Towards Better Forecasting by Fusing Near and Distant Future Visions

Jiezhu Cheng,[1,2] **Kaizhu Huang,**[3,4] **Zibin Zheng**[1,2]
[1]Sun Yat-sen University, School of Data and Computer Science, Guangzhou, China
[2]Sun Yat-sen University, National Engineering Research Center of Digital Life, Guangzhou, China
[3]Xi'an Jiaotong-Liverpool University, Department of Electrical and Electronic Engineering, Suzhou, China
[4]Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Hangzhou, China
chengjzh@mail2.sysu.edu.cn, kaizhu.huang@xjtlu.edu.cn, zibinzheng2@yeah.net

Abstract

Multivariate time series forecasting is an important yet challenging problem in machine learning. Most existing approaches only forecast the series value of one future moment, ignoring the interactions between predictions of future moments with different temporal distance. Such a deficiency probably prevents the model from getting enough information about the future, thus limiting the forecasting accuracy. To address this problem, we propose *Multi-Level Construal Neural Network* (MLCNN), a novel multi-task deep learning framework. Inspired by the Construal Level Theory of psychology, this model aims to improve the predictive performance by fusing forecasting information (i.e., future visions) of different future time. We first use the Convolution Neural Network to extract multi-level abstract representations of the raw data for near and distant future predictions. We then model the interplay between multiple predictive tasks and fuse their future visions through a modified Encoder-Decoder architecture. Finally, we combine traditional Autoregression model with the neural network to solve the scale insensitive problem. Experiments on three real-world datasets show that our method achieves statistically significant improvements compared to the most state-of-the-art baseline methods, with average 4.59% reduction on RMSE metric and average 6.87% reduction on MAE metric.

Introduction

Multivariate time series consists of experimental data with multiple variables observed at different points in time. They occur everywhere in our daily life, from the energy consumption, the traffic flow, to the stock prices. In such fields, effective decision often requires accurate prediction on relevant time series data. For example, knowing demand for electricity in the next few hours could help us devise a better energy use plan, and forecasting of stock market in the near or distant future could produce more profit.

Multivariate time series forecasting focuses on predicting the future outcomes of each variable given their past. As it is difficult to estimate exact future values, it is generally considered that the future observations can be subject to a conditional probability distribution of the past observations. In this case, the conditional expectation of the distribution can be given as a function of the past values:

$$\mathbb{E}[X_{t+h}|X_t, ..., X_{t-p+1}] = f(X_t, ..., X_{t-p+1}) \ . \quad (1)$$

For simplicity, we use X_{t+h} to represent the conditional mean $\mathbb{E}[X_{t+h}|X_t, ..., X_{t-p+1}]$ in later descriptions.

Researchers have been studying the forecasting problem (1) for years, developing all kinds of linear, non-linear and hybrid models for better predictions (Adhikari and Agrawal 2013; Khashei and Bijari 2011). However, given the past values, most of these models only estimate the conditional mean at the future moment X_{t+h} or in a continuous future window $\{X_{t+1}, ..., X_{t+h}\}$, using a single model architecture without considering the link between predictions of different future moments. This drawback may limit the models' generalization ability, since only one kind of vision about the future is obtained. Figure 1 shows three predictive tasks on X_{t+h}, X_{t+h-i} and X_{t+h+i}, where $0 < i < h$. Although they are performed based on the same past observations, different temporal distances from future observations give distinct future vision to each task. Current forecasting methods such as AR (Box and Jenkins 1970), AE-CRNN (Cirstea et al. 2018), DARNN (Qin et al. 2017) and LSTNet (Lai et al. 2017) perform these tasks independently with a single model architecture, ignoring the interplay between them. To our knowledge, there are few methods that model the interactions between multiple predictive tasks and fuse their future visions to improve the main task.

In this paper, we investigate whether the fusion of near and distant future visions could improve the performance of the main predictive task, as shown in Figure 1. Inspired by the Construal Level Theory (CLT) (Liberman and Trope 1998) revealing that people use different levels of abstract construals to predict future events, we propose a novel multi-task deep learning framework called *Multi-Level Construal Neural Network* (MLCNN) to perform multivariate time series forecasting. It first leverages a Convolution Neural Network (CNN) (Lecun et al. 1998) to extract multi-level feature abstractions from the raw time series and engages them for multiple predictive tasks. Next, the extracted abstractions are fed into a shared Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997), which captures complex long-term dependencies of the time series and

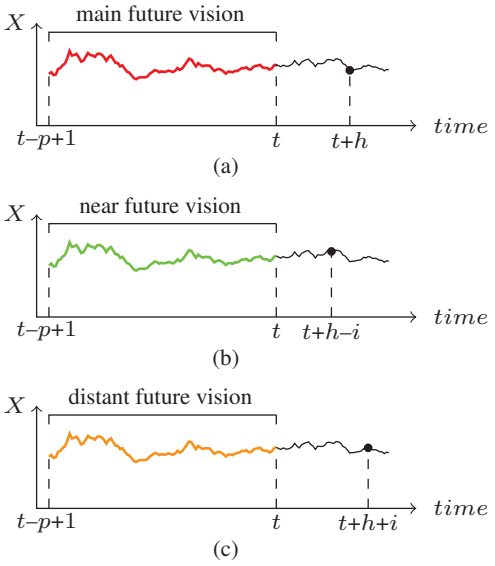

Figure 1: (a) Main predictive task on X_{t+h}. (b) Auxiliary predictive task on X_{t+h-i} with near future vision. (c) Auxiliary predictive task on X_{t+h+i} with distant future vision.

fuses the future visions of different predictive tasks. In addition, we design another main LSTM for the main predictive task, utilizing the feature abstraction and the shared information of the primary task to make more accurate prediction. In this case, the shared LSTM works as an encoder for the features of the primary task and the main LSTM works as a decoder. Finally, similar to the method proposed by (Lai et al. 2017), to deal with the scale changing problem of input data, we combine traditional autoregressive linear models with the non-linear part of neural networks to make our MLCNN more robust. Our contributions are of three-folds:

- Based on the Construal Level Theory about human predictive behavior, we design an effective extraction-sharing mechanism to construct and fuse the future visions of different forecasting tasks, and demonstrate its capabilities of improving the main predictive task.
- We develop a novel multi-task deep learning model with good generalization ability for multivariate time series forecasting.
- We conduct extensive experiments on three real-world datasets and show the advantages of our model against most state-of-the-art baseline methods, demonstrating new benchmark on the public datasets. All the data and experiment codes of our model are available at Github[1].

[1]https://github.com/smallGum/MLCNN-Multivariate-Time-Series

Related Work

Time Series Forecasting Research on time series forecasting has a long history. One of the most popular models is the Autoregression (AR) model. The variants of AR model such as the Moving Average (MA), Autoregressive Integrated Moving Average (ARIMA), and Vector Autoregression (VAR) models are also widely used (Box and Jenkins 1970). However, the AR model and its variants fall short in capturing the non-linear features of the time series signals due to their linear assumption about the data (Adhikari and Agrawal 2013). To address this problem, various non-linear models have been proposed, such as Factorization Machine (FM) (Chen et al. 2018; Wu et al. 2017; Yang et al. 2018) and Support Vector Regression (SVR) (Yang et al. 2009). Nevertheless, the number of parameters in these models grows quadratically over the temporal window size and the number of variables, implying large computational cost and high risk of overfitting when dealing with high dimensional multivariate time series.

Recently, Deep Neural Networks (DNNs) have attracted increasing attentions in the domain of time series forecasting, due to their great success in capturing non-linear data features. The first widely used models are Multi-Layer Perceptrons (MLPs) (Zhang, Patuwo, and Hu 1998), which learn the non-linear relationships of the input series through fully connected hidden layers. Later, Recurrent Neural Networks (RNNs) are known for their advantages in sequence learning (Sutskever, Vinyals, and Le 2014). In order to solve the vanishing gradients problem (Bengio, Simard, and Frasconi. 1994) when using RNNs to learn long-term dependencies, the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) and the Attention (Bahdanau, Cho, and Bengio 2014) models have been proposed and achieved thrilling results on univariate time series forecasting with multiple driving series (Liang et al. 2018; Qin et al. 2017). Besides, Convolutional Neural Networks (CNNs) (Lecun et al. 1998) have also found their significance on asynchronous time series prediction (Binkowski, Marti, and Donnat 2018). Furthermore, both theoretical and empirical findings have suggested that combining autoregressive linear models with non-linear DNNs can effectively improve the predictive performance (Khashei and Bijari 2011; Lai et al. 2017). Such a hybrid method is also adopted by our MLCNN model.

Construal Level Theory During the centuries, psychologists have conducted a large amount of researches on how individuals predict the future and the factors that influence those predictions (Griffin, Dunning, and Ross 1990). Particularly, Construal Level Theory (CLT) and its following study have been trying to reveal how temporal distance from future outcomes affect people's predictions (Liberman and Trope 1998; Trope and Liberman 2000). CLT assumes that individuals' predictions of future events depend on how they mentally construe those events. According to CLT, people tend to use higher level, more abstract construals to represent distant future events than to represent near future events (Nussbaum, Liberman, and Trope 2006). For multivariate time series forecasting, CLT inspires us to extract more abstract representations of data for distant future predictions and more specific features for near future predictions.

CLT is the core of the proposed architecture in this paper. Our MLCNN model uses a multi-layer CNN to extract discriminative features of the raw time series at different convolutional layers, forming the *construals* of different abstraction levels. The low- and high-level construals are respectively used for near and distant future predictions, thus producing near and distant future visions for the fusion model.

Multi-Task Deep Learning Multi-task learning (MTL) (Caruana. 1997) aims to train multiple tasks in parallel, so as to improve the performance of the main task with training signals from other related tasks. MTL in deep neural networks, called multi-task deep learning, have achieved significant results in many areas of artificial intelligence (Ruder 2017). However, literature on multi-task deep learning for time series prediction is still scarce, mainly due to the difficulties of finding proper auxiliary tasks. (Cirstea et al. 2018) proposed an MTL model AECRNN to perform univariate time series forecasting with related driving series, while few literature apply it on multivariate time series forecasting.

Our MLCNN model is a natural multi-task deep learning framework for multivariate time series forecasting. We choose the near and far future predictive tasks defined in Figure 1 as the auxiliary tasks and fuse their forecasting information to improve the main task. We demonstrate the superiority of this method through extensive experiments.

Model Architecture

In this section, we first formulate the problem at hand, and then present the proposed MLCNN architecture. Finally, we introduce the loss function and the optimization algorithm used by our model.

Problem Statement

In this paper, we focus on the task of multivariate time series forecasting. More formally, given time series $\mathcal{X} = \{X_{t-p+1}, ..., X_t\}$, where $X_i \in \mathbb{R}^n$ and n is the variable dimension, we are interested in predicting the value of the series at a certain future moment, that is, predicting the value of X_{t+h}, where $h \geq 1$ is the desirable horizon ahead of the current time stamp. In practice, the horizon h is chosen according to the demands of the environmental settings.

Besides, we define two notations fsp, namely *future span* and fst, namely *future stride* to help specify auxiliary forecasting tasks, where $0 < fsp \cdot fst < h$. Therefore, while performing prediction on the series value at the future moment $t+h$ as the main task, we also perform predictions at future moments $\{t+h-(fsp \cdot fst), ..., t+h-fst, t+h+fst, ..., t+h+(fsp \cdot fst)\}$ as auxiliary tasks. Without loss of generality, we set $fsp = 2$ and $fst = 1$ by default. That being said, assuming $\{X_{t-p+1}, ..., X_t\}$ are available, we predict the values of $\{X_{t+h-2}, X_{t+h-1}, X_{t+h}, X_{t+h+1}, X_{t+h+2}\}$ in parallel, forming a five-task learning problem. Among the five tasks, predictions on X_{t+h-2} and X_{t+h-1} have near future visions while predictions on X_{t+h+1} and X_{t+h+2} have distant future visions.

Convolutional Component

The first part of MLCNN is a multi-layer CNN (Lecun et al. 1998), where different layers extract different abstract features from the input data and deeper layers produce more abstract information. The CNN component aims to learn the local dependencies between variables and manufacture construals of different abstraction levels for multiple predictive tasks. As shown in Figure 2, for the five-task forecasting problem described above, we use the CNN to create five different construals:

$$\begin{aligned} C_{t+h-2} &= f_1(\mathbf{X_t^{-P}}) \\ C_{t+h-1} &= f_2(C_{t+h-2}) \\ C_{t+h} &= f_3(C_{t+h-1}) \\ C_{t+h+1} &= f_4(C_{t+h}) \\ C_{t+h+2} &= f_5(C_{t+h+1}) \ , \end{aligned} \tag{2}$$

where

- $\mathbf{X_t^{-P}} = [X_{t-p+1}; X_{t-p+2}; ...; X_t] \in \mathbb{R}^{p \times n}$ is the matrix of the given multivariate time series. n is the number of variables and p denotes the number of time points.
- $f_i : \begin{cases} \mathbb{R}^{p \times n} \to \mathbb{R}^{p \times m} & i = 1 \\ \mathbb{R}^{p \times m} \to \mathbb{R}^{p \times m} & i = 2, 3, 4, 5 \end{cases}$ are one-dimensional convolutional layers (Conv1D) with m filters in the CNN, and layer f_{i+1} is deeper than layer f_i.
- $C_{t+h-2}, ..., C_{t+h+2} \in \mathbb{R}^{p \times m}$ are extracted construals used for predictive tasks on $X_{t+h-2}, ..., X_{t+h+2}$, respectively. Dropout operation (Wu and Gu 2015) is also applied on every construal to avoid overfitting.

In addition, each filter in the CNN is $W_k \in \mathbb{R}^{w \times n}$ (the height of the filter is set to be the same as the variable dimension). The k-th filter sweeps through the input matrix X and produces:

$$c_k = Act(W_k * X + b_k) \ , \tag{3}$$

where $*$ denotes the convolution operation and c_k is the output vector. Act could be any activation function. In this paper, we empirically found that the $LeakyReLU$ function $LeakyReLU(x) = \begin{cases} x & x \geq 0 \\ \alpha x & \text{otherwise} \end{cases}$ with leak rate $\alpha = .01$ fits most data well. We make each vector c_k of length p by zero-padding on the matrix X.

Shared Recurrent Component

The construals $C_{t+h-2}, ..., C_{t+h+2}$ of multiple abstraction levels are then fed into a shared RNN one after another. The recurrent component is an LSTM (Hochreiter and Schmidhuber 1997) with the $\tanh$ function as the hidden update activation function. It captures the long-term dependencies of the time series and models the interactions between different predictive tasks, as shown in Figure 3 Part I. The hidden state of recurrent units at time τ for the k-th construal is

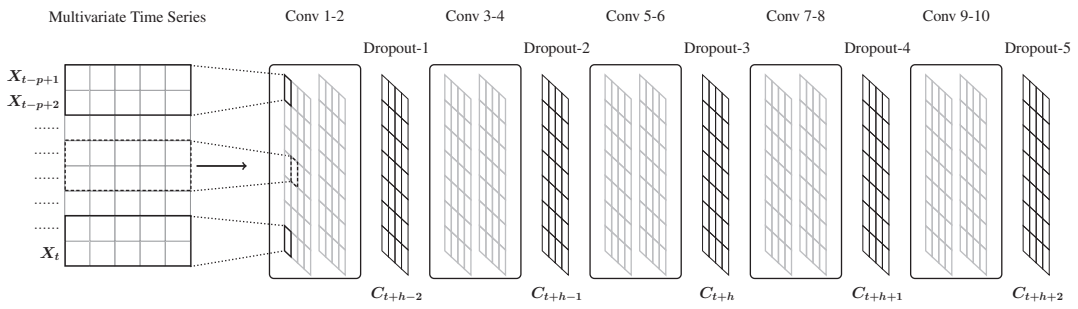

Figure 2: A 10-layer CNN to extract multi-level construals of the raw data

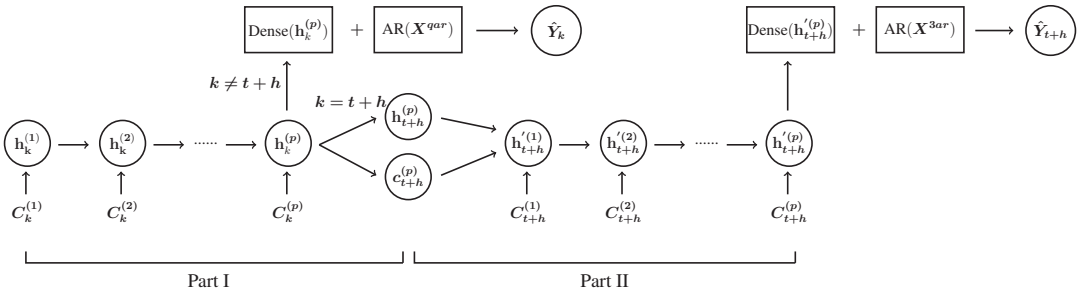

Figure 3: Part I: The shared LSTM for all tasks; Part II: The main LSTM for main task

computed as:

$$
\begin{aligned}
i_k^{(\tau)} &= \sigma(W_{ii}C_k^{(\tau)} + b_{ii} + W_{hi}\mathbf{h}_k^{(\tau-1)} + b_{hi}) \\
f_k^{(\tau)} &= \sigma(W_{if}C_k^{(\tau)} + b_{if} + W_{hf}\mathbf{h}_k^{(\tau-1)} + b_{hf}) \\
g_k^{(\tau)} &= \tanh(W_{ig}C_k^{(\tau)} + b_{ig} + W_{hg}\mathbf{h}_k^{(\tau-1)} + b_{hg}) \\
o_k^{(\tau)} &= \sigma(W_{io}C_k^{(\tau)} + b_{io} + W_{ho}\mathbf{h}_k^{(\tau-1)} + b_{ho}) \\
c_k^{(\tau)} &= f_k^{(\tau)} \odot c_k^{(\tau-1)} + i_k^{(\tau)} \odot g_k^{(\tau)} \\
\mathbf{h}_k^{(\tau)} &= o_k^{(\tau)} \odot \tanh(c_k^{(\tau)}) \,,
\end{aligned}
\tag{4}
$$

where $k \in \{t+h-2, t+h-1, ..., t+h+2\}, 1 \leq \tau \leq p, C_k^{(\tau)}$ denotes the τ-th row of the construal C_k, $\odot$ is the element-wise product and σ is the sigmoid function. The initial hidden state $\mathbf{h}_k^{(0)}$ and the initial cell state $c_k^{(0)}$ are set to zero by default. This shared LSTM fuses all kinds of future visions by sharing its weights and biases across all predictive tasks. Therefore, the fusion information is stored in shared parameters after the training phase and produces fusion features $\mathbf{h}_{t+h-2}^{(p)}, \mathbf{h}_{t+h-1}^{(p)}, \mathbf{h}_{t+h}^{(p)}, \mathbf{h}_{t+h+1}^{(p)}, \mathbf{h}_{t+h+2}^{(p)}$ for each forecasting task during the testing phase.

Main Recurrent Component

Similar in spirit to the Encoder-Decoder architecture (Cho et al. 2014), for the main predictive task (i.e. the forecasting of X_{t+h}), we use the shared LSTM to encode its fusion sequence, and devise another main LSTM to predict the output sequence. As shown in Figure 3 Part II, the output sequence $\mathbf{h}_{t+h}^{'(p)}$ is computed as:

$$
\mathbf{h}_{t+h}^{'(p)} = MainLSTM(C_{t+h}, \mathbf{h}_{t+h}^{(p)}, c_{t+h}^{(p)}) \,, \tag{5}
$$

where $MainLSTM$ has the same architecture as the shared LSTM but with the initial hidden state and cell state set to $\mathbf{h}_{t+h}^{(p)}$ and $c_{t+h}^{(p)}$, respectively. In our experiments, we empirically found that such a Fusion-Encoder-Main-Decoder architecture can boost the model performance in most cases.

We use a dense layer to align the outputs of the shared and main LSTMs:

$$
r_k^D = \begin{cases} W_k^D \mathbf{h}_k^{'(p)} + b_k^D & k = t+h \\ W_k^D \mathbf{h}_k^{(p)} + b_k^D & \text{otherwise} \end{cases} \,, \tag{6}
$$

where $k \in \{t+h-2, ..., t+h+2\}, r_k^D \in \mathbb{R}^n$ is the prediction result of the neural network for the predictive task on X_k and W_k^D, b_k^D are weights and biases of the dense layer.

Autoregressive Component

As is pointed out by (Lai et al. 2017), the non-linear nature of CNN and LSTM leads to their poor performance in capturing the scale changes of inputs, which significantly lowers the forecasting accuracy of the neural network model. To address this deficiency, Lai et al. decomposes their model into a linear part (i.e. the Autoregressive model) and a non-linear part (i.e. the neural network model). In this paper, we adopt the same method but change the Autoregressive (AR) model architecture to fit the neural network component of the MLCNN model. Typically, the scale of near future values is sensitive to the scale of recent past values, while the scale of distant future values is sensitive to the scale of both recent and further past values. Hence, we denote $s^{ar} \in \mathbb{N}$ as the *autoregressive stride* and define $X^{qar} = [X_t; X_{t-1}; ...; X_{t-qs^{ar}+1}] \in \mathbb{R}^{qs^{ar} \times n}$, where

$q \in \{1, 2, 3, 4, 5\}$. The forecasting result of the AR component for each predictive task is computed as follows:

$$r^L_{k,i} = \sum_{j=0}^{qs^{ar}} W^L_{k,j} X^{qar}_{j,i} + b^L_k \ , \tag{7}$$

where $k = t + h - 3 + q, 1 \leq i \leq n$. $r^L_{k,i}$ and $X^{qar}_{j,i}$ denote the i-th element of vectors r^L_k and X^{qar}_j, respectively. And W^L_k, b^L_k are weights and biases of the AR model. Note that all dimensions share the same set of linear parameters in each task.

We obtain the final prediction by combining the result of the neural network component and the AR component:

$$\hat{Y}_k = r^D_k + r^L_k \ , \tag{8}$$

where $k \in \{t+h-2, ..., t+h+2\}$ and $\hat{Y}_k \in \mathbb{R}^n$ is the final prediction of X_k. Thus $\hat{Y} \in \mathbb{R}^{5\times n}$ is the final prediction matrix of the five-task learning problem described previously.

Loss Function

L_2 error is the default loss function for most of the time series forecasting tasks:

$$L_2(Y, \hat{Y}) = \sum_{\Omega_{\text{train}}} \sum_{k=1}^{l} \sum_{j=1}^{n} (Y_{k,j} - \hat{Y}_{k,j})^2 \ , \tag{9}$$

where l is the number of tasks, n is the number of variables, Y is the ground truth, $\hat{Y}$ is the model's prediction and Ω_{train} denotes the training set. However, researchers have found that the absolute loss (L_1-loss) function:

$$L_1(Y, \hat{Y}) = \sum_{\Omega_{\text{train}}} \sum_{k=1}^{l} \sum_{j=1}^{n} |Y_{k,j} - \hat{Y}_{k,j}| \tag{10}$$

works better than L_2 loss function in some cases. In the experiment part of this paper, we use the validation set to decide which of the two loss functions is better for our model. The goal of training is to minimize the loss function given the parameter set of our model, which can be achieved by using the Stochastic Gradient Decent (SGD) method or its variants. In this paper, We utilize the Adam (Kingma and Ba 2014) algorithm to optimize the parameters of our model.

Experiments

In this section, we conduct extensive experiments on three real-world datasets for multivariate time series forecasting, and compare the result of proposed MLCNN model against 5 baselines. To demonstrate the efficiency of our model, we also perform time complexity analysis and ablation study.

Datasets

As depicted in Table 1, our experiments are based on three publicly available datasets:

- Traffic (Lai et al. 2017): This dataset consists of 48 months (2015-2016) *hourly data* from the California Department of Transportation. It describes the road occupancy rates (between 0 and 1) measured by different sensors on San Francisco Bay area freeways.

Table 1: Dataset statistics

Dataset	Traffic	Energy	NASDAQ
#Instances	17544	19735	40560
#Attributes	862	26	82
Sample rate	1 h	10 min	1 min
Train size	60%	80%	90%
Valid size	20%	10%	5%
Test size	20%	10%	5%

- Energy (Candanedo, Feldheim, and Deramaix 2017): This UCI appliances energy dataset contains measurements of 29 different quantities related to appliances energy consumption in a single house, recorded *every 10 minutes* for 4.5 months. We select 26 relevant attributes for our experiments.
- NASDAQ (Qin et al. 2017): This dataset includes the stock prices of 81 major corporations and the index value of NASDAQ 100, collected *minute-by-minute* for 105 days.

Metrics

To evaluate the performance of different methods for multivariate time series prediction, we use two conventional evaluation metrics (1) Root Mean Squared Error: $RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(Y_{t+h,j} - \hat{Y}_{t+h,j})^2}$ and (2) Mean Absolute Error: $MAE = \frac{1}{n}\sum_{j=1}^{n}|Y_{t+h,j} - \hat{Y}_{t+h,j}|$, where n is variable dimension, $Y_{t+h} \in \mathbb{R}^n$ is the ground truth of the time series value at the future moment $t + h$ and $\hat{Y}_{t+h} \in \mathbb{R}^n$ is the model's prediction. For both metrics, lower value is better. Note that for our multi-task forecasting model, we only use RMSE and MAE of the *main predictive task* for evaluation.

Baselines

We compare the MLCNN model with 5 baselines as follows:

- **VAR** (Box and Jenkins 1970; Hamilton 1994): The well-known Vector Autoregression model for multivariate time series forecasting.
- **RNN-LSTM** (Hochreiter and Schmidhuber 1997): The Recurrent Neural Network using LSTM cell. We combine an LSTM layer and a dense layer to perform multivariate time series forecasting.
- **MTCNN** (Lecun et al. 1998): The classical Convolution Neural Network exploiting the same multi-task idea as the MLCNN model. We use a simple multi-layer CNN to perform multiple predictive tasks of different future time.
- **AECRNN** (Cirstea et al. 2018): A multi-task learning model combining additional auto-encoders with a unified framework of Convolution and Recurrent Neural Networks. AECRNN is originally designed to perform univariate time series forecasting given other correlated time series. Here we extend this model to perform multivariate

Table 2: Forecasting results of all methods over the three datasets (best results displayed in **boldface**)

Dataset		Traffic			Energy			NASDAQ		
		Future			Future			Future		
Metrics	Models	t+3	t+6	t+12	t+3	t+6	t+12	t+3	t+6	t+12
RMSE	VAR	0.0370±9E-04	0.0373±4E-04	0.0364±6E-04	15.514±0.002	16.253±0.007	16.950±0.004	2.725±0.104	3.049±0.143	3.048±0.043
	RNN-LSTM	0.0298±5E-05	0.0304±2E-04	0.0299±2E-04	15.820±0.002	16.758±0.095	17.289±0.027	4.529±0.314	4.946±0.181	5.353±0.141
	MTCNN	0.0295±3E-04	0.0297±2E-04	0.0304±2E-04	15.841±0.154	16.549±0.028	17.481±0.203	4.197±0.174	3.928±0.217	4.341±0.327
	AECRNN	0.0286±2E-04	0.0291±3E-04	0.0295±3E-04	15.705±0.124	16.259±0.152	17.173±0.165	9.785±0.438	9.893±0.304	9.727±0.351
	LSTNet	0.0269±1E-04	0.0278±3E-04	0.0280±2E-04	15.506±0.049	**15.795±0.074**	16.890±0.105	0.366±0.006	0.522±0.010	0.754±0.022
	MLCNN	**0.0258±1E-04**	**0.0264±1E-04**	**0.0267±8E-05**	**15.130±0.087**	15.994±0.047	**16.782±0.125**	**0.365±0.002**	**0.516±0.003**	**0.739±0.004**
MAE	VAR	0.0255±1E-03	0.0256±4E-04	0.0246±7E-04	2.898±0.001	3.321±0.026	3.872±0.009	1.834±0.069	2.075±0.072	2.008±0.015
	RNN-LSTM	0.0134±6E-05	0.0138±1E-04	0.0136±2E-04	2.733±0.072	3.049±0.079	3.668±0.128	2.205±0.091	2.344±0.051	2.452±0.064
	MTCNN	0.0135±3E-04	0.0139±2E-04	0.0143±4E-04	3.415±0.098	3.896±0.062	4.312±0.131	2.433±0.038	2.375±0.081	2.397±0.070
	AECRNN	0.0121±1E-04	0.0124±2E-04	0.0131±2E-04	2.269±0.078	3.013±0.072	3.395±0.057	4.370±0.361	4.500±0.267	4.370±0.350
	LSTNet	0.0116±8E-05	0.0123±3E-04	0.0124±2E-04	**1.795±0.014**	2.386±0.030	3.112±0.043	0.093±0.003	0.135±0.006	0.195±0.012
	MLCNN	**0.0110±1E-04**	**0.0113±1E-04**	**0.0114±8E-05**	1.879±0.033	**2.378±0.017**	**3.036±0.044**	**0.091±0.001**	**0.130±0.001**	**0.186±0.001**

time series forecasting and compare it with the multi-task learning framework proposed in this paper.

- **LSTNet** (Lai et al. 2017): A novel multivariate time series forecasting framework which achieves great performance improvements by catching the long-term and short-term patterns of the time series data.

Training Details

We conduct a grid search over all hyperparameters for each method and dataset. Specifically, for the length of input window size T, we set $T \in \{1, 6, 12, 24, 24 \times 7\}$-hour and all methods share the same grid search range. For RNN-LSTM, we vary the number of hidden state size in $\{10, 25, 50, 100, 200\}$. For MTCNN, the filter number of CNN is chosen from $\{5, 10, 25, 50, 100\}$. For AECRNN, we use the default settings by its author. For LSTNet and MLCNN, the filter number of CNN is chosen from $\{5, 10, 25, 50, 100\}$ and the hidden state size of RNN is chosen from $\{10, 25, 50, 100, 200\}$. For simplicity, we use the same hidden state size for shared LSTM and main LSTM of our MLCNN model. The dropout rate of our model is chosen from $\{0.2, 0.3, 0.5\}$. During the training phase, the batch size is 128 and the learning rate is 0.001. We test different hyperparameters and find the best settings for each method.

Main Results

Table 2 summaries the experimental results of all the methods on the three datasets. Following the testing settings of (Lai et al. 2017) and (Qin et al. 2017), we use each model to predict the future value of the time series at the future moment $\{t + 3, t + 6, t + 12\}$, respectively, which means the future moment is set from 3 to 12 hours for the forecasting over the Traffic data, from 30 to 120 minutes over the Energy data and from 3 to 12 minutes over the NASDAQ data. To be fair, we train each model under different parameter settings for 10 times and report their best average performance and standard deviations for comparison[2].

[2]We first use the validation set to select hyper-parameters that obtain similar good predictive results. And then we simply split the dataset into training and testing sets to retrain the model and show the best testing performance of those good hyper-parameters.

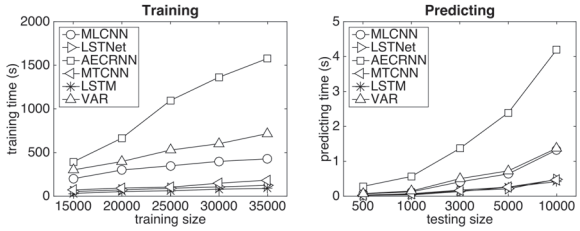

Figure 4: Result of time complexity comparison

Clearly, our method outperforms most of the baselines on both metrics, demonstrating the effectiveness of the framework design for fusing different future visions. The most significant improvements are achieved on the Traffic dataset. Specifically, MLCNN outperforms the state-of-the-art baseline LSTNet by **4.09%**, **5.04%**, **4.64%** on RMSE and **5.17%**, **8.13%**, **7.32%** on MAE on the Traffic dataset. Remarkably, our model achieves significantly improvements beyond the state-of-the-art multi-task learning framework AECRNN over all of the three datasets, showing the advantages of utilizing multi-level construals for multi-task forecasting. On the other hand, we observe that the RMSE and MAE of RNN-LSTM, MTCNN and AECRNN models are much worse than other models on the NASDAQ dataset. This is mainly because the three models are not sensitive to the scale of the input data due to their lack of the AR component. Therefore, the non-periodic scale changing of input signals will dramatically lower their predictive performance. In Figure 5, we also show the failure of MLCNN and LSTNet on NASDAQ dataset without the AR component. Furthermore, we conduct the two-sample t-test (Cressie and Whitford 1986) at 5% significance level on the forecasting results of MLCNN and LSTNet models. Overall, the small p-value of the test shows that our model achieves statistically significant improvements beyond the LSTNet model.

In summary, the evaluation results demonstrate the success of our MLCNN framework in fusing near and distant future visions to improve the forecasting accuracy.

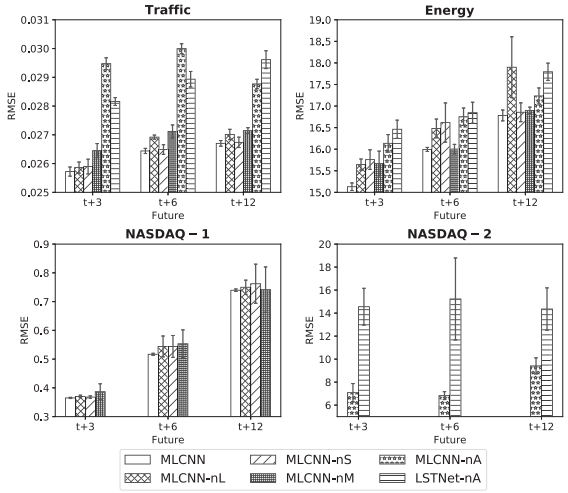

Figure 5: Results of variant comparison

Time Complexity Analysis

Although the proposed MLCNN architecture appears a little complex, we believe that the sharing mechanism of multi-task learning helps to reduce the training and predicting complexity. In the convolutional component, all preditive tasks share the low layers of a single multi-layer CNN. Also, in the recurrent component, weights and biases of the fusion LSTM are shared by all tasks. Sharing parameters among different tasks ensures that model complexity will not increase too much while performing multiple tasks. To prove this, we compare the behavior of all models as a function of the sample size and show the result over the NASDAQ dataset in Figure 4. The training and predicting time of our MLCNN model is close to that of other baselines. Significantly, MLCNN outperforms the VAR and the AECRNN models when dealing with high dimensional time series, proving the efficiency of our multi-task learning design.

Variant Comparison

To demonstrate the effectiveness of each model component, we compare MLCNN with 5 variants as follows:

- **MLCNN-nL**: We remove differences of abstraction levels between the construals for multiple tasks. Instead, we use independent CNNs with the same number of convolutional layers to construct each construal and thus fail the CLT in our model.
- **MLCNN-nS**: We remove the shared LSTM component (i.e., the fusion encoder) such that there is no fusion for the future visions of different forecasting tasks.
- **MLCNN-nM**: We remove the main LSTM component (i.e., the main decoder) and use a single LSTM for fusion and prediction.
- **MLCNN-nA**: We remove the AR component and test the predictive performance of the neural network part.
- **LSTNet-nA**: We also remove the AR component of the LSTNet model and compare it with MLCNN-nA.

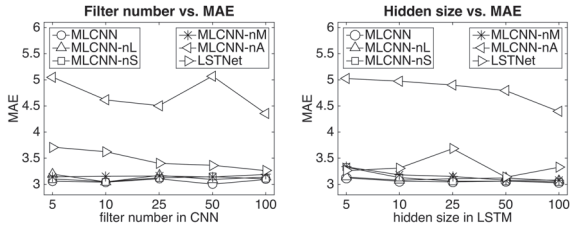

Figure 6: Results of parameter sensitivity tests

For all the variants, we tune their hidden dimension to make them have similar numbers of model parameters to the completed MLCNN model, eliminating the influences of different model complexity.

Figure 5 presents the results of comparison. Important observations from these results are listed as follows:

- MLCNN achieves the best result on all the datasets.
- Removing any component from MLCNN not only causes the performance drops but also increases the variances, showing the robustness of our MLCNN architecture design.
- Removing the AR component (in MLCNN-nA) from MLCNN causes the most significant performance drops on most of the datasets, which verifies the scale insensitive problem proposed by (Lai et al. 2017).
- MLCNN-nA achieves better performance than LSTNet-nA on most of the datasets, demonstrating the advantages of the neural network component of our MLCNN model even without the AR component.

In conclusion, the full MLCNN architecture is the most effective and robust forecasting model across all experiment settings.

Furthermore, we try different filter number of CNN and hidden state size of LSTM in both MLCNN as well as its variants and LSTNet. Figure 6 shows the comparison results of prediction on X_{t+12} on the Energy dataset. We can observe that MLCNN generally achieves best results under different parameter settings. Besides, compared to the LSTNet and the variants, our model is less sensitive to the parameter changes, showing the effectiveness of our multi-task deep learning framework.

Conclusion

In this paper, we propose a novel multi-task deep learning framework (MLCNN) for multivariate time series forecasting. In the first level, based on the Construal Level Theory of psychology, we design a multi-layer Convolution Neural Network to produce multi-level abstract construals for multiple predictive tasks. In the second level, we devise a Fusion-Encoder-Main-Decoder architecture to fuse the future visions of all tasks. Moreover, we combine the autoregressive model with the neural network to boost predictive performance. Experiments on three real-world datasets show that our model achieves the best performance against 5 baselines in terms of the two metrics (RMSE and MAE). In ad-

dition, we demonstrate the efficiency and robustness of the MLCNN architecture through in-depth analysis.

For the future research, the proposed model can be extended further by adding weighting machanism to the fusion encoder of different future visions, such as the Attention machanism (Bahdanau, Cho, and Bengio 2014). Besides, how to dynamically choose the temporal distances from the future (i.e., the fsp and fst parameters) instead of setting their values to default is another challenging problem.

Acknowledgments

This paper was supported by the National Key Research and Development Program (2016YFB1000101), the National Natural Science Foundation of China (61722214, U1811462, 61876155), the Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2016) and Key Program Special Fund in XJTLU under no. KSF-A-01, KSF-E-26 and KSF-P-02.

References

Adhikari, R., and Agrawal, R. K. 2013. An introductory study on time series modeling and forecasting. *CoRR* abs/1302.6613.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Bengio, Y.; Simard, P.; and Frasconi., P. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5:157–166.

Binkowski, M.; Marti, G.; and Donnat, P. 2018. Autoregressive convolutional neural networks for asynchronous time series. In *Proceedings of the 35th International Conference on Machine Learning*, 579–588.

Box, G., and Jenkins, G. 1970. *Time series analysis: forecasting and control*. Holden-Day series in time series analysis. Holden-Day.

Candanedo, L. M.; Feldheim, V.; and Deramaix, D. 2017. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings* 140:81 – 97.

Caruana., R. 1997. Multitask learning. *Machine learning* 28(1):41–75.

Chen, L.; Liu, Y.; Zheng, Z.; and Yu, P. 2018. Heterogeneous neural attentive factorization machine for rating prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, 833–842. New York, NY, USA: ACM.

Cho, K.; van Merriënboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Cirstea, R.; Micu, D.; Muresan, G.; Guo, C.; and Yang, B. 2018. Correlated time series forecasting using deep neural networks: A summary of results. *CoRR* abs/1808.09794.

Cressie, N. A. C., and Whitford, H. J. 1986. How to use the two sample t-test. *Biometrical Journal* 28(2):131–148.

Griffin, D. W.; Dunning, D.; and Ross, L. 1990. The role of construal processes in overconfident predictions about self and others. *Journal of Personality and Social Psychology* 59:1128–1139.

Hamilton, J. D. 1994. *Time series analysis*, volume 2. Princeton: Princeton university press.

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

Khashei, M., and Bijari, M. 2011. Which methodology is better for combining linear and nonlinear models for time series forecasting? *Journal of Industrial and Systems Engineering* 4(4):265–285.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Lai, G.; Chang, W.; Yang, Y.; and Liu, H. 2017. Modeling long- and short-term temporal patterns with deep neural networks. *CoRR* abs/1703.07015.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86:2278–2324.

Liang, Y.; Ke, S.; Zhang, J.; Yi, X.; and Zheng, Y. 2018. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI-18*, 3428–3434.

Liberman, N., and Trope, Y. 1998. The role of feasibility and desirability considerations in near and distant future decisions: A test of temporal construal theory. *Journal of Personality and Social Psychology* 75:5–18.

Nussbaum, S.; Liberman, N.; and Trope, Y. 2006. Predicting the near and distant future. *Journal of Experimental Psychology: General* 135(2):152–161.

Qin, Y.; Song, D.; Chen, H.; Cheng, W.; Jiang, G.; and Cottrell, G. W. 2017. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI-17*, 2627–2633.

Ruder, S. 2017. An overview of multi-task learning in deep neural networks. *CoRR* abs/1706.05098.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc. 3104–3112.

Trope, Y., and Liberman, N. 2000. Temporal construal and time-dependent changes in preference. *Journal of Personality and Social Psychology* 79:876–889.

Wu, H., and Gu, X. 2015. Towards dropout training for convolutional neural networks. *Neural Networks* 71:1 – 10.

Wu, Y.; Xie, F.; Chen, L.; Chen, C.; and Zheng, Z. 2017. An embedding based factorization machine approach for web service qos prediction. In *Service-Oriented Computing*, 272–286.

Yang, H.; Huang, K.; King, I.; and Lyu, M. R. 2009. Localized support vector regression for time series prediction. *Neurocomputing* 72(10):2659 – 2669.

Yang, Y.; Zheng, Z.; Niu, X.; Tang, M.; Lu, Y.; and Liao, X. 2018. A location-based factorization machine model for web service qos prediction. *IEEE Transactions on Services Computing* 1–1.

Zhang, G.; Patuwo, B. E.; and Hu, M. Y. 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* 14(1):35 – 62.