Adjusting for Autocorrelated Errors in Neural Networks for Time Series Regression and Forecasting

Preprint · January 2021

3 authors, including:

Duane Boning
Massachusetts Institute of Technology
303 PUBLICATIONS **4,092** CITATIONS

SEE PROFILE

Adjusting for Autocorrelated Errors in Neural Networks for Time Series Regression and Forecasting

Fan-Keng Sun [1] **Christopher I. Lang** [1] **Duane S. Boning** [1]

Abstract

In many cases, it is difficult to generate highly accurate models for time series data using a known parametric model structure. In response, an increasing body of research focuses on using neural networks to model time series approximately. A common assumption in training neural networks on time series is that the errors at different time steps are uncorrelated. However, due to the temporality of the data, errors are actually autocorrelated in many cases, which makes such maximum likelihood estimation inaccurate. In this paper, we propose to learn the autocorrelation coefficient jointly with the model parameters in order to adjust for autocorrelated errors. For time series regression, large-scale experiments indicate that our method outperforms the Prais-Winsten method, especially when the autocorrelation is strong. Furthermore, we broaden our method to time series forecasting and apply it with various state-of-the-art models. Results across a wide range of real-world datasets show that our method enhances performance in almost all cases.

1. Introduction

Time series data are easier to collect and increasingly ubiquitous as sensors become smaller and cheaper. For example, more and more industrial internet of things (IIoT) sensors are being installed around manufacturing equipment to collect time series data that can help to increase the efficiency of day-to-day manufacturing operations. Analysts seek to model such data to perform various tasks, such as time series classification, regression, and forecasting. However, several issues arise during the collection and modeling procedure.

First, which variables are essential and how many of them are enough? To illustrate, if we want to forecast household electricity consumption, temperature should be an important variable to consider. But the available electricity datasets, which many previous works are compared upon (Yu et al., 2016; Lai et al., 2018; Shih et al., 2019; Li et al., 2019; Salinas et al., 2020), do not include such information. Even if we were to consider weather, there are still many other variables that might be influential. In many real-world cases, in order to have satisfactory prediction results, it is either too complicated to decide which and how many variables are required, or impossible to collect all the required variables.

Secondly, measurement almost always involves errors that degrade the performance of the model. Importantly, the common assumption that the errors are uncorrelated in time might be incorrect, especially for time series data. For example, when taking time-sampled measurements in semiconductor fabrication equipment, noise is inevitable. If the rate of measurement is faster than the rate of fluctuation of the noise, then measurement errors are no longer uncorrelated.

Finally, we need to decide what model to use. In most instances, the exact form of the underlying system is unknown, so we can only approximate it. However, if we formulate an incorrect model, model mis-specification will greatly weaken performance. The advancement of neural networks (NNs) helps to alleviate this issue and enable us to achieve good approximations in many cases. Nonetheless, there is still no guarantee that a specific NN is representative enough on a specific dataset.

All three aforementioned issues can lead to *autocorrelated errors* (Montgomery et al., 2012; Greene, 2017), in which errors at the current time step are correlated with errors at previous time steps.

In ordinary least squares (OLS), autocorrelated errors violate the assumption that the errors are independent, which implies that the Gauss-Markov theorem is not applicable. Specifically, the variance of the coefficient estimates increases but the estimated standard error is underestimated. Thus, if neglected, prediction accuracy is reduced, and an outcome that appears to be statistically significant may actually be insignificant. As for nonlinear data, autocorrelated errors impair the standard maximum likelihood estimation (MLE) and thus weaken model performance.

Adjusting for autocorrelated errors in linear or nonlinear time series data has been studied extensively, especially in

[1]EECS, MIT, Cambridge, MA. Correspondence to: Fan-Keng Sun <fankeng@mit.edu>.

econometrics (Cochrane & Orcutt, 1949; Prais & Winsten, 1954; Hildreth & Lu, 1960; Gallant & Goebel, 1976; Beach & MacKinnon, 1978; Glasbey, 1979; Frydman, 1980). However, those methods are applicable only when the exact form of the underlying system is known. On the other hand, NNs for time-series-related tasks (Gamboa, 2017; Chalapathy & Chawla, 2019; Fawaz et al., 2019; Benidis et al., 2020) have become a popular research direction due to NNs' effectiveness of approximating unknown, nonlinear systems. However, to the best of our knowledge, none of the existing NN-based methods adjust for autocorrelated errors. Thus, we introduce a method to account for autocorrelated errors in NNs and show that the proposed method improves the performance in both time series regression and forecasting. The implementation of our method can be found at https://github.com/anonymous.

To be specific, our main contributions are:

- We propose to learn the autocorrelation coefficient jointly with model parameters via gradient descent in order to adjust for autocorrelated errors in neural networks for time series regression.
- Our large-scale experimental results on time series regression show that our method outperforms the modified Prais-Winsten method with statistical significance, especially when the autocorrelation is strong.
- We broaden our method for adjusting autocorrelated errors in neural network for time series regression to time series forecasting.
- Our extensive experimental results on time series forecasting show that our method improves forecasting performances across a wide range of real-world datasets and neural network models.
- By ablation study and grid-search over autocorrelation coefficient and model hyperparameters, we validate the strength of our method.

2. Preliminaries

2.1. Time series regression

In time series regression, similar to typical regression, we have the input matrix $\mathbf{X} \in \mathbb{R}^{T \times N}$ and the target vector $\mathbf{y} \in \mathbb{R}^T$, where T is the number of samples and N is the number of independent variables. The goal is to learn a model to predict y_t, the t-th entry in $\mathbf{y}$, given $\mathbf{X}_t$, the t-th row in $\mathbf{X}$, assuming y_t is an *unknown* function of $\mathbf{X}_t$ only.

The difference between time series regression and typical regression is that the $\{\mathbf{X}_1, \dots, \mathbf{X}_T\}$ and $\{y_1, \dots, y_T\}$ in time series regression are ordered chronologically, whereas in typical regression there is no clear order. In other words, the subscript t denotes relative time step and the sampling rate (in this work) is uniform from $t = 1$ to $t = T$.

Mathematically, after defining a model f and the trainable parameters θ, we want to find the best θ to minimize the residual sum of squares (RSS) on testing set by optimizing the RSS on training set, where RSS is defined as:

$$\text{RSS} = \sum_t \hat{e}_t^2 = \sum_t (y_t - \hat{y}_t)^2, \tag{1}$$

and $\hat{y}_t = f(\mathbf{X}_t; \theta)$ is the prediction of the model.

Let $\theta^* \in \arg\min_\theta$ RSS on testing set; then we can assume that $f(\bullet; \theta^*)$ is the data-generating function with errors e_t:

$$y_t = f(\mathbf{X}_t; \theta^*) + e_t \tag{2}$$

because the true underlying function is unknown. Notice the subtle difference between our definition of the residuals

$$\hat{e}_t = y_t - f(\mathbf{X}_t; \theta), \tag{3}$$

which is the "estimation of errors," and the true errors

$$e_t = y_t - f(\mathbf{X}_t; \theta^*) \tag{4}$$

in the context of this work.[1]

2.2. Time series forecasting

Similar to time series regression, we have an input matrix $\mathbf{X} = \{\mathbf{X}_1, \dots, \mathbf{X}_t, \dots, \mathbf{X}_T\} \in \mathbb{R}^{T \times N}$ representing N variables sampled at the same rate at the same time for T time steps where $\mathbf{X}_t \in \mathbb{R}^N$ is the t-th sample. The goal is to forecast the value of $\mathbf{X}_t$ given the histories $\{\mathbf{X}_1, \dots, \mathbf{X}_{t-1}\}$. In practice, only the W most recent histories $\{\mathbf{X}_{t-W}, \dots, \mathbf{X}_{t-1}\}$ are fed into a model. This is a common approach (Qin et al., 2017; Lai et al., 2018; Shih et al., 2019; Bai et al., 2020) that assumes older histories are less informative, establishes fair comparisons between different methods, and makes the memory usage plausible.

Analogous to the derivation in time series regression, given the model f, we want to find the best θ to minimize the RSS on the testing set:

$$\text{RSS} = \sum_t \|y_t - \hat{y}_t\|_2^2, \tag{5}$$

where $y_t = \mathbf{X}_t$ and $\hat{y}_t = \hat{\mathbf{X}}_t = f(\mathbf{X}_{t-W}, \dots, \mathbf{X}_{t-1}; \theta)$.

Also, we have $\theta^* \in \arg\min_\theta$ RSS on testing set and

$$y_t = f(\mathbf{X}_{t-W}, \dots, \mathbf{X}_{t-1}; \theta^*) + e_t. \tag{6}$$

2.3. Autocorrelated Errors

In most of the machine learning literature, the errors e_t are assumed to be mutually independent:

$$\text{cov}(e_t, e_{t+h}) = 0, \forall h \neq 0. \tag{7}$$

However, as discussed in Section 1, there are three reasons why the assumption may be violated and the errors are thus

[1] In most machine learning papers, the term error actually refers to the residual in our definition.

autocorrelated, especially in time series data. In general, a p-th order autocorrelated error has the form

$$e_t = \rho_1 e_{t-1} + \cdots + \rho_p e_{t-p} + \epsilon_t,\ |\rho_i| < 1, \forall i \quad (8)$$

where $\rho_1, \ldots, \rho_p$ are autocorrelation coefficients and $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ is white noise. Notice that the magnitude of every ρ_i should be strictly smaller than 1.

In this work, we only focus on first-order autocorrelation because it is the single most significant term. Hence, we simplify the notation to $e_t = \rho e_{t-1} + \epsilon_t$. Nevertheless, our method can be extended to higher order autocorrelation.

When there exists first-order autocorrelated errors,

$$\text{cov}(e_t, e_{t+h}) = \frac{\rho^h}{1-\rho^2}, \forall h = 0, \pm 1, \pm 2, \ldots, \quad (9)$$

i.e., errors are no longer mutually independent, and thus the standard MLE becomes untenable. Alternatively, one should use the following form of MLE:

$$y_t - \rho y_{t-1} = f(\bullet_t; \theta) - \rho f(\bullet_{t-1}; \theta) + \epsilon_t, \quad (10)$$

so the errors are uncorrelated. Practically, the true ρ value is unknown, so an estimate $\hat{\rho}$ is used instead. Per Section 3.1, there are several methods to obtain the estimate $\hat{\rho}$ with known system, but none for NNs with unknown systems.

3. Related Work

3.1. Autocorrelated errors in time series regression

Adjusting for autocorrelated errors in time series regression has been studied extensively, especially in econometrics. Typically, after collecting the data and determining the formulation of the underlying system, the Durbin-Watson statistic (Durbin & Watson, 1951) is calculated to detect the presence of first-order autocorrelated errors. If there is statistical evidence that first-order autocorrelated errors exist, then one of the following methods can be applied.

The Cochrane-Orcutt method (Cochrane & Orcutt, 1949) is the most basic approach. It first estimates the autocorrelation coefficient of the residuals, then transforms the series to weaken the autocorrelation and fits OLS to the transformed series. The procedure can be done once or iteratively until convergence; both versions have the same asymptotic behavior (Gallant & Goebel, 1976), but the performance may differ with finite samples. During the transformation of the time series, the first sample is discarded — a large information loss when the sample size is small. The Prais-Winsten method (Prais & Winsten, 1954) solves this issue by retaining the first sample with appropriate scaling. Finally, the Beach-Mackinnon method (Beach & MacKinnon, 1978) formulates the exact likelihood function that incorporates not only the first sample but also an additional term that constrains the autocorrelation coefficient to be stationary.

These methods update the estimate of the autocorrelation coefficient starting from 0. However, multiple local minima might exist and the procedure might converge to a bad local minimum (Sargan, 1964; Dufour et al., 1980; Oxley & Roberts, 1982; Dufour et al., 1983). Thus, the Hildreth-Lu method (Hildreth & Lu, 1960) grid-searches over possible autocorrelation coefficients and picks the best one.

For autocorrelation of higher orders, the Ljung-Box test (Ljung & Box, 1978) or Breusch-Godfrey test (Breusch, 1978; Godfrey, 1978) can be applied to detect their presence. Aforementioned methods can then be extended for adjustment (Box et al., 1976; Abraham & Ledolter, 1983).

When dealing with nonlinear data, as long as the underlying system is known, prior methods are applicable by changing OLS to nonlinear least squares or employing other nonlinear optimization techniques such as BFGS (Fletcher, 1987). However, real-world time series data are often difficult to formulate with a known-correct model structure, especially when there are multiple series. This is where neural networks come into play.

Neural networks (NNs) (Hopfield, 1982) are inherently designed to learn arbitrary nonlinear relationships between input-target pairs from the data. In (Hornik et al., 1989), it has been proven that a nonlinear NN with sufficient number of hidden units is capable of approximating any function to any desired degree of accuracy. Pairing this with the exponentially increasing availability of data, computational power, and new algorithms, NNs can learn complex functionalities. For instance, NNs have greatly outperformed statistical, filtering, or subspace methods on speech enhancement (Loizou, 2013; Xu et al., 2013), which can be considered as a time series regression task with multiple target variables. Although NNs show promising outcomes, as of today, there is no paper dedicated to adjusting autocorrelated errors in NNs for time series regression.

3.2. Autocorrelated errors in time series forecasting

For modeling linear, univariate time series, autocorrelated errors can be represented by the moving average (MA) model, a component of the widely-used autoregressive integrated moving average (ARIMA) (Box et al., 1976) formalism. ARIMA can also be extended to multivariate time series.

Nevertheless, real-world data are usually nonlinear and complex. Thus, NN-based models have been the most prominent ones for time series forecasting. The four popular building blocks for these models are recurrent neural networks (RNNs), convolutional neural networks (CNNs), graph neural networks (GNNs), and Transformers.

Long Short-term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) networks are the most basic NN for time series forecasting. These are an improved version of RNNs, but these often fail at learning complex temporal (intra-

series) and spatial (inter-series) patterns. Temporal Pattern Attention (Shih et al., 2019) networks add convolutional attention to LSTMs to help capture temporal patterns. Adaptive Graph Convolutional Recurrent Networks (Bai et al., 2020) combine RNNs and GNNs to learn not only temporal but also spatial patterns. The Temporal Convolutional Networks (Bai et al., 2018) are dilated CNNs with residual connection that are good at modeling long sequences. The Convolutional Transformer (Li et al., 2019) plugs an input convolution into the original Transformer (Vaswani et al., 2017) to capture local temporal patterns. Dual Self-Attention Networks (Huang et al., 2019) use CNNs for temporal patterns and self-attention for spatial patterns.

There are many other works using NN-based models for time series forecasting (Qin et al., 2017; Lai et al., 2018; Sen et al., 2019; Salinas et al., 2020), most of which include at least one of the four building blocks above. However, to the best of our knowledge, none of the previous work addresses the issue of autocorrelated errors.

4. Our Method

4.1. For time series regression

Following the work of (Cochrane & Orcutt, 1949; Prais & Winsten, 1954; Gallant & Goebel, 1976), we design a similar method for NNs to adjust for autocorrelated errors:

1. Initialize model parameter θ randomly and $\hat{\rho}$, the estimation of ρ, at 0.
2. Fix $\hat{\rho}$ and train the model sufficiently to minimize RSS on training data, where
$$\begin{aligned}\hat{e}_1 &= \sqrt{1-\hat{\rho}^2}(y_1 - f(\mathbf{X}_1;\theta)),\\ \hat{e}_t &= (y_t - \hat{\rho}y_{t-1})\\ &\quad - (f(\mathbf{X}_t;\theta) - \hat{\rho}f(\mathbf{X}_{t-1};\theta)), \forall t \geq 2,\end{aligned} \tag{11}$$
and obtain the new model parameter θ'.
3. Compute the residuals $\hat{e}_t = y_t - f(\mathbf{X}_t;\theta')$.
4. Use the residuals to update $\hat{\rho}$ by linearly regressing $\hat{e}_t$ on $\hat{e}_{t-1}$, i.e.,
$$\hat{\rho} = \frac{\sum_{t=2}^{T}\hat{e}_t\hat{e}_{t-1}}{\sum_{t=1}^{T-1}\hat{e}_t^2}. \tag{12}$$
5. Go back to step 2 or stop if sufficiently converged.

In step 2, the model is trained with stochastic gradient descent (SGD) and we say a model is sufficiently trained if the RSS on the validation set does not improve for 50 consecutive epochs; and in step 5, the model is considered sufficiently converged after 750 total epochs of training. Although step 2 employs Adam, the whole method can be considered as using coordinate descent (Wright, 2015) to optimize $\hat{\rho}$ and θ alternately.

We call this method the modified Prais-Winsten method as it closely resembles the original Prais-Winsten method for linear data (Prais & Winsten, 1954). However, a major shortcoming of the original Prais-Winsten method is that it will only converge to a local minimum because ρ and θ are not optimized simultaneously. This happens on linear data (Sargan, 1964; Dufour et al., 1980; Oxley & Roberts, 1982; Dufour et al., 1983), not to mention nonlinear data. Hence, the modified Prais-Winsten method also suffers from this shortcoming.

To improve the optimization, we further propose to treat $\hat{\rho}$ as a *trainable* parameter and update it with θ jointly using SGD. That is, we optimize $(\theta, \hat{\rho})$ to minimize the negative log-likelihood function (Beach & MacKinnon, 1978)
$$\begin{aligned}&\text{const.} - \frac{1}{2}\log(1-\hat{\rho}^2)\\ &+ \frac{T}{2}\log[(1-\hat{\rho}^2)(y_1 - f(\mathbf{X}_1;\theta))^2\\ &\quad + \sum_{t=2}^{T}(y_t - \hat{\rho}y_{t-1} - f(\mathbf{X}_t;\theta) + \hat{\rho}f(\mathbf{X}_{t-1};\theta))^2].\end{aligned} \tag{13}$$

Originally, the term $\log(1-\hat{\rho}^2)$ helps to constrain $|\hat{\rho}| < 1$. But in practice, we found that this term is inconsequential. We omit this term and instead use the hyperbolic tangent to constrain it, i.e., $\hat{\rho} = \tanh(\tilde{\rho}), \tilde{\rho} \in \mathbb{R}$. Thus, by ignoring the shifting, scaling, and logarithm that does not affect the monotonicity in equation (13), our loss function is
$$\begin{aligned}&(1-\hat{\rho}^2)(y_1 - f(\mathbf{X}_1;\theta))^2\\ &\quad + \sum_{t=2}^{T}(y_t - \hat{\rho}y_{t-1} - f(\mathbf{X}_t;\theta) + \hat{\rho}f(\mathbf{X}_{t-1};\theta))^2.\end{aligned} \tag{14}$$

There are two advantages of jointly optimizing $(\hat{\rho}, \theta)$:

- Most deep NNs trained on large-scale datasets employ SGD, not coordinate descent. This has the benefit of escaping shallow local minima.
- Joint training relieves us from deciding when to switch to $\hat{\rho}$ after updating θ as in step 2 in the modified Prais-Winsten method.

4.2. For time series forecasting

In time series forecasting, the input and output are the same series, which means that not only the output of the model involves autocorrelated errors, but also the input of the model.

Following equation (6), we can adjust the autocorrelated errors in the output by
$$\begin{aligned}\mathbf{X}_t - \hat{\rho}\mathbf{X}_{t-1} &= f(\mathbf{X}_{t-W},\dots,\mathbf{X}_{t-1};\theta)\\ &\quad - \hat{\rho}f(\mathbf{X}_{t-W-1},\dots,\mathbf{X}_{t-2};\theta) + \epsilon_t\\ &:= f(\mathbf{X}_{t-W-1},\dots,\mathbf{X}_{t-1};\theta,\hat{\rho}) + \epsilon_t.\end{aligned} \tag{15}$$

In this way, the series $\{\mathbf{X}_t - \hat{\rho}\mathbf{X}_{t-1}\}$ is free of autocorre-

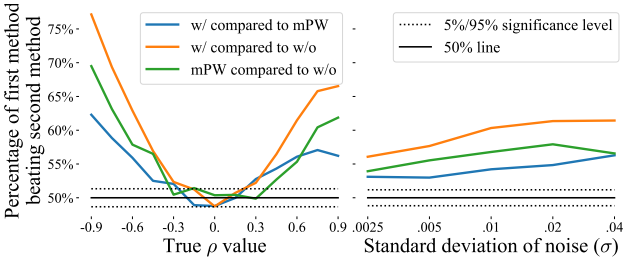

Figure 1. Pairwise comparison of three methods of training NN on the synthesized data with different true ρ value (left) and different standard deviation of noise (right).

lated errors. However, after this adjustment, the input and output series are not the same, which will complicate the learning process. Thus, we adjust the autocorrelated errors in the input as well. That is,

$$\mathbf{X}_t - \hat{\rho}\mathbf{X}_{t-1} = f(\mathbf{X}_{t-W} - \hat{\rho}\mathbf{X}_{t-W-1}, \ldots, \mathbf{X}_t - \hat{\rho}\mathbf{X}_{t-1}; \theta) + \epsilon_t, \tag{16}$$

which results in the same series that is free of autocorrelated errors for both input and output series. In addition, subtracting lag-1 values decrease the scale of the series and facilitates training of NNs on data with large scale differences, similar to normalization in data preprocessing.

In practice, $\mathbf{X}_{t-W-1}$ is replaced by the mean vector $\bar{X} \in \mathbb{R}^N$ in training set, so exactly W histories are used and the scale of each history is similar. $\hat{\rho}$ can be a scalar or N-dimensional vector depending on the dataset. Notice the optimal $\hat{\rho}$ given a fixed θ cannot be precisely calculated as in equation (12) because $\hat{\rho}$ is an input to the function f. This gives us another reason to update $\hat{\rho}$ using SGD.

5. Experiments on Time Series Regression

Our experiments on time series regression are conducted entirely on synthesized datasets following previous works (Cochrane & Orcutt, 1949; Gallant & Goebel, 1976; Beach & MacKinnon, 1978). Due to space limitations, only the important results are discussed here and the rest are located in Appendix A. The results show the superiority of our method of adjustment over both the standard method without adjustment and the modified Prais-Winsten method, especially when the autocorrelation is strong.

5.1. Data setup

We synthesize our data using Monte Carlo simulations. The data-generating function is

$$\begin{aligned} y_t &= \tanh(\frac{\mathbf{X}_t\theta + 1}{\sqrt{N}}) + e_t, \quad \mathbf{X}_t \in \mathbb{R}^N, \theta = \mathbf{1} \in \mathbb{R}^N, \\ \mathbf{X}_t &\sim \mathcal{N}(\mathbf{0}, \sigma_x^2\mathbf{I}), \quad \sigma_x = 0.2, \\ e_t &= \rho e_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2). \end{aligned} \tag{17}$$

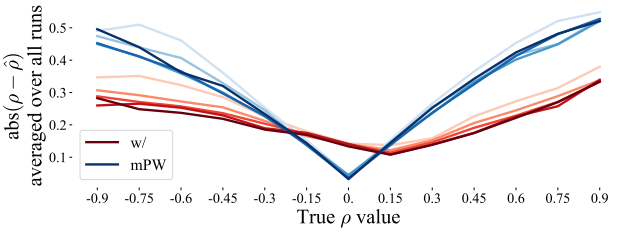

Figure 2. abs$(\rho - \hat{\rho})$ of neural network averaged over all runs for "w/" and "mPW" on the synthesized data. Lines with higher opacity represent results that are run on data with larger variance of noise.

To compare different methods across a diverse set of datasets, we synthesized 30 random datasets for each combination of all the following values: $N \in \{2, 3, 6, 12, 24\}$, $\sigma \in \{0.0025, 0.005, 0.01, 0.02, 0.04\}$, $T \in \{25, 50, 100, 200, 400\}$, and $\rho \in \{-0.9, -0.75, \ldots, 0.9\}$. Thus, there are a total of $48,750$ runs for each method.

5.2. Compared models, methods and metrics

The NN we used has six fully-connected layers with ReLU activation function and three residual connections. The learning rate is $5{\cdot}10^{-3}$ for θ and 10^{-2} for $\hat{\rho}$, both with Adam optimizer (Kingma & Ba, 2015). For each synthesized training set with T samples, we synthesize $100T$ samples as the testing set. 20% of the training set are split into the validation set, and the model with the best validation RSS is chosen.[2] We trained the NN with at most 750 epochs using our method with adjustment, modified Prais-Winsten, and standard MLE without adjustment, denoted as w/, mPW, and w/o, respectively.

Methods are compared in a pairwise manner. The winner of each run is the one with the smaller RSS on the testing set. The method that wins the most with statistical significance is considered as the better method. We also show the value of abs$(\rho - \hat{\rho})$ averaged over all runs for w/ and mPW.

5.3. Experimental results

Pairwise comparison of w/, mPW, and w/o are shown in the subfigures in Figure 1. From the figures, we see that on average, w/ is better than mPW, which is better than w/o. Moreover, looking at the left subfigure, we see that the outperformance is greater when the true ρ value has larger magnitude. Particularly, when abs$(\rho) > 0.15$, w/ beats mPW and w/o with statistical significance, and when abs$(\rho) \leq 0.15$, mPw is the best method on average, but all three methods have similar performances. Finally, observing the right subfigure in Figure 1, we see that adjusting for autocorrelated errors, using either w/ or mPW, has greater improvement when the noise has a larger scale.

In Figure 2, the value of abs$(\rho - \hat{\rho})$ averaged over all runs

[2] When N is small, to avoid good or bad luck, the validation RSS of the first 5 epochs are ignored.

Table 1. Root relative squared residual and average relative improvement of all combinations of models and datasets averaged over five runs. "w/o" implies without adjustment for autocorrelated errors, whereas "w/" implies with adjustment. Best performance in boldface and is superscribed with † if the p-value of paired t-test is lower than 5%. Average relative improvement is the percentage of improvement of "w/" over "w/o" averaged over all datasets for each model.

Models	LSTM		TPA		AGCRN		TCN		Conv-T		DSANet	
Datasets	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
PeMSD4	.2304	**.1960**†	.1742	**.1737**	.1718	**.1709**	.2203	**.1965**†	.2111	**.2055**†	.1775	**.1762**
PeMSD8	.1960	**.1586**†	.1392	**.1388**	.1370	**.1366**	.1809	**.1587**†	.1679	**.1516**†	.1409	**.1398**
Traffic	.4936	**.3643**†	.3559	**.3517**†	**.3326**†	.3339	.4657	**.3678**†	.4645	**.3711**†	.3454	**.3444**
ADI-920	.0469	**.0432**†	.0436	**.0419**†	.0491	**.0474**†	.0438	**.0421**†	.0681	**.0561**†	.0992	**.0599**†
ADI-945	.0060	**.0060**	.0545	**.0368**†	.0095	**.0084**†	.0065	**.0064**	.0358	**.0355**	.0709	**.0581**
M4-Hourly	.0444	**.0328**†	.0520	**.0416**†	**.0282**	.0284	.0358	**.0317**†	.0422	**.0414**	.0902	**.0773**†
M4-Daily	.8886	**.0429**†	.0485	**.0321**†	.0334	**.0311**†	.6802	**.0933**†	.6961	**.2942**†	.0309	**.0302**
M4-Weekly	1.241	**.5826**†	.2247	**.1758**	.5406	**.4431**†	1.065	**.4958**†	1.214	**.9989**†	.0412	**.0396**†
M4-Monthly	.9112	**.3868**†	.2777	**.1564**†	.1540	**.1308**†	.5715	**.3601**†	.8964	**.6939**†	.1171	**.1096**†
M4-Quarterly	.6536	**.4115**†	.2779	**.2067**†	.1776	**.1637**†	.4946	**.4184**†	.5207	**.4679**†	.1624	**.1439**†
M4-Yearly	.6177	**.4207**†	.4024	**.2662**	.3241	**.2446**†	.5910	**.4112**†	.4657	**.4203**	.1319	**.1135**
M5-L9	.2760	**.2260**†	.1898	**.1854**	.1959	**.1940**	.2754	**.2260**†	.2890	**.2616**†	**.1823**	.1824
M5-L10	.6035	**.3787**†	.3227	**.3155**	.3029	**.2999**	.5577	**.4471**†	.5603	**.4127**†	.3066	**.3043**†
Air-quality	.2014	**.1764**†	.1715	**.1677**†	.1700	**.1696**	.1926	**.1787**†	.1934	**.1706**†	.1695	**.1687**
Electricity	.0840	**.0793**†	.0648	**.0639**†	.0759	**.0719**†	.0799	**.0730**†	.0790	**.0741**†	**.0663**	.0664
Exchange	.0815	**.0188**†	.0509	**.0354**	.0124	**.0119**	.0822	**.0266**†	.0394	**.0366**	.0115	**.0109**†
Solar	.1517	**.1055**†	.1061	**.1052**†	.0994	**.0992**	.1407	**.1057**†	.1389	**.1071**†	.1064	**.1032**†
avg. rel. improv.		32.4%		15.1%		5.79%		25.3%		15.0%		7.12%

are shown. Similar to the outcomes in Figure 1, we see that w/ has better estimates of ρ when $\text{abs}(\rho) > 0.15$, and mPW has the upper hand otherwise. The reason, we believe, lies in step 2 in the mPW method. When the scale of ρ is small, there is little benefit in adjusting for autocorrelation, so the first iteration of step 2 in mPW will essentially train the model to near convergence just like the w/o method and will result in small-scale $\hat{\rho}$. In contrast, training $(\theta, \hat{\rho})$ together from the onset, as in w/, might result in a larger-scale $\hat{\rho}$ because the NN is not yet converged in the beginning.

6. Experiments on Time Series Forecasting

Our experiments on time series forecasting are conducted on five models and seventeen datasets, a total of eighty-five combinations. We show that by applying our method, in almost all combinations, the performance of the model improves. We also conduct an ablation study and grid-search over autocorrelation coefficient and model parameters to further illustrate the strength of our method.

6.1. Models

To demonstrate that our method is model-agnostic, we apply it on six different models introduced in Section 3.2, including basic and state-of-the-art models: (1) Long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997); (2) Temporal Pattern Attention (TPA) (Shih et al., 2019); (3) Adaptive Graph Convolutional Recurrent Network (AGCRN) (Bai et al., 2020); (4) Temporal Convolutional Network (TCN) (Bai et al., 2018); (5) Convolutional Transformer (Conv-T) (Li et al., 2019); and (6) Dual Self-Attention Network (DSANet) (Huang et al., 2019). These five models are chosen deliberately to include RNN, CNN, GNN, and Transformer, the four building blocks of NN for time series forecasting.

Table 2. Root relative squared residual for the best grid-searched hyperparameters averaged over ten runs. Best performance in boldface and is superscribed with † if the p-value of paired t-test is lower than 5%.

Datasets	Models	w/o	w/
M4-Hourly	AGCRN	.0275	**.0267**†
Traffic	AGCRN	**.3325**†	.3332
M5-L9	DSANet	.1765	**.1763**
Electricity	DSANet	.0651	**.0641**†

6.2. Datasets

A wide range of datasets are explored to show that the benefit of applying our method is not dependent on the dataset. We categorize these datasets into five categories:

1. Traffic (PeMSD4, PeMSD8, Traffic) (Lai et al., 2018; Guo et al., 2019): road occupancy rate. PeMSD4 and PeMSD8 are used in AGCRN. Traffic is used in both TPA and Conv-T.

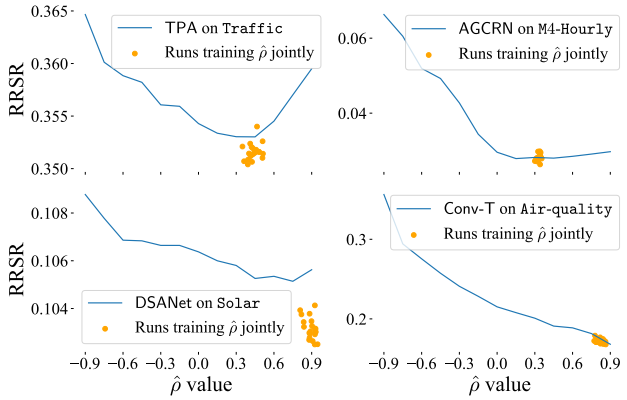

Figure 3. RRSR of four combinations of models and datasets averaged over 20 runs when $\hat{\rho}$ is fixed at a value ranging from -0.90 to 0.90. Orange dots are the results when $\hat{\rho}$ is not fixed and is instead learned jointly with model parameter θ.

2. Manufacturing (`ADI-920`, `ADI-945`): sensor values during semiconductor manufacturing process.
3. M4-competition (`M4-Yearly`, `M4-Quarterly`, `M4-Monthly`, `M4-Weekly`, `M4-Daily`, `M4-Hourly`) (Makridakis et al., 2020a): re-arranged data from the M4-competition where each dataset consists of series from microeconomics, macroeconomics, financial, industry, demographic, and other. `M4-Hourly` is used in Conv-T.
4. M5-competition (`M5-L9`, `M5-L10`) (Makridakis et al., 2020b): aggregated data from the M5-competition representing Walmart sales.
5. Miscellaneous (`Air-quality`, `Electricity`, `Exchange-rate`, `Solar`) (Lai et al., 2018): other datasets. `Electricity` and `Solar` are used in both Conv-T and TPA. `Exchange-rate` is used in TPA.

Each dataset is split into training (60%), validation (20%), and testing (20%) set in chronological order. In data preprocessing, the data is normalized by the mean and variance of the whole training set. The detailed description and statistics of the datasets can be found in Appendix B.

6.3. Compared metrics

Our compared metrics is the root relative squared residual (RRSR) on the testing set, defined as

$$\text{RRSR} = \frac{\sqrt{\sum_{t\in\text{testing}} \|y_t - \hat{y}_t\|_2^2}}{\sqrt{\sum_{t\in\text{testing}} \|y_t - \bar{y}\|_2^2}}, \tag{18}$$

where $y_t = \mathbf{X}_t$, $\hat{y}_t$ is the prediction of the model, and $\bar{y}$ is the mean value of the whole testing set. The benefit of using RRSR is to scale the residuals so the outcomes are more readable, regardless of the scale of the dataset.

To aggregate results over multiple datasets, we also calculate

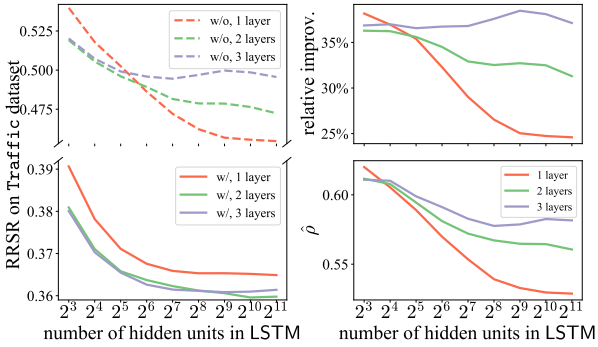

Figure 4. Results of training LSTM on the `Traffic` dataset with different numbers of layers and hidden units averaged over 30 runs using both the w/o and w/ methods. In the upper right subfigure, y-axis is the relative improvement of using w/ over w/o.

the averaged relative improvement, defined as

$$\frac{1}{D}\sum_{d=1}^{D} \frac{(\overline{\text{RRSR}}_{\text{w/o, d}} - \overline{\text{RRSR}}_{\text{w/, d}})}{\overline{\text{RRSR}}_{\text{w/o, d}}} \cdot 100\%, \tag{19}$$

where D is the number of datasets, w/o denotes training without adjustment, w/ denotes with adjustment, and $\overline{\text{RRSR}}$ is the averaged RRSR over multiple runs.

6.4. Evaluation on all combinations

We run all combinations of models and datasets with or without our method of adjusting for autocorrelated errors. Due to physical limitations, we cannot fine-tune or do grid-search for every model and dataset. Hence, across all runs, number of epochs is 750 with early-stopping if validation does not improve for 25 consecutive epochs, batch size is 64, window size W is 60, learning rate is $3 \cdot 10^{-3}$ for model parameters and 10^{-2} for $\hat{\rho}$, both with Adam (Kingma & Ba, 2015) optimizer. Two exceptions are AGCRN on `M5-L10`, which is trained with $W = 30$ due to GPU memory limit, and all Conv-T, which are trained with 10^{-3} learning rate that yields much better outcomes. Other model-specific hyperparameters, which mostly follow each original paper for that model if possible, are also fixed across all datasets. For simplicity, we set $\hat{\rho}$ as an N-dimensional vector when $N \geq 300$ and $\hat{\rho}$ as a scalar otherwise. Loss function is mean RSS on the normalized data[3]. Detailed listing of hyperparameters can be found in Appendix C.

The results are shown in Table 1. In all but four cases, our method improves the performance of the model. Even in the event that our method is ineffective, the deterioration is small. Notice that the averaged relative improvement is more significant on LSTM than on AGCRN or DSANet. We conjecture that AGCRN and DSANet are better designed for time series than LSTM, so the autocorrelated errors resulting from model mis-specification are less severe. Experiments in Section 6.7 further support this conjecture.

[3]Equivalent to the mean squared error.

Table 3. Averaged relative improvement of three adjustment types compared to no adjustment at all. Best performance in boldface and is superscribed with † if the p-value of paired t-test between the first and second best method is lower than 5%.

Models	TPA				AGCRN				DSANet			
Datasets	w/o	w/ inp.	w/ out.	w/	w/o	w/ inp.	w/ out.	w/	w/o	w/ inp.	w/ out.	w/
PeMSD8	.1392	.1392	.1392	**.1388**	.1370	.1375	.1373	**.1366**†	.1409	.1408	.1408	**.1398**†
ADI-945	.0545	.0452	.0425	**.0368**†	.0095	**.0082**	.0085	.0084	.0709	.0592	.0714	**.0581**
M4-Hourly	.0520	**.0394**†	.0415	.0416	.0282	.0277	**.0272**†	.0284	.0902	.0886	.0910	**.0773**†
M5-L9	.1898	.1915	.1922	**.1854**†	.1959	.1949	.1945	**.1925**†	.1823	**.1822**	.1823	.1824
Exchange	.0509	.0437	.0401	**.0354**†	.0124	.0136	.0130	**.0119**†	.0115	.0115	.0121	**.0109**†
Solar	.1061	.1054	.1061	**.1052**†	.0994	.1002	.0999	**.0992**	.1064	.1033	.1063	**.1032**
avg. rel. improv.		9.23%	10.4%	**14.4%**		1.00%	1.54%	**2.86%**		3.52%	-1.21%	**6.86%**

6.5. Grid-searching hyperparameters

To further verify the advantages of our method, we select the four cases in Table 1 where our method underperformed, and grid-search hyperparameters on them. For all models, the choices are $\{10^{-3}, 3 \cdot 10^{-3}\}$ for learning rate and $\{32, 64\}$ for batch size. For AGCRN, hyperparameters are searched over $\{32, 64\}$ for hidden units and $\{2, 10\}$ for embedding dimension. For DSANet, filter length in local convolution are chosen from $\{3, 5, 7\}$ and number of layers from $\{1, 2\}$. Other hyperparameters follow those in Section 6.4.

The best achieved RRSRs are shown in Table 2. Out of the four combinations, our method produces better outcomes in three, which implies that the improvement is consistent across different hyperparameters. Notice that AGCRN is specifically designed for traffic forecasting, which explains why it is challenging to improve on top of it when the target dataset is traffic-related, as also shown in Table 1.

6.6. Grid-searching $\hat{\rho}$

Similar to the Hildreth-Lu method as described in Section 3.1, we grid-search over possible values of $\hat{\rho}$ and then fix it during training in order to find the optimal $\hat{\rho}$. The search range is $\{-0.9, -0.75, \ldots, 0.9\}$ and the values are the same across all dimensions of $\hat{\rho}$. Notice that fixing $\hat{\rho} = 0$ is equivalent to the w/o method.

We run four combinations and the results are shown in Figure 3. It is clear that fixing $\hat{\rho}$ at some positive number yields better results compared to the w/o method. This shows that adjusting for autocorrelated errors do enhance the model performance. In addition, when $\hat{\rho}$ are trained jointly with θ, the learned $\hat{\rho}$ is close to the optimal $\hat{\rho}$ searched. Plus, jointly training $\hat{\rho}$ and θ may result in even better RRSR compared to grid-search because all trainable parameters are optimized simultaneously.

6.7. Effect of model mis-specification

Here, we train LSTM with different numbers of layers and hidden units on the Traffic dataset to explore the effect of model mis-specification. We expect that, with the same number of layers, LSTM with more hidden units is more expressive so the adjustment for autocorrelation is less beneficial. The results are shown in Figure 4.

In the left subfigures, we see that the RRSR decreases when the number of hidden units increases in both w/o and w/ methods. However, the relative improvement of applying w/ over w/o decreases, as shown in the upper right subfigure. Meanwhile, in the bottom right subfigure, $\hat{\rho}$ also decreases. All these observations combined indicate that when LSTM is more expressive, the need for adjustment reduces as $\hat{\rho}$ decreases, and the advantage of using w/ over w/o weakens as relative improvement decreases.

6.8. Ablation study

In our adjustment for autocorrelated errors for time series forecasting, we adjust for both the input and the output. Here, we study the effect of adjusting for only input or output – that is, fixing $\hat{\rho}$ to 0 for the input or output part. The results are shown in Table 3, which is a subset of Table 1 because of limited computational resources. From the table, we see that adding either input or output adjustment for autocorrelated errors improves performance. Nevertheless, adding both results in the best overall improvement.

7. Conclusion

In this paper, we propose to adjust for autocorrelated errors by learning the autocorrelation coefficient jointly with the model parameters. Our method can be applied on any neural network to improve its performance on any time series dataset. Experimental results on time series regression show the benefits of using our method especially when the autocorrelation is strong. Additionally, results on time series forecasting demonstrate that applying our method in existing state-of-the-art models further enhances their performance across a wide variety of datasets. Experiments are conducted to showcase the need for adjustment and to support the superiority of our method. For future research directions, we can explore our method on other time series tasks, such as time series classification, and also extend it to forecast more than one time step ahead.

Acknowledgements

We thank John Yamartino and Ramana Veerasingam from Lam Research for their helpful discussions and insights. This work has been supported in part by gifts from Lam Research.

References

Abraham, B. and Ledolter, J. *Statistical methods for forecasting*. John Wiley & Sons, New York, 1983.

Bai, L., Yao, L., Li, C., Wang, X., and Wang, C. Adaptive graph convolutional recurrent network for traffic forecasting. In *Advances in Neural Information Processing Systems*, 2020.

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

Beach, C. M. and MacKinnon, J. G. A maximum likelihood procedure for regression with autocorrelated errors. *Econometrica*, 46:51–58, 1978.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., and Januschowski, T. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:1803.01271*, 2020.

Box, G. E., Jenkins, G. M., and Reinsel, G. C. *Time Series Analysis: Forecasting and Control (2nd ed.)*. John Wiley & Sons, San Francisco, 1976.

Breusch, T. S. Testing for autocorrelation in dynamic linear models. *Australian Economic Papers*, 17:334–355, 1978.

Chalapathy, R. and Chawla, S. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1803.01271*, 2019.

Cochrane, D. and Orcutt, G. H. Application of least squares regression to relationships containing auto-correlated error terms. *Journal of the American Statistical Association*, 44:32–61, 1949.

Dufour, J. M., Gaudry, M. J. I., and Liem, T. C. The cochrane-orcutt procedure numerical examples of multiple admissible minima. *Economics Letters*, 6:43–48, 1980.

Dufour, J. M., Gaudry, M. J. I., and Hafer, R. W. A warning on the use of the cochrane-orcutt procedure based on a money demand equation. *Empirical Economics*, 8:111–117, 1983.

Durbin, J. and Watson, G. S. Testing for serial correlation in least squares regression. *Biometrika*, 38:159–177, 1951.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33: 917–963, 2019.

Fletcher, R. *Practical Methods of Optimization (2nd ed.)*. John Wiley & Sons, New York, 1987.

Frydman, R. A proof of the consistency of maximum likelihood estimators of nonlinear regression models with autocorrelated errors. *Econometrica*, 48:853–860, 1980.

Gallant, A. R. and Goebel, J. J. Nonlinear regression with autocorrelated errors. *Journal of the American Statistical Association*, 71:961–967, 1976.

Gamboa, J. Deep learning for time-series analysis. *arXiv preprint arXiv:1803.01271*, 2017.

Glasbey, C. A. Correlated residuals in non-linear regression applied to growth data. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28:251–259, 1979.

Godfrey, L. G. Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables. *Econometrica*, 46: 1293–1301, 1978.

Greene, W. H. *Econometric Analysis (8th ed.)*. Pearson, New York, 2017.

Guo, S., Lin, Y., Feng, N., Song, C., , and Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proc. of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 922–929, 2019.

Hildreth, C. and Lu, J. Y. Demand relations with autocorrelated disturbances. *Technical Bulletin, Michigan State University, Agricultural Station*, 276, 1960.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences*, 79:2554–2558, 1982.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Network*, 2:359–366, 1989.

Huang, S., Wang, D., Wu, X., and Tang, A. Dsanet: Dual self-attention network for multivariate time series forecasting. In *ACM International Conference on Information and Knowledge Management*, pp. 2129–2132, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In *The International ACM SIGIR Conference*, pp. 95–104, 2018.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, pp. 5243–5253, 2019.

Ljung, G. M. and Box, G. E. P. On a measure of a lack of fit in time series models. *Biometrika*, 65:297–303, 1978.

Loizou, P. C. *Speech enhancement: theory and practice*. CRC Press, 2013.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36:54–74, 2020a.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. The m5 accuracy competition: results, findings and conclusions. *URL https://www. researchgate. net/publication/344487258_The_M5_Accuracy_ competition_Results_findings_and_conclusions*, 2020b.

Montgomery, D. C., Peck, E. A., and Vining, G. *Introduction to Linear Regression Analysis (5th ed.)*. Wiley, New York, 2012.

Oxley, L. T. and Roberts, C. J. Pitfalls in the application of the cochrane-orcutt technique. *Oxford Bulletin of Economics and Statistics*, 44:227–240, 1982.

Prais, S. J. and Winsten, C. B. Trend estimators and serial correlation. *Cowles Commission Discussion Paper No. 383*, 1954.

Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., and Cottrell, G. W. A dual-stage attention-based recurrent neural network for time series prediction. In *International Joint Conference on Artificial Intelligence*, pp. 2627–2633, 2017.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36:1181–1191, 2020.

Sargan, J. D. Wages and prices in the united kingdom: a study in econometric methodology. In *Econometric Analysis for National Economic Planning*, 1964.

Sen, R., Yu, H.-F., and Dhillon, I. S. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Advances in Neural Information Processing Systems*, pp. 4837–4846, 2019.

Shih, S.-Y., Sun, F.-K., and Lee, H.-Y. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108:1421–1441, 2019.

Vaswani, A., Parmar, N. S. N., Uszkoreit, J., Jones, L., Gomez, A. N., ukasz Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Wright, S. J. Coordinate descent algorithms. *Mathematical Programming*, 151:3–34, 2015.

Xu, Y., Du, J., Dai, L.-R., and Lee, C.-H. An experimental study on speech enhancement based on deep neural networks. *IEEE Signal processing letters*, 21:65–68, 2013.

Yu, H.-F., Rao, N., and Dhillon, I. S. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in Neural Information Processing Systems*, pp. 847–855, 2016.

Appendices

A. Details of NNs applied to the synthesized data

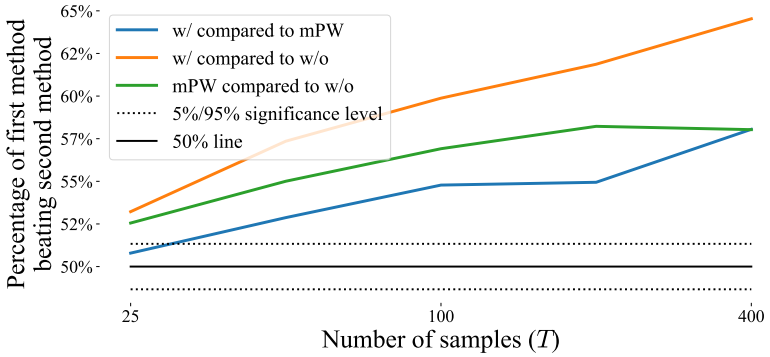

Figure 5. Pairwise comparison of three methods of training NN on the synthesized data with different number of samples T. With more samples, the outperformance is more consistence, comparing to small-sample training where there are huge variances.

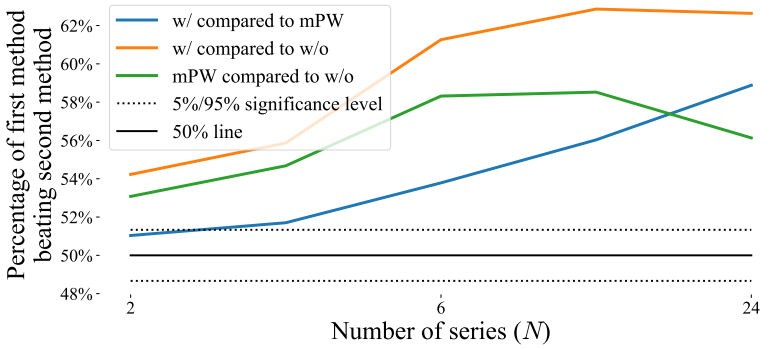

Figure 6. Pairwise comparison of three methods of training NN on the synthesized data with different number of series N. With more series, more overfitting when autocorrelated errors are not adjusted.

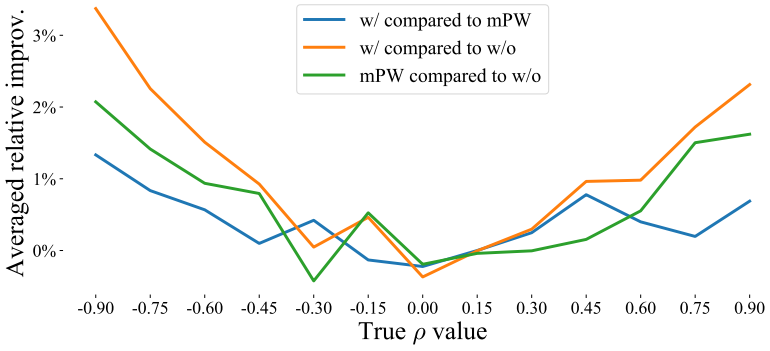

Figure 7. Pairwise comparison of averaged relative improvement of three methods of training NN on the synthesized data with different true ρ value. Results are similar to the one where y-axis is percentage of outperformance.

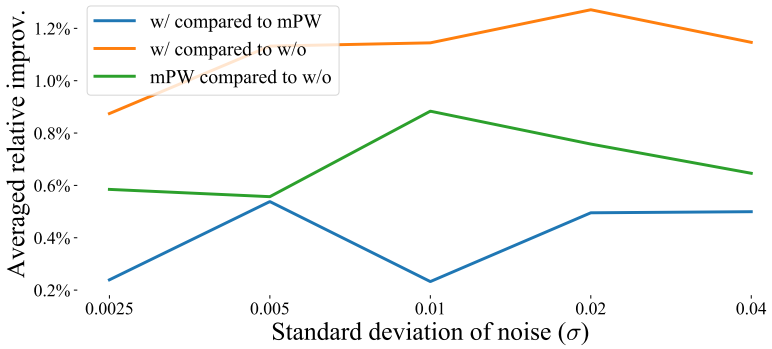

Figure 8. Pairwise comparison of averaged relative improvement of three methods of training NN on the synthesized data with different standard deviation of noise (σ). Results are similar to the one where y-axis is percentage of outperformance.

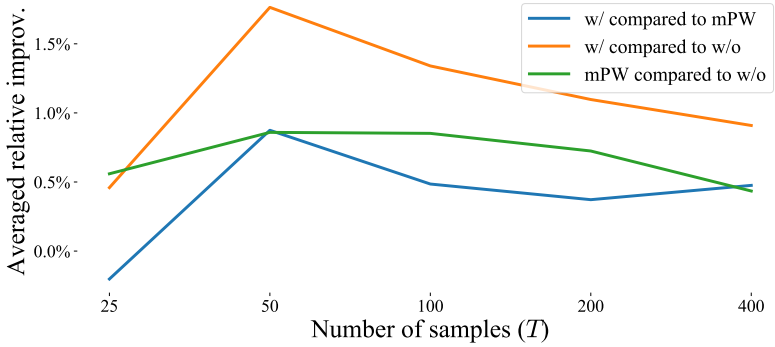

Figure 9. Pairwise comparison of averaged relative improvement of three methods of training NN on the synthesized data with different number of samples (T). Results are similar to the one where y-axis is percentage of outperformance, except that when the number of samples increases, the improvement decreases because all methods have similar performances when number of samples approaches infinity.

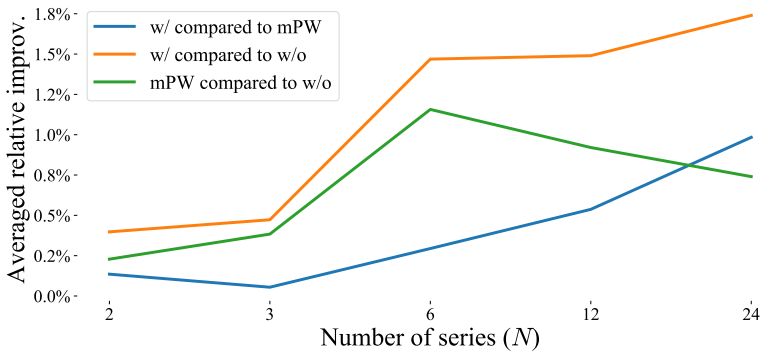

Figure 10. Pairwise comparison of averaged relative improvement of three methods of training NN on the synthesized data with different number of series (N). Results are similar to the one where y-axis is percentage of outperformance.

B. Detailed descriptions and statistics of real-world datasets

- `PeMSD4`: Traffic data in San Francisco Bay Area from January 2018 to February 2018. Only the total traffic flow series are used.
- `PeMSD8`: Similar to `PeMSD4`, but recorded in San Bernardino from July 2016 to August 2016.
- `Traffic` [4]: Data from the California Department of Transportation describing road occupancy rates, a number between 0 and 1, of the San Francisco Bay area freeways from 2015 to 2016.
- `ADI-920`, `ADI-945`: Sensor values recorded from the plasma etcher machine in Analog Device Inc. The raw data is in three-dimension because it is split up by wafer cycles, so we concatenate all wafer cycles together and batch it without using data points from different wafer cycles. `ADI-920` and `ADI-945` represent two different recipes.
- `M4-Hourly`, `M4-Daily`, `M4-Weekly`, `M4-Monthly`, `M4-Quarterly`, `M4-Yearly`: Six datasets from the M4 competition with different sampling rate. Each dataset contains miscellaneous series, categorized into six domains (micro, industry, macro, finance, demographic, other). Originally, each series has different length and start time, but we crop out part of each dataset so every series has the same start time and length without missing value.
- `M5-L9`, `M5-L10`: Raw data are Walmart unit sales of 3,049 products sold in ten stores in three States (CA, TX, WI). The products can be categorized into 3 product categories (Hobbies, Foods, and Household) and 7 product departments. `M5-L9` is the level-9 aggregation and `M5-L10` is the level-10 aggregation [5].
- `Air-quality` [6]: Recorded by gas multisensor devices deployed on the field in an Italian city.
- `Electricity` [7]: Electricity consumption in kWh from 2012 to 2014.
- `Exchange`: exchange rate of eight countries (Australia, British, Canada, Switzerland, China, Japan, New Zealand, and Singapore) from 1990 to 2016.
- `Solar` [8]: Solar power production records in 2006 from photovoltaic power plants in Alabama State.

Datasets	length of dataset T	number of series N	sampling rate
`PeMSD4`	16,992	307	5 minute
`PeMSD8`	17,856	170	5 minute
`Traffic`	17,544	862	1 hour
`ADI-920`	1,440,493	30	500 millisecond
`ADI-945`	515,366	30	500 millisecond
`M4-Hourly`	744	121	1 hour
`M4-Daily`	4,208	1,493	1 day
`M4-Weekly`	2,191	43	1 week
`M4-Monthly`	816	203	1 month
`M4-Quarterly`	674	27	1 quarter
`M4-Yearly`	618	9	1 year
`M5-L9`	1,941	70	1 day
`M5-L10`	1,941	3049	1 day
`Air-quality`	9,357	13	1 hour
`Electricity`	26,304	321	1 hour
`Exchange`	7,588	8	1 day
`Solar`	52,560	137	10 minutes

[4] http://pems.dot.ca
[5] Please see the competition guidelines (https://mofc.unic.ac.cy/m5-competition) for definition of the aggregations.
[6] https://archive.ics.uci.edu/ml/datasets/Air+quality
[7] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams2011
[8] http://www.nrel.gov/grid/solar-power-data

C. Hyperparameters of all NNs for time series forecasting

- LSTM: number of layers = 2, hidden size = 64.
- TPA: number of layers = 1, hidden size = 64, linear autoregressive size = 24.
- AGCRN: number of layers = 1, hidden size = 64, embedding dimension = 10.
- TCN: number of layers = 9, hidden size = 64.
- Conv-T: number of layers = 3, number of head in attention = 8, hidden size = 256, filter kernel size = 6.
- DSANet: number of layers = 1, local temporal filter size = 3, number of channels = 32, dropout = 0.1.