of TCAN is evaluated on three solar power data sets as a case study. The results show that TCAN outperforms the state-of-the-art deep learning models DeepAR, LogSparse Transformer, N-BEATS and TCNN and a persistent baseline in terms of accuracy. TCAN requires less number of convolutional layers than TCNN to cover the input sequence and is also much faster to train. The sparse attention maps facilitate understanding and interpretability of the results by showing the most relevant timesteps for the prediction of each instance.

In future work, we will investigate (1) the importance of the input time steps identified by the attention mechanism and the covariates to further improve interpretability, and (2) the application of TCAN to other time series forecasting tasks.

References

[1] 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.

[2] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[4] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181 – 1191, 2020.

[5] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of Transformer on time series forecasting," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[6] 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.

[7] 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.

[8] Y. Lin, I. Koprinska, and M. Rana, "Temporal convolutional neural networks for solar power forecasting," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2020.

[9] 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.

[10] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, "Probabilistic forecasting with temporal convolutional neural network," *Neurocomputing*, vol. 399, pp. 491 – 501, 2020.

[11] D1, "Sanyo dataset," http://dkasolarcentre.com.au/source/alice-springs/dka-m4-b-phase, 2020.

[12] D2, "Hanergy dataset," http://dkasolarcentre.com.au/source/alice-springs/dka-m16-b-phase, 2020.

[13] D3, "Solar dataset," https://www.nrel.gov/grid/solar-power-data.html, 2014.

[14] Y. Lin, I. Koprinska, and M. Rana, "SpringNet: Transformer and Spring DTW for time series forecasting," in *Proceedings of the International Conference on Neural Information Processing (ICONIP)*, 2020.

[15] 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.

[16] 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*, 1989.

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

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

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[20] A. Martins and R. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in *Proceedings of The International Conference on Machine Learning (ICML)*, 2016.

[21] B. Peters, V. Niculae, and A. F. T. Martins, "Sparse sequence-to-sequence models," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[22] G. M. Correia, V. Niculae, and A. F. T. Martins, "Adaptively sparse transformers," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[23] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[24] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2016.