A Parallel Electrical Optimized Load Forecasting Method Based on Quasi-Recurrent Neural Network

Caiming Yang¹, Wenxing Wang², Xinxin Zhang¹, Qinhui Guo², Tianyi Zhu³* and Qian Ai³

¹State Grid Shaoxing Power Supply Company, Shaoxing, China
²State Grid Zhejiang Shengzhou Power Supply Co. ,Ltd., Shengzhou, China
³Department of Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China

*Corresponding author email: zty0912@sjtu.edu.com

Abstract. Based on massive power big data resources, this paper establishes a new model for short-term load forecasting based on quasi-recurrent neural network (QRNN). QRNN combines the structural advantages of recurrent neural network (RNN) and convolutional neural network (CNN). It takes advantage of RNN’s cyclic connections to deal with the temporal dependencies of the load series, while implementing parallel calculations in both timestep and minibatch dimensions like CNN. The paper detailly describes the design and construction of QRNN, as well as the pre-processing and training steps of the forecasting model. Then, the algorithm is deployed to the big data platform, and an integrated load prediction system integrating data extraction, offline training, online forecasting and data visualization is developed. Finally, the proposed model is compared with some widely used machine learning load forecasting models. The results show that the QRNN based method achieves better prediction accuracy, and greatly improves the computational efficiency of training and testing, which is more practical for real-time and large-scale load forecasting.

Keywords: Short-term load forecasting; Quasi-Recurrent Neural Networks; Parallel computing; Integrated load forecasting system.

1. Introduction
Load forecasting refers to the process of calculating and predicting future loads through analysing and mining historical data. As the basis of power system planning and dispatching, power load forecasting can be used for power plant generation capacity planning and grid dispatching, which can improve the stability of power system operation. Therefore, load forecasting has high social and economic benefits, and is the foundation of power grid safety management[1]. Methods for short-term load forecasting (STLF) mainly include statistical methods and artificial intelligence methods. Among them, load forecasting methods based on statistics include time series method[2], multiple linear regression model[3], Kalman filtering model, etc. Artificial intelligence methods mainly include fuzzy prediction method, wavelet transform method, support vector regression (SVR)[4], etc. In recent years, deep learning, as an emerging method in the field of machine learning, is a powerful tool for dealing with nonlinear and dynamic uncertainties in smart power grids. With strong automatic feature extraction capability and advantages in processing high-dimensional and nonlinear data, it has been widely applied in the field of STLF. Typical methods include Deep Neural Network (DNN)[6], Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), etc. As a typical kind of RNNs, Long Short-Term Memory (LSTM)[5] can reflect the impact of historical information on
the current state, so it is very suitable for the field of sequence modelling. LSTM has been widely used in language modelling, machine translation, speech recognition and other fields, and gradually applied in the field of load prediction. Nevertheless, the LSTM’s loop structure also prevents parallel computing on the network, which limits the speed of training and testing.

In recent years, with the development of power system towards intelligent direction, the penetration rate of distributed energy is increasing gradually. The increase of adjustable flexible loads, such as electric vehicles, makes the randomness of grid loads gradually increase[7], which brings greater challenges to the accurate prediction of loads. On the other hand, so as to achieve differentiated planning and lean management requirements on the distribution network side, load forecasting is developing towards refined granularity. The refined load forecasting for mass data greatly increases the data size to be processed (from the original GB to TB[8]), and it still shows a growing trend. Under this new background, it is of great significance to effectively utilize the load big data to improve the precision of load prediction, while improving the processing efficiency to meet the real-time requirements in engineering applications.

Therefore, this paper proposes a new load prediction method using quasi-recurrent neural networks. The QRNN combines the structural advantages of RNN and CNN, which can utilize the temporal correlation of loads like RNN as well as realizing parallel optimization in timestep and minibatch dimensions like CNN. The paper introduces the steps of model construction and training in detail. Furthermore, the algorithm is deployed on the big data platform, and an integrated prediction system is established and applied in practical engineering application. Finally, experiments prove that the QRNN based method can achieve higher prediction accuracy than other popular machine learning methods, and effectively improve the processing efficiency of load prediction, which can better handle real-time and massive load forecasting.

2. Methodology
In this section, we first introduce a typical RNN called LSTM and argue why LSTM is not suitable for parallel training. Next, we propose and explain Quasi-Recurrent Neural Network (QRNN), which has proven to be an effective alternative to LSTM. Furthermore, we analyse the computational complexity.

2.1. Long Short-Term Memory
RNNs contain cyclic connections which allow retaining historical information in a long-range sequence, and thus are well suited for time series modelling and predictive modelling like power load forecasts. LSTM is one of the most notably used types of RNNs. It ameliorates RNN by using a memory gate and an extra forget gate[9], which allow LSTM to store long-term memory and avoid issues of gradient explosion and vanishing in RNNs.

In an LSTM unit, the basic unit that forms a deep LSTM network, the state at tth time step is defined in Eq.(1)[9].

\[
\begin{align*}
    f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
    i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
    o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
    c_t &= \sigma(W_c x_t + U_c h_{t-1} + b_c) \\
    h_t &= o_t \circ \tanh(c_t)
\end{align*}
\]  

(1)

In which \(i, f, o, c\) stand for input gate, forget gate, output gate and memory gate respectively; \(W\) and \(U\) are weight matrices and \(b\) is a bias vector of the corresponding gate; Symbol \(\circ\) stands for dot product; The activation function is in practice \(\tanh\) or sigmoid function \(\sigma\).

As indicated in Eq.(1), an LSTM unit has eight multiply-vector multiplications that is much more computationally expensive than the dot product. Of these, \(W_f x_t, W_i x_t, W_o x_t, W_c x_t\) can be precomputed through multi time processing approach. Nevertheless, the parts containing \(h_{t-1}\) rely on previous output, which makes it difficult for fully precomputing, thus creating an execution bottleneck. The LSTMs should be calculated sequentially and can hardly be accelerated by parallel computing. To sum up,
LSTMs operates slowly, which restrict the algorithm in big data volume, fast and real-time load forecasting scenes, and thus the algorithm is suitable for engineering applications.

2.2. Quasi-Recurrent Neural Networks

We present QRNNs to reduce the computational effort of the recurrent step in LSTMs. QRNNs are hybrid neural networks of LSTMs and CNNs, combining the advantages of both. Compared to origin LSTMs, QRNNs can be highly parallelizable like CNNs [10][11]. The block diagram of QRNN architecture compared with typical LSTM and CNN is shown in Figure 1. Brown refers to matrix multiplications or convolutions. Blue refers to a parameter-less functions running in parallel along the feature or channel dimension. Contiguous blocks mean that computations can be performed in parallel. As the figure shown, LSTMs can be decomposed into brown linear blocks and blue element blocks, while computation at each time step has dependency with the previous results.

![Figure 1. Block diagram of QRNN compared with typical LSTM and CNN.](image)

Each layer of QRNN combines two seed components, similar to convolutional layers and pooling layers in CNN. These two types of layers both allow fully parallel computation: the convolutional layers support parallelization across mini batches and spatial dimensions (i.e. sequence dimensions); the pooling layers support parallelization across mini batches and feature dimensions [10].

The equation of QRNN unit is shown in Eq.(2).

\[
\begin{align*}
\hat{x}_t &= \tanh(W \ast X_t) \\
f_t &= \sigma(W_f \ast X_t) \\
o_t &= \sigma(W_o \ast X_t) \\
c_t &= f_t \odot c_{t-1} + (1 - f_t) \odot \hat{x}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{align*}
\]

Where \(X_t \in \mathbb{R}^{k \times n}\) is the input sequence of \(k\ n\)-dimensional vectors \(x_{t-k+1}, \ldots, x_t\). The symbol \(\ast\) represents the mask convolution along the timestep dimension. \(W, W_f, W_o\) are all convolutional filter banks in \(\mathbb{R}^{d \times n \times k}\), and \(k\) is the width of filters. The first three expressions are the convolution part of QRNN, and these convolution operations produce \(m\)-dimensional sequences \(\hat{x}_t, f_t, o_t\). The symbol \(\odot\) represents elementwise multiplication. The last two expressions are the pooling part of QRNN, just as the familiar elementwise gates do in LSTM unit. Distinctively, QRNN only uses a forget gate, which term “dynamic average pooling” in [12].

As indicated in Eq.(2), a single QRNN only performs 3 multiply-vector operations that depend just on the input sequence \(X\), without dependency on previous outputs like \(h_{t-1}\). With known input, these multiply-vector operations \(W \ast X_t\) could be pre-calculated in multiple timesteps. Therefore, weight matrices with a large amount of memory do not need to be loaded at each timestep. In this approach the cost of DRAM is reduced when the timesteps required to conduct grows. Hence, these operations can be parallelized in one matrix-matrix multiplication, as can be seen in Eq.(3).
In which \( U \in R^{L \times 3d} \) is the combined result matrix; \( d \) is the number of hidden layer neurons; \( L = T - k + 1 \) represents the input sequence length. When considering minibatch \( B, U \in R^{L \times B \times 3d} \) will become a tensor. The QRNN network construction process is as Algorithm 1.

**Algorithm 1: QRNN Network Construction**

**Input:** merged matrix \( U[k, j, l'] \), initial hidden state \( c_0[i, j] \), bias vector \( b_f[i] \), \( b_r[i] \)

**Output:** hidden state tensor \( c[\cdot, \cdot, \cdot] \), output tensor \( h[\cdot, \cdot, \cdot] \)

**Parameter settings:** mini batch \( B \), number of hidden layer neurons \( d \), output sequence length \( L \)

for \( j = 1, ..., B; i = 1, ..., d \) do

\[
\begin{align*}
\tilde{x} &= \text{tanh}(U[k, j, l]) \\
f &= \sigma(U[k, j, i + d] + b_f[i]) \\
r &= \sigma(U[k, j, i + d \times 2] + b_r[i]) \\
c &= f \times c + (1 - f) \times \tilde{x} \\
h &= r \times \text{tanh}(c) \\
c[k, j, i] &= c \\
h[k, j, i] &= h
\end{align*}
\]

return \( h[\cdot, \cdot, \cdot] \) and \( c[\cdot, \cdot, \cdot] \)

### 3. Implementation

#### 3.1. Feature Selection

We have selected several features as the input and output of one QRNN unit, as listed in Table 1. The selected input features consist of historical load, weather characteristics including average temperature and relative humidity, date type and timestamp, with a total of five-dimensional array. The selected output is the prediction load value at the next moment. Finally, a series of QRNN units are spliced together for load sequence prediction.

**Table 1. Selected Input and Output Features.**

| Feature       | Symbol            |
|---------------|-------------------|
| Input         |                   |
| Historical load| Load\((t)\)       |
| Time          | \( t \)           |
| Date category | DayType\((i)\)     |
| Weather factors| weather\((1:2)\)  |
| Output        |                   |
| Load forecast | Load\((t+1)\)     |

#### 3.2. Data Pre-processing

Considering the possible deviated or missing outliers, data pre-processing of the original data set is required. These missing data and outliers (null values of no more than 10% of the data volume and "NAN" outliers) are supplemented by sliding window averaging.

Furthermore, in order to eliminate the influence of different dimensions in the expression, the data set \( x(i) \) needs to be normalized, so that the input feature vectors are limited within the range of \([0,1]\). Normalization of input features can also improve convergence speed and prediction accuracy. The expression for normalized input data \( x_1(i) \) is

\[
x_1(i) = \frac{x(i) - x_{min}}{x_{max} - x_{min}}
\]

In which \( x_{max} \) and \( x_{min} \) are the maximum and minimum values of input respectively.
3.3. Evaluation Index
The mean absolute percentage error (MAPE) and root mean square error (RMSE) are used to evaluate the forecasting effect, as shown in Eq. (5) and Eq. (6).

\[
\delta_{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - d_i}{y_i} \right| \times 100 \% \tag{5}
\]

\[
\delta_{RMSE} = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (y_i - d_i)^2 \tag{6}
\]

Where \( n \) is the length of prediction sequence; \( d_i \) and \( y_i \) represent the forecasts and the real data at timestep \( i \) respectively.

3.4. Modular Load Prediction System Based on Big Data Platform
The STLF model proposed can be applied to practical projects such as power grid dispatching and marketing. Based on Alibaba cloud big data platform, which integrates the cloud server and the enterprise-level data warehouse, we developed a set of STLF software. The software can be applied to the intelligent management and control platform of urban distribution network. The software combines off-line training with on-line predicting so as to realize fast and real-time load forecasts, and finally forms an integrated load forecasts system from data extraction to prediction and then to data visualization. The above process is described as Figure 2, and specifically includes the following steps:

1) Data synchronization. Continuous data protection (CDP) is adopted to synchronize the data in business data system to Open Data Processing Service (ODPS) data warehouse, so as to realize data fusion.

2) On-line data extraction and preprocessing. Extract user attributes, load data and weather data from enterprise database for offline calculation and pre-processing. Then, we divide the data set into three subsets, namely, training set, validation set and testing set.

3) STLF model establishment and training. A short-term load forecasting model based on QRNN network was established, and the model application was deployed through elastic compute service (ECS). Select optimal hyperparameters and store them. Thereafter, these model parameters are periodically fine-tuned and updated.

4) Visual display and system-level application. The system will invoke the deployed model for real-time load forecasting and the forecast results will be synchronized to the RDS for data visualization. Furthermore, the prediction results can support advanced applications, such as power grid dispatch and power marketing strategy formulation.

4. Experiment
The experimental datasets contain power load data and weather data. The former is provided by a power company in Zhejiang Province, with the time span from 2019-05-27 to 2020-05-27 and the step length of 15 minutes. The latter including temperature and relative humidity is downloaded from the National Meteorological Data Center. We split the datasets for training, validation, and testing at a scale of 0.7: 0.15: 0.15. Then the day-ahead load was predicted at the step length of 96 points. A total of 431 groups of data are were modelled, and the test results of a typical distribution transformer with one-week data were analysed.

The hardware platform is an cloud computing platform of ECS.gn6i-c4g1.Xlarge with 2.5GHz Intel Xeon Platinum 8163 CPU and NVIDIA T4 GPU. The QRNN model is implemented through TensorFlow-1.11.0(GPU version) architecture, and CUDA-9.0 is used to realize GPU parallel computing. The LSTM and SVR methods are used for comparison. In order to verify the performance of the proposed method, analysis is carried out from the perspectives of forecasting accuracy and operational performance.

4.1. Comparison of Prediction Accuracy
For all the models, Bayesian Optimization\textsuperscript{[13]} is applied to search for the optimal hyperparameters. To regularize, L2 regularization of \( 5 \times 10^{-6} \) is applied and a dropout rate of 0.2 is used for each layer in
LSTM and QRNN models. Day-ahead load prediction was carried out with the data of June 10 solstice and June 16 as the test set. Figure 3 shows the one-week load prediction curve as well as its true distribution, and Table 2 gives detailed RMSE and MAPE results.

As the experimental results shown, the prediction accuracy of QRNN and LSTM in load forecasting is obviously better than that of the traditional SVR. Compared with LSTM, QRNN has a slight improvement in the prediction effect, and the overall forecasting accuracy is similar. In general, QRNN, as the fusion version of LSTM and CNN, has excellent performance in prediction accuracy.

![Diagram](https://example.com/diagram.png)

**Figure 2.** A Comprehensive Load Prediction System Based on Cloud Computing Platform.

**Table 2.** Errors On The Testing Set.

| Day     | QRNN  | LSTM  | SVR   |
|---------|-------|-------|-------|
| 5/20    | MAPE/%|       |       |
| 2.796   | 2.531 | 4.061 |
| 5/21    | RMSE/MW| 21.7129| 16.7619| 26.1836|
| 5/22    | MAPE/%|       |       |
| 2.512   | 2.342 | 4.719 |
| 5/23    | RMSE/MW| 17.4424| 25.3202| 48.3500|
| 5/24    | MAPE/%|       |       |
| 2.838   | 2.133 | 3.594 |
| 5/25    | RMSE/MW| 17.1850| 14.6369| 29.1321|
| 5/26    | MAPE/%|       |       |
| 2.998   | 3.415 | 3.335 |
| 5/27    | RMSE/MW| 19.3844| 13.8555| 22.7862|
| 5/28    | MAPE/%|       |       |
| 2.991   | 2.449 | 6.431 |
| 5/29    | RMSE/MW| 17.7984| 14.4143| 39.0186|
| 5/30    | MAPE/%|       |       |
| 2.401   | 3.394 | 3.674 |
| 5/31    | RMSE/MW| 15.3025| 20.1422| 22.2410|
| Total   | MAPE/%|       |       |
| 2.806   | 2.861 | 4.724 |
|         | RMSE/MW| 18.5870| 18.8831| 32.1879|
Figure 3. Comparison of Forecasts Result.

4.2. Comparison of Execution Efficiency
We have measured the execution time of QRNN, LSTM and SVR models on each sequence, as shown in Table 3 and Figure 4. The training time including forward and backward propagation as well as the testing time have been calculated and compared experimentally.

As far as the training time, the SVR method has the fastest execution speed. With similar prediction accuracy, the training time of the QRNN based load forecasting model is only 23.588 seconds, which is slightly longer than that of SVR, while the training time of LSTM is as long as 221.877 seconds, which is several times that of QRNN. In terms of the testing time, the QRNN shows the fastest speed with a time of 0.272 seconds, followed by LSTM with a time of 1.103 seconds, and finally by SVR.

Further, the acceleration ratio of QRNN than LSTM with different data volumes is given in Table 4. The acceleration rate is calculated as Eq. (7) shows.

\[ S = \frac{t_d - t_d^d}{t_d^d} \]  

(7)
Where $s$ is the acceleration rate, $t_d$ and $t_q$ are the average execution times of LSTM and QRNN networks when the data volume is $d$, respectively.

As is shown in Table 4, the training speed of the QRNN based model on a single group of data is about 89.92% faster than LSTM based model. In terms of execution speed on the test set, the time difference between QRNN and LSTM is reduced, whereas the QRNN based model is still 75.34% faster than LSTM based model. The training speed of the SVR method is slightly faster than that of the QRNN method, whereas its testing speed is much slower than the other two deep recurrent neural networks, which makes it less effective when applied to real-time load forecasting.

With the increase of data volume, the acceleration rate of QRNN compared with LSTM gradually increases, which proves that QRNN can better utilize computing resources and is more prominent in the case of large data volumes. On the actual load forecasting system we developed, the models were usually updated over a fixed period of time. In the need of refined load forecasting and analysis, fine tuning many different prediction models will consume more memory resources, which poses great challenges to the operational efficiency of deep learning models such as LSTM. As the experimental results show, the proposed prediction model based on QRNN performs well in both prediction accuracy and computational efficiency. On the one hand, the RNN like structure of the QRNN model enables it reflect the connection of load data on time series, and thus the time correlation of load series will be deeply mined. On the other hand, the QRNN model uses the idea of CNN to improve the network structure, and combines with parallel optimization calculation to greatly improve the prediction efficiency. Thus, the QRNN method makes recurrent neural network more practical for engineering, and provides a feasible scheme for real-time and massive power load forecasts scenarios.

**Table 3. Execution Time.**

| Method | Training(s) | Testing(s) |
|--------|-------------|------------|
|        | Forward     | Backward   |            |
| QRNN   | 10.687      | 12.901     | 0.272      |
| LSTM   | 95.756      | 126.121    | 1.103      |
| SVR    | 5.342       | 6.316      | 3.305      |

**Table 4. Speed Up Rate of QRNN Compared to LSTM.**

| Data volumes | Training acceleration rate | Testing acceleration rate |
|--------------|----------------------------|---------------------------|
| 1.6          | 89.92%                     | 75.34%                    |
| 8.0          | 90.51%                     | 75.71%                    |
| 20.1         | 90.92%                     | 76.07%                    |
| 40.2         | 91.06%                     | 76.18%                    |
| 60.2         | 91.10%                     | 76.23%                    |

**Figure 4. Execution Time of QRNN, LSTM and SVR.**

5. Conclusion

This paper proposes a short-term load forecasting model based on quasi-recurrent neural networks. QRNN combines the structure of CNN and LSTM, and can give play to the advantages of the two models simultaneously. Similar to RNNs, QRNNs can reflect and mind the correlation of load data on time series. Similar to CNNs, QRNNs allows parallel computing acceleration in both time-step and mini-batch dimensions, thus can handle long sequence and high throughput data well. Furthermore, the
algorithm is deployed to the big data platform, and an integrated power load prediction system from data extraction, offline training, online forecasting to data visualization is developed. Finally, comparison experiments are conducted with SVR based method and LSTM based method. The comparison is made from the perspective of prediction accuracy and prediction speed. Results show that the QRNN based method performs well in both prediction accuracy and computational efficiency. The prediction accuracy of QRNN model is slightly better than that of LSTM model, and better than that of SVR. Simultaneously, the QRNN based method greatly improves the prediction efficiency through parallel computing. In comparison with LSTM-based model, the training speed of QRNN-based model is increased by 89.92%, and the average testing speed is increased by 75.34%. As the data volume increases, the acceleration rate increases. To sum up, the proposed model has a certain value to handle the problem of real-time and large-scale load forecasts in engineering.

References
[1] Ranaweera D K, Karady G G, Farmer R G. Economic impact analysis of load forecasting[J]. IEEE Transactions on Power Systems, 2002, 12(3):1388-1392.
[2] Zhang Z, Li H, Zhao Y, et al. Short-term load forecasting based on the grid method and the time series fuzzy load forecasting method[C]// International Conference on Renewable Power Generation (RPG 2015). IET, 2016.
[3] Hong T, Wang P, Willis H L. A Naïve multiple linear regression benchmark for short term load forecasting[C]// IEEE Power & Energy Society General Meeting. IEEE, 2011.
[4] Xing Z. Short-Term Load Forecasting for Electric Bus Charging Stations Based on Fuzzy Clustering and Least Squares Support Vector Machine Optimized by Wolf Pack Algorithm[J]. Energies, 2018, 11(6):1449-1458.
[5] Yudantaka K, Kim J S, Song H. Dual Deep Learning Networks Based Load Forecasting with Partial Real-Time Information and Its Application to System Marginal Price Prediction[J]. Energies, 2019, 12(6):1449-1458.
[6] Suryanarayana G, Lago J, Geysen D, et al. Thermal load forecasting in district heating networks using deep learning and advanced feature selection methods[J]. Energy, 2018.
[7] LU Jixiang, ZHANG Qipei, YANG Zhihong, et al. Short-term Load Forecasting Method Based on CNN-LSTM Hybrid Neural Network Model[J]. Automation of Electric Power Systems, 2019, 43(8).
[8] LI Gang, LIU Yan, SONG Yu, et al. Visualization pretreatment method for electric power big data based on information fusion[J]. Guangdong Electric Power, 2016, 29(12): 10-14.
[9] Sak H, Senior A, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling[J]. computer science, 2014.
[10] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. arXiv preprint arXiv:1611.01576 (2016).
[11] Frédéric Godin, Jonas Degrange, Joni Dambre, Wesley De Neve. "Dual Rectified Linear Units (DReLUs): A replacement for tanh activation functions in Quasi-Recurrent Neural Networks"[J]. Pattern Recognition Letters, 2018.
[12] Balduzzi D, Ghifary M. Strongly-Typed Recurrent Neural Networks[J]. 2016.
[13] Snoek J, Larochelle H, Adams R P. Practical Bayesian Optimization of Machine Learning Algorithms[J]. Advances in neural information processing systems, 2012.