Periodic-CRN: A Convolutional Recurrent Model for Crowd Density Prediction with Recurring Periodic Patterns

Ali Zonoozi[1], **Jung-jae Kim**[2], **Xiaoli Li**[2], **Gao Cong**[1],
[1] School of Computer Science and Engineering, Nanyang Technological University
[2] Institute for Infocomm Research, A-STAR
s130052@e.ntu.edu.sg, {jjkim, xlli}@i2r.a-star.edu.sg, gaocong@ntu.edu.sg

Abstract

Time-series forecasting in geo-spatial domains has important applications, including urban planning, traffic management and behavioral analysis. We observed recurring periodic patterns in some spatio-temporal data, which were not considered explicitly by previous non-linear works. To address this lack, we propose novel 'Periodic-CRN' (PCRN) method, which adapts convolutional recurrent network (CRN) to accurately capture spatial and temporal correlations, learns and incorporates explicit periodic representations, and can be optimized with multi-step ahead prediction. We show that PCRN consistently outperforms the state-of-the-art methods for crowd density prediction across two taxi datasets from Beijing and Singapore.

1 Introduction

Spatio-temporal forecasting, which aims to predict future events and trends given the spatial and temporal dynamics in historical data, is one of the most important problems in machine learning and AI research. One typical application is future crowd density estimation. Today, with the large amount of location reporting sensors available in mobile devices, vehicles and other moving objects, it is possible to build accurate models for a wide range of applications such as estimating traffic flow and congestion, predicting taxi demand, and forecasting people crowd density in public transportation stations.

Formally, the forecasting problem can be defined as follows: Given a sequence $\{X^{(1)}, X^{(2)}, ..., X^{(t)}\}$ of observed values, the goal is to predict future values of $\{X^{(t+1)} ... X^{(t+\tau)}\}$ for an arbitrary horizon $\tau \geq 1$. In recent years, numerous spatio-temporal forecasting techniques and applications have been studied, including road traffic forecast [Lippi *et al.*, 2013; Lv *et al.*, 2015], human mobility pattern prediction for urban planning [Zheng *et al.*, 2014] as well as next frame prediction in videos [Srivastava *et al.*, 2015; Finn *et al.*, 2016].

Many spatio-temporal time-series data in geo-spatial domains show *periodic patterns* over different time scales such as days or weeks. For instance, the number of cars passing a pre-defined bounded region usually increases in rush hours and decreases over night, and this pattern repeats everyday. On a weekly scale, the traffic patterns in weekends are similar to each other, compared to the weekday patterns. Some examples of these periodic patterns can be observed in Figure 1. In this paper, we will leverage these recurring patterns for our predictions.

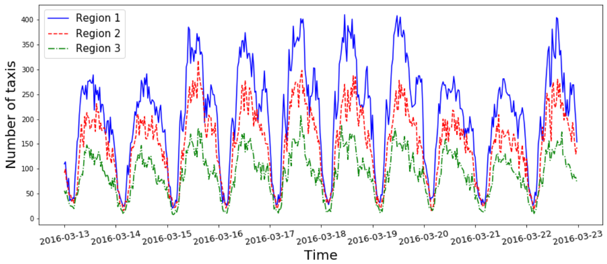

Figure 1: In-flow of taxis over 30 min intervals into 3 different regions of size $750m * 750m$ over period of 10 days in Beijing. The time-series data show clear *daily* and *weekly* periodic patterns.

Classic linear time-series forecasting models such as ARIMA and its variants [Box *et al.*, 2015] are only applicable to stationary time-series where statistical properties of data should remain approximately constant over time. Moreover, finding the right set of parameters for these models are challenging. Seasonal ARIMA (SARIMA) is able to incorporate seasonal parameters into the ARIMA model [Williams and Hoel, 2003], but it becomes ineffective and not applicable for modeling recurring patterns with long seasonal periods in practice [Hyndman, 2010].

Recently, deep learning models have become popular for spatio-temporal prediction problems, due to their inherent ability to model complex non-linearities in data. Emergence of convolutional recurrent networks (CRNs) such as ConvLSTM [Xingjian *et al.*, 2015] and ConvGRU [Ballas *et al.*, 2016] was a breakthrough in spatio-temporal forecasting, due to their ability to capture spatial and temporal correlations in end-to-end trained models. However, these models have not explicitly considered periodic patterns. Previous attempts to do so in deep learning based models for other time-series forecasting applications have been limited; for example, [Zhang *et al.*, 2017a] additionally used raw input data from previous periods in their model.

In this paper, we propose **Periodic-CRN (PCRN)**, a novel deep learning based model for crowd density prediction in spatio-temporal domain. Specifically, our model predicts the next state of a geo-spatial space, given a sequence of previous states. Our key idea is using *periodic representations* of previously observed recurring patterns with tensors, rather than using raw input data from those periods in the learning process as in previous work [Zhang *et al.*, 2017a]. In addition to relieving practitioners from providing redundant input to the model, it lets the model learn effective representations from *sequence of inputs* in previous periods, as compared to single snapshots from raw inputs. Our specific contributions can be summarized as follows:

- We introduce periodic representations, into a spatio-temporal prediction model, and show their effectiveness in producing more accurate predictions. To the best of our knowledge, this is the first work that makes use of periodic patterns in a convolutional recurrent network architecture, for a spatio-temporal forecasting problem.
- We propose three different methods for updating and reusing periodic representation and introduce a weighting based fusion mechanism to weigh the periodic tensors of different scales (e.g. daily, weekly) based on their relevance to the current state.
- We use a pyramidal architecture for multi-layer convolutional recurrent network, for efficient and enhanced capturing of spatial and temporal correlations in spatio-temporal data.
- We adapt our proposed method and show its effectiveness for multi-step ahead prediction as compared to other linear and non-linear methods. We also show that prediction accuracy for a desired time interval could be enhanced by using multi-step ahead prediction with a shorter time interval.

2 Related Work

In time-series literature, models based on Auto Regressive Moving Average (ARMA) [Box *et al.*, 2015] focus on linear temporal modeling of time-series data [Williams *et al.*, 1997; Lee and Fambro, 1999]. ARIMA and its variants such as Vector ARIMA (VARMA) and Seasonal ARIMA (SARIMA) have been extensively used in traffic flow forecasting problems [Lippi *et al.*, 2013; Min and Wynter, 2011; Kamarianakis and Prastacos, 2003; Williams and Hoel, 2003].

Recently, deep neural network models have been used for next frame prediction problem in video analysis. [Srivastava *et al.*, 2015] introduced an encoder-decoder LSTM network for video prediction, and [Donahue *et al.*, 2015] introduced a recurrent convolutional network for visual learning. The works of ConvLSTM [Xingjian *et al.*, 2015] and ConvGRU [Ballas *et al.*, 2016] replaced the full connections of state-state and input-state transitions in GRU and LSTM, respectively, with convolution operations, which had the two following implications: (1) better modeling of spatial regularities and (2) reducing the number of parameters significantly. Based on this architecture, several studies further improved next frame prediction in videos [Finn *et al.*, 2016; Mathieu *et al.*, 2016; Byeon *et al.*, 2017; Lotter *et al.*, 2016; Oh *et al.*, 2015]. In geo-spatial forecasting domain, [Ziat *et al.*, 2017] used latent representations to capture spatio-temporal dynamics and RNNs for decoding states into observations. However, these approaches did not consider periodically recurring patterns.

CW-RNN [Koutnik *et al.*, 2014] addressed the challenge of modeling long-term dependencies in sequences; however, their model is not applicable for capturing patterns spanning over days or weeks where the input is highly dimensional. Several studies focused on traffic density prediction using deep learning without considering either spatial, temporal or both correlations [Lv *et al.*, 2015; Zhang *et al.*, 2017b]. Closest to our work is the study [Zhang *et al.*, 2017a] which uses deep residual network for crowd flow prediction. However, their model only consists of convolutional layers, which is not effective for modeling temporal correlations.

3 The Proposed Method

In this section, we first define the problem and then introduce the proposed PCRN model and its components: (1) Pyramidal convolutional recurrent network, (2) Periodic representations and (3) Fusion process in detail. Finally we explain the implementation and design decisions related to our model.

3.1 Problem Statement

We define the geo-spatio-temporal prediction problem similar to next frame prediction problem in a video sequence. The observations of crowd density or any other feature of interest for each pre-defined geographic region during a fixed time interval are aggregated and mapped into a 2-D dimensional multi-channel image $X^{(t)} \in \mathbb{R}^{M*W*H}$ for discrete time point t, referred to as 'spatial image' hereafter. Here W, H, M refer to the width, height and number of channels of spatial images, respectively. Therefore, $\{X^{(1)}, X^{(2)}, ..., X^{(T)}\}$ represents the set of all spatial images used for model creation.

As the input, our proposed model receives a sequence of spatial images $\{X^{(t-\tau+1)}, ..., X^{(t)}\}$, where τ is a pre-defined sequence length. As the outcome of our model, we are interested in predicting a spatial image for the next time interval $X^{(t+1)}$. Since the temporal information of the spatial images is implicated in spatio-temporal setting, we also treat this meta-data information as an additional input to the model. Altogether, PCRN is proposed as an optimization model to learn an approximation function f, based on a deep neural network architecture and formulated as:

$$\hat{X}^{(t+1)} = f(X^{(t)}, ..., X^{(t-\tau+1)}, X^{(t+1)}_{meta}; \theta) \qquad (1)$$

In Eq.1, $X^{(t+1)}_{meta}$ denotes the temporal meta-data information for the predicted frame, and θ are the parameters of the model to be learned. $\hat{X}^{(t+1)}$ is generated by an approximation function over previously observed spatial images defined by a deep neural network architecture, which will be explained later.

In Eq.2, we define the objective function in least-square form to minimize the Euclidean distance between the ground truth and the predicted image at the next time point.

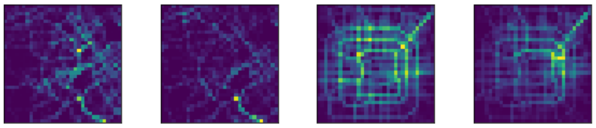

Figure 2: Examples of 32 × 32 partitioned spatial images. Lighter colors represent higher density. Left to right: volume of all taxis and taxis with passengers on board in downtown Singapore, in-flow and outflow of taxis in Beijing.

$$\mathcal{L}(\theta) = \frac{1}{T-\tau} \sum_{t=\tau}^{T-1} \parallel X^{(t+1)} - \hat{X}^{(t+1)} \parallel_2^2 \tag{2}$$

Spatial Images: We create spatial images from the records generated by independent actors, which contain both temporal (time-stamp) information and geo-spatial coordinates.

Formally each record $r^{(t',l)} \in \mathbb{R}^D$ is a D dimensional record generated at time t', and at location l where $l = \langle lat, lon \rangle$ indicates latitude and longitude coordinates of the generated record respectively. $r_d^{(t',l)}$ indicates the d-th dimensional value of the record $r^{(t',l)}$.

To create spatial images from records, we partition the area to be predicted into a $W \times H$ grid map, and define mapping function $g(l)$, to map records located inside the partitioned area into their corresponding grids in the grid map. Finally, for a discrete time point t, representing a fixed interval, we create spatial image $X^{(t)}$ as follows:

$$X_{(d,i,j)}^{(t)} = |\{r^{(t',l)} | g(l) = \langle i,j \rangle, t \leq t' < t+v, \mathcal{I}(r_d^{(t',l)})\}| \tag{3}$$

The number of records satisfying an arbitrary requirement based on feature $r_d^{(t',l)}$, generated during interval of length v from discrete time point t, and located at grid (i,j) are aggregated together, and set as the pixel value of $X_{(d,i,j)}^{(t)}$. Therefore, the multi-channel image $X^{(t)}$ will represent the crowd density in discrete time interval t. Figure 2, shows examples of spatial images used as input of the model.

3.2 Periodic Convolutional Recurrent Network

Periodic-CRN (PCRN) model proposed in this paper consists of three main components: (1) Convolutional recurrent network (CRN) to capture spatio-temporal representation of an input sequence, (2) Periodic representation dictionaries to dynamically reuse and update representations from multiple periodic patterns, and (3) Fusion process to merge periodic representations with current input sequence representation and temporal meta-data. The overall model architecture is illustrated in Figure 3.

3.2.1 Pyramidal CRN Model

Capturing spatial and temporal correlations is crucial for building accurate spatio-temporal prediction models. As in many geo-spatial problems such as crowd density or traffic congestion prediction, the state of traffic in one region inevitably influences nearby regions.

Specifically, we consider using Convolutional GRU [Ballas *et al.*, 2016] as our recurrent neural network architecture, as it has shown good performance, but with fewer parameters to be optimized. The equations for ConvGRU gates, candidate state $\tilde{h}^{(t)}$ and hidden state $h^{(t)}$ are shown in Equation 4.

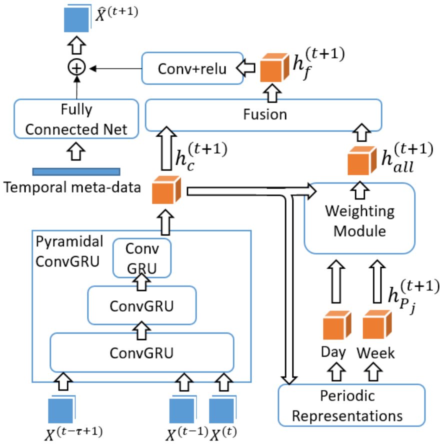

Figure 3: Weighting based fusion model architecture

$$\begin{aligned} z^{(t)} &= \sigma(W_z * [x^{(t)}, h^{(t-1)}]) \\ r^{(t)} &= \sigma(W_r * [x^{(t)}, h^{(t-1)}]) \\ \tilde{h}^{(t)} &= tanh(W_h * [x^{(t)}, r^{(t)} \circ h^{(t-1)}]) \\ h^{(t)} &= (1 - z^{(t)}) \circ h^{(t-1)} + z^{(t)} \circ \tilde{h}^{(t)} \end{aligned} \tag{4}$$

In Equation 4, $(*)$ symbol denotes convolution operation and $(\circ)$ denotes element-wise multiplication. Here $h^{(t)}$ is the hidden state tensor preserving the sequential information up to time point t.

Stacking convolutional RNN layers would result in a wider receptive field over nearby spatial regions and take their influence into account in upper layers, which leads to better representations and more accurate predictions. At the same time, stacking more layers with the same number of time steps would increase computational complexity and memory consumption significantly, resulting in very slow convergence.

Inspired by the RNN structure for speech recognition proposed in [Chan *et al.*, 2016], we adapt a pyramidal structure for our multi-layer Convolutional GRUs, where the consecutive outputs from lower layers are concatenated before being passed to upper layer. This modification reduces the sequence length of the upper layer by a factor of 2, which results in significant speedup without negatively affecting performance. Formally the input and hidden state in layer $i > 0$ are derived as follows:

$$X_i^{(t)} = [h_{i-1}^{(2t)}, h_{i-1}^{(2t+1)}] \tag{5}$$

$$h_i^{(t)} = f(X_i^{(t)}, h_i^{(t-1)}) \tag{6}$$

The inputs at layer $i = 0$ ($X_0^{(t)}$) are the raw spatial images and the function $f(.)$ refers to the set of operations in Eq. 4.

3.2.2 Periodic Representations

Recurrent neural networks generate a hidden state vector at each cell, forget unnecessary information and update them accordingly by recurrently processing their inputs and previous states (Eq. 6). Consequently, at the last cell, the hidden vector (tensor) preserves the sequential information of input in a dense representation. We aim to reuse these representation tensors for each period by introducing a loop-back mechanism in our learning procedure.

The key idea in this model is to dynamically maintain memory-based dictionary of periodic representations learned by the stacked pyramidal ConvGRUs, by implementing the two following procedures: (1) Loading periodic representations for current time point t from all available dictionaries, and (2) saving representation of the current time point in the dictionaries after the forward pass through stacked ConvGRUs. We denote the current representation tensor at the last ConvGRU layer as $h_c^{(t)}$, where t refers to the data point as in $X^{(t)}$.

Definition 1: For a spatio-temporal time-series data X, let P_j be a periodic pattern, i.e. a set of periodic representations with the same scale, e.g. P_1:daily pattern, P_2:weekly pattern, etc. We define l_{P_j} as the size of the set P_j i.e. the number of discrete time intervals within the period scale of P_j. For instance, $l_{P_{daily}} = 48$ when the time interval duration is 30 minutes.

We build a dictionary of periodic representations for each P_j. When predicting $X^{(t+1)}$, the pyramidal ConvGRU outputs a current representation tensor $h_c^{(t+1)}$, given the input sequence $(X^{(t-\tau+1)}, ..., X^{(t)})$. This representation tensor is stored in a key-value dictionary data structure $\mathcal{D}_j = (\mathcal{K}_j, \mathcal{V}_j)$, where $\mathcal{K}_j$ is the set of keys, and $\mathcal{V}_j$ is the set of periodic representation tensors h_{P_j}, for period P_j. For each periodic pattern P_j, the keys $\mathcal{K}_j$ of $\mathcal{D}_j$ are generated using a bucketing function defined as follows:

Definition 2: We define bucketing function $b_j(t)$ to map the time point t into bucket k, such that $\{k \in \mathbb{N} | k \le l_{P_j}\}$.

For instance, assuming the interval is given in minutes, and 60 is divisible by the interval duration, we define bucketing function $b_{daily}(t)$ as follows:

$$\begin{aligned} b_{daily}(t) = t.hour * (60/interval) \\ + \lfloor t.minute/interval \rfloor + 1 \end{aligned} \tag{7}$$

We propose 3 different approaches for retrieving and updating periodic representations stored in $\mathcal{V}_j$. (1) Using sequential periodic representation, (2) Using estimated average of periodic representations and (3) Using temporally ordered representations. The representations are initialized to zero, to represent lack of knowledge from periodic patterns at early stages.

Sequentially Ordered Representation

Assuming $X^{(t+1)}$ is the data point to be predicted, representation $h_{P_j}^{(t+1)}$ is loaded from $\mathcal{D}_j$ as shown in Eq. 8. Here, the bucketing function b_j, outputs the lookup key for dictionary $\mathcal{D}_j$. In the saving procedure, the current representation output $h_c^{(t+1)}$ generated from ConvGRU, replaces the previously stored representation in $\mathcal{D}_j$ as shown in Eq. 9. This procedure is performed sequentially on training and testing data, however since the data might be randomly shuffled prior to training, the periodic patterns are not temporally ordered necessarily.

$$h_{P_j}^{(t+1)} \leftarrow \mathcal{D}_j[b_j(t+1)] \tag{8}$$

$$\mathcal{D}_j[b_j(t+1)] \leftarrow h_c^{(t+1)} \tag{9}$$

Estimated Average Representations

In this approach, instead of the last periodic representation, a set of last n representations are stored in a FIFO queue. Here $\mathcal{D}_j[b_j(t+1)]$ retrieves the queue of representation tensors for period P_j and the average of all tensors is provided to the model as shown in Eq. 10. In saving phase, the current representation $h_c^{(t+1)}$ replaces the oldest representation tensor in $\mathcal{D}_j[b_j(t+1)]$, as shown in Eq. 11.

$$h_{P_j}^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{D}_j[b_j(t+1)]_i \tag{10}$$

$$\mathcal{D}_j[b_j(t+1)].enqueue(h_c^{(t+1)}) \tag{11}$$

Temporally Ordered Representations

In this approach, we keep a list of representation tensors in the exact temporal order for each periodic pattern P_j. Using the lists, we can identify the representation tensor of the previous period for the time point t; for instance, the representation tensors of $(t - l_{P_1})$ and $(t - l_{P_2})$ indicate the daily and weekly representations for t, respectively.

For this purpose, we use exact time-stamps as the keys for dictionary $\mathcal{D}_j$ instead of using the bucketing functions b_j. The retrieving and saving procedures are shown in Eq. 12 and 13 respectively.

$$h_{P_j}^{(t+1)} \leftarrow \mathcal{D}_j[(t+1) - l_{P_j}] \tag{12}$$

$$\mathcal{D}_j[(t+1) - l_{P_j}] \leftarrow h_c^{(t+1)} \tag{13}$$

3.2.3 Fusion Process

In order to effectively make use of periodic representations in the model, we should merge them with current representation tensor $h_c^{(t+1)}$ generated by ConvGRU. In order to estimate relevance of each periodic representation to the current state, we propose a weighting based fusion method inspired by attention mechanism in encoder-decoder networks [Bahdanau *et al.*, 2015].

Weighting Based Fusion: In this method, we consider current representation tensor h_c as the context, and during training, we learn scalar weights a_j for each representation tensor h_{P_j}, which indicates importance and relevance of that periodic representation with respect to current context h_c. Through softmax operation, we require that $a_j \in [0, 1]$ and $\sum_j a_j = 1$. The final context tensor is derived by the weighted sum of all representation tensors by their importance factor. This procedure is shown in Eq. 14-17.

$$e_j^{(t+1)} = W_f.flatten(W_e * [h_c^{(t+1)}, h_{P_j}^{(t+1)}]) + b_f \tag{14}$$

$$a_j^{(t+1)} = \frac{exp(e_j^{(t+1)})}{\sum_{j'} exp(e_{j'}^{(t+1)})} \tag{15}$$

$$h_{all}^{(t+1)} = \sum_j a_j^{(t+1)} h_{P_j}^{(t+1)} \tag{16}$$

$$h_f^{(t+1)} = relu(W_c \circ h_c^{(t+1)} + W_P \circ h_{all}^{(t+1)}) \tag{17}$$

In Eq.14, W_e is a $(1 * 1)$ convolutional kernel that acts as a feature pooling mechanism in filter space [Lin *et al.*, 2014]. A single tensor h_{all} represents all periodic tensors weighted by their importance.

After the fusion process, a convolution layer transforms h_f into the same shape as input spatial image i.e. $\mathbb{R}^{M*W*H}$. In Eq. 18, W_f is a $W \times H$ convolutional kernel with M number of filters.

$$h_f^{(t+1)} = relu(h_f^{(t+1)} * W_f) \qquad (18)$$

Meta-data Fusion: In spatio-temporal applications, timestamps of records are available and could be used to provide additional information to the model. For this purpose, temporal information, such as time of day and day of week are one-hot encoded, concatenated and passed to two fully connected layers. After reshaping the output, they are merged with $h_f^{(t+1)}$ to create spatial image $\hat{X}^{(t+1)}$, as shown in Equations 19, 20.

$$h_{meta}^{(t+1)} = relu(relu(X_{meta}^{(t+1)}.W_{fc1} + b_{fc1}).W_{fc2} + b_{fc2}) \qquad (19)$$

$$\hat{X}^{(t+1)} = relu(h_f^{(t+1)} + h_{meta}^{(t+1)}) \qquad (20)$$

3.3 Implementation and Optimization

To optimize the model, stochastic gradient descent (SGD) method, using back propagation through time (BPTT) was used. The least-square error term introduced in Eq.2 was used as the objective function to be minimized. To ensure proportional influence of all pixels, spatial images were down-scaled using min-max normalization, mapping all records into range $[0, 1]$.

Using grid search method, three ConvGRU layers were used in hierarchical architecture with 32, 64 and 96 number of filters respectively. The input sequence length in the lowest ConvGRU layer was set to 28 time steps. Tensorflow library was used as the optimization framework, and the model was trained on a server with 16GB GPU memory. The learning rate was set to $1e^{-4}$ and the model was trained in 500 epochs, with early stopping mechanism.

4 Experiments

4.1 Experimental Settings

In this section, we describe the datasets, various baselines and the evaluation metric used in the experiments.

4.1.1 Datasets

The following two datasets were used for evaluation.

Singaopore taxi trajectories (TaxiSG): This dataset contains trajectory log records generated from taxis in Singapore in period of January to April 2015. The CBD area in downtown Singapore were partitioned into a 32*32 grid map and taxi log records were aggregated in each cell for intervals of 5 minutes. Density of all taxis and taxis with passengers on board were used to create set of $X^{(t)} \in \mathbb{R}^{2*32*32}$ spatial images according to Eq. 3. Last 12 day's data were used for validation and testing.

Dataset	Type	Period (months)	Interval (mins)	Total Images	Test days
TaxiBJ	Taxi Flow	18	30	21360	28
TaxiSG	Taxi Density	4	5	34560	12

Table 1: The characteristic of two benchmark datasets

Beijing taxi trajectories (TaxiBJ): The dataset contains inflow and out-flow of taxis in different regions in Beijing in four separate time intervals: July-Oct. 2013, March-June 2014, March-June 2015 and Nov. 2015-April 2016. Beijing is partitioned into 32*32 grids, and for each grid cell, in-flow and out-flow of vehicles in intervals of 30 minutes are calculated using the procedure explained in [Zhang *et al.*, 2017a], resulting in set of spatial images $X^{(t)} \in \mathbb{R}^{2*32*32}$.

4.1.2 Baseline Methods

The following 6 baselines are used in our experiments:

Snaive: Seasonal naive model, which sets the predicted value equal to the last observed seasonal pattern, where the season duration is set as week.

HA: Historical average over observed weekly patterns.

ARIMA: ARIMA is a powerful linear model to fit and forecast time-series data.

VAR: Vector Autoregressive model that captures linear interdependencies between multiple univariate time-series for modeling and forecasting multivariate time-series data [Kamarianakis and Prastacos, 2003].

ConvGRU: Convolutional GRU architecture introduced in [Ballas *et al.*, 2016] modified to have pyramidal architecture but without fusion with periodic representations.

ST-ResNet: A deep learning model for crowd flow prediction by learning deep convolutional neural networks using residual learning [Zhang *et al.*, 2017a]. Learned representations are merged in a fusion process along with external information such as weather condition and holidays. Raw inputs from previous periods need to be provided during runtime.

4.1.3 Evaluation Metric

We employ the widely used root mean squared error $rmse(X, \hat{X}) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(X_t - \hat{X}_t)^2}$ as the evaluation metric for similarity between predictions and ground-truth $\hat{X}, X \in \mathbb{R}^{M*W*H}$.

4.2 Next Frame Prediction

First, we compare 6 baselines with 3 different variants of our model, namely (1) sequentially ordered periodic patterns $PCRN_{seq}$, (2) moving average patterns $PCRN_{avg}$, and (3) temporally ordered periodic representations $PCRN_{temp}$, for crowd density prediction at the next interval. The results for prediction performance from baselines and the best performing variant of our model $PCRN_{temp}$ are shown in Table 2. Each entry in the Table shows RMSE score obtained by the particular method on the test set of a particular dataset. Note that a lower value of RMSE means a better performance.

For parametric models such as ARIMA and VAR, best set of parameters were found for each time-series using grid search and Akaike Information Criterion (AIC). For ST-ResNet we used their best performing variant, with 12 residual units. Please note that, unlike our model, ST-ResNet uses

Method	Taxi BJ	Taxi SG
Snaive	44.83	6.946
HA	40.796	8.685
ARIMA	22.44	4.600
VAR	22.05	4.56
ST-ResNet	16.69	4.52
ConvGRU	17.35	4.341
$PCRN_{temp}$	**15.85**	**4.252**

Table 2: Next step prediction results

additional raw input from previous periods on the runtime. As shown in Table 2, generally non-linear models outperform linear models. We observe that our proposed $PCRN_{temp}$ (indeed all the variants of our model) outperforms 6 baselines consistently.

Results obtained from ConvGRU model, which has the same architecture as our model but without merging with periodic representations, demonstrate that convolutional RNN architecture is effective for modeling spatio-temporal time-series data as it outperforms other linear and non-linear models, though not consistently outperforming ST-ResNet. Comparing results obtained from PCRN models with ConvGRU shows the effectiveness of using periodic representations in our model.

4.3 Representation Update Method and Fusion Mechanism

To compare different periodic representation update methods, we compare three different variants of our model, namely, $PCRN_{seq}$, $PCRN_{avg}$ and $PCRN_{temp}$. The results in Table 3 show the superiority of $PCRN_{temp}$ over $PCRN_{seq}$ and $PCRN_{avg}$ which can be attributed to leveraging the exact temporal order of time point representations.

Compared to fusion method used in previous work [Zhang *et al.*, 2017a], our weighting-based fusion method is able to take relevance of each periodic pattern into account, and in fact, showed better performance in almost all cases in our experiments. It also reduced number of parameters by learning a single weight tensor for aggregated representations.

4.4 Multi-Step Ahead Prediction

PCRN can perform multi-step ahead prediction, by appending the output for a time point t to the input for $t + 1$, and running the model for an arbitrary number of steps. We compare various methods for predicting next 6 steps ahead. The results in Figure 4, show that RMSE of ARIMA and VAR linearly grows as time step increases. PCRN shows lowest rate of error increase among all, and consistently outperforms deep learning based ST-ResNet, over all steps especially further steps ahead. This can be attributed to better capturing of temporal correlations through ConvGRUs, and using periodic representations instead of raw inputs from previous periods in our model.

Method	TaxiBJ	TaxiSG
$PCRN_{seq}$	16.06	4.262
$PCRN_{avg}$	16.74	4.265
$PCRN_{temp}$	15.85	4.252

Table 3: Comparison over different PCRN variants

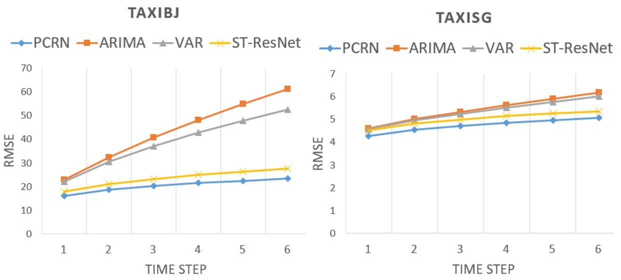

Figure 4: Comparison of $PCRN_{seq}$ with ARIMA, VAR and ST-ResNet on multi-step prediction errors up to 6 steps ahead.

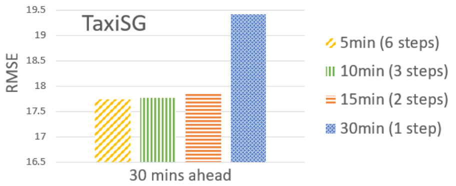

Figure 5: Multi-step and one step ahead prediction errors with variable interval lengths

Interval	5mins	10mins	15mins	30mins
Images	31104	15552	10368	5184

Table 4: Number of training images for each interval length for TaxiSG dataset

We also do an experiment on multi-step ahead prediction with different interval durations. As the prediction interval duration grows, the size of the training set reduces, making it harder for deep models with a large number of parameters to get optimized. To measure the effect of interval length in practice, we perform an experiment on multi-step ahead prediction using variable interval lengths, to predict up to fixed time point ahead on TaxiSG dataset. As illustrated in Figure 5, 6 steps ahead prediction with 5-min interval on TaxiSG dataset has lower error than one step prediction with 30-min interval. This can be attributed to lower training instances in 30-min case. This shows the improvement of results when multi-step ahead prediction with a shorter interval is used. The number of images in each training set for each interval length is shown in Table 4.

5 Conclusion

In this paper, we proposed an optimization based model for crowd density prediction, using pyramidal convolutional recurrent network architecture. By leveraging recurring periodic patterns observed in geo-spatio-temporal domains, and dynamically incorporating them in our prediction model with a loop-back mechanism, our model outperforms other linear and non-linear baselines in the next step and multi-step predictions. Moreover, by introducing a weighting based fusion mechanism, the importance and relevance of periodic representation were taken into account.

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[Ballas *et al.*, 2016] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016.

[Box *et al.*, 2015] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[Byeon *et al.*, 2017] Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. Fully context-aware video prediction. *arXiv preprint arXiv:1710.08518*, 2017.

[Chan *et al.*, 2016] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

[Donahue *et al.*, 2015] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR2015*, pages 2625–2634, 2015.

[Finn *et al.*, 2016] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, pages 64–72, 2016.

[Hyndman, 2010] Rob J Hyndman. Forecasting with long seasonal periods. `https://robjhyndman.com/hyndsight/longseasonality/`, 2010. Accessed: 2018-01-10.

[Kamarianakis and Prastacos, 2003] Yiannis Kamarianakis and Poulicos Prastacos. Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches. *Transportation Research Record: Journal of the Transportation Research Board*, (1857):74–84, 2003.

[Koutnik *et al.*, 2014] Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. 2014.

[Lee and Fambro, 1999] Sangsoo Lee and Daniel Fambro. Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, (1678):179–188, 1999.

[Lin *et al.*, 2014] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.

[Lippi *et al.*, 2013] Marco Lippi, Matteo Bertini, and Paolo Frasconi. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on ITS*, 14(2):871–882, 2013.

[Lotter *et al.*, 2016] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

[Lv *et al.*, 2015] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on ITS*, 16(2):865–873, 2015.

[Mathieu *et al.*, 2016] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.

[Min and Wynter, 2011] Wanli Min and Laura Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606–616, 2011.

[Oh *et al.*, 2015] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *NIPS*, pages 2863–2871, 2015.

[Srivastava *et al.*, 2015] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852, 2015.

[Williams and Hoel, 2003] Billy M Williams and Lester A Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672, 2003.

[Williams *et al.*, 1997] BM Williams, PK Durvasula, and DE Brown. Urban freeway travel prediction: application of seasonal arima and exponential smoothing models. In *77th Annual Transportation Research Board Meeting*, 1997.

[Xingjian *et al.*, 2015] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[Zhang *et al.*, 2017a] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, pages 1655–1661, 2017.

[Zhang *et al.*, 2017b] Shanghang Zhang, Guanhang Wu, Joao P. Costeira, and Jose M. F. Moura. Understanding traffic density from large-scale web camera data. In *CVPR*, July 2017.

[Zheng *et al.*, 2014] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM TIST*, 5(3):38, 2014.

[Ziat *et al.*, 2017] Ali Ziat, Edouard Delasalles, Ludovic Denoyer, and Patrick Gallinari. Spatio-temporal neural networks for space-time series forecasting and relations discovery. In *ICDM*, pages 705–714. IEEE, 2017.