IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 10, 2020, accepted April 23, 2020, date of publication April 30, 2020, date of current version May 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2991462

City-Wide Traffic Congestion Prediction Based on CNN, LSTM and Transpose CNN

NAVIN RANJAN, SOVIT BHANDARI, HONG PING ZHAO, HOON KIM, AND PERVEZ KHAN
IoT and Big-Data Research Center, Department of Electronics Engineering, Incheon National University, Incheon 22012, South Korea

Corresponding author: Pervez Khan (pervaizkanju@hotmail.com)

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1B07050418) and the Post-Doctoral Research Program of Incheon National University 2016.

ABSTRACT Traffic congestion is a significant problem faced by large and growing cities that hurt the economy, commuters, and the environment. Forecasting the congestion level of a road network timely can prevent its formation and increase the efficiency and capacity of the road network. However, despite its importance, traffic congestion prediction is not a hot topic among the researcher and traffic engineers. It is due to the lack of high-quality city-wide traffic data and computationally efficient algorithms for traffic prediction. In this paper, we propose (i) an efficient and inexpensive city-wide data acquisition scheme by taking a snapshot of traffic congestion map from an open-source online web service; Seoul Transportation Operation and Information Service (TOPIS), and (ii) a hybrid neural network architecture formed by combing Convolutional Neural Network, Long Short-Term Memory, and Transpose Convolutional Neural Network to extract the spatial and temporal information from the input image to predict the network-wide congestion level. Our experiment shows that the proposed model can efficiently and effectively learn both spatial and temporal relationships for traffic congestion prediction. Our model outperforms two other deep neural networks (Auto-encoder and ConvLSTM) in terms of computational efficiency and prediction performance.

INDEX TERMS Convolutional neural network, long short-term memory, partially convolutional neural network, spatiotemporal feature, traffic congestion forecasting, transport network.

I. INTRODUCTION

With the increase in the economy, rapid urbanization, and desire toward a private traveling [1], the traffic congestion level of most of the large and growing cities around the world has increased drastically, which directly affects the growth, development, and environment of the cities. Besides, it also increases both commuting time and the tendency of road rage, increasing the frequency of road accidents [2]–[4]. Thus, the study of traffic management is of high importance among researchers. The high congestion can be alleviated either by increasing the transportation infrastructure, which is the expensive method, or by deploying feasible traffic strategies, such as congestion pattern analysis or short-term traffic information prediction that can efficiently be employed to the existing road network in a fraction of cost. Compared to pattern analysis, which determines the road networks having reoccurring congestion [5], [6], predicting the accurate short-term traffic information [7]–[10] such as traffic speed, traffic flow volume, and congestion level, is more informative to the commuters and traffic management agencies. Among them, the traffic congestion level gives the status of the road network (Jam, Slow, or Free) more desirable parameter for the drives to make better route choices circumventing the congested roads, and traffic managers to operate efficiently by systematically responding to the supply and demand change of transportation network.

The early forecasting models focused on the prediction of traffic parameters such as speed, volume, and traffic flow on a single road, group of roads, or small road networks mainly due to lack of data availability. Due to its partial prediction capabilities on road networks, these works were not desirable to both commuters and traffic agencies. The existing work uses data from a fixed sensor (road sensor, inductive loop, traffic camera, etc.) installed on every road or the network of vehicles (VANET, Floating Car) operating on each route. These kinds of data are difficult to collect because installation, operating and, maintenance are expensive as well as

The associate editor coordinating the review of this manuscript and approving it for publication was Gustavo Callico.

it is difficult to access from the third party, as it requires special permission. Recently the web service like Google Traffic [11], Bing Map [12], Seoul Transportation Operation and Information Service (TOPIS) [13], and Baidu Map [14], publicly started to provide the accurate city-wide real-time traffic information such as congestion level and the average speed of the road segment. Although these web services are public, easily accessible, and provide traffic information for most of the cities in the world, there is only a handful number of studies based on them. The one reason is the curse of dimensionality, as the prediction problem is a time-series analysis which takes multiple inputs, so processing multiple traffic map image is costly.

Long Short-Term Memory (LSTM) recurrent neural network is known for its capability of mining and remembering temporal relationship over a long sequence of historical data and has been very successful in varieties of the area such as recognition [15], translation and time series prediction [16], [17]. But, LSTM can be challenging to use and is slow at the processing when input and output are the sequences of high-resolution two-dimensional data. Similarly, the Convolutional Neural Network (CNN) has gained its fame in spatial learning by extracting a low-resolution feature image from the high-resolution input image and has been successful in the domain of image understanding, object detection, and segmentation [18], [19]. Unlike LSTM, CNN doesn't have any difficulty in processing high-resolution multidimensional data because of its unique ability like local connectivity, weight sharing, and pooling. Image segmentation work in [18] uses convolutional encoder and decoder architecture, where convolutional encoder encodes the input image into a low-resolution image consisting of important spatial features and convolutional decoder decodes the latent representation back to its original size, in the process, the network learns to segment the image into various groups.

Inspired by the successful application of convolutional autoencoder and LSTM mentioned above, in this paper, we present an approach, which can learn both spatial and temporal relationships between the sequences of historical image data for traffic congestion prediction. In the proposed architecture, we add the LSTM network between the convolutional encoder and the convolutional decoder. The convolutional encoder at first converts the sequence of the input image into sequences of low-resolution latent state. The LSTM network then learns the time series representation from the sequences, and finally, the convolutional decoder converts the latent state back to its original resolution. The contribution of the paper can be summarized as follows:

- We develop a new prediction model PredNet. The model exploits the advantage of various deep learning architecture, including convolutional neural network, LSTM network, and transposed convolutional neural networks to learn the spatiotemporal sequences of historical data for efficient end-to-end traffic congestion prediction on the transportation network.
- The proposed model can be generalized to large-scale traffic prediction problems while retaining trainability on resource constraint devices because of the implementation of convolutional, downsampling, and upsampling layers.
- Our extensive experiments on the Seoul city transportation network demonstrate the efficiency and effectiveness of the proposed approach.

In this section, we describe the background and motivation of the study and briefly highlight the inspiration for modeling the proposed algorithm. The rest of the paper is structured as follows: Section II presents the literature related to work on traffic prediction. Section III presents the methodology of traffic prediction, which includes problem statement, data source, preprocessing and database design, and explain the architecture of our hybrid deep learning neural network which learns both spatial and temporal features for traffic congestion prediction. Section IV presents the data description for the model, metrics used for testing the effectiveness of the proposed model, detailed explanation for model construction, and performance comparison of the proposed model with two state-of-art prediction models like ConvLSTM and Autoencoder. Finally, Section V presents the conclusion of our work and provide the future direction of this study.

II. RELATED WORK

In the past, the researchers used the data-driven approach, mainly focused on the development of the statistical and mathematical model to analyze the time-series relation in the traffic data, also referred to as a parametric approach. Primarily, the work was based on the assumption of linearity and stationarity to capture future trends, such as the historical average model [20], smoothing techniques [21], and error component model [22]. Later, a typical parametric time-series model autoregressive integrated moving average (ARIMA) model [23] was introduced to identify the pattern by decomposing long-term trends and seasonal patterns. However, it suffers from the tendency to concentrate on the mean value of the time-series and unable to predict the extremes [24]. The family of the ARIMA-based model, such as seasonal ARIMA models [25], and Kalman filter model [26], use a vast historical database for model development and are also very sensitive to the traffic data.

Due to limitations in a parametric approach, the researchers start to pay attention to nonparametric models, k-nearest neighbor (KNN), support vector machine (SVM), bayesian network (BN), and artificial neural network (ANN). Unlike the parametric approach, the nonparametric model relies on the training data to determine the model structure and number of parameters. A KNN model [27] searches a close neighbor matching to the current data from the historical database to predict the traffic flow. The SVM model [28] is based on the structural risk minimization principle, has a unique advantage in the fields of small samples and high-dimensional nonlinear data. The family of SVM-based mod-

els, such as seasonal SVM, least-square SVM [29], online-SVM [30], and wavelet-SVM [31] were proposed for traffic prediction, and they can solve the problem of the curse of dimensionality, overfitting, and local minima. The BN model [32], [33] takes account of the causal relationship between the random variables statistically and can address the problem of incomplete data based on the message passing mechanism.

All the aforementioned work requires significant prior domain knowledge and feature engineering to achieve better performance. ANN model in [34] have the advantage over the previous algorithms due to its capability to work with multi-dimensional data without any feature engineering, and also due to its potential to perceive the non-linear relationship between input and output features to provide generalized solutions. However, due to its shallow depth in architecture, the accuracy of the model was not satisfactory, so the researchers shift their research directed toward the deep machine learning architecture. Long short-term memory (LSTM), a special recurrent neural network (RNN), can learn the temporal relations from the time-series sequence because of its build-in memory cells. It has shown remarkable results for traffic speed [35], traffic flow [36], [37] and congestion prediction [38], [39]. A deep autoencoder neural network [10] uses the temporal relationship between the input image sequences to predict short-term traffic congestion. All these attempts only consider the temporal relationship between the image sequences.

To make use of both spatial and temporal features of traffic data, the researchers started to build a hybrid model, i.e., by combining two or more independent models as a one. In [40], the KNN-LSTM model mines both spatial features by selecting the most related neighbor and temporal variability to predict the flow. In [41] Autoencoder-LSTM were combined, the internal relationship of the traffic flow was obtained by Autoencoder, and the LSTM network predicts complex linear traffic flow. In [8], traffic data was converted into a two-dimensional time-space matrix (spatiotemporal traffic data) and use a convolutional neural network model to predict large-scale network-wide traffic speed with high accuracy. In [42], the researcher proposed a novel model called the LC-RNN to predict road traffic speed, which consists of a look-up convolutional layer and recurrent layers. Look-up operation selects all the adjacent road, convolutional operation extracts the spatial correlations, and recurrent layers learn the long-term temporal patterns. The researcher in [43] used the convLSTM model, which is just like the LSTM model, where internal matrix multiplications are exchanged with convolutional operations. Convolutional operation extracts the spatial information, and LSTM learns the temporal information of traffic flow. Furthermore, in [44], the researcher proposed a deep learning model called SCRN, which is a combination of CNN and LSTM. At first, CNN extracts the spatial features of the traffic network for all the periods, and then the LSTM network learns the time-series temporal relation to predict traffic speed of 278 road links.

The hybrid neural network achieves better results than simple neural networks and traditional methods due to its ability in mining both spatial and temporal features from the traffic data. Although the hybrid model shows encouraging results in the domain of traffic prediction, there are very few works in the discipline of traffic congestion prediction based on deep neural networks, mainly due to the unavailability of high-quality city-wide congestion data. Some recent works in the congestion prediction are, as mentioned. In [10], the researcher trained the Autoencoder model with unnaturally compressed snapshots of traffic images from the open-source website to predict traffic congestion. The predicted images are not visually intuitive as a lot of road information is lost during image compression. In [38], the researchers collected road-based congestion information for few roads from an online source to learn and predict the traffic congestion on those roads using the LSTM network. In [39], the researchers used bus driving time data during peak periods to train the LSTM network to predict the traffic congestion time on six road segments. In [45], the researcher proposed a novel model called PCNN, which uses vehicle passage records from surveillance cameras on roads, to model periodic traffic congestion patterns to predict the short-term traffic congestion. In [46], the researchers used the congestion information from GPS data from each link to learn and predict traffic congestion evolution. In [47], the researchers used machine learning techniques (logistic regression, random forest, and neural networks) on vehicle trajectories data available through connected vehicle technology to identify and predict congestion formation. This study has a prediction horizons of 10 and 20 seconds, intended for warning drivers of upcoming traffic conditions. All these congestion studies predict congestion on a single road or a few major roads in the city or lack in predicting traffic image in fine granularity as in [10]. So, in this paper, we used a city-wide transportation Image data from TOPIS website to predict city-wide short-long term (prediction horizons of 10, 30, and 60 minutes) traffic congestion with fine granularity based on a hybrid structure containing CNN, LSTM, and Transposed CNN.

III. METHODOLOGY

In this section, we first define the problem statement for time-series traffic congestion prediction, then discuss on data acquisition from an open-source online platform, and finally, describe the component of the proposed Prediction Network (PredNet) architecture.

A. PROBLEM STATEMENT

Based on past observation of traffic congestion data, the proposed deep neural network is designed to predict the short-term congestion level. Let's consider $N \in \{1, 2, \ldots, n\}$ be the chronological order of n image data in our database. The i^{th} past observation of the time series data for the period of t is given by, $X_i = \{x_{i+0}, x_{i+1}, x_{i+2} \ldots, x_{i+t}\} \in \mathbb{R}^{t+1}$. The primary object of this study is to develop the prediction model f, which uses previous observation X_i to predict the

congestion level at time t, for k prediction horizon, i.e., $Y_i = x_{i+t+k}$. The model f can be defined, as in (1)

$$Y_i = f\,(X_i, \theta) \tag{1}$$

here, θ are the model parameter of our model. We divided our database N into 3 parts train, validation, and test. We generate m set historical time-series data from the train dataset, $X_{test} = \{X_0, X_1, X_2, \ldots, X_m\}$, such that the corresponding forecast data point associating X_{test} given by, $Y_{test} = \{x_t, x_{t+1}, x_{t+2} \ldots, x_{t+m}\}$ also lies in the training dataset. Hence, we can use supervised learning to train our neural network.

B. DATABASE

Google Traffic Map and Bing Map provided traffic information for almost all the cities in the world (except for few countries like China and South Korea for Google Traffic Map). Baidu Map provides information for all the cities in China, Naver Map provides traffic information for all the cities in South Korea, TOPIS focused on traffic conditions of Seoul, South Korea, and Sigalert serve for the bay area in California, USA. For our research, we captured the congestion map from TOPIS online web service, which provides the city-wide accurate real-time congestion level of the road network. Figure 1(a) shows the raw image of central Seoul (South Korea) on 20th September 2019 at 15:05, which consists of road networks along with background and text. TOPIS traffic image has three congestion levels, namely: Jam, Slow, and Free State. Here, color 'Red' represents the Jam, 'Yellow' represents the Slow, and 'Green' represents the free congestion level. The congestion level (C.L.) is categorized based on the average speed of the road, given in (2).

$$C.L. = \begin{cases} Free & if\ v > 25km/hr \\ Slow & if\ 10km/hr \leq v \leq 25km/hr \\ Jam & if\ v < 10km/hr \end{cases} \tag{2}$$

Since the digital image is a collection of the pixels which can be represented as a matrix (2D for grayscale and 3D for color image), extracting road network with congestion information can be easily performed by the mathematical operation. As congestion level in the image data has unique color composition, with upper and lower boundary value as ([0, 28, 160], [170, 117, 250]), ([0, 150, 195], [100, 220, 255]), and ([20, 160, 50], [110, 240, 120]) for Red, Yellow and Green color respectively in BGR format. The image masking operation is performed to extract the congestion level from the image. At first, the mask is generated for only road network by comparing the image data with the boundary value of each congestion color, and then 'bitwise and' operation is performed between mask image and raw image to generate the image with the only road network. The resulting image is shown in Figure 1(b) has red, yellow, green color representing the congestion level, and black color is the background. Matrix form representation of the image is shown in (3); each element has three values representing the

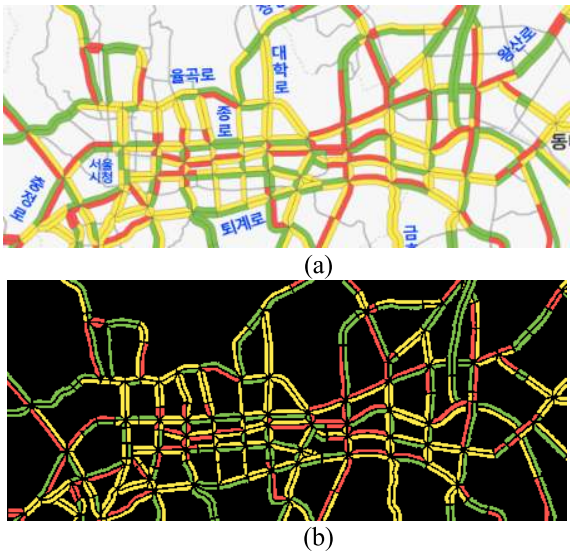

FIGURE 1. Road network of Seoul city with the Jam, Slow, and Free congestion level denoted in red, yellow, and green color, respectively. (a) A sample of raw image data capture from the TOPIS website. (b) A sample of the image after masking operation where black color represents the background.

congestion level in [B, G, R] format.

$$x_t = \begin{bmatrix} c^t_{1,1} & c^t_{1,2} & \cdots & c^t_{1,C} \\ c^t_{2,1} & c^t_{2,2} & \cdots & c^t_{2,C} \\ \vdots & \vdots & \cdots & \vdots \\ c^t_{R,1} & c^t_{R,2} & \cdots & c^t_{R,C} \end{bmatrix} \tag{3}$$

C. MODEL ARCHITECTURE

In this section, we discuss the architecture for predicting short-term city-wide traffic congestion using the sequences of the historical Image dataset. A schematic of our framework is shown in Figure 2. Our proposed model is the combination of 3 networks, namely: 1) Feature Extraction Network (FEN), which perform convolutional and pooling operation and convert the image into a lower-dimensional feature space, 2) the Recurrent Network, which consists of stacked LSTM layers responsible for learning time-series information on data from the previous layer, and 3) Reconstruction Network, which performs convolutional and transposes convolution operation on data from recurrent layer to produce predicted image. The architecture is designed to learn the spatial-temporal relationship of traffic congestion levels among roads in the transportation network.

As stated in section 3.1, our primary objective is to predict the congestion level of the transportation network based on the historical time-series sequence of the image, which can be solely performed by the LSTM model. The need for the CNN model and Transpose Convolutional arises because of 2 reasons: i) LSTM is a recurrent neural network, compute slowly for the large input parameter, and ii) LSTM only consider temporal features. In the following sections, we discuss each network in detail.

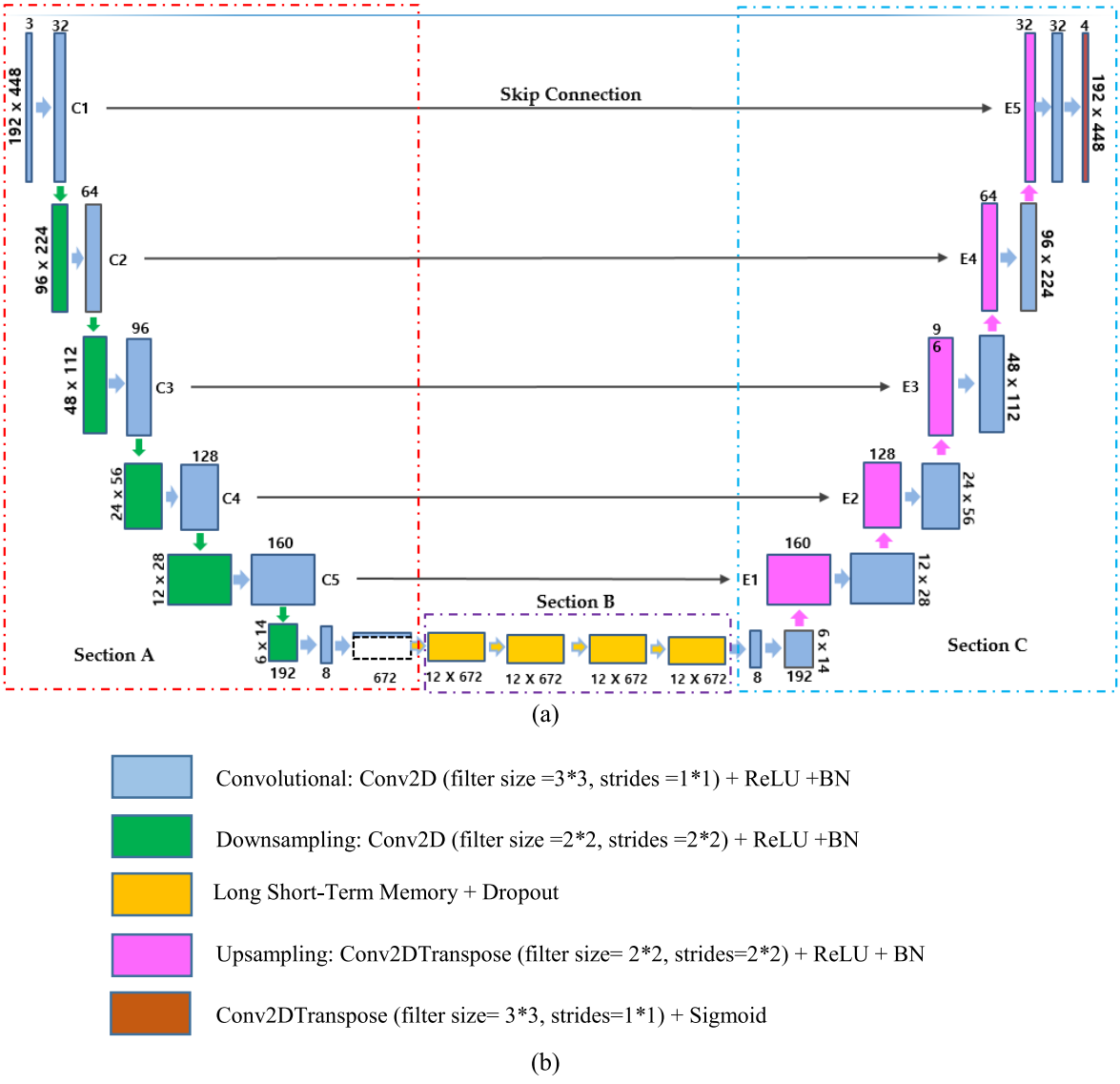

FIGURE 2. PredNet model architecture. (a) The model is divided into three sections, section A is Feature Extraction Network, section B is Recurrent Network, and section C is Reconstruction Network, the arrow from section A to section C represent the skip connection and number along the block represent the number of filters and shape of the matrix. (b) The color code represents the type of operation.

1) FEATURE EXTRACTION NETWORK

Convolutional Neural Network is a special type of Deep Neural Network (DNN) that is inspired by Hubel and Wiesel's work in neuroscience [48]. Since it first proposed in the work of handwritten zip code recognition [49], numerous variations of CNN architectures have been proposed. However, the unique aspects of CNN are the same, i.e., local connectivity and weight sharing. CNN has the superior ability in feature representation of an input image as compared to other deep learning architectures like auto-encoder and multilayer perceptron, as it handles the spatial correlations between the nearby pixels.

CNN consists of two layers, namely convolutional and pooling layer. The main objective of the convolutional layer is to learn the feature representation of the input image. It is a locally connected neuron, i.e., each neuron of output layers only receives input from a small local group of the neuron from the previous layer. It is composed of several convolution kernels which convolute with the image or previous layer to learn different feature representation. Mathematically, the f^{th} feature map of l^{th} convolutional layer y_f^l is can be obtained by first convoluting input image or previous layer output with the convolutional filter and then applying bit-wise non-linear activation, as in (4)

$$y_f^l = \sigma\left(\sum_{k=1}^{f_{l-1}} y_k^{l-1}, W_{kf}^l + b_f^l\right), \quad f \in [1, f_l] \tag{4}$$

where y_k^{l-1} is the k^{th} feature map of $(l-1)^{th}$ layer, W_{kf}^l is the kernel weight at position k connected to the f^{th} feature map of

l^{th} layer, b_f^l is the bias of f^{th} filter of l^{th} layer, f_l is the number of filters in l^{th} layer and σ represent element-wise non-linear activation function.

The function of the pooling layer is to achieve shift-invariance by progressively reduce the spatial size of the feature map but retains essential information. Pooling layers decrease the complexity in the network leading to faster convergence rate. Each pooled feature map corresponds to the feature map from the previous layer. The common pooling operations are max-pooling [50] and subsampling [51]. These pooling operations doesn't have any trainable parameter and computes one value from (m, n) rectangular region of the feature map, which decreases the resolution by the factor of m and n along each direction. Max-pooling selects the superior invariant features from the patch, and subsampling takes the average over the patch and passes through the non-linearity. In our experiment, we replace the traditional pooling operation by standard convolution operation with the stride of 2∗2. This approach is also mention in [52], the strided convolution increase the model expressiveness ability as it has a learnable parameter while reducing the feature map resolution.

The Feature Extraction Network is shown in Figure 2, section A. This architecture is formed by stacking convolutional and pooling layer and have one flatten layer at the end. The convolution operation is performed by the kernel of 3∗3 over the input image of 192∗448∗3 using unit stride and zero padding (to conserve the dimension of input). The (i, j) location of f^{th} feature map of l^{th} convolutional layer, $y_f^l(i, j)$ can be obtained by first convoluting previous $l-1^{th}$ layer output with the convolution filter of size (m, m) and then applying bit-wise non-linear activation, it is the detailed version of Equation 4, and is given as in (5)

$$y_f^l(i,j) = \sigma\left(\sum_{k=1}^{f_{l-1}}\sum_{a=0}^{m-1}\sum_{b=0}^{m-1}\left(W_{kf}^l(a,b)\otimes y_k^{l-1}(i+a, j+b)+b_f^l\right)\right), \quad f \in [1, f_l] \tag{5}$$

where (a, b) is kernel location.

The convolution layer is followed by the pooling layer, the (i, j) location of f^{th} feature map of $l+1^{th}$ pooling layer, $y_f^{l+1}(i, j)$ can be obtained by first convoluting previous lth layer output with the convolution filter of size (2, 2) and then applying bit-wise non-linear activation, and is given in (6)

$$y_f^{l+1}(i,j) = \sigma\left(\sum_{k=1}^{f_l}\sum_{a=0}^{m-1}\sum_{b=0}^{m-1}\left(W_{kf}^{l+1}(a,b)\otimes y_k^l(2i+a, 2j+b)+b_f^{l+1}\right)\right), \quad f \in [1, f_{l+1}] \tag{6}$$

where y_k^l is the location of k^{th} feature map of location of l^{th} layer, W_{kf}^{l+1} is the kernel weight at position k connected to the location of f^{th} feature map of $l+1^{th}$ layer, b_f^{l+1} is the bias of f^{th} filter of $l+1^{th}$ layer, f_{l+1} is the number of filter in l^{th} layer and σ represent element-wise non-linear activation function.

The output of the feature extracted network is connected to the recurrent network, which is stacked LSTM layers, which only take vector input. Hence, the output of CNN architecture is converted in vector by the flattening layer. Let L be the previous layer before flattening layer, having f_L number of feature maps of resolution (x, y), then the output of $L+1$ layer, y^{L+1} is given, as in (7)

$$y^{L+1} = flatten\left(\left[y_1^L, y_2^L, \ldots, y_{f_L}^L\right]\right) \tag{7}$$

where $y_1^L, y_2^L, \ldots, y_{f_L}^L$ are the feature maps of L^{th} layers, each feature maps contains $x * y$ elements, Hence, (7) can be rewritten as given in (8)

$$y^{L+1} = \left(\left[y_1^1, y_2^1, \ldots, y_e^1, y_1^2, y_2^2, \ldots, y_e^2, y_1^3, \ldots, y_1^{f_L}, y_2^{f_L}, \ldots, y_e^{f_L}\right]\right), \quad e = x * y \tag{8}$$

here, e is the number of elements in each feature map. Equation 8 shows the vector representation of L^{th} layer. Besides, it represents the high-level feature extraction of the input image.

The proposed architecture is for traffic congestion prediction based on the sequence of the historical data, but FEN extracts high-level features for only one image at a time. Therefore, the entire FEN network has to be encapsulated in the time distribution layer, i.e., FEN works in the loop to extract the feature from all the time-series input images before going to the Recurrent Network. As stated in section 3.1, t number of the past image is used for prediction, then the output of the FEN network has t number of vector representations of the input images. In figure 2, section A, there is a dotted block with blue and white color, blue represents the vector of the first map, and white represents the feature vector of other input images in the sequence. Each input with the resolution of 192∗448∗3 is compressed to the vector of 672 elements.

2) RECURRENT NETWORK

Figure 2, section B, shows the recurrent network, which is a primary prediction module in the architecture, it is made up of four stack of LSTMs. LSTM network was first mention in [53], which solves the vanishing and exploding gradient problem seen while training conventional recurrent Neural Networks (RNNs) [54] with the gradient-based back-propagation through time technique.

An LSTM unit contains a cell state, the memory part of LSTM and three gates input gate, forget gate and output gate, to protect and control the cell state. LSTM unit undergoes multiple operations at each gate to compute the output of LSTM called hidden state. At time t hidden state (h^t) is computed by the following operation: At input gate, new information is added to cell state which is completed in two-part, first, a sigmoid layer decide which input value is to be updated (i^t) given, as in (9), and then tanh layer creates a

vector of new candidate values ($\tilde{c^t}$) given, as in (10)

$$i^t = \sigma\left(\sum_{u=1}^{U} W_{ui}x_u^t + \sum_{v=1}^{V} W_{vi}h_v^{t-1} + b_i\right) \quad (9)$$

$$\tilde{c^t} = \tanh\left(\sum_{u=1}^{U} W_{uc}x_u^t + \sum_{v=1}^{V} W_{vc}h_v^{t-1} + b_c\right) \quad (10)$$

At forget gate, LSTM decides what information to forget from the cell state, computed, as in (11). Based on the update at input gate and forget gate, the old cell state at $t-1$, (c^{t-1}), is updated toc^t as in (12)

$$f^t = \sigma\left(\sum_{u=1}^{I} W_{uf}x_u^t + \sum_{k=1}^{H} W_{vf}h_v^{t-1} + b_f\right) \quad (11)$$

$$c^t = f^t * c^{t-1} + i^t * \tilde{c^t} \quad (12)$$

At output gate, the LSTM decide what parts of the cell state go to output, it is given, as in (13)

$$o^t = \sigma\left(\sum_{u=1}^{U} W_{uo}x_u^t + \sum_{v=1}^{V} W_{vo}h_v^{t-1} + b_o\right) \quad (13)$$

The final output of the LSTM unit, h^t is the function of cell state and the output gate, it is computed, as in (14)

$$h^t = o^t \tanh(c^t) \quad (14)$$

where $\sigma(z)$ and tanh(z) are sigmoid activation function and hyperbolic tangent activation function are defined as follows:

$$\sigma(z) = \frac{1}{1+e^{-Z}} \quad (15)$$

$$\tanh(z) = \frac{e^{Z} - e^{-Z}}{e^{Z} + e^{-Z}} \quad (16)$$

Here, x^t is the input at time t. W_i, W_f, and W_o represent the weight matrices of the input gate, forget gate, and output gate.

3) RECONSTRUCTION NETWORK

The output from the recurrent layer is passed through the reconstruction layer, where the compressed representation of spatiotemporal learned data is enlarged into the original resolution of the input data. The reconstruction network consists of series of convolutional and partially convolutional operation, is shown in Figure 2(a), section C. During transposed convolutional operations, we set stride and kernel size same to prevent artifacts such as checkboard patterns at final layer, due to overlap in the kernels. To further support the better decoding, the skip connection from FEN is connected, shown in Figure 2(a). Mathematically, reconstruction is performed similarly to as presented in the feature extraction layer. The detailed implementation of the network is explained in the next section.

IV. EXPERIMENT AND RESULT ANALYSIS

A. DATA SOURCE

In this research, we choose Seoul (the capital city of) South Korea and among the largest city in the world, where the traffic congestion is very high, especially in the central region, as shown in Figure 1(a). The figure is an example of a raw snapshot of a road network of central Seoul from the TOPIS website. Each snapshot is 192 × 448 pixels in size, covering about an area of 7.5km × 17km (scale 1 cm = 1.3km). These online web services use multiple sources of data collections such as inductive loop, crowdsourcing, etc. to provide accurate real-time data for the entire city.

In this paper, we are focused on traffic congestion level prediction from 07:00 to 12:00 on weekdays. We collect the snapshots of traffic data from 19th September 2019 to 31st December 2019, a total of 104 days, at an interval of 5 minutes (60 samples per day). Out of 104 days, there is partial or no data collection for 26 days, so we remove all the missing day's data to generate the database. Samples from 19th September to 25th November are used as the training set, samples from 25th November to 30th November are used for the validation set, and samples from 1st December to 31st December are used for the prediction of the trained model. Data preprocessing and database generation is explained in section III-B.

B. PERFORMANCE COMPARISON AND METRICS

In order to verify the effectiveness and superiority of the proposed architecture, PredNet, two state-of-art deep learning neural networks, namely: ConvLSTM [43], and Autoencoder [10] are selected for comparison. ConvLSTM is just like LSTM, but internal matrix multiplication is exchanged with convolution operations, which can mine both spatial and temporal information from the input image sequence. The deep Autoencoder is a neural network having an encoding layer, which aims to learn a representation (encoding) from the set of data and decoding layer, which tries and learns to generate from the reduced encoding as close as possible to its original input. We performed the traffic congestion prediction for three-time horizons (10, 30, and 60 minutes) for comparison and analysis of proposed PredNet. As the model predicts the image with congestion levels information represented by a color on each road, pixel-wise classification based on categorical entropy loss function is a more desirable parameter instead of evaluating the model based on mean square error or mean absolute error as in literature [10] and [43]. In this paper, we present the performance result based on precision, recall, and accuracy of the model for traffic prediction. Equation 17, 18, and 19 define precision, recall, and accuracy, respectively, and categorical cross-entropy loss function, as in (20).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (17)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (18)$$

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total} \tag{19}$$

$$L\left(y, y'\right) = -\sum_{j=0}^{M}\sum_{i=0}^{N}\left(y_{ij} * log\left(y'_{ij}\right)\right) \tag{20}$$

Here, y is true value, y' is predicted value, (i, j) is the row and column in the image.

C. IMPLEMENTATION OF PROPOSED MODEL

As mention in section III-A, the objective of the model is to take the chronological sequence of traffic image to predict the congestion level of transportation network at different prediction horizons (10, 30, and 60 minutes). To achieve the objective, the deep neural network, as explained in section III-C, is used. As the network is very deep, i.e., a large number of hidden layers, this might bring the problem of vanishing gradient. Because with the increase in the number of hidden layers, the gradient shrinks towards zero during backpropagation, which results in the weight never updating its value. The solution to vanishing gradient is, skip connection -a connection from initial layers of the network to later layers- which enables the gradient to flow directly through the skip connections backward from later layers to initial layers. Apart from vanishing gradient, the other parameters to consider while implementing the model are the number of past images for input, the number of LSTM layers in between convolutional encoding and decoding, consecutive filter number for both convolutional encoder and decoder, and hyper-parameters.

At first, we evaluate our prediction model for vanishing gradient problem by monitoring change in performance under different combinations of the skip connections from the feature extraction network (Figure 2(a) Section A) to the reconstruction network (Figure 2(a) Section C). We select five upsampling layers in the reconstruction network, namely: E1, E2, E3, E4, and E5, as shown in Figure 2(a), where skip connection could be beneficial as it carries the feature maps with much image details, which would help upsampling layer to recover a clean version of the image. We select five layers, presented just before the downsampling layer in the feature extraction network, namely: C1, C2, C3, C4, and C5, as shown in Figure 2(a), for skip connection to reconstruction network. There are 5 cases of skip connection C1 to E5, C2 to E4, C3 to E3, C4 to E2, and C5 to E1 and a large number of possible architecture based on a combination of skip connections. We investigate 6 model scenarios to find the best architecture.

Table 1 shows the prediction performance (for 10 min horizon) for the training and validation dataset. As seen from the table, the performance of configuration 7 (without any skip connection) is boosted significantly with the addition of skip connections. With the addition of one skip connection, Configuration No. (C.N) 5 and 6, the training and validation mean square error is decreased by 40% and 23% respectively; the training and validation pixel to pixel accuracy is improved by 1.85% and 1.60% respectively, and the training and validation categorical cross-entropy loss is decreased by 30% and 22% respectively. Further investigation shows the skip connection between the early and late layers (C1 to E5) in C.N 5 achieves higher performance gain than skip connection between intermediate layers (C5 to E1) in C.N 6, which suggests the model is suffering from vanishing gradient problem. With the addition of more skip connections, the model continues to perform better, and the best result is achieved with C.N 1, which have skip connection to all the upsampling layer in reconstruction network from feature extraction network, as shown in Figure 2(a). Although the training accuracy in C.N 1 is lower than C.N 2, the C.N 1 has better representation capability as its validation accuracy, MSE, and loss is best among all the configuration. All the other model parameters, such as the number of filters, sequence of input images, and hyper-parameters, are kept same for testing all the configuration mention in Table 1.

Choosing the optimal number of historical images as an input to the model is very crucial. Selecting a large number of images than optimal will consume unwanted computing resources, and it will be difficult to train on resource-constrained devices. Whereas, choosing a small sequence input image will hinder the performance of the model, as there wouldn't be enough information to exploit the time-series relationship among the data. Since the time interval between data is 5 minutes, we experiment with 13, 12, 11, and 9 historical data (i.e., 65, 60, 55, and 45 minutes respectively) to train and predict the traffic congestion.

As seen from Table 2, C.N 2, with 11 input samples, achieves exceptional performance for the training dataset but fails to be superior on the validation data set. C.N 3 produces better MSE and accuracy results than other configurations and similar loss as C.N 4. Hence, we choose a C.N 3 with 12 image samples as input for implementing our prediction model as it has better generalization capability compared to other configurations, and with less computing time compared to C.N 3 with 13 input samples.

Furthermore, the number of filters in the convolution neural networks plays a vital role in model performance. As each filter extract different feature map from the same layer, increasing the number of filters insure more learning but increase beyond some optimal filter number does not affect the performance gain but increase the resource consumption. Hence, we experimented on three configurations of the number of filters for the convolutional layer to find a proper one. The feature extraction network and reconstruction network mention in Figure 2(a) are symmetric in terms of the number of filters. From Table 3, we can see that C.N 3 has better MSE and accuracy, and C.N 1 has a better loss in the training dataset. However, for the validation dataset, C.N 1 outperforms all other configurations, which suggests that C.N 1 has better generalization capability. Other parameters like skip connection, no. of the input image, and hyper-parameters were same for generating results for all three configurations.

The details of the PredNet is explained in this subsection. As shown in Figure 2(a), the proposed model has 30 layers

TABLE 1. Comparison of prediction metrics (10 min horizon) for training and validation dataset for different configurations of skip connections. The best result is marked in bold.

C.N	Skip Connection Configuration	Training			Validation		
		MSE	Accuracy	Loss	MSE	Accuracy	Loss
1	All	**0.0130**	0.9622	**0.0932**	**0.0144**	**0.9610**	**0.1014**
2	C1-E5, C3-E3, C5-E1	0.0136	**0.9632**	0.0953	0.0148	0.9601	0.1054
3	C1-E5, C3-E3	0.0138	0.9621	0.0967	0.0145	0.9603	0.1017
4	C3-E3, C5-E1	0.0140	0.9618	0.1005	0.0146	0.9604	0.1034
5	C1-E5	0.0142	0.9611	0.1007	0.0147	0.9599	0.1025
6	C5-E1	0.0152	0.9585	0.1109	0.0149	0.9598	0.1075
7	No connection	0.0250	0.9410	0.1410	0.0194	0.9446	0.1322

TABLE 2. Comparison of prediction metrics (10 min horizon) for training and validation dataset for the different numbers of the historical image as input data. The best result is marked in bold.

C.N	Sequence of Input Image	Training			Validation		
		MSE	Accuracy	Loss	MSE	Accuracy	Loss
1	9	0.0131	0.9637	0.0916	0.0147	0.9577	0.1037
2	11	**0.0128**	**0.9640**	**0.0909**	0.0149	0.9594	0.1056
3	12	0.0130	0.9628	0.0932	**0.0144**	**0.9610**	0.1014
4	13	0.0129	0.9624	0.0927	0.0146	0.9608	**0.1013**

TABLE 3. Comparison of prediction metrics (10 min horizon) for training and validation dataset for the different numbers of filter configuration for convolutional layers. The best result is marked in bold.

C.N	No. Filter Configuration	Training			Validation		
		MSE	Accuracy	Loss	MSE	Accuracy	Loss
1	32_64_96_128_160_192_8	0.0131	0.9622	**0.0932**	**0.0144**	**0.9610**	**0.1014**
2	16_32_64_96_128_160_8	0.0158	0.9575	0.1104	0.0154	0.9572	0.1065
3	16_32_64_128_256_512_8	**0.0129**	**0.9624**	0.0934	0.0150	0.9591	0.1016

consisting of 12 convolutional layers; 5 downsampling layers; 5 upsampling layers; 4 LSTM layers; and one of each flatten, reshape, input and output layers. The model input has four dimensions (12, 192, 448, 3), where the first number indicates sequences of 12 images are taken as input, second and third number indicates the row and column of image and fourth number represent the channel of the image. The input layer is followed by the convolutional layer with 32 convolutional filters of size (3 × 3), strides of (1 × 1), and padding 'same'. This layer is followed by a downsampling layer, performed by convolutional operation with strides of (2 × 2), and padding 'valid'. The combination of convolutional and downsampling layer encode the input (12, 192, 448, 3) to (12, 6, 14, 8). The flatten layer further converts to (12, 672) and feed to Recurrent Network, where LSTM learn the features by unfolding the times series and capturing the pattern. The output of LSTM (12, 672) is reshaped into (12, 6, 14, 8) by Reshape layers, which is followed by series of convolutional layers and upsampling layers -performed by transposed convolutional operation with filter size (2 × 2) and stride (2 × 2)-, to regenerate the encoded representation back to original image resolution (12, 192, 448, 4). All the layer have ReLU activation except the last layer which have softmax activation. All the convolutional, downsampling and upsampling layer have dropout of 0.1, and batch normalization, whereas LSTM layer, has 0.2 dropouts.

In our experiment, we implemented the ConvLSTM model with six layers having configuration [48, 36, 24, 24, 12, and 4], with the filter size of (3 × 3), strides of (1 × 1), and 'same' padding. Each layer except last has ReLU activation, 0.1 dropouts, and a batch normalization layer, whereas the last layer has softmax activation. The input to ConvLSTM is a sequence of 12 images with the resolution of (192 × 448 x3). For Autoencoder, we adopt the configuration [512, 384, 256, and 128] with ReLU activation of each layer, except softmax activation in the last layer. The input to Autoencoder is (12, 345), i.e., the congestion level of each road for 12 input samples. Besides, the loss function for both models is changed from MSE to categorical cross-entropy for a fair comparison.

All the models mentioned were trained on a real-world traffic congestion data of Seoul city (South Korea) using Adaptive Moment Estimation (Adam) optimizer with the learning rate of 1e-4, learning decay rate of 0.95, variables moving average decay of 0.999, and a batch size of 1. We train all the models using a categorical cross-entropy loss function. Also, all the model is implemented using Keras deep learning library on an Ubuntu 18.04.4 machine with 4 NVIDIA TITAN Xp Graphics Cards.

D. RESULT AND ANALYSIS

Table 4 presents the performance metrics of our proposed model PredNet along with ConvLSTM and Autoencoder in terms of precision, recall, and accuracy on a training dataset at different prediction horizons (10, 30, and 60 minutes). Here, instead of using the entire pixels of the image, we randomly choose a single pixel for each road (road-wise value) to evaluate the model. The road-wise prediction performance gives the true evaluation of the model as the performance calculation is not affected by background pixels and road length. The proposed model PredNet achieves around 2 to 12% performance gain, for all prediction horizons compared to ConvLSTM and Autoencoder.

Table 5 shows the road-wise per hour average prediction accuracy for eight working days from 3rd December 2019 to

TABLE 4. Performance comparison on training dataset for the prediction horizon of 10, 30, and 60 minutes. The best result is marked in bold.

Prediction	Precision			Recall			Accuracy		
Horizons	P.N	C.L	A.E	P.N	C.L	A.E	P.N	C.L	A.E
10 minutes	**0.8724**	0.8360	0.7661	**0.8564**	0.8116	0.7255	**0.8835**	0.8573	0.7722
30 minutes	**0.8487**	0.8210	0.7192	**0.8525**	0.8100	0.7120	**0.8595**	0.8392	0.7537
60 minutes	**0.8474**	0.7461	0.7298	**0.8455**	0.7309	0.7053	**0.8422**	0.7567	0.7389

TABLE 5. Road-wise accuracy comparison for the prediction horizon of 10, 30, and 60 minutes. The best result is marked in bold.

Date & Time	10 minutes			30 minutes			60 minutes		
	P.N	C.L	A.E	P.N	C.L	A.E	P.N	C.L	A.E
12-03 08:00	**0.8735**	0.8359	0.7709	**0.8541**	0.8296	0.7063	**0.8293**	0.7302	0.6978
12-03 09:00	**0.8726**	0.8236	0.7845	0.8294	**0.8357**	0.7292	**0.8293**	0.7118	0.7263
12-03 10:00	**0.8793**	0.8554	0.7936	**0.8500**	0.8286	0.7514	**0.8489**	0.7341	0.6954
12-03 11:00	**0.8784**	0.8490	0.7748	0.8347	**0.8361**	0.7164	**0.8375**	0.7447	0.6882
12-04 08:00	**0.8716**	0.8264	0.7605	**0.8340**	0.8250	0.7232	**0.8407**	0.7638	0.7041
12-04 09:00	**0.8743**	0.8417	0.7466	**0.8243**	0.8212	0.7338	**0.8400**	0.7425	0.6935
12-04 10:00	**0.8639**	0.8412	0.7699	**0.8437**	0.8296	0.7227	**0.8400**	0.7512	0.6908
12-04 11:00	**0.8716**	0.8427	0.7630	**0.8383**	0.8282	0.7152	**0.8317**	0.7408	0.7162
12-05 08:00	**0.8714**	0.8320	0.6998	0.8301	**0.8344**	0.7355	**0.8194**	0.7239	0.6908
12-05 09:00	0.8473	**0.8536**	0.7942	0.8262	**0.8265**	0.7329	**0.8358**	0.7348	0.6882
12-05 10:00	**0.8677**	0.8398	0.7884	**0.8420**	0.8217	0.7203	**0.8230**	0.7415	0.6949
12-05 11:00	**0.8645**	0.8349	0.7527	**0.8335**	0.8187	0.7237	**0.8354**	0.7215	0.7068
12-06 08:00	**0.8576**	0.8238	0.7976	**0.8262**	0.8105	0.7208	**0.8335**	0.7430	0.6995
12-06 09:00	**0.8662**	0.8373	0.7692	**0.8369**	0.8190	0.7338	**0.8402**	0.7075	0.6459
12-06 10:00	**0.8762**	0.8373	0.7104	**0.8371**	0.8311	0.7324	**0.8303**	0.7568	0.6560
12-06 11:00	**0.8788**	0.8419	0.6973	**0.8566**	0.8344	0.6886	**0.8358**	0.7447	0.7089
12-09 08:00	**0.8703**	0.8456	0.7213	**0.8226**	0.8190	0.6664	**0.8286**	0.7300	0.6989
12-09 09:00	**0.8628**	0.8354	0.7693	0.8313	**0.8335**	0.6635	**0.8337**	0.7275	0.7075
12-09 10:00	**0.8757**	0.8284	0.6964	**0.8386**	0.8361	0.7309	**0.8378**	0.7522	0.6838
12-09 11:00	**0.8885**	0.8335	0.8078	**0.8282**	0.8139	0.7157	**0.8390**	0.7290	0.6839
12-10 08:00	**0.8631**	0.8320	0.7830	**0.8347**	0.8185	0.7329	**0.8356**	0.7063	0.7065
12-10 09:00	**0.8521**	0.8300	0.7627	**0.8347**	0.8185	0.7329	**0.8457**	0.7389	0.7048
12-10 10:00	0.8587	**0.8698**	0.7764	**0.8323**	0.8127	0.7200	**0.8346**	0.7135	0.6986
12-10 11:00	0.8535	**0.8608**	0.7473	0.8301	**0.8306**	0.7167	**0.8421**	0.7140	0.7188
12-11 08:00	**0.8616**	0.8260	0.7022	**0.8484**	0.8014	0.7278	**0.8371**	0.7539	0.6978
12-11 09:00	**0.8708**	0.8341	0.7546	0.8279	**0.8397**	0.7312	**0.8404**	0.7471	0.7162
12-11 10:00	**0.8711**	0.8494	0.7300	**0.8221**	0.8006	0.7442	**0.8346**	0.7331	0.7101
12-11 11:00	**0.8793**	0.8242	0.7827	**0.8303**	0.8282	0.7220	**0.8349**	0.7394	0.6989
12-12 08:00	**0.8676**	0.8441	0.8029	**0.8547**	0.8270	0.7396	**0.8385**	0.7065	0.7152
12-12 09:00	**0.8560**	0.8313	0.7768	0.8344	**0.8352**	0.7150	**0.8385**	0.7176	0.6915
12-12 10:00	**0.8588**	0.8455	0.7587	**0.8422**	0.8423	0.7478	**0.8363**	0.7256	0.6903
12-12 11:00	0.8522	**0.8675**	0.7700	0.8378	**0.8405**	0.7338	**0.8274**	0.7430	0.7060
Average	**0.8674**	0.8398	0.7599	**0.8359**	0.8259	0.7227	**0.8355**	0.7335	0.6979

* P.N= PredNet *C.L= ConvLSTM *A.E= Autoencoder

12th December 2019, in a period of 08:00 to 12:00, for all three congestion prediction models. The proposed network, PredNet, shows that the model accuracy performance is better for all prediction horizons, i.e., 10, 30, and 60 minutes. For 10 minutes horizons, the average accuracy for PredNet is in the range of lowest being 0.8473 to highest being 0.8793. Out of 32 hours of prediction, PredNet achieves the best accuracy for 28 hours, and ConvLSTM achieves best for 4 hours. Similarly, for 30 minutes prediction horizon, PredNet achieves the best accuracy for 23 hours, and ConvLSTM achieves the highest value for the other 9 hours. Whereas, for 60 minutes prediction horizon, the PredNet outperforms all other models by delivering the best result for all periods. The highest accuracy for 30 and 60 minutes prediction for PredNet is 0.8566 and 0.8489, respectively.

Figure 3 shows the detailed representation of prediction accuracy on 3rd December 2019, from 08:00 to 12:00, for prediction horizons of 10 and 30 minutes, respectively. It shows the prediction accuracy for every 5 minutes. Figure 3(a) shows that the PredNet achieves the highest accuracy value for 46 times out of 48 samples, the highest being 0.9325 at 10:00. Similarly, the PredNet makes high accuracy for 42 times out of 48 samples for 30 minutes prediction, as shown in Figure 3(b). The accuracy of ConvLSTM is slightly lower than our proposed PredNet in most of the instance. However, the prediction accuracy for Autoencoder based on architecture from the literature [10] is inferior. Both PredNet and ConvLSTM perform well for traffic congestion prediction as they use spatial and temporal information. In contrast, Autoencoder uses only temporal data for forecasting; this could be one of the reasons for its poor performance.

From Table 5 and Figure 3, we can say the proposed PredNet performs better than the other two state-of-art prediction models in all prediction horizons. Even though the models

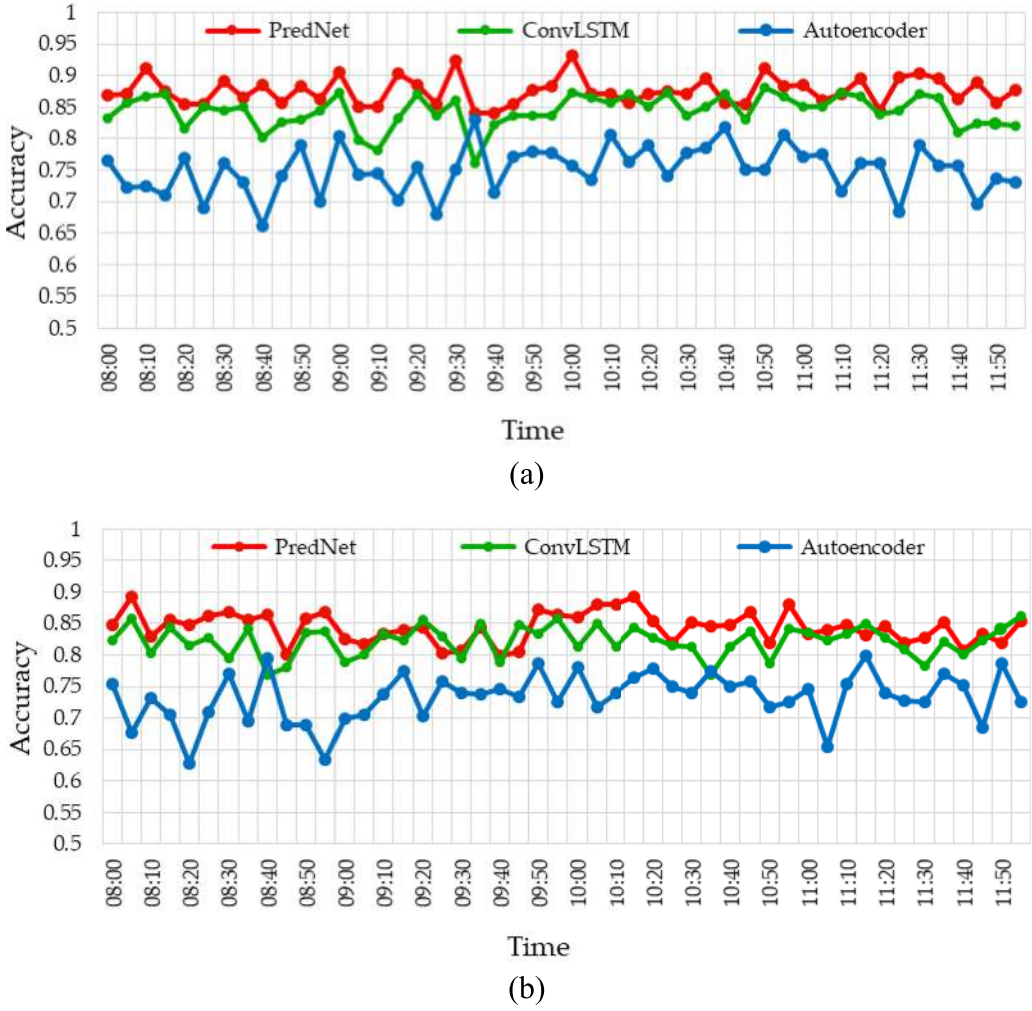

FIGURE 3. Road-wise accuracy for traffic congestion prediction on 3rd December 2019. (a) Prediction accuracy for prediction horizon of 10 minutes. (b) Prediction accuracy for prediction horizon of 30 minutes.

attain high performance in terms of accuracy, there is no guarantee that the trained model has better representation capacity for predicting the different congestion levels. There might be a case that the model predicts some congestion level accurately and have a poor prediction for others. In Table 6, we present Precision and Recall metrics for Jam, Slow, and Free congestion levels for all three models, for all three prediction horizon of 10, 30, and 60 minutes. This metrics shows the exactness and sensitivity of models in learning and representing congestion levels.

Table 6 shows the comparison on 3rd December 2019 from 09:00 to 10:00 at every 5 minutes. The proposed, PredNet, shows prediction ability with high precision for all the congestion levels in all three prediction horizons. For 10 minutes, prediction horizons, the PredNet achieves precision ranges from 77% to 94% with an average value of 86%, which is 10% and 12% more than ConvLSTM and Autoencoder for Jam congestion levels. For the Slow congestion level, the PredNet attains an average precision of 86.7%, which is 3% and 14% higher compared to ConvLSTM and Autoencoder, respectively. Furthermore, for the Free congestion level, PredNet reaches an average precision of 82.9%, which is 0.6% and 10% higher than ConvLSTM and Autoencoder, respectively. Similarly, the PredNet achieves a precision value of 80.8%, 86.9%, and 80.7% for 30 minutes prediction and 82.4%, 85.3%, and 80.5% for 60 minutes prediction, for Jam, Slow, and Free congestion levels, which is higher than other two models. As shown in Table 6, the PredNet dominates other models in terms of average recall value for predicting all three congestion levels at all prediction horizons. The highest recall value achieved by the PredNet is 0.876, 0.919, and 0.926 for 10 minutes horizons, 0.874, 0.944, and 0.918 for 30 minutes horizons, and 0.839, 0.903, and 0.904 for 60 minutes horizons, for Jam, Slow, and Free congestion levels. In terms of recall, for Jam congestion level, PredNet achieves performance gain of around 4-10% compared to ConvLSTM and 10-16% compared to Autoencoder, for predicting Jam congestion levels, for prediction horizons of 10, 30, and

TABLE 6. Precision and Recall metrics of all the models at different prediction horizons of 10 and 30 minutes. Best performance values are marked in bold.

Date & Time	Precision									Recall								
	Jam			Slow			Free			Jam			Slow			Free		
	P.N	C.L	A.E	P.N	C.L	A.E	P.N	C.L	A.E	P.N	C.L	A.E	P.N	C.L	A.E	P.N	C.L	A.E
10 minutes																		
12-03 09:00	**0.823**	0.783	0.770	**0.910**	0.868	0.676	**0.840**	0.838	0.868	**0.815**	0.759	0.746	**0.896**	0.886	0.892	**0.869**	0.855	0.753
12-03 09:05	**0.870**	0.594	0.708	**0.825**	0.767	0.715	0.838	**0.843**	0.704	**0.722**	0.617	0.652	**0.862**	0.861	0.677	**0.821**	0.774	0.650
12-03 09:10	**0.807**	0.695	0.820	**0.899**	0.767	0.732	**0.795**	0.773	0.762	**0.765**	0.489	0.610	0.787	**0.839**	0.620	**0.871**	0.767	0.784
12-03 09:15	**0.773**	0.664	0.707	0.909	**0.916**	0.877	**0.841**	0.826	0.480	0.750	**0.798**	0.637	**0.852**	0.819	0.590	**0.870**	0.84	0.561
12-03 09:20	**0.882**	0.816	0.620	0.821	**0.840**	0.802	0.861	**0.871**	0.652	0.833	**0.851**	0.756	0.886	**0.894**	0.572	**0.834**	0.833	0.723
12-03 09:25	**0.817**	0.786	0.705	0.844	**0.879**	0.931	0.819	**0.835**	0.428	**0.876**	0.795	0.684	**0.860**	0.820	0.511	**0.822**	0.846	0.676
12-03 09:30	**0.946**	0.863	0.759	**0.920**	0.880	0.643	**0.859**	0.837	0.805	**0.814**	0.762	0.720	**0.882**	0.881	0.812	**0.926**	0.873	0.646
12-03 09:35	**0.827**	0.733	0.820	**0.880**	0.759	0.722	0.768	0.735	**0.840**	**0.574**	0.409	0.684	0.801	**0.835**	0.818	**0.883**	0.763	0.862
12-03 09:40	**0.822**	0.759	0.645	**0.883**	0.874	0.623	0.827	**0.865**	0.743	**0.862**	0.783	0.707	**0.896**	0.892	0.725	0.840	**0.843**	0.685
12-03 09:45	**0.929**	0.837	0.820	0.740	**0.785**	0.654	**0.811**	0.785	0.803	0.780	**0.845**	0.646	**0.838**	0.811	0.785	**0.815**	0.802	0.734
12-03 09:50	**0.887**	0.834	0.753	**0.886**	0.819	0.692	**0.837**	0.832	0.807	0.778	**0.805**	0.623	**0.899**	0.859	0.812	**0.834**	0.825	0.759
12-03 09:55	**0.938**	0.780	0.838	**0.882**	0.859	0.673	**0.850**	0.834	0.787	**0.857**	0.810	0.690	**0.919**	0.867	0.801	**0.853**	0.824	0.785
Average	**0.860**	0.762	0.747	**0.867**	0.834	0.728	**0.829**	0.823	0.723	**0.786**	0.727	0.680	**0.865**	0.855	0.718	**0.853**	0.820	0.718
30 minutes																		
12-03 09:00	**0.891**	0.881	0.747	0.819	**0.855**	0.579	**0.818**	0.811	0.717	**0.786**	0.815	0.604	**0.885**	0.835	0.720	0.833	**0.836**	0.787
12-03 09:05	**0.854**	0.811	0.920	**0.913**	0.782	0.643	**0.787**	0.745	0.842	**0.686**	0.543	0.652	0.841	**0.890**	0.750	**0.904**	0.753	0.854
12-03 09:10	**0.776**	0.756	0.734	0.801	**0.825**	0.645	**0.766**	0.766	0.646	**0.815**	0.793	0.748	**0.762**	0.756	0.553	**0.793**	0.769	0.761
12-03 09:15	0.803	0.848	**0.877**	**0.911**	0.865	0.666	0.801	**0.803**	0.626	**0.851**	0.684	0.725	0.857	**0.874**	0.501	**0.898**	0.843	0.863
12-03 09:20	**0.864**	0.861	0.520	**0.828**	0.827	0.789	**0.836**	0.822	0.441	0.797	**0.800**	0.720	**0.913**	0.878	0.518	**0.918**	0.802	0.610
12-03 09:25	0.643	0.586	**0.734**	**0.902**	0.857	0.734	**0.816**	0.781	0.610	**0.808**	0.629	0.658	**0.792**	0.784	0.511	**0.848**	0.799	0.761
12-03 09:30	**0.802**	0.717	0.163	**0.863**	0.860	0.758	**0.804**	0.738	0.399	**0.854**	0.695	0.591	**0.864**	0.831	0.443	0.784	**0.811**	0.721
12-03 09:35	0.729	0.725	**0.757**	**0.913**	0.908	0.606	0.838	**0.849**	0.816	0.698	**0.722**	0.626	**0.944**	0.930	0.752	**0.836**	0.810	0.691
12-03 09:40	**0.751**	0.729	0.687	0.894	**0.909**	0.684	**0.770**	0.749	0.753	**0.794**	0.662	0.677	**0.839**	0.802	0.687	**0.832**	0.819	**0.931**
12-03 09:45	**0.835**	0.826	0.620	**0.891**	0.886	0.603	**0.854**	0.804	0.718	**0.874**	0.828	0.802	**0.854**	0.828	0.692	**0.853**	0.816	0.786
12-03 09:50	**0.886**	0.878	0.810	**0.886**	0.830	0.627	0.805	**0.831**	0.775	**0.828**	0.811	0.752	0.818	**0.859**	0.755	**0.817**	0.811	**0.819**
12-03 09:55	0.857	**0.873**	0.827	**0.810**	0.800	0.658	**0.789**	0.780	0.722	**0.729**	0.699	0.703	**0.854**	0.849	0.690	**0.900**	0.811	0.867
Average	**0.808**	0.791	0.700	**0.869**	0.850	0.666	**0.807**	0.790	0.672	**0.793**	0.723	0.688	**0.852**	0.843	0.631	**0.851**	0.804	0.788
60 minutes																		
12-03 09:00	**0.806**	0.671	0.678	0.890	0.657	0.661	**0.841**	0.792	0.753	**0.760**	0.660	0.648	**0.810**	0.807	0.745	**0.856**	0.675	0.692
12-03 09:05	**0.835**	0.717	0.778	0.857	0.695	0.660	**0.812**	0.794	0.577	**0.770**	0.652	0.628	**0.888**	0.859	0.743	**0.820**	0.675	0.762
12-03 09:10	0.772	0.660	**0.889**	0.853	0.804	0.677	**0.767**	0.667	0.838	**0.771**	0.745	0.643	**0.871**	0.763	0.791	**0.793**	0.742	0.731
12-03 09:15	**0.874**	0.698	0.700	0.808	0.762	0.690	**0.821**	0.717	0.677	**0.810**	0.747	0.637	**0.835**	0.596	0.603	**0.838**	0.807	0.771
12-03 09:20	**0.806**	0.705	0.667	0.839	0.776	0.806	0.752	**0.794**	0.447	**0.783**	0.678	0.525	**0.824**	0.586	0.469	**0.835**	0.781	0.634
12-03 09:25	0.865	0.686	0.600	0.779	**0.795**	0.770	**0.819**	0.648	0.644	**0.791**	0.630	0.614	**0.821**	0.764	0.564	**0.800**	0.787	0.704
12-03 09:30	**0.822**	0.733	0.400	**0.873**	0.658	0.884	**0.744**	0.598	0.432	**0.815**	0.681	0.739	**0.813**	0.737	0.497	**0.806**	0.767	0.667
12-03 09:35	**0.781**	0.607	0.778	**0.882**	0.850	0.670	**0.846**	0.725	0.778	**0.810**	0.655	0.628	**0.869**	0.732	0.748	**0.867**	0.828	0.800
12-03 09:40	**0.842**	0.729	0.750	**0.889**	0.754	0.687	**0.821**	0.699	0.807	**0.822**	0.787	0.611	**0.809**	0.675	0.752	**0.904**	0.782	0.868
12-03 09:45	**0.806**	0.797	0.636	**0.903**	0.606	0.594	**0.796**	0.627	0.748	**0.836**	0.737	0.721	0.849	**0.857**	0.679	**0.889**	0.743	0.744
12-03 09:50	0.879	0.728	**0.889**	0.810	**0.868**	0.625	**0.820**	0.716	0.772	**0.839**	0.738	0.643	**0.913**	0.707	0.709	**0.795**	0.793	0.745
12-03 09:55	**0.803**	0.703	0.714	**0.848**	0.687	0.641	**0.816**	0.642	0.750	**0.828**	0.687	0.738	**0.815**	0.793	0.789	0.770	0.634	**0.753**
Average	**0.824**	0.703	0.707	**0.853**	0.743	0.697	**0.805**	0.702	0.685	**0.803**	0.700	0.648	**0.843**	0.740	0.674	**0.831**	0.751	0.734

*P.N PredNet *C.L ConvLSTM *A.E Autoencoder

TABLE 7. Computing resources for training with prediction horizons of 10, 30, and 60 minutes.

	PredNet	ConvLSTM[43]	Autoencoder
10 minutes			
Input Parameter	(12, 192, 448, 3)		(12, 345)
Epochs	22	32	1735
Training time(s)	7920	60800	1680
30 minutes			
Epochs	25	37	1795
Training time(s)	8000	61050	1615
60 minutes			
Epochs	23	34	1755
Training time(s)	7130	56400	1579

60 minutes. For Slow congestion level, PredNet attains performance gain of around 1 % gain for prediction horizons of 10 and 30 minutes whereas 10% for 60 minutes prediction horizon compared to ConvLSTM and about 15-18% for all three prediction horizons compared to Autoencoder. Similarly, a significant gain in the recall value is attained compared to other models for predicting the Free congestion level. Our proposed model shows the consistent prediction for all three prediction horizons, ConvLSTM shows a reliable forecast for 10 and 30 minutes horizons but fails for 60 minutes prediction horizons.

Figure 4 shows the end-to-end result of PredNet, which shows the comparison of ground truth and its corresponding prediction congestion level on 3rd December 2019 with a prediction horizon of 10 minutes. In Figure 4, Column A denotes the ground truth image of every 5 minutes, and column B indicates the predicted image with its precision (P), recall (R), and accuracy (A). The PredNet predicts the congestion level of the city at excellent granularity, which is visually intuitive compared to work in [10].

The computational resource requirement for any deep neural network solely depends on the type of connection between the layers. Local connectivity and weight sharing nodes use fewer resources compared to fully connected nodes, between the layers. For the input dimension of (12, 192, 448, 3) and architecture mentioned in section IV-C ConvLSTM takes 0.307 million parameters, the proposed PredNet takes 16.5 million parameters, and Autoencoder takes 1,718.5 million parameters (575 Million for gray image). It shows that the PredNet is efficient than Autoencoder in terms of resource utilization but not compared to ConvLSTM. Besides, the PredNet model is more computational efficiency than the other two in terms of computing time, as shown in Table 7. We can see our proposed network takes less number of epochs and training time to converge compared to ConvLSTM. PredNet takes around 1.9 to 2.3 hours to

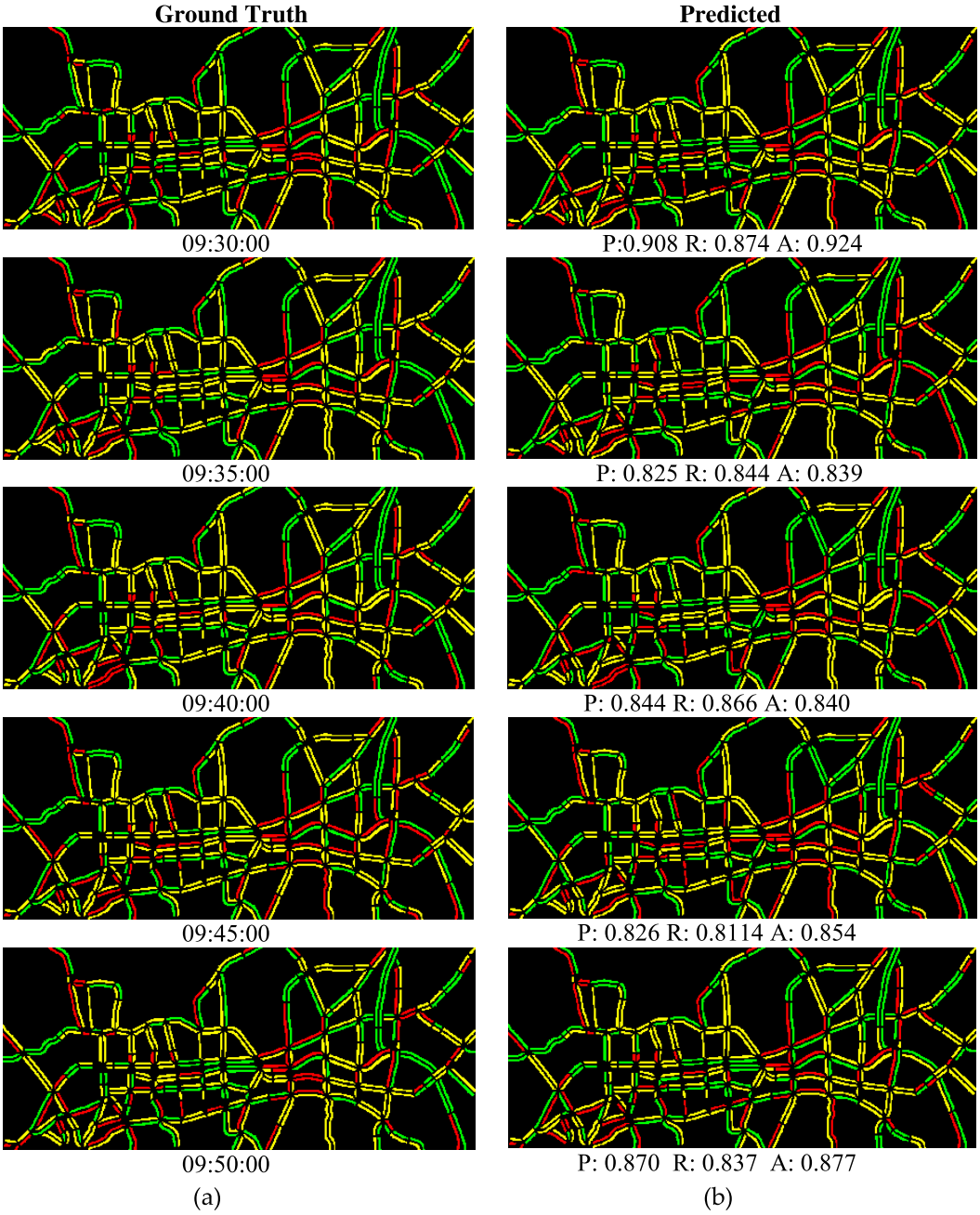

FIGURE 4. Comparison of ground truth congestion levels in column (a) vs. prediction congestion levels in column (b) on 3rd December 2019 with the prediction horizons of 10 minutes.

converge, and ConvLSTM around 15.67 to 16.95 hours to converge. PredNet is eight times faster than ConvLSTM. For original input resolution, Autoencoder is practically impossible to model and train on same device where PredNet and ConvLSTM are trained. So, in this research, we have reformatted the input matrix for Autoencoder by taking one-pixel value per road from the image instead of every pixel. The input dimension for Autoencoder decrease to (12, 345), i.e., 12 image samples with 345 road congestion level on each image. The trainable parameter decreases to 2.9 million and training time down to around 40 to 50 minutes. Even though resource utilization and training time are very low compared to the PredNet, in terms of performance, the Autoencoder model is very poor. Hence, in regard to the high-performance result, the proposed model is efficient in terms of training time compared to both models and efficient in terms of resource utilization compared to the Autoencoder.

V. DISCUSSION AND CONCLUSION

In this work, we present a deep learning model architecture to predict the city-wide traffic congestion prediction using an image data source from the online traffic portal. We develop the hybrid model by combining the Convolutional Neural Network, LSTM, and Transposed Convolutional Network,

which can learn both spatial and temporal relation of the input data effectively. The model was trained for three prediction horizons 10, 30, and 60 minutes. Unlike in the previous studies where researchers compared the predicted image with the ground truth in terms of MSE or MAE showing the average error between the images rather than the road-wise prediction, in this paper, we compare our proposed PredNet performance with two other state-of-art algorithms ConvLSTM and Autoencoder, in terms of precision and recall for predicting each congestion level, and accuracy based on road-wise prediction.

As discussed in section IV-C, we conclude the optimal architecture for the proposed prediction model consists of 5 skip connection from feature extraction to reconstruction layers to prevent vanishing gradient, and the sequence of 12 historical images provide an excellent prediction result. From the result and analysis section IV-D, we can see our proposed model achieves the best average accuracy for all three prediction horizons, and precision and recall values are highest for PredNet for all congestion levels, for all prediction horizons. Besides, our proposed PredNet beats the ConvLSTM by 8 folds in terms of computing time and can train image data with a large resolution on a smaller resource compared to Autoencoder, as it incorporates convolutional and downsampling layers rather than fully connected layers as in Autoencoder. However, with the encouraging prediction performance, there is still room for improvement in the model in terms of computational efficiency as a lot of resources and computing time is being wasted in learning the background area.

For future work, we can include external factors like weather information (rain, snow, foggy) for each road, which can improve the model performance. In addition, we will try to enhance the computing capability by removing all the background during our training and also try to add more information from the different data sources for more accurate predictions.

REFERENCES

[1] C. Onyeneke, C. Eguzouwa, and C. Mutabazi, "Modeling the effects of traffic congestion on economic activities—Accidents, fatalities and casualties," *Biomed. Statist. Informat.*, vol. 3, no. 2, pp. 7–14, Aug. 2018.

[2] C. Wang, M. A. Quddus, and S. G. Ison, "Impact of traffic congestion on road accidents: A spatial analysis of the M25 motorway in england," *Accident Anal. Prevention*, vol. 41, no. 4, pp. 798–808, Jul. 2009.

[3] P. Hao, C. Wang, G. Wu, K. Boriboonsomsin, and M. Barth, "Evaluating the environmental impact of traffic congestion based on sparse mobile crowd-sourced data," in *Proc. IEEE Conf. Technol. Sustainability (SusTech)*, Phoenix, AZ, USA, Nov. 2017, pp. 1–6.

[4] S. Ye, "Research on urban road traffic congestion charging based on sustainable development," *Phys. Procedia*, vol. 24, pp. 1567–1572, 2012.

[5] F. Rempe, G. Huber, and K. Bogenberger, "Spatio-temporal congestion patterns in urban traffic networks," in *Proc. Int. Symp. Enhancing Highway Perform. (ISEHP)*, Berlin, Germany, 2016, pp. 513–524.

[6] L. Xu, Y. Yue, and Q. Li, "Identifying urban traffic congestion pattern from historical floating car data," *Procedia Social Behav. Sci.*, vol. 96, pp. 2084–2095, Nov. 2013.

[7] J. Park, D. Li, Y. L. Murphey, J. Kristinsson, R. McGee, M. Kuang, and T. Phillips, "Real time vehicle speed prediction using a neural network traffic model," in *Proc. Int. Joint Conf. Neural Netw.*, San Jose, CA, USA, Jul. 2011, pp. 2991–2996.

[8] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, Apr. 2017, doi: 10.3390/s17040818.

[9] H. Chang, Y. Lee, B. Yoon, and S. Baek, "Dynamic near-term traffic flow prediction: System-oriented approach based on past experiences," *IET Intell. Transp. Syst.*, vol. 6, no. 3, p. 292, 2012, doi: 10.1049/iet-its.2011.0123.

[10] S. Zhang, Y. Yao, J. Hu, Y. Zhao, S. Li, and J. Hu, "Deep autoencoder neural networks for short-term traffic congestion prediction of transportation networks," *Sensors*, vol. 19, no. 10, p. 2229, May 2019, doi: 10.3390/s19102229.

[11] *Google Maps*. Accessed: May 4, 2019. [Online]. Available: https://www.google.com/maps/place/Delhi,+India/@28.6471948,76.9531797,11z/data=!3m1!4b1!4m5!3m4!1s0x390cfd5b347eb62d:0x37205b715389640!8m2!3d28.7040592!4d77.1024902

[12] Bing. *Bing Maps*. Accessed: May 5, 2019. [Online]. Available: https://www.bing.com/maps/traffic

[13] Seoul TOPIS. *Seoul Transport Opearation & Information Service Center*. Accessed May 5, 2019. [Online]. Available: https://topis.seoul.go.kr/prdc/openPrdcMap.do

[14] Baidu. *Baidu Maps*. Accessed: May 10, 2019. [Online]. Available: https://map.baidu.com/13036895.494262943,4748316.384998233,11.52z/maplayer%3Dtrafficrealtime

[15] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Vancouver, BC, Canada, May 2013, pp. 6645–6649, doi: 10.1109/ICASSP.2013.6638947.

[16] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Wuhan, China, Nov. 2016, pp. 324–328, doi: 10.1109/YAC.2016.7804912.

[17] W. Wei, H. Wu, and H. Ma, "An AutoEncoder and LSTM-based traffic flow prediction method," *Sensors*, vol. 19, no. 13, p. 2946, Jul. 2019, doi: 10.3390/s19132946.

[18] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, doi: 10.1109/TPAMI.2016.2644615.

[19] X. Liu, Z. Deng, and Y. Yang, "Recent progress in semantic image segmentation," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 1089–1106, Aug. 2019, doi: 10.1007/s10462-018-9641-3.

[20] B. L. Smith and M. J. Demetsky, "Traffic flow forecasting: Comparison of modeling approaches," *J. Transp. Eng.*, vol. 123, no. 4, pp. 261–266, 1997, doi: 10.1061/(ASCE)0733-947X(1997)123:4(261).

[21] M.-C. Tan, S. C. Wong, J.-M. Xu, Z.-R. Guan, and P. Zhang, "An aggregation approach to short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 1, pp. 60–69, Mar. 2009, doi: 10.1109/TITS.2008.2011693.

[22] E. Frejinger and M. Bierlaire, "Capturing correlation with subnetworks in route choice models," *Transp. Res. B, Methodol.*, vol. 41, no. 3, pp. 363–378, Mar. 2007, doi: 10.1016/j.trb.2006.06.003.

[23] S. V. Kumar and L. Vanajakshi, "Short-term traffic flow prediction using seasonal ARIMA model with limited input data," *Eur. Transp. Res. Rev.*, vol. 7, Sep. 2015, Art. no. 21, doi: 10.1007/s12544-015-0170-8.

[24] Z. Zhu, B. Peng, C. Xiong, and L. Zhang, "Short-term traffic flow prediction with linear conditional Gaussian Bayesian network," *J. Adv. Transp.*, vol. 50, no. 6, pp. 1111–1123, Oct. 2016, doi: 10.1002/atr.1392.

[25] G. Shi, J. Guo, W. Huang, and B. M. Williams, "Modeling seasonal heteroscedasticity in vehicular traffic condition series using seasonal adjustment approach," *J. Transp. Eng.*, vol. 140, no. 5, pp. 1053–1058, 2014.

[26] J. Guo, W. Huang, and B. M. Williams, "Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification," *Transp. Res. C, Emerg. Technol.*, vol. 43, pp. 50–64, Jun. 2014.

[27] L. Zhang, Q. Liu, W. Yang, N. Wei, and D. Dong, "An improved K-nearest neighbor model for short-term traffic flow prediction," *Procedia-Social Behav. Sci.*, vol. 96, pp. 653–662, Nov. 2013, doi: 10.1016/j.sbspro.2013.08.076.

[28] M. Castro-Neto, Y. Jeong, M. K. Jeong, and L. D. Han, "AADT prediction using support vector regression with data-dependent parameters," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 2979–2986, Mar. 2009, doi: 10.1016/j.eswa.2008.01.073.

[29] H. Su, L. Zhang, and S. Yu, "Short-term traffic flow prediction based on incremental support vector regression," in *Proc. 3rd Int. Conf. Natural Comput. (ICNC)*, Haikou, China, 2007, pp. 640–645, doi: 10.1109/ICNC.2007.661.

[30] X. Wang, N. Zhang, Y. Zhang, and Z. Shi, "Forecasting of short-term metro ridership with support vector machine online model," *J. Adv. Transp.*, vol. 2018, pp. 1–13, Jun. 2018, doi: 10.1155/2018/3189238.
[31] Y. Sun, B. Leng, and W. Guan, "A novel wavelet-SVM short-time passenger flow prediction in beijing subway system," *Neurocomputing*, vol. 166, pp. 109–121, Oct. 2015, doi: 10.1016/j.neucom.2015.03.085.
[32] S. Sun, C. Zhang, and G. Yu, "A Bayesian network approach to traffic flow forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 124–132, Mar. 2006, doi: 10.1109/TITS.2006.869623.
[33] P. Kuan Hoong, O. Kok Chien, I. K. T. Tan, and C.-Y. Ting, "Road traffic prediction using Bayesian networks," in *Proc. IET Int. Conf. Wireless Commun. Appl. (ICWCA)*, Kuala Lumpur, Malaysia, 2012, pp. 1–5.
[34] B. Sharma, S. Kumar, P. Tiwari, P. Yadav, and M. I. Nezhurina, "ANN based short-term traffic flow forecasting in undivided two lane highway," *J. Big Data*, vol. 5, Dec. 2018, Art. no. 48, doi: 10.1186/s40537-018-0157-0.
[35] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 187–197, May 2015, doi: 10.1016/j.trc.2015.03.014.
[36] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Nov. 2016, pp. 324–328, doi: 10.1109/YAC.2016.7804912.
[37] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, "LSTM-based traffic flow prediction with missing data," *Neurocomputing*, vol. 318, pp. 297–305, Nov. 2018, doi: 10.1016/j.neucom.2018.08.067.
[38] Y.-y. Chen, Y. Lv, Z. Li, and F.-Y. Wang, "Long short-term memory model for traffic congestion prediction with online open data," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Rio de Janerio, Brazil, Nov. 2016, pp. 132–137, doi: 10.1109/ITSC.2016.7795543.
[39] Z. Huang, J. Xia, F. Li, Z. Li, and Q. Li, "A peak traffic congestion prediction method based on bus driving time," *Entropy*, vol. 21, no. 7, p. 709, Jul. 2019, doi: 10.3390/e21070709.
[40] X. Luo, D. Li, Y. Yang, and S. Zhang, "Spatiotemporal traffic flow prediction with KNN and LSTM," *J. Adv. Transp.*, vol. 2019, pp. 1–10, Feb. 2019, doi: 10.1155/2019/4145353.
[41] W. Wei, H. Wu, and H. Ma, "An AutoEncoder and LSTM-based traffic flow prediction method," *Sensors*, vol. 19, no. 13, p. 2946, Jul. 2019, doi: 10.3390/s19132946.
[42] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, "LC-RNN: A deep learning model for traffic speed prediction," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3470–3476, doi: 10.24963/ijcai.2018/482.
[43] Y. Liu, H. Zheng, X. Feng, and Z. Chen, "Short-term traffic flow prediction with conv-LSTM," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2017, pp. 1–6, doi: 10.1109/WCSP.2017.8171119.
[44] H. Yu, Z. Wu, Y. Wang, and X. MA, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," *Sensors*, vol. 17, no. 7, 1501, Jun. 2017, doi: 10.3390/s17071501.
[45] M. Chen, X. Yu, and Y. Liu, "PCNN: Deep convolutional networks for short-term traffic congestion prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 11, pp. 3550–3559, Nov. 2018.
[46] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PLoS ONE*, vol. 10, no. 3, Mar. 2015, Art. no. e0119044, doi: 10.1371/journal.pone.0119044.
[47] A. Elfar, A. Talebpour, and H. S. Mahmassani, "Machine learning approach to short-term traffic congestion prediction in a connected environment," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2675, no. 45, pp. 185–195, Dec. 2019, doi: 10.1177/0361198118795010.
[48] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, Jan. 1962.
[49] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.
[50] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella, "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *Proc. IEEE Int. Conf. Signal Image Process. Appl. (ICSIPA)*, Kuala Lumpur, Malaysia, Nov. 2011, pp. 342–347, doi: 10.1109/ICSIPA.2011.6144164.
[51] D. Scherer, A. Muller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks—ICANN*. Berlin, Germany: Springer, 2010, p. 6354.
[52] S. J. Tobias, A. Dosovitskiy, T. Brox, and M. A. Riedmiler, "Striving for simplicity: The all convolutional net," 2014, *arXiv:1412.6806v3*. [Online]. Available: https://arxiv.org/abs/1412.6806
[53] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[54] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994, doi: 10.1109/72.279181.

NAVIN RANJAN received the bachelor's degree from Kathmandu University, Nepal, in 2016. He is currently working as a Graduate Research Assistant with Incheon National University, South Korea. His research interests include the Internet of Things (IoT), image processing, computer vision, scene understanding, and reinforcement learning.

SOVIT BHANDARI received the bachelor's degree from Kathmandu University, Nepal, in 2016. He worked as a Core Network Engineer with Huawei Technologies Nepal Company, Pvt. Ltd., from January 2018 to August 2018. He has been working as a Graduate Research Assistant with Incheon National University, South Korea, since September 2018. His research interests include radio resource management, 5G and beyond mobile communication, machine learning, the Internet of Things, image processing, and computer vision.

HONG PING ZHAO received the bachelor's degree from Jilin Jianzhu University, China, in 2016. He has been working as a Graduate Research Assistant with Incheon National University, South Korea, since March 2018. His research interests include machine learning, big data, and networking.

HOON KIM has been working as an Associate Professor with the Department of Electronics Engineering, Incheon National University, South Korea, since 2008. His research interests include radio resource management, optimization techniques, 5G mobile communication systems, machine learning, and the Internet of Things. Dr. Kim is a member of KICS, IEIE, and IEICE.

PERVEZ KHAN has been working as an Assistant Professor with the Department of Computer Science and Information Technology, University of Malakand, Pakistan, since March 2018. His research interests include wireless communications, wireless sensor networks, wireless ad-hoc networks, wireless body area networks, 5G networks, machine learning, fuzzy logic, and MAC protocol design.