A Short-Term Rainfall Prediction Model using Multi-Task Convolutional Neural Networks

Minghui Qiu
Alibaba Group
minghuiqiu@gmail.com

Peilin Zhao
Ant Financial Services Group
peilin.zpl@antfin.com

Ke Zhang
Alibaba Group
xiping.zk@alibaba-inc.com

Jun Huang
Alibaba Group
huangjun.hj@alibaba-inc.com

Xing Shi
Alibaba Group
xing.shix@alibaba-inc.com

Xiaoguang Wang
Alibaba Cloud
xiaoguang.wx@alibaba-inc.com

Wei Chu
Alibaba Group
weichu.cw@alibaba-inc.com

***Abstract*—Precipitation prediction, such as short-term rainfall prediction, is a very important problem in the field of meteorological service. In practice, most of recent studies focus on leveraging radar data or satellite images to make predictions. However, there is another scenario where a set of weather features are collected by various sensors at multiple observation sites. The observations of a site are sometimes incomplete but provide important clues for weather prediction at nearby sites, which are not fully exploited in existing work yet. To solve this problem, we propose a multi-task convolutional neural network model to automatically extract features from the time series measured at observation sites and leverage the correlation between the multiple sites for weather prediction via multi-tasking. To the best of our knowledge, this is the first attempt to use multi-task learning and deep learning techniques to predict short-term rainfall amount based on multi-site features. Specifically, we formulate the learning task as an end-to-end multi-site neural network model which allows to leverage the learned knowledge from one site to other correlated sites, and model the correlations between different sites. Extensive experiments show that the learned site correlations are insightful and the proposed model significantly outperforms a broad set of baseline models including the European Centre for Medium-range Weather Forecasts system (ECMWF).**

I. Introduction

Precipitation prediction, such as short-term rainfall prediction, is the task to predict rainfall amount in the near future (e.g., 0-6 hours) based on current observations. It has been a very important problem in the field of meteorological services. An accurate weather prediction can not only support causal usages such as outdoor activities but also provide early warnings of floods or traffic accidents. However, the design of an effective prediction system for accurate short-term rainfall prediction is still a challenging task for researchers and meteorological centers, due to the difficulty of collecting spatial-temporal weather features and the complex nature of weather nowcasting modeling.

In the literature, there are mainly two lines of existing methods for short-term precipitation forecasting: the numerical weather prediction (NWP) models and the radar echo extrapolation based methods [1]. The NWP models simulate atmosphere movements by physical equations, while the extrapolation based methods mostly utilize radar information [2]. However, it is worth noting that, many meteorology observatory stations have collected another type of data – a set of weather features at different time slots from multiple observation sites. For example, Guangdong (a province in China) Meteorological Bureau has a lot of sensors to collect weather variables such as wind speed, dew point, temprature, humidity etc, from many observation sites. These observations of a site are sometimes incomplete but provide important clues to the weather at nearby sites, which are not fully exploited in existing work yet. To bridge this gap, we study a model to predict rainfall levels based on the multi-site features collected from a network of rain gauges.

Furthermore, over the years, researchers have demonstrated the effectiveness of Convolutional Neural Network (CNN) in automatically extracting deep features from images, speech, and time series data [3], [4], [5], [6]. However, CNN has not been studied in the task of rainfall prediction yet. Inspired by this, we adapt convolutional neural networks to extract deep features from our raw multi-site features.

Last but not least, although some sites have rich weather features, many others sites do not. In this case, it is desirable to leverage knowledge learned from one rich feature site to help other sites for prediction. Thus we propose a *multi-task* setting for our model. Since sites with similar environment tend to perform similar, while sites with different environments may be different. It will be helpful to study the *commonalities* and *differences* between different sites. Hence we consider to model site correlations to uncover the inter-relationships between different sites in the multi-task model. Experiments show that our multi-task formulation can boost the model prediction performance and the learned site correlation is insightful.

In a nutshell, our full model has three main components: a feature transformation network (CNN model) to represent the input features, an output model to predict rainfall amount, and a multi-task module to incorporate site correlations. In addition to this, the input features are extracted from nearby sites or a network of rain gauges, referred as multi-site features.

We summarize our contributions as follows:

- **Multi-site Features.** Our model considers multi-site features collected from a network of rain gauges. Experi-

2374-8486/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDM.2017.49

ments show that our multi-site features are much more helpful than single-site features.
- **Deep Convolutional Features.** We consider deep CNN for extracting deep features from raw multi-site features. A deep CNN with two convolution layers have shown to be superior over its competing baseline models.
- **Multi-task Setting.** To the best of our knowledge, this is the first work that generalizes rainfall prediction task from independent single-task learning to multi-task relationship learning. Experiments show our approach can boost rainfall prediction performance and the learned site correlation is positively correlated to site geospatial distances.
- **Evaluation.** We evaluated our methods against several baseline models including state-of-the-art neural network models on two real-world datasets and a public weather prediction system, and found our methods have shown to have better results.
- **Cloud based Weather Service.** Our prediction models are built on a Cloud machine learning platform PAI[1]. We are working with many weather prediction centers to deploy our weather prediction, and plan to build a cloud based weather service in the platform.

The remainder of the paper is organized as follows: Section II presents related work of our methods. Section III defines the problem and presents our model. Section IV illustrates the effectiveness of the proposed model in our experimental results. Section V presents two case studies to illustrate the learned site correlation and the importance of multi-site features. Section VI discusses our future work. Finally, Section VII concludes the work.

II. Related Work

Recent studies for rainfall prediction mainly focus on modeling the temporal evolution of rainfall. These work can be grouped into two classes: one is over continuous spatial regions which is referred to as *spatial-temporal model*, and the other is using discrete sets of locations that is refereed to as *multi-site models* [7]. The commonality shared by these two classes of models is that all models are constructed in continuous time.

The spatial-temporal models are appropriate when radar data are available. Extensive researches have been conducted on the physical properties of weather radars [8], on models for calibrating the outputs with land measurements [9], and on integrating commercial data with other weather modalities [10]. Klein et al., applied a new deep network layer called "dynamic convolutional layer" to radar image sequences for the application of weather prediction [11].

Further, instead of building a spatial-temporal model, an alternative approach is to build a multi-site model to explore the inter-site properties. For instance, the authors in [10], [12] studied the cross-relation function of the rainfall intensity at a pair of sites. In contrast, our approach considers to aggregate raw weather features from nearby sites or a network of rain gauges, and use a neural network to extract deep features from them to build a prediction model.

Recently machine-learning techniques have been explored to tackle the challenges of rainfall forecasting. Time series analysis such as ARIMA and simple classifiers based on artificial neural networks have been applied to weather forecasting [13]. In particular, Kuligowski and Barros (1998) applied artificial neural networks for localized precipitation forecast [14]. Bayesian networks [15] and restricted Boltzmann machines (RBM) [16] have also been applied to weather prediction challenges. Further, Grover et al., explored a deep neural network that combined pre-trained predictive models to model joint statistics of a set of weather-related variables [17]. Another architecture based on deep learning is introduced to predict the accumulated daily precipitation in [18]. A similar model is also used in [19]. Among all these machine-learning techniques, thesetwo studies [18], [19] are most closest to ours. However, it is worth noting that, their models are using autoencoder based perception, while we use convolutional neural networks to extract deep features. Furthermore, we incorporate a multi-task formulation to automatically learn the site correlation.

Precipitation prediction problem is a quite challenging and active research topic in the meteorology research community [20]. Typically, they use statistical approaches [21], [22], longer-term numerical weather prediction (NWP) models [2], computer vision techniques [23] and deep learning techniques [24], [25], [26]. Longer-term NWP requires a complex simulation of the physical equations, while computer vision and deep learning methods are usually on radar maps. Convolutional neural networks have been widely used in images, speech, and time series tasks [3], [5], [6] as CNNs are able to extract deep features from the raw data. To the best of our knowledge, CNN has not been applied in the task of leveraging rainfall features for rainfall prediction. We thus consider to use CNN for our study.

Multi-task learning is a learning framework that seeks to boost generalization performance by jointly learning a task and other tasks using a shared representation [27], [28], [29], [30], [31]. To the best of our knowledge, this work is the first to generalize the rainfall task from independent single-task to multi-task learning. Inspired by the work [32], we consider a multi-task relation learning method to automatically learn the task (site) relationships. Experiments show the multi-task relation learning setting helps further boost model performances.

III. Model

Suppose we have S sites. For a site $s \in \{1, 2, \dots, S\}$, the training set contains n_s data points (x_i^s, y_i^s), where $i \in \{1, 2, \dots, n_s\}$. Here x_i^s is an input instance in $\mathbb{R}^{F \times T}$, i.e. multivariate time series data, where F is its feature dimension and T denotes the number of time slots. As we seek to predict the rainfall amount, the task here is a regression problem where $y_i^s \in \mathbb{R}$. Our aim is to train a model to predict label y_i for input x_i.

[1]PAI: Platform of Artificial Intelligence. https://pai.base.shuju.aliyun.com/

A simple way to solve this task is to train an individual model for each site s. Usually the model transforms the raw features to hidden representations and use an output layer (e.g., softmax) for prediction. Let θ_s be the feature transformation network, U_s be the output layer. The objective is defined as:

$$\min_{\{\theta,U\}} \sum_{s=1}^{S}\frac{1}{n_s}\sum_{i=1}^{n_s} l(f(x_i^s,\theta_s,U_s),y_i^s) + \lambda_1\sum_{s=1}^{S}||\theta_s||_F^2 + \lambda_2\sum_{s=1}^{S}||U_s||_F^2, \tag{1}$$

where $f(\cdot)$ denotes the feature transformation function, and l denotes our loss function. The first term illustrates the loss and the second is the regularization. F denotes the Frobenius norm. We refer to this model as *individual model*. Another simple approach is to train one unified model for all sites, i.e. $\forall s, \theta_s = \theta, U_s = U$. The objective is modified as:

$$\min_{\theta,U} \sum_{s=1}^{S}\frac{1}{n_s}\sum_{i=1}^{n_s} l(f(x_i^s,\theta,U),y_i^s) + \lambda_1||\theta||_F^2 + \lambda_2||U||_F^2. \tag{2}$$

We refer to this model as *unified model*.

It is straightforward to see that the individual model does not consider any inherent correlations between different sites, while the unified model assumes all sites are the same. As both of these models fail to incorporate the site correlations, here we propose a multi-task model to address this problem, which will be detailed in the next section.

A. Multi-Task Model

The unified model involves the feature transformation network θ and an output layer U. The process is illustrated as follows.

$$l\big(f(x_i^s,\theta,U),y_i^s\big) = l\big(f(g(x_i^s,\theta),U),y_i^s\big) \tag{3}$$

$$= \big(U^\top g(x_i^s,\theta) - y_i^s\big)^2, \tag{4}$$

where $g : \mathbb{R}^{F\times T} \rightarrow \mathbb{R}^{H\times 1}$ is a feature transformation function, which can be a neural network model or any feature representation methods. H is the hidden representation dimension. As our problem is a regression problem, the output layer U is in $\mathbb{R}^{H\times 1}$. Finally, we take the mean square error as the loss function.

However, the above formulation does not takes the difference of the different sites into consideration. To utilize the multi-site observations, we convert the above model to a multi-task model where we assume each site s has its own $W_s \in \mathbb{R}^{H\times 1}$ as output layer. Note that we use the same feature transformation process for all the models as this encourages the model to learn universal feature representations. As a result, the above model is modified as:

$$l\big(f(x_i^s,\theta,U,W_s),y_i^s\big) = l\big(f(g(x_i^s,\theta),U+W_s),y_i^s\big) \tag{5}$$

$$= \big((U+W_s)^\top g(x_i^s,\theta) - y_i^s\big)^2. \tag{6}$$

Lastly, we explicitly model a *site correlation matrix* $\Omega \in \mathbb{R}^{S\times S}$, where $\Omega_{i,j}$ is a value that indicates the similarity between sites i and j. We adopt the matrix-variate distribution setting from [32], where we confine W_s with the site correlation matrix Ω by using $\mathrm{tr}(W\Omega^{-1}W^\top)$. This means, if W_i and W_j are close, $\Omega_{i,j}$ should be large. In all, we train an unified model θ and U for all sites in our multi-task model, and consider different W_s to model site specific variations. The former is to model *commonalities* and the latter is about *differences*. Formally, the model is defined as follows:

$$\begin{aligned}\min_{W,U,\theta,\Omega}\quad & \sum_{s=1}^{S}\frac{1}{n_s}\sum_{i=1}^{n_s} l(f(x_i^s,\theta,U,W_s,y_i^s)) + \lambda_1\,\mathrm{tr}(W\Omega^{-1}W^\top)\\ & + \frac{1}{2}\lambda_2||\theta||_F^2 + \frac{1}{2}\lambda_3||W||_F^2 + \frac{1}{2}\lambda_4||U||_F^2.\\ s.t.\quad & \Omega \succeq 0,\\ & \mathrm{tr}(\Omega) = 1.\end{aligned} \tag{7}$$

where $\mathrm{tr}(\cdot)$ is matrix trace, $W = [W_s]_{s=1}^S$ are model parameters for all the sites. The two constraints on Ω are used to restrict its complexity. Note that, if there is only one site and Ω is given as a positive scalar, our model is degenerated to the unified model.

In our model, the feature transformation network θ is a *two-layer CNN* and the output layer U is a *fully-connected layer*.

B. Deep Convolutional Neural Network

As shown in Eqn. 7, our model mainly contains two important terms: the first term illustrates the model prediction loss, and the second term models the site correlation. In this work, for the prediction model, we use a deep Convolutional Neural Network (CNN) for extracting deep features and use a fully-connected layer to predict the final labels, which is illustrated in Figure 1. The former is our *feature transformation network*, and the latter is our *output model*. Below we describe the whole architecture.

Input Features. Our input features are either single-site features or multi-site features, both are collected at different time slots. Hence an input instance x belongs to the space $\mathbb{R}^{F\times T}$, where F is the input feature dimension and T is the time slots. In our experiments, T is set as a small number ($T = 6$) as the most recent weather features usually are more important for rainfall prediction.

Sliding Window Strategy. A sliding window strategy is used to segment the time series weather data into a collection of short-time data frames. Let K be the window size, the input x will be split into $T - K$ data frames, each with $F \times K$ features. Each data frame is fed into a convolution layer to extract high-level features. After an empirical analysis, we set K as 2.

Convolution Layer 1. In the convolution layer, the data frames are convolved with several convolutional kernels. The kernel sizes are usually set as $d \times d$ in image processing, as a rectangle region in an image provides important local features. Differently, we set kernel size as $F \times 2$ in our case, as it is beneficial to consider convolution over all the features. This setting is close to the study in [6]. After convolution, we apply a max-pooling on all the extracted convolutional features to

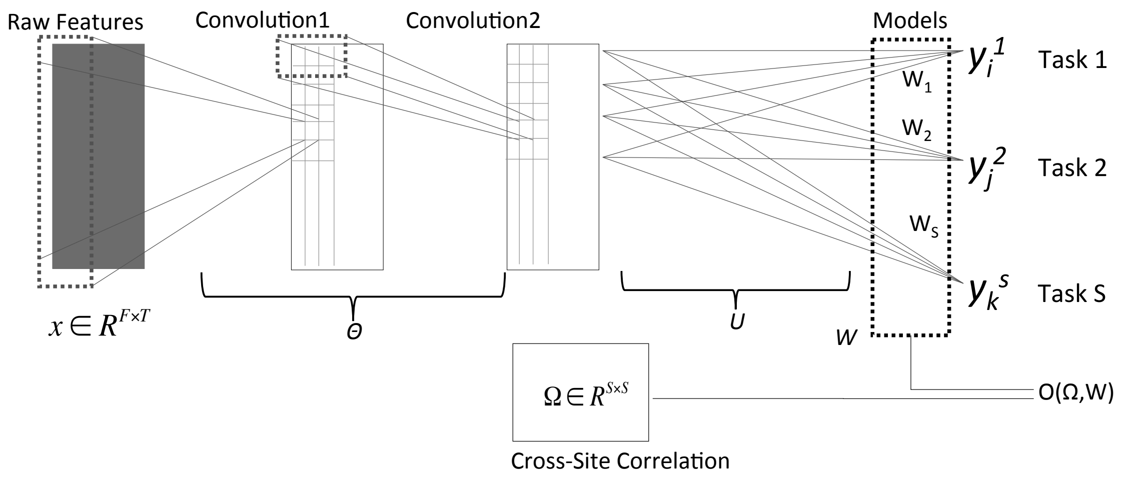


Fig. 1. Our rainfall prediction model. θ is a deep convolutional neural network and U is a fully-connected layer. We set $O(\Omega, W) = \text{tr}(W\Omega^{-1}W^\top)$ in our model.

reduce the length of the feature map. This process can help to minimize the number of model parameters. Our pooling filter sizes are set as 1×2. The whole process is illustrated as follows.

$$z^i = Conv(x^{i-1}), \tag{8}$$
$$h^i = g(z^i), \tag{9}$$
$$x^i = MaxPool(h^i), \tag{10}$$

where i is a layer ID, $g(\cdot)$ is an activation function which is set as Rectified Linear Units (ReLUs) [33] in our model.

Convolution Layer 2. We consider two convolution layers to extract deep features. In this second convolution layer, we apply a different kernel size of 1×3. After convolution, again we apply a pooling layer with filter sizes of 1×2.

The above two convolution layers and two pooling layers correspond to feature transformation network θ in the Eqn. 7.

Fully-connected layer. After the above feature learning process, we concatenate all the features and feed them into a fully-connected layer to get the output. In our case, the output layer is with a shape of $H \times 1$, where H is the feature dimension after previous steps. A deeper network such as multiple-layer perceptron can be applied here to boost the performance. This layer corresponds to feature transformation network U in the Eqn. 7.

C. Inference

As discussed in [32], it is not easy to optimize the above objective function with respect to Ω, W, U, and θ. Alternatively, we use a stochastic alternating method to first optimize θ, W, and U through fixing Ω, and then optimize Ω by fixing θ, W and U. The procedure is repeated within a given iteration or until the convergence criterion is met. Now, we derive the formulations to optimize W, θ and Ω in the following.

Optimizing W, U and θ: While Ω is fixed, we can use gradient descent method to optimize W and θ jointly. Our problem is reformulated as:

$$\begin{aligned} p(W, U, \theta|\Omega) = &\sum_{s=1}^{S} \frac{1}{n_s} \sum_{i=1}^{n_s} l(f(x_i^s, \theta, U, W_s, y_i^s)) \\ &+ \lambda_1 \text{tr}(W\Omega^{-1}W^\top) \\ &+ \frac{1}{2}\lambda_2 ||\theta||_F^2 + \frac{1}{2}\lambda_3 ||W||_F^2 + \frac{1}{2}\lambda_4 ||U||_F^2. \end{aligned}$$

To update W, U and θ, we need to compute the gradients w.r.t W, U and θ, i.e., $\frac{\partial p(\cdot)}{\partial W}$, $\frac{\partial p(\cdot)}{\partial U}$ and $\frac{\partial p(\cdot)}{\partial \theta}$, respectively. Let $l' = \sum_{s=1}^{S} \frac{1}{n_s} \sum_{i=1}^{n_s} l(f(x_i^s, \theta, U, W_s, y_i^s))$, we derive the formulas as follows:

$$\frac{\partial p(W, U, \theta|\Omega)}{\partial \theta} = \frac{\partial l'}{\partial \theta} + \lambda_2 \theta, \tag{11}$$

$$\begin{aligned} \frac{\partial p(W, U, \theta|\Omega)}{\partial W} &= \frac{\partial l'}{\partial W} + \lambda_3 W + \lambda_1 \frac{\partial \, \text{tr}(W\Omega^{-1}W^\top)}{\partial W} \\ &= \frac{\partial l'}{\partial W} + \lambda_3 W + \lambda_1 W \Omega^{-1}, \end{aligned} \tag{12}$$

$$\frac{\partial p(W, U, \theta|\Omega)}{\partial U} = \frac{\partial l'}{\partial U} + \lambda_4 U. \tag{13}$$

These three terms $\frac{\partial l'}{\partial W}$, $\frac{\partial l'}{\partial U}$ and $\frac{\partial l'}{\partial \theta}$ can be computed by BackPropagation. More details can be found in [3], [34]. With these, we can then update W, U and θ accordingly.

Optimizing Ω: With the fixed W, θ, our objective here is to solve the following optimization problem.

$$\begin{aligned} \min_{\Omega} \quad & \text{tr}(W\Omega^{-1}W^\top) \\ s.t. \quad & \Omega \succeq 0, \quad \text{tr}(\Omega) = 1. \end{aligned} \tag{14}$$

According to [32], the above convex problem has a closed-form solution: $\Omega = \frac{(W^\top W)^{\frac{1}{2}}}{\text{tr}\left((W^\top W)^{\frac{1}{2}}\right)}$. Clearly, if W_i and W_j are closer, $\Omega_{i,j}$ tends to be larger.

Initializing Ω: A simple way to initialize Ω is to set it as an identity matrix $\Omega = \frac{1}{S}\mathbf{I}_S$. Our intuition is that those sites that are close to each other should share similar models, i.e. if site i and j are with small geospatial distance, $\Omega_{i,j}$ should be large. Motivated by this, we use sites' geospatial distance to initialize Ω.

Let Φ be a matrix that indicates the affinity scores between different sites. We define $\Phi_{i,j}$ as follows:

$$\Phi_{i,j} = \exp^{-\frac{d_{i,j}^2}{2\sigma^2}}, \tag{15}$$

where $d_{i,j}$ is the geospatial distance between site i and site j, and σ is a given scalar.

We can treat Φ as an site correlation matrix and assume that sites physically close to each other should have high correlation. In our model training, we first set $\Omega = \Phi$ and let the model tune Ω afterwards.

Finally, we present the inference algorithm in Algorithm 1. Note that the presented algorithm is a simple stochastic gradient method where learning rate η is fixed. In our implementation, we use Adam [35] to compute adaptive learning rates for each parameter.

```
Data: Input instances $\left\{\{x_i^s, y_i^s\}_{i=1}^{n_s}\right\}_{s=1}^{S}$
Result: Model parameters $\theta, W, U, \Omega$
Initialize $\theta, W, U$ using random values
Initialize $\Omega = \Phi$
while not reach convergence do
    count = 0
    Let $\eta$ be the learning rate
    while count $<$ Threshold do
        Read a minibatch
        Update $\theta \leftarrow \theta - \eta\frac{\partial p(W,U,\theta|\Omega)}{\partial \theta}$
        Update $U \leftarrow U - \eta\frac{\partial p(W,U,\theta|\Omega)}{\partial U}$
        Update $W \leftarrow W - \eta\frac{\partial p(W,U,\theta|\Omega)}{\partial W}$
        count += 1
    end
    Update $\Omega \leftarrow \frac{(W^\top W)^{\frac{1}{2}}}{\mathrm{tr}\left((W^\top W)^{\frac{1}{2}}\right)}$
end
```

Algorithm 1: Inference algorithm for our final model (Multi-Task Convolutional Neural Network).

The overall system is implemented on the PAI platform [2] that supports distributed training and evaluating using multi-CPU and multi-GPU. Our method and all the baseline methods are trained on machines with Nvidia Tesla K40 GPU.

IV. Experiments

In this section, we begin with a quantitative evaluation for the proposed methods, and then present some discussions on our models. We organize our experiment results as below.

- Sec. IV-A $\sim$ IV-C describe experiment setup details.
- Sec. IV-D shows the proposed multi-site features are more helpful than single-site features in rainfall prediction;
- Sec. IV-E shows the proposed convolutional architecture helps to find more advanced deep features that contribute to a better prediction result;
- Sec. IV-F illustrates the effectiveness of the proposed multi-task setting of our model;
- Sec. IV-H examines the importance of initializing site correlation Ω;
- Sec. IV-I shows our final model can outperform a public rainfall prediction service significantly.

A. Datasets

We evaluated the proposed methods on two real-world datasets. The first one is the daily accumulated rainfall in a meteorological station located in Manizales city (MCdata for short) [3], first used in [18]. Another is a large-scale rainfall dataset collected by the observation sites of Guangdong province [4], China, denoted as GDdata. The GDdata is recorded mostly at a 5-minute granularity. That is, the data instances are generated at every 5 minutes, but the features are from the most recent three hours. Hence two consecutive data instances share some overlapped temporal information. The label for each data instance is the total rainfall amount in the future 3 hour.

Note that, the first dataset MCdata is a single-task dataset, while the other GDdata is a multi-task dataset. We used the first dataset to test our model feature extraction power and the second to evaluate our full model performance. The statistics of our datasets are shown as follows.

TABLE I
Statistics of our datasets

	MCdata	GDdata
# Sites	N.A.	187
# Training Records	2952	2,429,375
# Validation Records	632	626,077
# Testing Records	632	626,077
Label	next day	next 3 hours
Duration	2002 $\sim$ 2013	2013 $\sim$ 2015

The GDdata contains eight types of weather variables as features, which is shown in Table II. The wind speed β with angle α is represented by a two-dimensional vector with the dimensions corresponding to the speeds of two spatial directions, i.e. $\beta \times cos(\alpha), \beta \times sin(\alpha)$. The other weather variables are scalars. Totally we have 17 features. For each type of feature, we record the maximum and the sum of the values within three time intervals from the current time point (last 1, 2, and 3 hours) as our features. In all, we have $17 \times 2 \times 3 = 102$ features.

B. Multi-site Features

The performance of regression methods depends on the evidential weather features. Also with more spatiotemporal features of the site, it is possible to obtain more accurate

[2]https://pai.base.shuju.aliyun.com/

[3]The dataset can be downloaded from this platform called CDIAC, and its link is: http://cdiac.manizales.unal.edu.co/IDEA/resultados.php

[4]http://www.grmc.gov.cn/

TABLE II
17 FEATURE DIMENSIONS OF GDDATA.

Feature (Dimension)	Details
Rain condition (2)	rainy or not; rain amount
Observatory height (1)	height of our observatory site
Wind speed (2)	$\beta \times cos(\alpha), \beta \times sin(\alpha)$, where β and α are the speed and angle
Wind direction (8)	N,NE,E,SE,S,SW,W,NW (N:North,S:South, E:East,W:West)
Dew point (1)	dew point value
Temperature (1)	temperature of surface
Air pressure (1)	air pressure of the sites
Humidity (1)	air humidity of the sites

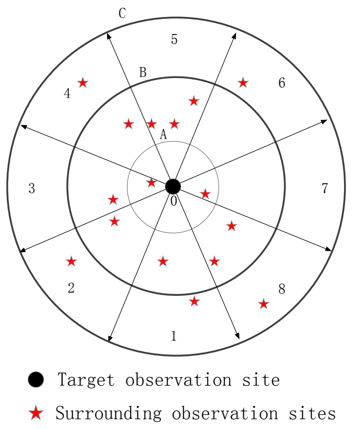


Fig. 2. Multi-site features from an observation site of interest (labeled as 0) and eight surrounding regions (1 to 8) with three distance spans (A,B,C). All together we have 25 cells (1 self + 8× 3 surrounding cells), from each cell we extract 102 features.

prediction results. This is justified by the fact that the weather of a site can provide important clues to the weather of nearby sites. For example, site A is 50km away from site B to the east. If it is rainy with a northeast wind in site A, possibly site B will be rainy in near future. According to this, the ensemble features of different weather variables are introduced to capture the relationships among various sites. We divide the spatial region around the site of interest into normally distributed spatial cells by different directions, distances, and heights. Without loss of generality, we tried eight surrounding directions (labeled as 1 to 8) with three with spans (labeled as A,B,C) as in Fig. 2. We end up having 8×3 cells and one self-cell, 25 cells in total. The ensemble features are then obtained by concatenating the weather features in the 25 cells. All together we have $102 \times 25 = 2550$ features. The resulting features are referred as *multi-site features*.

All the features are normalized to $[0, 1]$ via $\frac{V[:,i]-\min(V[:,i])}{\max(V[:,i])-\min(V[:,i])}$ where $V[:, i]$ is the ith features for all the instances.

C. Models and Criteria

We compared our approach with the following methods:

- **QPF:** Quantitative Precipitation Forecast is the expected amount of melted precipitation accumulated over a specified time period over a specified area [36]. The method is used by Guangdong meteorological center. We directly obtain the QPF results from the center.
- **LR:** We consider a liner model - linear regression (LR) as our baseline method.
- **MLP:** We consider a multiple-layer perceptron as another baseline model. This model has been used in [18].
- **AE-MLP:** We also consider a denoising autoencoder based multiple-layer perceptron. This model has shown to have state-of-the-art results in [18], [19].
- **RNN:** We consider a Recurrent Neural Network (RNN) as another baseline model. We consider LSTM as our RNN unit [37].
- **CNN:** We propose to use CNN architecture for the task.
- **MT-MLP:** The multi-task MLP method.
- **MT-RNN:** The multi-task RNN method.
- **MT-CNN:** The multi-task CNN method, our final model.

For the MLP and AE-MLP, we used the same structure as used in [18] on MCdata. For GDdata, we used a two layer network and choose the best structure from: $[1024, 256]$, $[512, 128]$, $[256, 64]$, $[128, 32]$, and $[64, 16]$ based on their performance. We've also tested three layers as well and find comparable results. For CNN, the kernels in the first convolution layer and second convolution layer are of sizes 2550×2 and 1×3, respectively. The fully-connected layer of CNN and MT-CNN is a single layer. The hidden layer size is chosen from $1024, 512, 256$, and 128 based on the performance. For RNN, the hidden state size is chosen from $1024, 512, 256$, and 128.

To evaluate the performance of these methods, we used the Mean Square Error (MSE). Let X be the predicted labels and Y be the vector of observed labels, then

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i)^2. \quad (16)$$

We also evaluated these model using several commonly used precipitation forecasting metrics: critical success index (CSI) and correlation (Corr). According to [26], CSI is computed based on these three metrics: hits, misses, false alarms. We first converted the prediction and ground truth to a 0/1 matrix using a threshold of 0.5mm/h rainfall rate (indicating raining or not) and then calculated the hits (prediction = 1, truth = 1), misses (prediction = 0, truth = 1) and false alarms (prediction = 1, truth = 0). Then, we compute:

$$\text{CSI} = \frac{hits}{hits + misses + falsealarms} \quad (17)$$

$$\text{Corr(X,Y)} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (18)$$

Note that for MSE the smaller the better, while for Corr and CSI the larger the better.

D. Single-site Features vs. Multi-site Features

We first evaluated the performance of the base models, i.e. LR, MLP, RNN and CNN, on both single-site and multi-site

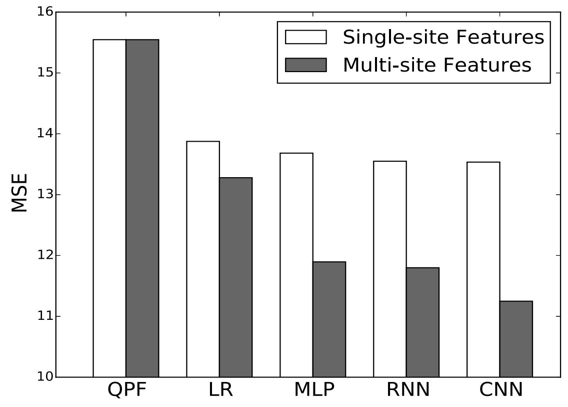


Fig. 3. The MSE of using multi-site features vs. single-site features.

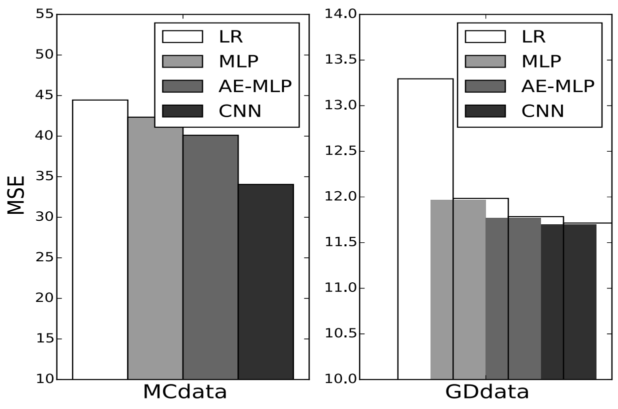


Fig. 4. The Comparison of Our Final Model and Other Methods.

features in our GDdata. We omitted the study of MCdata here as it only has single-site features.

Firstly, as shown in Fig. 3, those data-driven methods like LR, MLP, RNN and CNN significantly outperform QPF. This shows the extracted features from meteorological features can help to build a more accurate rainfall prediction model.

Secondly, deep learning methods like MLP, RNN and CNN outperform the linear method LR. This shows the fact that non-linear methods are better on handing complex relationships among meteorological features than basic linear models.

Thirdly, the CNN model shows better performance than other competing baselines including RNN. A reason why CNN has better performance than RNN is that our multi-site features already contain aggregated features from different time spans. Furthermore, CNN model is good at extracting local patterns from a small time window, which shows to beneficial for the short-term rainfall prediction task. We suspect that the RNN model may have better results for long-term rainfall prediction task as it is able to mine longer time dependent features. We leave this study as future work, as for such tasks, it is necessary to mine richer features from a longer time span.

Last, all the methods achieve better performances with multi-site features than single-site features. This shows that weather conditions from surrounding sites can provide richer information for predicting a target site's rainfall level.

E. Deep Convolutional Features

We further compared the effectiveness of our deep convolutional features on both MCdata and GDdata in Table 4.

First, neural network based methods significantly outperform linear model (LR in our experiment) in all the criteria.

Second, AE-MLP that encodes raw features into a hidden vector can help boost the performance of MLP model.

Last but not least, our CNN model achieves the best performance when comparing with all the baseline methods including AE-MLP, the state-of-the-art method in this work [18], on both MCdata and GDdata. We conducted one-tailed paired t-test on the results. We found CNN significantly outperforms other models, with $p < 1E-7$ in MCdata and $p < 0.05$ in GDdata. This shows the CNN model can find more meaningful features that contribute to better predictions.

F. Multi-task settings.

We considered multi-tasks settings of the proposed CNN architecture, and compared the resulting final MT-CNN model with other baseline methods.

TABLE III
A COMPARISON OF OUR FINAL MODEL AND OTHER METHODS.

GDdata	MSE	Correlation	CSI
QPF	15.547	0.399	0.866
LR	13.28	0.469	0.875
MLP	11.966	0.472	0.711
MT-MLP	11.894	0.473	0.824
RNN	11.809	0.480	0.881
MT-RNN	11.801	0.488	0.884
CNN	11.699	0.492	**0.886**
MT-CNN	**11.253**	**0.521**	0.872

The results are shown in Table III. A few important observations are discussed in order.

1. QPF. Although this method has been widely used in many weather forecasting centers, we found data-driven methods have better results. This is encouraging as it shows that it is beneficial to train a prediction model based on collected weather features.

2. Multi-task vs. single-task. MT-CNN, MT-RNN and MT-MLP have better results than CNN, RNN and MLP respectively, this shows that treating all the sites as one task is not optimal. Multi-task model, on the other hand, can incorporate cross-site (cross-task) correlation, which helps to boost the performance.

3. Our final model. Our final model MT-CNN has the best performance compared with other baseline models in terms of MSE and Corr. A minor downgraded performance has been found on CSI while compared with CNN model. Since CSI is computed in terms of rainfall amounts within 0.5mm/h, this means CNN model has a bit better performance in terms of small rainfall amount prediction. Close examinations show that our model has better performance than CNN model in terms of predicting rainfall amount larger than 0.5mm/h. As MSE and Corr represent an overall evaluation results, our final model enjoys clear advantages over other competing methods in general, with $p < 0.05$ using one-tailed paired t-test.

G. Individual model vs. unified model vs. multi-Task model

As discussed in Section III, there are two degenerate variants of our multi-task model: one is an individual model where each site has its own model, the other is an unified model where all sites share the same model. We evaluated their performance with our final model. Note that all these three models use the same CNN model as base.

TABLE IV
THE COMPARISON BETWEEN INDIVIDUAL MODEL, UNIFIED MODEL AND MULTI-TASK MODEL IN GDDATA.

GDdata	Individual	Unified	Multi-Task
RMSE	11.987	11.699	**11.253**

The individual model has the worst performance, the potential reason is that each site has its own model thus has fewer training samples than the other two models. The unified model does not consider the differences between different sites which is not ideal for our task. Our multi-task model has better performance than the other two models as it models both commonalities and differences between different sites.

H. Site correlation

Recall that in our model, we used Φ, computed based on geospatial distances, to initialize the site correlation matrix Ω. To illustrate the importance of this initialization, we compared it with initialization with identity matrix, i.e. $\Omega = \frac{1}{S}\mathbf{I}_S$, referred to as MT-CNN+No Prior.

TABLE V
THE IMPORTANCE OF USING Φ TO INITIALIZE Ω.

GDdata	MSE	Correlation	CSI
MT-CNN+No Prior	11.533	0.501	0.871
MT-CNN	**11.253**	**0.521**	0.872

Clearly, with the help of Φ, the final model improves a little bit. This Φ provides a good starting point for Ω that makes the learning of Ω a bit easier, which in turn benefits the final model. Note that, even without this initialization, our final model still achieves the best results against the competing methods in Table III.

I. Comparison with a public system

We further evaluated our model's performance with a public rainfall prediction system, European Centre for Medium-Range Weather Forecasts (ECMWF), in terms of Corr and CSI. As shown in Fig. 5, the proposed model significantly outperforms the ECMWF results in terms of all the results with

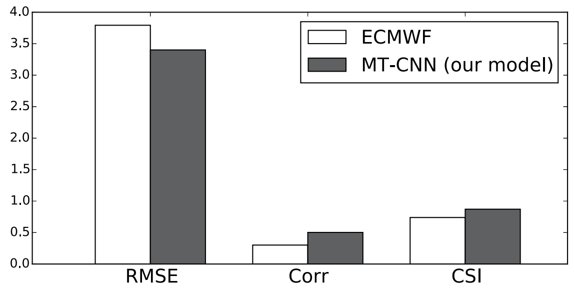


Fig. 5. A Comparison with ECMWF. RMSE is the Root MSE, Corr is Correlation, and CSI is Critical Success Index.

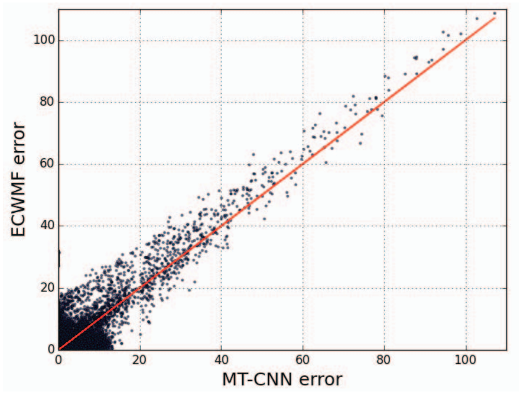


Fig. 6. The MT-CNN prediction errors vs. ECMWF prediction errors.

$p < 1E-40$ using one-tailed paired t-test. This shows that by using the raw features from observation sites, our model can achieve reasonable good rainfall prediction results.

To further explore the results obtained by the proposed model, we compared the relationship between the MT-CNN and the ECMWF prediction error in Fig. 6. The horizontal axis represents the error between the MT-CNN predicted value and the actual observed value, while the vertical axis represents the error between the ECMWF predicted value and the actual observed value. Those data points lie in the straight line with slope 1 and intercept 0 are tie results, while those above the line are wining results from MT-CNN, and below the line are wining results from ECMWF. Clearly, there are more data points above the line, which shows our MT-CNN has better results. We also found that the ECMWF suffers from more large errors than our method.

V. CASE STUDIES

In this section, we present two case studies to examine the learnt site correlation and the importance of multi-site features.

A. Site correlation.

Recall that in our experiment, we used Φ as the prior of the site correlation matrix Ω. To illustrate the rational behind this, we removed such prior and set Ω as an identity matrix initially. We then examined the correlation between the learned Ω and Φ.

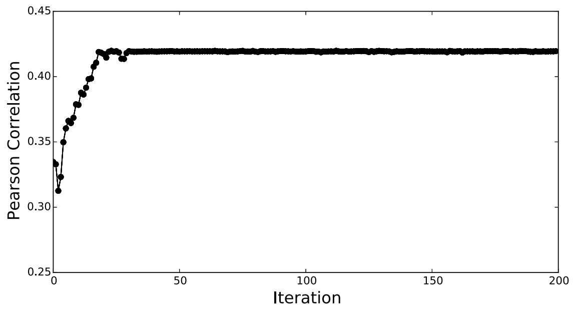


Fig. 7. A case study of Pearson correlation between learned site correlation and geospatial distance (the larger the better). 0.42 means a medium positive correlation.

We measured the closeness of our learned Ω and Φ using Pearson correlation score, and monitor the correlation score in different iterations. As shown in Fig 7, the results show a medium positive correlation between learned site correlation and geospatial distance, i.e. Ω and Φ, at the end of iterations. Initially the correlation score is around 0.32, the score improves sequentially and finally reach 0.42. Note that Ω and Φ do not guarantee to be strongly positive correlated, as in reality the site correlation would be affected by not only geolocations but also other weather and environment variables, e.g. topography, weather condition, and even climate changes. The results are still encouraging as this shows geolocations can help to learn site correlation.

B. Model results and multi-site features.

We also considered a real case study to examine the effectiveness of our model results in Fig. 8. At 9pm, the area in circle was still sunny, while at midnight it started raining. For this case, our method successfully predicted the raining case, while ECMWF and QPF failed. This shows our model is able to make use of multi-site features to make better predictions.

We also considered an empirical study to examine the importance of multi-site features. There exists many other samples that are close to the above raining case. Based on the multi-site features, we carried out k-means clustering to find such similar samples in training set. Among the similar samples found by our clustering method, we found that 54% of them are in raining cases, against 16% on average on all the training samples. This shows that our multi-site features help to identify such raining cases.

VI. Discussion

Radar Images. The convolutional LSTM [26] has a good performance in precipitation prediction using spatial-temporal radar image sequence. Our dataset only consists of weather variables collected by sensors of observation sites. In the near future, we will work on datasets with both weather variables and radar images to test these methods. Furthermore, it is also interesting to test multi-task setting of these models. Our framework is a general multi-task learning framework, where the learning method (CNN in our study) can be replaced by any other deep learning methods.

Multi-site Features. In this work, we propose to extract multi-site features to help our prediction task. It is also interesting to automatically learn such features. For example, to develop a special kind of convolution kernel to extract multi-site features.

Multiple Data Sources. We seek to collaborate with more weather forecasting centers to get more rich data sources such as topological information, cloud graph data, and so on.

Long-term Rainfall Prediction. In the near future, we will work on long-term precipitation forecasting, for example weather prediction for next week or even month. In long-term forecasting, richer features with a longer time period need to be explored.

VII. Conclusion

In this work, we studied how to use a set of weather features collected from multiple surrounding observation sites for rainfall prediction. We proposed a neural network based approach, which leverages the advantage of convolutional neural networks to automatically extract deep features from multi-site observation data. We explicitly incorporated site correlations on a multi-task setting in our model. Extensive experiments showed that the proposed approach performs significantly better than the baseline methods including several state-of-the-art deep learning based rainfall prediction models. We further compared the proposed approach with a public weather forecast center results and demonstrated the effectiveness of our model.

Acknowledgment

The authors would like to thank Emilcy Juliana Hernandez Leal and Victor Sanchez-Anguix for sharing the data set. The authors would also like to thank for the helpful feedback and comments provided by Yaliang Li, Liang Feng, and Cen Chen, as well as for the valuable comments from the anonymous reviewers.

References

[1] J. Sun, M. Xue, J. W. Wilson, I. Zawadzki, S. P. Ballard, J. Onvlee-Hooimeyer, P. Joe, D. M. Barker, P.-W. Li, B. Golding, M. Xu, and J. Pinto, "Use of nwp for nowcasting convective precipitation: Recent progress and challenges," *Bulletin of the American Meteorological Society*, vol. 95, no. 3, pp. 409–426, 2014.

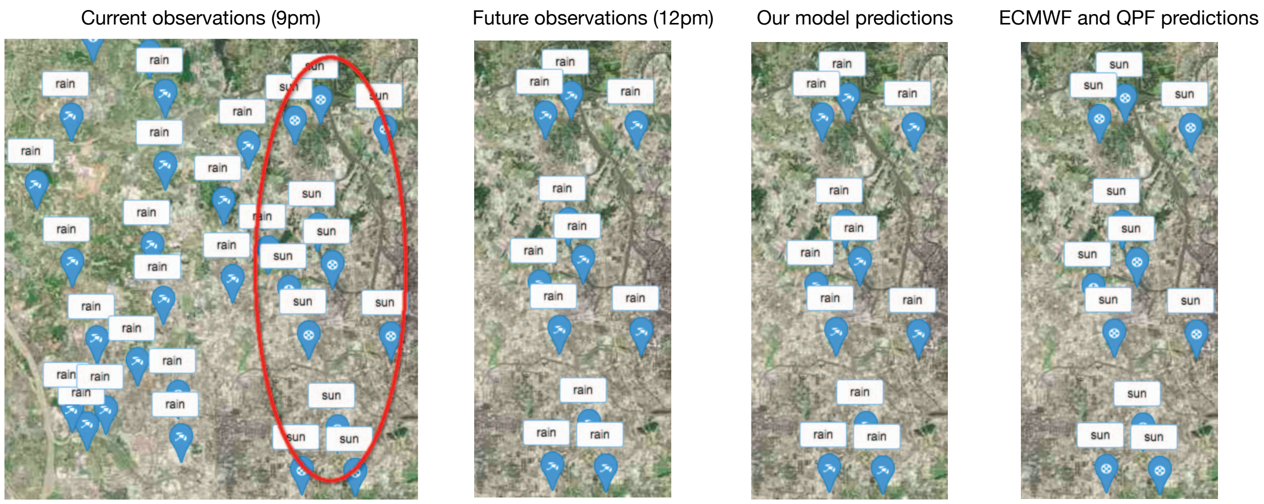
[2] M. Reyniers, "Quantitative precipitation forecasts based on radar observations: Principles, algorithms and operational systems," *Institut Royal Met eorologique de Belgique*, 2008.

[3] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 1995.

[4] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, G. Orr and M. K., Eds. Springer, 1998.

[5] Y. LeCun, K. Kavukvuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. International Symposium on Circuits and Systems (ISCAS'10)*. IEEE, 2010.

[6] G. Sateesh Babu, P. Zhao, and X.-L. Li, *Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life*, 2016, pp. 214–228.



Current observations (9pm)
Future observations (12pm)
Our model predictions
ECMWF and QPF predictions
rain
sun

Fig. 8. A case study of our model's prediction results.

[7] H. S. Wheater, V. S. Isham, D. R. Cox, R. E. Chandler, A. Kakou, P. J. Northrop, L. Oh, C. Onof, and I. Rodriguez-Iturbe, "Spatial-temporal rainfall fields: modelling and statistical aspects," *Hydrology and Earth System Sciences Discussions*, vol. 4, no. 4, pp. 581–601, 2000.

[8] O. Caumont, A. Foray, L. Besson, and J. Parent du Châtelet, "An observation operator for radar refractivity change: Comparison of observations and convective-scale simulations," *Boundary-Layer Meteorology*, vol. 148, no. 2, pp. 379–397, 2013.

[9] L. Besson, C. Boudjabi, O. Caumont, and J. Parent du Chatelet, "Links between weather phenomena and characteristics of refractivity measured by precipitation radar," *Boundary-Layer Meteorology*, vol. 143, no. 1, pp. 77–95, 2012.

[10] C. G. Collier, "Flash flood forecasting: What are the limits of predictability?" *Quarterly Journal of the Royal Meteorological Society*, vol. 622, no. 133, pp. 3–23, 2007.

[11] B. Klein, L. Wolf, and Y. Afek, "A dynamic convolutional layer for short range weather prediction," in *CVPR*, 2015.

[12] A. Kakou, "Point process based models for rainfall," *University College London (University of London), PhD Thesis*, 1997.

[13] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Comp. Int. Mag*, vol. 4, no. 2, pp. 24–38, 2009.

[14] R. J. Kuligowski and A. P. Barros, "Localized precipitation forecasts from a numerical weather prediction model using artificial neural networks," *Weather and Forecasting*, pp. 1194–1204, 1998.

[15] A. S. Cofiño, R. Cano, C. Sordo, and J. M. Gutiérrez, "Bayesian networks for probabilistic weather prediction," in *Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002*, F. van Harmelen, Ed. IOS Press, 2002, pp. 695–699.

[16] R. Mittelman, B. Kuipers, S. Savarese, and H. Lee, "Structured recurrent temporal restricted boltzmann machines," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, 2014, pp. 1647–1655.

[17] A. Grover, A. Kapoor, and E. Horvitz, "A deep hybrid model for weather forecasting," in *Proceedings of the 21th ACM SIGKDD 2015*, 2015, pp. 379–386.

[18] E. Hernández, V. Sánchez-Anguix, V. Julián, J. P. Cámara, and N. D. Duque, "Rainfall prediction: A deep learning approach," in *11th International Conference on Hybrid Artificial Intelligent Systems, HAIS 2016*. Springer, 2016, pp. 151–162.

[19] J. N. Liu, Y. Hu, J. J. You, and P. W. Chan, "Deep neural network based feature representation for weather forecasting," in *Proceedings of International Conference on Artificial Intelligence (ICAI)*, 2014.

[20] B. T. Olsen, U. S. Korsholm, C. Petersen, N. W. Nielsen, B. H. Sass, and H. Vedel, "On the performance of the new nwp nowcasting system at the danish meteorological institute during a heavy rain period," *Meteorology and Atmospheric Physics*, vol. 127, no. 5, pp. 519–535, 2015.

[21] D. R. Cox and V. Isham, "A simple spatial-temporal model of rainfall," in *The Royal Society A*, 1988.

[22] P. Northrop, "Modeling and statistical analysis of spatial-temporal rainfall fieldsph.d. thesis, department of statistical science," *University College London (University of London), PhD Thesis*, 1996.

[23] P. Cheung and H. Yeung, "Application of optical-flow technique to significant convection nowcast for terminal areas in hong kong." in *In the 3rd WMO International Symposium on Nowcasting and Very ShortRange Forecasting (WSN12)*, August 2012, p. 610.

[24] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," 2016, book in preparation for MIT Press. [Online]. Available: http://www.deeplearningbook.org

[25] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[26] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *NIPS 2015*, 2015, pp. 802–810.

[27] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, Jul. 1997.

[28] B. Bakker and T. Heskes, "Task clustering and gating for bayesian multitask learning," *J. Mach. Learn. Res.*, vol. 4, pp. 83–99, Dec. 2003.

[29] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proceedings of the Tenth KDD*, 2004, pp. 109–117.

[30] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in *NIPS 19*, P. B. Schölkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 41–48.

[31] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient l2, 1-norm minimization," in *UAI 2009*, 2009, pp. 339–348.

[32] Y. Zhang and D.-Y. Yeung, "A convex formulation for learning task relationships in multi-task learning," in *UAI'10*, 2010, pp. 733–742.

[33] G. E. Hinton, "Rectified linear units improve restricted boltzmann machines vinod nair," *ICML*, 2010.

[34] D. Rumelhart, G. Hintont, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[35] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[36] J. S. Bushong, "Quantitative precipitation forecast: Its generation and verification at the southeast river forecast center," *Georgia Institute of Technology*, 1999.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, Nov. 1997.