CloudLSTM: A Recurrent Neural Model for Spatiotemporal Point-cloud Stream Forecasting

CloudLSTM: A Recurrent Neural Model for Spatiotemporal Point-cloud Stream Forecasting

Chaoyun Zhang,[1*] **Marco Fiore,**[2] **Iain Murray,**[3] **Paul Patras**[3]
[1] Tencent Lightspeed & Quantum Studios
[2] IMDEA Networks Institute
[3] The University of Edinburgh
vyokkyzhang@tencent.com, marco.fiore@imdea.org, i.murray@ed.ac.uk, paul.patras@ed.ac.uk

Abstract

This paper introduces CloudLSTM, a new branch of recurrent neural models tailored to forecasting over data streams generated by geospatial point-cloud sources. We design a Dynamic Point-cloud Convolution (DConv) operator as the core component of CloudLSTMs, which performs convolution directly over point-clouds and extracts local spatial features from sets of neighboring points that surround different elements of the input. This operator maintains the permutation invariance of sequence-to-sequence learning frameworks, while representing neighboring correlations at each time step – an important aspect in spatiotemporal predictive learning. The DConv operator resolves the grid-structural data requirements of existing spatiotemporal forecasting models and can be easily plugged into traditional LSTM architectures with sequence-to-sequence learning and attention mechanisms. We apply our proposed architecture to two representative, practical use cases that involve point-cloud streams, i.e., mobile service traffic forecasting and air quality indicator forecasting. Our results, obtained with real-world datasets collected in diverse scenarios for each use case, show that CloudLSTM delivers accurate long-term predictions, outperforming a variety of competitor neural network models.

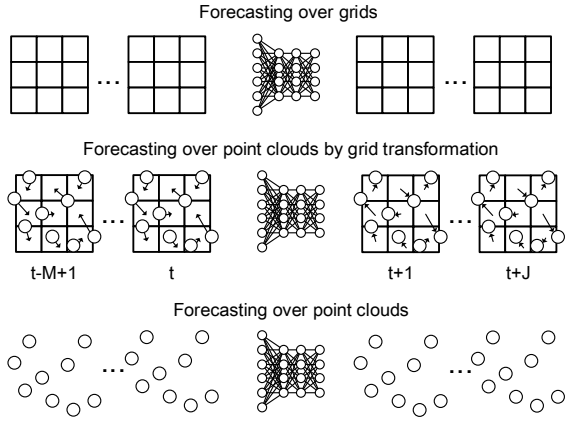

Figure 1: Different approaches to geospatial data stream forecasting: predicting over inputs that are inherently grid-structured, e.g., video frames using ConvLSTM (top); mapping of point-clouds to grids, e.g., mobile network traffic collected at different base stations, to enable forecasting using existing neural models (middle); forecasting directly over point-cloud streams using historical information (as above, but w/o pre-processing), as proposed in this paper (bottom).

Introduction

Point-cloud stream forecasting aims at predicting the future values and/or locations of data streams generated by a geospatial point-cloud $\mathcal{S}$, given sequences of historical observations (Shi and Yeung 2018). Example data sources include mobile network base stations that serve the traffic generated by ubiquitous mobile services at city scale (Zhang et al. 2019), sensors that monitor the air quality of a target region (Cheng et al. 2018), or moving crowds that produce individual trajectories. Unlike traditional spatiotemporal forecasting on grid-structural data, like precipitation nowcasting (Shi et al. 2015) or video frame prediction (Wang et al. 2018), point-cloud stream forecasting needs to operate on geometrically scattered sets of points, which are irregular and unordered, and encapsulate complex spatial correlations. While vanilla Long Short-term Memories (LSTMs) have modest abilities to exploit spatial features (Shi et al. 2015), convolution-based recurrent neural network (RNN) models, such as ConvLSTM (Shi et al. 2015) and PredRNN++ (Wang et al. 2018), are limited to grid-structural data inputs, and thus inappropriate to handle scattered point-clouds.

Learning the spatiotemporal features that are relevant to the forecasting task, from the location information embedded in such irregular data sources, is in fact challenging. Existing approaches that tackle the point-cloud stream forecasting problem can be categorized into two classes, both bearing significant shortcomings: *(i)* methods that transform point-clouds into data structures amenable to processing with mature solutions, such as the grids exemplified in Figure1 (Zhang et al. 2019); and *(ii)* models that ignore the exact locations of each data source and inherent spatial correlations (Liang et al. 2018). The transformations required by the former not only add data preprocessing overhead, but also introduce spatial displacements that distort relevant correlations among points (Zhang et al. 2019). The latter are largely location-invariant, while recent literature suggests spatial correlations should be revisited over time, to suit series prediction tasks (Shi et al.

[*]This work was conducted while the author was with The University of Edinburgh.

2017). In essence, overlooking dynamic spatial correlations risks leading to modest forecasting performance.

Contributions. In this paper, we introduce Convolutional Point-cloud LSTMs (CloudLSTMs), a new branch of recurrent neural network models tailored to geospatial point-cloud stream forecasting. The CloudLSTM builds upon a Dynamic Point-cloud Convolution (DConv) operator, which takes raw point-cloud streams (both data time series and spatial coordinates) as input, and performs dynamic convolution over these, to learn spatiotemporal features over time, irrespective of the topology and permutations of the point-cloud. This eliminates the need for input data structure preprocessing, and avoids the distortions thereby introduced. The proposed CloudLSTM takes into account the locations of each data source and performs *dynamic positioning* at each time step, to conduct a deformable convolution operation over point-clouds (Dai et al. 2017). This allows revising the spatiotemporal correlations and the configuration of the data points over time, and guarantees the location-invariant property is met at different steps. Importantly, the DConv operator is flexible, as it can be easily plugged into different existing neural network models, such as RNNs, LSTMs, Seq2seq learning (Sutskever et al. 2014), and attention mechanisms (Luong et al. 2015).

We evaluate our proposed architectures on four benchmark datasets, covering two spatiotemporal point-cloud stream forecasting applications: *(i)* base station-level forecasting of data traffic generated by mobile services (Zhang and Patras 2018; Bega et al. 2019), leveraging metropolitan-scale mobile traffic measurements collected in two European cities for 38 popular mobile apps; and *(ii)* forecasting six air quality indicators on two city clusters in China (Zheng et al. 2015). These tasks represent important use cases of geospatial point-cloud stream forecasting. We combine our CloudLSTM with Seq2seq learning and an attention mechanism, then undertake a comprehensive evaluation on all datasets. The results demonstrate that our architecture can deliver precise long-term point-cloud stream forecasting in different settings, outperforming 13 different benchmark neural models in terms of four performance metrics, without any data preprocessing requirements. To our knowledge, the proposed CloudLSTM is the *first dedicated neural architecture* for spatiotemporal forecasting that *operates directly on point-cloud streams*.

Related Work

Spatiotemporal Forecasting. Convolution-based RNN architectures have been widely employed for spatiotemporal forecasting, as they simultaneously capture spatial and temporal dynamics of the input. Shi et al., incorporate convolution into LSTMs, building a ConvLSTM for precipitation nowcasting (Shi et al. 2015). This approach exploits spatial information, which in turn leads to higher prediction accuracy. The ConvLSTM is improved by constructing a subnetwork to predict state-to-state connections, thereby guaranteeing location-variance and flexibility of the model (Shi et al. 2017). PredRNN (Wang et al. 2017) and PredRNN++ (Wang et al. 2018) evolve the ConvLSTM by constructing spatiotemporal cells and adding gradient highway units. These improve long-term forecasting performance and mitigate the gradient vanishing problem in recurrent architectures. Although these solution work well for spatiotemporal forecasting, they can not be applied directly to point-cloud streams, as they require point-cloud-to-grid data structure preprocessing (Zhang et al. 2019). STGCN takes a different approach to forecasting, by relying on graph convolutional and convolutional sequence learning layers, to capture spatial/temporal dependencies (Yu, Yin, and Zhu 2018); this approach can prove superior, yet forecasting performance can still be improved significantly, as we will demonstrate.

Feature Learning on Point-clouds. Deep neural networks for feature learning on point-cloud data are advancing rapidly. PointNet performs feature learning and maintains input permutation invariance (Qi et al. 2017a). PointNet++ upgrades this structure by hierarchically partitioning point-clouds and performing feature extraction on local regions (Qi et al. 2017b). VoxelNet employs voxel feature encoding to limit inter-point interactions within a voxel (Zhou and Tuzel 2018). This effectively projects cloud-points onto sub-grids, which enables feature learning. Li et al., generalize the convolution operation on point-clouds and employ $\mathcal{X}$-transformations to learn the weights and permutations for the features (Li et al. 2018). Through this, the proposed PointCNN leverages spatial-local correlations of point-clouds, irrespective of the order of the input. Although these architectures can learn the spatial features of point-clouds, they are designed to work with static data, and thus have limited ability to discover temporal dependencies.

Convolutional Point-cloud LSTM

Next, we formalize the problem and properties of forecasting over point-cloud-streams. We then introduce the DConv operator, which is at the core of our proposed CloudLSTM architecture. Finally, we present CloudLSTM and its variants, and explain how to combine CloudLSTM with Seq2seq learning and attention mechanisms, to achieve precise forecasting over point-cloud streams.

Forecasting over Point-cloud Streams

We formally define a point-cloud containing a set of N points, as $\mathcal{S} = \{p_1, p_2, \cdots, p_N\}$. Each point $p_n \in \mathcal{S}$ contains two sets of features, i.e., $p_n = \{\nu_n, \varsigma_n\}$, where $\nu_n = \{v_n^1, \cdots, v_n^H\}$ are value features (e.g., mobile traffic measurements, air quality indexes, etc.) of p_n, and $\varsigma_n = \{c_n^1, \cdots, c_n^L\}$ are its L-dimensional coordinates. At each time step t, we may obtain U different channels of $\mathcal{S}$ by conducting different measurements denoted by $\mathcal{S}_t^v = \{\mathcal{S}_t^1, \cdots, \mathcal{S}_t^U\}$, $\mathcal{S}_t^v \in \mathbb{R}^{U \times N \times (H+L)}$. Here, different U resemble the RGB channels in images. We can then formulate the J-step point-cloud stream forecasting problem, given M observations, as:

$$\widehat{\mathcal{S}}_{t+1}^v, \cdots, \widehat{\mathcal{S}}_{t+J}^v = \underset{\mathcal{S}_{t+1}^v, \cdots, \mathcal{S}_{t+J}^v}{\operatorname{argmax}} p(\mathcal{S}_{t+1}^v, \cdots, \mathcal{S}_{t+J}^v | \mathcal{S}_t^v, \cdots, \mathcal{S}_{t-M+1}^v). \quad (1)$$

In some cases, each point's coordinates may be unchanged, since the data sources have fixed locations. An ideal point-cloud stream forecasting model should embrace five key properties, similar to other point-cloud and spatiotemporal forecasting problems (Qi et al. 2017a; Shi et al. 2017):

(i) **Order invariance:** A point-cloud is usually arranged without a specific order. Permutations of the input points should not affect the forecasting output (Qi et al. 2017a).
(ii) **Information intactness:** The output of the model should have exactly the same number of points as the input, without losing any information, $i.e., N_{\text{out}} = N_{\text{in}}$.
(iii) **Interaction among points:** Points in $\mathcal{S}$ are not isolated, thus the model should allow interactions among neighboring points and capture local dependencies (Qi et al. 2017a).
(iv) **Robustness to transformations:** The model should be robust to correlation-preserving transformation operations on point-clouds, e.g., scaling and shifting (Qi et al. 2017a).
(v) **Location variance:** Spatial correlations among points may change over time. Such dynamic correlations should be revised and learnable during training (Shi et al. 2017).

In what follows, we introduce the Dynamic point-cloud Convolution (DConv) operator as the core module of the CloudLSTM, and explain how it satisfies these properties.

Dynamic Convolution over Point-clouds

The Dynamic point-cloud Convolution operator (DConv) generalizes the ordinary convolution on grids. Instead of computing the weighted summation over a small receptive field for each anchor point, DConv does so on point-clouds, while inheriting desirable properties of the ordinary convolution operation. The vanilla convolution takes U_{in} channels of 2D tensors as input, and outputs U_{out} channels of 2D tensors of smaller size (if without padding). Similarly, the DConv takes U_{in} channels of a point-cloud $\mathcal{S}$, and outputs U_{out} channels of a point-cloud, but with the same number of elements as the input, to fulfill the *information intactness* property *(ii)* discussed previously. For simplicity, we denote the i^{th} channel of the input set as $\mathcal{S}^i_{in}$ and the j^{th} channel of the output as $\mathcal{S}^j_{\text{out}}$. Both $\mathcal{S}^i_{in}$ and $\mathcal{S}^j_{\text{out}}$ are thus 2D tensors, of shape $(N, (H+L))$ and $(N, (H+L))$.

We also define $\mathcal{Q}^{\mathcal{K}}_n$ as a subset of points in $\mathcal{S}^i_{in}$, which includes the $\mathcal{K}$ nearest points with respect to p_n in the Euclidean space, i.e., $\mathcal{Q}^{\mathcal{K}}_n = \{p^1_n, \cdots, p^k_n, \cdots, p^{\mathcal{K}}_n\}$, where p^k_n is the k-th nearest point to p_n in the set $\mathcal{S}^i_{in}$. Note that p_n itself is included in $\mathcal{Q}^{\mathcal{K}}_n$ as an anchor point, i.e., $p_n \equiv p^1_n$. Recall that each $p_n \in \mathcal{S}$ contains H value features and L coordinate features, i.e., $p_n = \{\nu_n, \varsigma_n\}$, where $\nu_n = \{v^1_n, \cdots, v^H_n\}$ and $\varsigma_n = \{c^1_n, \cdots, c^L_n\}$. Similar to vanilla convolution, for each p_n in $\mathcal{S}^i_{in}$, DConv sums the element-wise product over all features and points in $\mathcal{Q}^{\mathcal{K}}_n$, to obtain the values and coordinates of a point p'_n in $\mathcal{S}^j_{\text{out}}$. Note that dynamic spatial correlations imply that the value features are related to their positions at the previous layer/state: hence, we aggregate the coordinate features $c(p^k_n)^l_i$ when computing the value features $v^{h'}_{n,j}$.

The resulting mathematical expression of the DConv is expounded in Eq. 2. We define learnable weights $\mathcal{W}$ as 5D tensors with shape $(U_{\text{in}}, \mathcal{K}, (H+L), (H+L), U_{\text{out}})$. The weights are shared across different anchor points in the input map. Each element $w^{m,m',k}_{i,j} \in \mathcal{W}$ is a scalar weight for the i-th input channel, j-th output channel, k-th nearest neighbor of each point corresponding to the m-th value and coordinate features for each input point, and m'-th value and coordinate features for output points. Similar to the convolution operator, we define b_j as a bias for the j-th output map. In the above, h and h' are the $h^{(\prime)}$-th value features of the input/output point set. Likewise, l and l' are the $l^{(\prime)}$-th coordinate features of the input/output. $\sigma(\cdot)$ is the sigmoid function, which limits the range of predicted coordinates to $(0, 1)$, to avoid outliers. Before feeding them to the model, the coordinates of raw point-clouds are normalized to $(0, 1)$ by $\varsigma = (\varsigma - \varsigma_{\min})/(\varsigma_{\max} - \varsigma_{\min})$, on each dimension. This improves the transformation robustness of the operator.

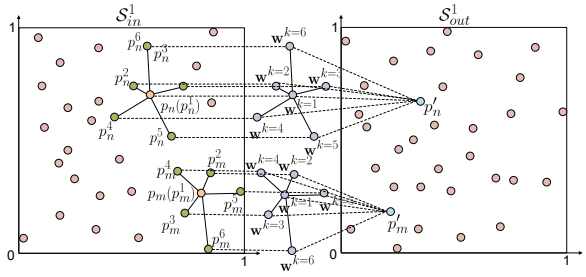

Figure 2: DConv operation with a single input channel and $\mathcal{K} = 6$ neighbors. For every $p \in \mathcal{S}^1_{in}$, DConv weights its $\mathcal{K}$ neighboring set $\mathcal{Q}^{\mathcal{K}}_n = \{p^1_n, \cdots, p^6_n\}$ to produce values and coordinate features for $p'_n \in \mathcal{S}^1_{\text{out}}$. Here, each $\mathbf{w}^k$ is a set of weights w with index k (i.e., k-th nearest neighbor) in Eq. 2, shared across different p.

The $\mathcal{K}$ nearest points can vary for each channel at each location, because the channels in the point-cloud dataset may represent different types of measurements. For example, channels in the mobile traffic dataset are related to the traffic consumption of different mobile apps. Instead, channels in the air quality dataset capture different air quality indicators (SO_2, CO, etc.). The spatial correlations will vary between different measurements (channels), due to the diverse nature of the phenomena producing them. For instance, Facebook or Instagram usage may show strong spatial correlations during a large social event spanning several neighborhoods of a city, as people communicate about the event; whereas Spotify traffic will probably be unaffected in this case. The same applies to air quality indicators, which are affected by, e.g., the unique geographical dynamics of road traffic over the street layout. As these spatial correlations must be learnable, we do not fix the locations of $\mathcal{K}$ across channels, but allow each channel to find the best neighbor set.

We provide a graphical illustration of DConv in Figure 2. For each point p_n, the DConv operator weights its $\mathcal{K}$ nearest

Algorithm 1 DConv implementation using 2D conv operator

1: **Inputs:**
   $\mathcal{S}'_{in}$, with shape $(N, \mathcal{K}, (H+L), U_{\text{in}})$.
2: **Initialise:**
   The weight tensor $\mathcal{W}$.
3: Reshape the input map $\mathcal{S}^{i'}_{in}$ from shape $(N, \mathcal{K}, (H+L), U_{\text{in}})$ to shape $(N, \mathcal{K}, (H+L) \times U_{\text{in}})$
4: Reshape the weight tensor $\mathcal{W}$ from shape $(U_{\text{in}}, \mathcal{K}, (H+L), (H+L), U_{\text{out}})$ to shape $(1, \mathcal{K}, U_{\text{in}} \times (H+L), U_{\text{out}} \times (H+L))$
5: Perform 2D convolution $\mathcal{S}_{\text{out}} = Conv(\mathcal{S}^{i'}_{in}, \mathcal{W})$ with step 1 without padding. $\mathcal{S}_{\text{out}}$ becomes a 3D tensor with shape $(N, 1, U_{\text{out}} \times (H+L))$
6: Reshape the output map $\mathcal{S}_{\text{out}}$ to $(N, (H+L), U_{\text{out}})$
7: Apply sigmoid function $\sigma(\cdot)$ to the coordinates feature in $\mathcal{S}_{\text{out}}$

$$
\begin{aligned}
v_{n,j}^{h'} &= \sum_{i\in U_{\text{in}}} \sum_{p_n^k \in \mathcal{Q}_n^{\mathcal{K}}} \Big(\sum_{h\in H} w_{i,j}^{h,h',k} v(p_n^k)_i^h + \sum_{l\in L} w_{i,j}^{(H+l),h',k} c(p_n^k)_i^l \Big) + b_j, \\
c_{n,j}^{l'} &= \sigma\Bigg(\sum_{i\in U_{\text{in}}} \sum_{p_n^k \in \mathcal{Q}_n^{\mathcal{K}}} \Big(\sum_{h\in H} w_{i,j}^{h,l',k} v(p_n^k)_i^h + \sum_{l\in L} w_{i,j}^{(H+l),l',k} c(p_n^k)_i^l \Big) + b_j \Bigg), \\
\mathcal{S}_{\text{out}}^j &= (p'_1, \cdots, p'_N) = \Big(((v_1^{1'}, \cdots, v_1^{H'}), (c_1^{1'}, \cdots, c_1^{L'})), \cdots, ((v_N^{1'}, \cdots, v_N^{H'}), (c_N^{1'}, \cdots, c_N^{L'})) \Big).
\end{aligned} \tag{2}
$$

neighbors across all features, to produce the values and coordinates in the next layer. Since the permutation of the input neither affects the neighboring information nor the ranking of their distances for any $\mathcal{Q}_n^{\mathcal{K}}$, DConv is a symmetric function whose output does not depend on the input order. This means that the property *(i)* discussed earlier is satisfied. Further, DConv is performed on every point in set $\mathcal{S}_{in}^i$ and produces exactly the same number of features and points for its output; property *(ii)* is therefore naturally fulfilled. In addition, operating over a neighboring point set, irrespective of its layout, allows capturing local dependencies. It also improves robustness to global transformations (e.g., shifting and scaling), jointly with the normalization over the coordinate features. Overall, the design meets the desired properties *(iii)* and *(iv)*. More importantly, DConv learns the layout and topology of the cloud-point for the next layer, which changes the neighboring set $\mathcal{Q}_n^{\mathcal{K}}$ for each point at output $\mathcal{S}_{\text{out}}^j$. This attains the "location-variance" property *(v)*, allowing the model to perform a dynamic positioning tailored to each channel and time step. *This is essential in spatiotemporal forecasting neural models*, which must capture spatial correlations that change over time (Shi et al. 2017).

DConv Implementation

The DConv can be efficiently implemented using a standard 2D convolution operator, by data shape transformation. We assume a batch size of 1 for simplicity. Recall that the input and output of DConv, $\mathcal{S}'_{in}$ and $\mathcal{S}_{\text{out}}$, are 3D tensors with shape $(N, (H+L), U_{\text{in}})$ and $(N, (H+L), U_{\text{out}})$, respectively. Note that for each p_n in $\mathcal{S}_{in}^i$, we find the set of top $\mathcal{K}$ nearest neighbors $\mathcal{Q}_n^{\mathcal{K}}$. Combining these, we transform the input into a 4D tensor $\mathcal{S}_{in}^{i'}$, with shape $(N, \mathcal{K}, (H+L), U_{\text{in}})$. To perform DConv over $\mathcal{S}_{in}^{i'}$, we split the operator into the steps outlined in Algorithm 1. This enables to translate the DConv into a standard convolution operation, which is highly optimized by existing deep learning frameworks.

Relations with PointCNN & Deformable Conv

The DConv operator builds upon the PointCNN (Li et al. 2018) and deformable convolution neural network (DefCNN) on grids (Dai et al. 2017), but introduces several variations tailored to point-cloud structural data. PointCNN employs the $\mathcal{X}$-transformation over point-clouds, to learn the weight and permutation on a local point set using multilayer perceptrons (MLPs), which introduces extra complexity. This operator guarantees the order invariance property, but leads to information loss, since it performs aggregation over points. In our DConv operator, the permutation is maintained by aligning the weight of the ranking of distances between point p_n and $\mathcal{Q}_n^{\mathcal{K}}$. Since the distance ranking is unrelated to the order of the inputs, the order invariance is ensured in a parameter-free manner, without extra complexity and loss of information.

Further, the DConv operator can be viewed as the DefCNN (Dai et al. 2017) over point-clouds, with the differences that *(i)* DefCNN deforms weighted filters, while DConv deforms the input maps; and *(ii)* DefCNN employs bilinear interpolation over input maps with a set of continuous offsets, while DConv instead selects $\mathcal{K}$ neighboring points for its operations. Both DefCNN and DConv have transformation flexibility, allowing adaptive receptive fields on convolution.

DConv Complexity Analysis

We study the complexity of DConv by separating the operation into two steps: *(i)* finding the neighboring set $\mathcal{Q}_n^{\mathcal{K}}$ for each point $p_n \in \mathcal{S}$, and *(ii)* performing the weighting computation in Eq. 2. We discuss the complexity of each step separately. For simplicity and without loss of generality, we assume the number of input and output channels are both 1. For step *(i)*, the complexity of finding $\mathcal{K}$ nearest neighbors for one point is close to $O(\mathcal{K} \cdot L \log N)$,[1] if using KD trees (Bentley 1975). For step *(ii)*, it is easy to see from Eq. 2 that the complexity of computing one feature of the output p'_n is $O((H+L)\cdot\mathcal{K})$. Since each point has $(H+L)$ features and the output point set $\mathcal{S}_{\text{out}}^j$ has N points, the overall complexity of step *(ii)* becomes $O(N\cdot\mathcal{K}\cdot(H+L)^2)$. This is equivalent to the complexity of a vanilla convolution operator, where both the input and output have $(H+L)$ channels, and the input map and kernel have N and $\mathcal{K}$ elements, respectively. This implies that, compared to the convolution operator whose inputs, outputs, and filters have the same size, DConv introduces extra complexity by searching the $\mathcal{K}$ nearest neighbors for each point $O(\mathcal{K} \cdot L \log N)$. Such complexity does not increase much even with higher dimensional point-clouds.

The CloudLSTM Architecture

The DConv operator can be plugged straightforwardly into LSTMs, to learn both spatial and temporal correlations over point-clouds. We formulate the Convolutional Point-cloud LSTM (CloudLSTM) as:

$$
\begin{aligned}
&i_t = \sigma(\mathcal{W}_{si} \circledast \mathcal{S}_t^v + \mathcal{W}_{hi} \circledast H_{t-1} + b_i), \\
&f_t = \sigma(\mathcal{W}_{sf} \circledast \mathcal{S}_t^v + \mathcal{W}_{hf} \circledast H_{t-1} + b_f), \\
&C_t = f_t \odot C_{t-1} + i_t \odot \tanh(\mathcal{W}_{sc} \circledast \mathcal{S}_t^v + \mathcal{W}_{hc} \circledast H_{t-1} + b_c), \\
&o_t = \sigma(\mathcal{W}_{so} \circledast \mathcal{S}_t^v + \mathcal{W}_{ho} \circledast H_{t-1} + b_o), \\
&H_t = o_t \odot \tanh(C_t).
\end{aligned} \tag{3}
$$

[1] $L \ll log(n)$ is required to guarantee efficiency. Practical point-clouds are of dimension 2 or 3 and we have significantly more than 3 points in the dataset. Hence this condition holds.

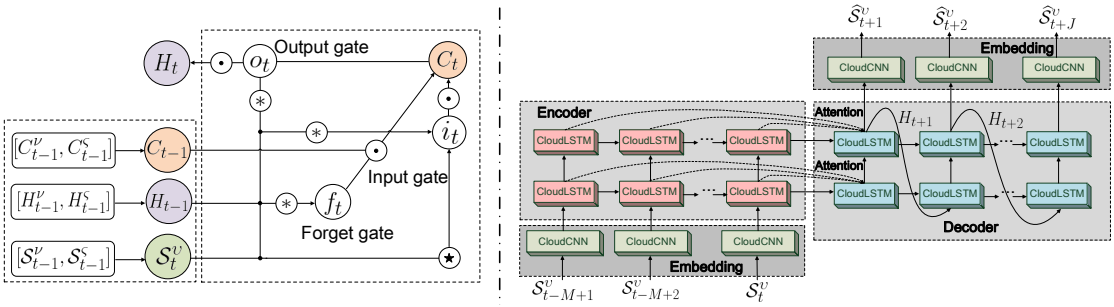

Figure 3: The inner structure of the CloudLSTM cell (left) and the overall Seq2seq CloudLSTM architecture (right). We denote by $(\cdot)^{\nu}$ and $(\cdot)^{\varsigma}$ the value and coordinate features of each input, while these features are unified for gates.

Similar to ConvLSTM (Shi et al. 2015), i_t, f_t, and o_t, are input, forget, and output gates respectively. C_t denotes the memory cell and H_t is the hidden states. Note that i_t, f_t, o_t, C_t, and H_t are all point-cloud representations. $\mathcal{W}$ and b represent learnable weight and bias tensors. In Eq. 3, '$\odot$' denotes the element-wise product, '$\circledast$' is the DConv operator formalized in Eq. 2, and '$\circledast$' a simplified DConv that removes the sigmoid function in Eq. 2. The latter only operates over the gates computation, as the sigmoid functions are already involved in outer calculations (first, second, and fourth expressions in Eq. 3). We show the structure of a basic CloudLSTM cell Figure 3 (left).

We combine our CloudLSTM with Seq2seq learning (Sutskever et al. 2014) and the soft attention mechanism (Luong et al. 2015), to perform forecasting, given that these neural models have been proven to be effective in spatiotemporal modelling on grid-structural data (e.g., (Shi et al. 2015; Zhang et al. 2018)). We show the overall Seq2seq CloudLSTM in the right part of Figure 3. The architecture incorporates an encoder and a decoder, which are different stacks of CloudLSTMs. The encoder encodes the historical information into a tensor, while the decoder decodes the tensor into predictions. The states of the encoder and decoder are connected using the soft attention mechanism via a context vector (Luong et al. 2015). We denote the j-th and i-th states of the encoder and decoder as H^j_{en} and H^i_{de}. The context tensor for state i at the encoder is represented as $c_i = \sum_{j \in M} a_{i,j} H^j_{en} = e^{i,j} / \sum_{j \in M} e_{i,j}$, where $e_{i,j}$ is a score function, which can be selected among many alternatives. In this paper, we choose $e_{i,j} = \mathbf{v}_a^T \tanh(\mathbf{W}_a * [H^j_{en}; H^i_{de}])$. Here $[\cdot;\cdot]$ is the concatenation operator and '$*$' is the convolution function. Both $\mathbf{W}_a$ and $\mathbf{v}_a$ are learnable weights. The H^i_{de} and c_i are concatenated into a new tensor for the following operations.

Before feeding the point-cloud to the model and generating the final forecasting, the data is processed by point-cloud Convolutional (CloudCNN) layers, which perform the DConv operations. Their function is similar to the word embedding layer in natural language processing tasks (Mikolov et al. 2013), which helps translate the raw point-cloud into tensors and *vice versa*. In this study, we employ a two-stack encoder-decoder architecture, and configure 36 channels for each CloudLSTM cell, as further increasing the number of stacks and channels did not entail significant gain.

We also combine DConv with RNN and Conv. GRU, introducing novel Conv. Point-cloud RNN (CloudRNN) and Conv. Point-cloud GRU (CloudGRU). Like CloudLSTM, the CloudRNN and CloudGRU employ a Seq2seq architecture, but without attention mechanism.

Experiments

To evaluate the performance of our architectures, we employ measurement datasets of traffic generated by 38 mobile services and recorded at individual network antennas, and of 6 air quality indicators collected at monitoring stations. We use the proposed CloudLSTM to forecast future mobile service demands and air quality indicators in the target regions. We provide a comprehensive comparison with 12 baseline deep learning models, over four performance metrics. All models are implemented using TensorFlow (Abadi et al. 2016) and TensorLayer (Dong et al. 2017). We train all architectures with a computing cluster with two NVIDIA Tesla K40M GPUs. We optimize all models by minimizing the mean square error (MSE) between predictions and ground truth using the Adam optimizer (Kingma and Ba 2015).

Datasets and Preprocessing

We conduct experiments on two spatiotemporal point-cloud stream forecasting tasks over 2D geospatial environments. As the data sources have fixed locations in these applications, the coordinate features are omitted in the final output. However, such features would be necessarily included in different use cases, such as crowd mobility forecasting.

Mobile Traffic Forecasting. We experiment with large-scale multi-service datasets collected by a major operator in two large European metropolitan areas with diverse topology and size during 85 consecutive days. The data describes the volume of traffic generated by devices associated to each of the 792 and 260 antennas in the two target cities, respectively. The antennas are non-uniformly distributed over the urban regions, thus they form 2D point-clouds over space. The traffic volume at each antenna is expressed in Megabytes and aggregated over 5-minute intervals, which leads to 24,482 traffic snapshots. These snapshots are gathered independently for each of 38 popular mobile services.

Air Quality Forecasting. We investigate air quality forecasting performance using a public dataset (Zheng et al. 2015), which comprises six air quality indicators (i.e., PM2.5, PM10,

Table 1: The mean±std of MAE, RMSE, PSNR, and SSIM across all models considered, evaluated on two datasets collected in different cities for mobile traffic forecasting.

Model	City 1				City 2			
	MAE	**RMSE**	**PSNR**	**SSIM**	**MAE**	**RMSE**	**PSNR**	**SSIM**
MLP	4.79±0.54	9.94±2.56	49.56±2.13	0.27±0.12	4.59±0.59	9.44±2.45	50.30±2.28	0.33±0.14
CNN	6.00±0.62	11.02±2.09	48.93±1.60	0.25±0.12	5.30±0.51	10.05±2.06	49.97±1.87	0.32±0.14
3D-CNN	4.99±0.57	9.94±2.44	49.74±2.13	0.33±0.14	5.21±0.48	9.97±2.03	50.13±1.85	0.37±0.16
DefCNN	6.76±0.81	11.72±2.57	48.43±1.82	0.16±0.08	5.31±0.51	9.99±2.13	49.84±1.87	0.32±0.14
PointCNN	4.95±0.53	10.10±2.46	49.43±2.06	0.27±0.12	4.75±0.56	9.55±2.32	50.17±2.16	0.35±0.15
CloudCNN	4.81±0.58	9.91±2.81	49.93±2.21	0.29±0.11	4.68±0.52	9.39±2.22	50.31±2.03	0.36±0.14
LSTM	4.20±0.66	9.58±3.17	50.47±3.29	0.36±0.10	4.32±1.64	9.17±3.03	50.79±3.26	0.42±0.12
ConvLSTM	3.98±1.60	9.25±3.10	50.47±3.29	0.36±0.10	4.09±1.59	8.87±2.97	51.10±3.33	0.42±0.12
PredRNN++	3.97±1.60	9.29±3.12	50.43±3.30	0.36±0.10	4.07±1.56	8.87±2.97	51.09±3.34	0.42±0.12
STGCN	3.88±1.30	9.11±2.99	50.53±3.10	0.37±0.10	3.99±1.36	8.68±2.27	51.19±3.00	0.43±0.11
PointLSTM	4.63±0.45	9.47±2.55	50.02±2.26	0.34±0.14	4.56±0.54	9.26±2.43	50.52±2.35	0.37±0.15
CloudRNN ($\mathcal{K}=9$)	4.08±1.66	9.19±3.17	50.45±3.23	0.32±0.12	4.08±1.65	8.74±3.03	51.10±3.26	0.39±0.14
CloudGRU ($\mathcal{K}=9$)	3.79±1.59	8.90±3.11	50.73±3.29	0.39±0.10	3.90±1.57	8.47±2.96	51.40±3.33	0.45±0.12
CloudLSTM ($\mathcal{K}=3$)	3.71±1.63	8.87±3.11	50.76±3.30	0.39±0.10	3.86±1.51	8.42±2.94	51.45±3.32	0.46±0.11
CloudLSTM ($\mathcal{K}=6$)	3.72±1.63	8.91±3.13	50.72±3.29	0.38±0.10	3.84±1.59	8.46±2.96	51.43±3.33	0.45±0.12
CloudLSTM ($\mathcal{K}=9$)	3.72±1.62	8.88±3.11	50.75±3.29	0.39±0.10	3.89±1.55	8.46±2.96	51.41±3.32	0.46±0.11
Attention CloudLSTM ($\mathcal{K}=9$)	**3.66±1.64**	**8.82±3.10**	**50.78±3.21**	**0.40±0.11**	**3.79±1.57**	**8.43±2.96**	**51.46±3.33**	**0.47±0.11**

NO_2, CO, O_3 and SO_2) collected by 437 air quality monitoring stations in China, over a span of one year. The monitoring stations are partitioned into two city clusters, based on their geographic locations, and measure data on an hourly basis. Clusters A and B have 274 and 163 stations, respectively.

Further details about all datasets can be found in (Zhang et al. 2020). Before feeding to the models, the measurements associated to each mobile service and air quality indicator are transformed into different input channels of the point-cloud $\mathcal{S}$. All coordinate features ς are normalized to the $(0, 1)$ range. In addition, for the baseline models that require grid-structural input (i.e., CNN, 3D-CNN, ConvLSTM and PredRNN++), the data are transformed into grids (Zhang et al. 2019) using the Hungarian algorithm (Kuhn 1955). The ratio of training plus validation, and test sets is 8:2.

Benchmarks and Performance Metrics

We compare the performance of our proposed CloudLSTM with a set of baseline models, as follows. PointCNN (Li et al. 2018) performs convolution over point-clouds and has been employed for point-cloud classification and segmentation. CloudCNN is an original benchmark we introduce, which stacks the proposed DConv operator over multiple layers for feature extraction from point-clouds. PointLSTM is another original benchmark, obtained by replacing the cells in ConvLSTM with the $\mathcal{X}$-Conv operator employed by PointCNN, which provides a fair term of comparison for other Seq2seq architectures. Beyond these models, we also compare the CloudLSTM with two of its variations, i.e., CloudRNN and CloudGRU, which were introduced earlier. Other baseline models we consider, include MLP (Goodfellow et al. 2016), CNN (Krizhevsky et al. 2012), 3D-CNN (Ji et al. 2013), LSTM (Hochreiter and Schmidhuber 1997), ConvLSTM (Shi et al. 2015) PredRNN++ (Wang et al. 2018). Among these, the first three are frequently used as benchmarks in mobile traffic forecasting (Zhang and Patras 2018; Bega et al. 2019). DefCNN learns the shape of the convolutional filters and has similarities with the DConv operator proposed in this study (Dai et al. 2017). LSTM is an advanced RNN frequently employed for time series forecasting (Hochreiter and Schmidhuber 1997). While ConvLSTM (Shi et al. 2015) can be viewed as a baseline model for spatiotemporal predictive learning, the PredRNN++ is the state-of-the-art architecture for spatiotemporal forecasting on grid-structural data and achieves the best performance in many applications (Wang et al. 2018). We also employ a spatio-temporal Graph CNN (STGCN) architecture to map point-clouds onto graphs, based on the distance between them, thereby addressing forecasting from a different perspective (Yu, Yin, and Zhu 2018). Detailed configuration of all models is discussed in (Zhang et al. 2020).

We quantify the accuracy of the proposed CloudLSTM in terms of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Since the mobile traffic snapshots can be viewed as "urban images" (Liu et al. 2015), we also select Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) (Hore and Ziou 2010) to quantify the fidelity of the forecasts and their similarity with the ground truth, as suggested by relevant recent work (Zhang et al. 2017). More details are discussed in (Zhang et al. 2020).

For the mobile traffic prediction task, we employ all neural networks to forecast city-scale mobile traffic consumption over a time horizon of $J = 6$ sampling steps, i.e., 30 minutes, given $M = 6$ consecutive past measurements. For RNN-based models, i.e., LSTM, ConvLSTM, PredRNN++, CloudLSTM, CloudRNN, and CloudGRU, we then extend the number of prediction steps to $J = 36$, i.e., 3 hours, to evaluate their long-term performance. In the air quality forecasting use case, all models receive a half day of measurements, i.e., $M = 12$, as input, and forecast indicators in the following 12 h, i.e., $J = 12$. As for the previous use case, the number of prediction steps is then extended to $J = 72$ (3 days), for all RNN-based models.

Result on Mobile Traffic Forecasting

We perform 6-step forecasting for 4,888 instances across the test set, and report in Table 1 the mean and standard

Table 2: The mean±std of MAE, RMSE, PSNR, and SSIM across all models considered, evaluated on two datasets collected in different city clusters for air quality forecasting.

Model	Cluster A				Cluster B			
	MAE	**RMSE**	**PSNR**	**SSIM**	**MAE**	**RMSE**	**PSNR**	**SSIM**
MLP	113.13±191.89	142.03±240.24	23.54±7.38	0.13±0.10	40.34±22.16	50.81±27.27	24.75±4.02	0.10±0.10
CNN	37.62±8.18	47.67±11.21	28.35±2.38	0.13±0.05	18.59±2.24	23.66±2.76	30.17±1.14	0.34±0.04
3D-CNN	37.09±7.63	48.01±10.36	28.36±2.17	0.32±0.08	19.84±2.20	25.30±2.55	29.58±0.99	0.35±0.05
DefCNN	37.51±8.34	47.69±11.44	28.40±2.41	0.13±0.05	19.46±2.45	25.58±2.80	29.55±1.07	0.26±0.05
PointCNN	39.60±7.63	51.61±10.35	27.63±2.02	0.19±0.04	19.25±2.38	24.60±2.99	29.89±1.20	0.17±0.03
CloudCNN	31.62±8.73	40.68±11.89	29.91±2.89	0.23±0.04	15.11±3.45	19.97±4.33	31.91±2.01	0.38±0.04
LSTM	30.62±8.97	40.83±11.88	29.87±2.79	0.31±0.10	14.38±3.37	19.10±4.29	32.16±2.03	0.41±0.07
ConvLSTM	22.91±8.09	31.62±11.40	31.98±3.24	0.50±0.10	10.39±2.82	14.20±3.87	34.79±2.45	0.60±0.06
PredRNN++	25.14±8.48	34.38±11.77	31.34±3.13	0.37±0.08	11.43±2.81	15.68±3.85	33.94±2.21	0.50±0.05
STGCN	25.01±8.40	33.98±11.54	31.41±3.08	0.39±0.08	11.22±2.49	15.36±3.59	34.00±2.20	0.52±0.05
PointLSTM	36.64±7.99	47.42±10.64	28.56±2.25	0.31±0.06	18.77±2.18	24.66±2.58	29.79±1.07	0.35±0.07
CloudRNN ($\mathcal{K}=9$)	33.09±8.23	42.16±11.38	29.53±2.66	0.13±0.07	14.82±3.75	19.70±4.54	31.93±2.18	0.14±0.06
CloudGRU ($\mathcal{K}=9$)	22.12±8.02	30.65±11.25	32.22±3.38	0.53±0.09	9.58±2.82	13.41±3.78	35.26±2.57	0.68±0.07
CloudLSTM ($\mathcal{K}=3$)	**20.84±7.88**	**29.16±11.06**	**32.64±3.40**	**0.57±0.10**	**9.12±2.75**	**12.95±3.72**	**35.59±2.62**	**0.69±0.07**
CloudLSTM ($\mathcal{K}=6$)	21.31±7.52	29.71±10.61	32.48±3.29	0.55±0.10	9.38±2.85	13.20±2.79	35.42±2.60	0.68±0.07
CloudLSTM ($\mathcal{K}=9$)	21.72±7.83	30.14±11.05	32.36±3.34	0.54±0.10	9.73±2.84	13.58±3.77	35.20±2.56	0.66±0.07
Attention CloudLSTM ($\mathcal{K}=9$)	21.72±7.78	30.04±10.95	32.38±3.29	0.56±0.10	9.38±2.69	13.41±3.78	35.26±2.57	**0.69±0.07**

deviation (std) of each metric. We also investigate the effect of a different number of neighboring points (i.e., $\mathcal{K}=3,6,9$), as well as the influence of the attention mechanism. The metrics are computed over 5,000 sample points and the Z-scores are always well above the significance threshold.

Observe that RNN-based architectures in general obtain superior performance, compared to CNN-based models and the MLP. In particular, our proposed CloudLSTM, and its CloudRNN, and CloudGRU variants outperform all other architectures, achieving lower MAE/RMSE and higher PSNR/SSIM on both urban scenarios. This suggests that the DConv operator learns features over geospatial point-clouds more effectively than vanilla convolution and PointCNN, as well as than the graph-based STGCN structure. In addition, CloudLSTM performs better than CloudGRU, which in turn outperforms CloudRNN.

Interestingly, the forecasting performance of the CloudLSTM seems fairly insensitive to the number of neighbors ($\mathcal{K}$); it is therefore worth using a small $\mathcal{K}$ in practice, to reduce model complexity. Further, we observe that the attention mechanism improves the forecasting performance, as it helps capturing better dependencies between input sequences and vectors in decoders, which is an effect also confirmed by other NLP tasks.

Results where the prediction horizon is extended to up to $J=36$ steps, i.e., 3 hours (long-term forecasting), for all RNN-based architectures are available in (Zhang et al. 2020).

Results on Air Quality Forecasting

We employ all models to deliver 12-step air quality forecasting on six indicators, given 12 snapshots as input. Results over 1,350 samples are in Table 2. Also in this use case, the proposed CloudLSTMs attain the best performance across all 4 metrics, outperforming state-of-the-art methods (ConvLSTM) by up to 12.2% and 8.8% in terms of MAE and RMSE, respectively. Unlike in the mobile traffic forecasting results, a lower $\mathcal{K}$ yields better prediction performance, though the difference appears subtle. Again, the CloudCNN always proves superior to the PointCNN, indicating that CloudCNNs are better feature extractors over point-clouds. Overall, the results demonstrate the effectiveness of the CloudLSTM models for modeling spatiotemporal point-cloud stream data, regardless of the tasks to which they are applied.

Note that we conduct our experiments using strict variable-controlling methodology, i.e., only changing one factor while keep the remaining the same. Therefore, it is easy to study the effect of each factor. For example, taking a look at the performance of LSTM, ConvLSTM, PredRNN++, PointLSTM and CloudLSTM, which employ dense layers, and CNN, PointCNN and D-Conv as core operators but using LSTM as the RNN structure, it is clear that the D-Conv contributes significantly to the performance improvements. Further, by comparing CloudRNN, CloudGRU and CloudLSTM, it appears that CloudRNN $\ll$ CloudGRU $<$ CloudLSTM. Similarly, by comparing the CloudLSTM and Attention CloudLSTM, we see that the effects of the attention mechanism are not very significant. Therefore, we believe the core operator $>$ RNN structure $>$ attention, ranked by their contribution.

Conclusions

We introduce CloudLSTM, a dedicated neural model for spatiotemporal forecasting tailored to point-cloud data streams. The CloudLSTM builds upon the DConv operator, which performs convolution over point-clouds to learn spatial features while maintaining permutation invariance. The DConv simultaneously predicts the values and coordinates of each point, thereby adapting to changing spatial correlations of the data at each time step. DConv is flexible, as it can be easily combined with various RNN models (i.e., RNN, GRU, and LSTM), Seq2seq learning, and attention mechanisms. We consider two application case studies, where we show that our proposed CloudLSTM achieves state-of-the-art performance on large-scale datasets collected in urban regions. CloudLSTM gives a new perspective on point-cloud stream modelling, and it can be easily extended to higher dimension point-clouds, without requiring changes to the model.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no.101017109 "DAEMON", and from the Cisco University Research Program Fund (grant no. 2019-197006).

References

Abadi, M.; et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *OSDI*, volume 16.

Bega, D.; et al. 2019. DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning. In *Proc. IEEE INFOCOM*.

Bentley, J. L. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM* 18(9): 509–517.

Cheng, W.; et al. 2018. A neural attention model for urban air quality inference: Learning the weights of monitoring stations. In *AAAI Conference on Artificial Intelligence*.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *IEEE ICCV*.

Dong, H.; et al. 2017. TensorLayer: A Versatile Library for Efficient Deep Learning Development. In *Proc. ACM Multimedia*, 1201–1204.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8): 1735–1780.

Hore, A.; and Ziou, D. 2010. Image quality metrics: PSNR vs. SSIM. In *Proc. IEEE International Conference on Pattern Recognition (ICPR)*, 2366–2369.

Ji, S.; et al. 2013. 3D convolutional neural networks for human action recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence* 35(1): 221–231.

Kingma, D.; and Ba, J. 2015. Adam: A method for stochastic optimization. *ICLR* .

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)* 2(1-2): 83–97.

Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; and Chen, B. 2018. PointCNN: Convolution on $\mathcal{X}$-transformed points. In *NeurIPS*.

Liang, Y.; et al. 2018. GeoMAN: Multi-level Attention Networks for Geo-sensory Time Series Prediction. In *IJCAI*, 3428–3434.

Liu, L.; Wei, W.; Zhao, D.; and Ma, H. 2015. Urban resolution: New metric for measuring the quality of urban sensing. *IEEE Transactions on Mobile Computing* 14(12): 2560–2575.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing*.

Mikolov, T.; et al. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. PointNet: Deep learning on point sets for 3d classification and segmentation. In *IEEE CVPR*.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*.

Shi, X.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.-K.; and Woo, W.-c. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NeurIPS*.

Shi, X.; Gao, Z.; Lausen, L.; Wang, H.; Yeung, D.-Y.; Wong, W.-k.; and Woo, W.-c. 2017. Deep learning for precipitation nowcasting: A benchmark and a new model. In *NeurIPS*.

Shi, X.; and Yeung, D.-Y. 2018. Machine Learning for Spatiotemporal Sequence Forecasting: A Survey. *arXiv preprint arXiv:1808.06865* .

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*.

Wang, Y.; Long, M.; Wang, J.; Gao, Z.; and Philip, S. Y. 2017. PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs. In *NeurIPS*.

Wang, Y.; et al. 2018. PredRNN++: Towards A Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning. In *ICML*.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 3634–3640.

Zhang, C.; Fiore, M.; and Patras, P. 2019. Multi-Service Mobile Traffic Forecasting via Convolutional Long Short-Term Memories. In *IEEE International Symposium on Measurements & Networking (M&N)*.

Zhang, C.; Ouyang, X.; and Patras, P. 2017. ZipNet-GAN: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network. In *ACM CoNEXT*, 363–375.

Zhang, C.; and Patras, P. 2018. Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks. In *Proc. ACM MobiHoc*.

Zhang, C.; Patras, P.; and Haddadi, H. 2019. Deep learning in mobile and wireless networking: A survey. *IEEE Comms. Surveys & Tutorials* .

Zhang, C.; Fiore, M.; Murray, I.; and Patras, P. 2020. CloudLSTM: A Recurrent Neural Model for Spatiotemporal Point-cloud Stream Forecasting. [Online] http://homepages.inf.ed.ac.uk/ppatras/pub/cloudlstm-extended.pdf

Zhang, L.; et al. 2018. Attention in Convolutional LSTM for Gesture Recognition. In *NeurIPS*.

Zheng, Y.; et al. 2015. Forecasting fine-grained air quality based on big data. In *ACM KDD*.

Zhou, Y.; and Tuzel, O. 2018. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *IEEE CVPR*.