sustainability

MDPI

Article

Large-Scale Road Network Congestion Pattern Analysis and Prediction Using Deep Convolutional Autoencoder

Navin Ranjan [1], Sovit Bhandari [1], Pervez Khan [1], Youn-Sik Hong [2] and Hoon Kim [1,*]

[1] IoT and Big Data Research Center, Department of Electronics Engineering, Incheon National University, Incheon 22012, Korea; ranjannavin07@gmail.com (N.R.); sovit198@gmail.com (S.B.); pervaizkanju@hotmail.com (P.K.)

[2] Department of Computer Science and Engineering, Incheon National University, Incheon 22012, Korea; yshong@inu.ac.kr

* Correspondence: hoon@inu.ac.kr

Abstract: The transportation system, especially the road network, is the backbone of any modern economy. However, with rapid urbanization, the congestion level has surged drastically, causing a direct effect on the quality of urban life, the environment, and the economy. In this paper, we propose (i) an inexpensive and efficient Traffic Congestion Pattern Analysis algorithm based on Image Processing, which identifies the group of roads in a network that suffers from reoccurring congestion; (ii) deep neural network architecture, formed from Convolutional Autoencoder, which learns both spatial and temporal relationships from the sequence of image data to predict the city-wide grid congestion index. Our experiment shows that both algorithms are efficient because the pattern analysis is based on the basic operations of arithmetic, whereas the prediction algorithm outperforms two other deep neural networks (Convolutional Recurrent Autoencoder and ConvLSTM) in terms of large-scale traffic network prediction performance. A case study was conducted on the dataset from Seoul city.

Keywords: traffic congestion; congestion pattern modelling; convolutional autoencoder; traffic prediction

Citation: Ranjan, N.; Bhandari, S.; Khan, P.; Hong, Y.-S.; Kim, H. Large-Scale Road Network Congestion Pattern Analysis and Prediction Using Deep Convolutional Autoencoder. *Sustainability* **2021**, *13*, 5108. https://doi.org/10.3390/su13095108

Academic Editors: Pan Lu and Marc A. Rosen

Received: 12 February 2021
Accepted: 24 April 2021
Published: 2 May 2021

1. Introduction

Traffic congestion has become a serious issue faced by cities around the world, developed or not, and the pattern indicates that the situation is going to get worse [1]. Traffic congestion directly affects the quality of urban life, by lengthening commuting time, promoting a rise in the tendency towards road rage, increasing the number of traffic-related accidents, causing health-related problems due to air pollution, etc. It also effects the economy, urban sustainable growth and development, and environmental pollution [2–4]. Thus, this study on transportation management is of high importance. Congestion related problems occur when there is an imbalance in supply and demand along the transportation network. Typically, regarding the nature of occurrence, traffic congestion is of two types: reoccurring and non-reoccurring. In reoccurring congestion, parts of the road network experience congestion very frequently or at a specific time of the day, mainly due to a bottleneck in the transportation infrastructure, a similar origin–destination pattern among commuters, inefficient management of supply–demand, heavy weather, inaccurate traffic signaling time, or unforeseen conditions, etc. Non-reoccurring congestion consists of unpredictable non-repeating delays that usually last for a short duration and are generally seen during special events, road accidents, vehicle breakdown, road maintenance, emergency conditions, or bad weather [5]. As non-reoccurring congestion is spontaneous, non-typical, and unplanned, it is extremely difficult to estimate its likelihood. Therefore, it is crucial for transportation planning to identify activities to reduce its occurrence, vulnerability, and exposure, especially during emergency conditions [6]. Based on the nature of the occurrence,

several approaches [6–12] have been implemented to assess and reduce non-reoccurring congestion.

The problem of reoccurring congestion can be resolved by applying congestion management measures either on the supply-side or the demand-side. Supply can be extended by building new roads and transportation infrastructures, which is an expensive approach that also causes damage to the environment, and these newly built resources could be under-utilized during non-peak hours. Constructing new resources also leads to the increased attractiveness of private vehicle use (i.e., generation of induced traffic), which will intensify the road traffic congestion, leading to unsustainable mobility patterns in the long run [13]. On the other hand, demand-side management measures are designed to reduce vehicle demand on the road network through numerous traffic strategies, such as congestion pricing, transportation allowance, transit and ridesharing, parking pricing, etc. [14]. This approach is cheap, smoothly deployable on the existing transportation network, and does not accentuate congestion conditions due to inducing increased traffic, as seen in supply-side management.

The primary focus of this paper is developing a methodology to mitigate road traffic congestion. The research aims to use demand-side congestion measures as a building block towards resilient and sustainable urban mobility. In this regard, we developed traffic strategies and conducted a case study using real-traffic data from Seoul city (South Korea) to evaluate their capabilities in reducing reoccurring traffic congestion. We propose efficient city-wide algorithms: (i) traffic congestion pattern analysis based on image processing (TCPIP) and (ii) a deep convolutional autoencoder-based grid congestion index prediction network (CIPNet). The TCPIP algorithm generates the city-wide congestion map pattern for any period (such as morning or evening rush hour, noon, etc.); it shows the likelihood of traffic jam occurrence on each road in a transportation network by monitoring the patterns from the historical data. The proper monitoring of the congestion pattern is the first step toward building a resilient and sustainable transportation system [13–17]. Based on the congestion pattern map, the traffic planning agencies, policy-makers, and transportation experts can pinpoint and investigate those networks with a high probability of being in traffic-jam states, examine the cause and, accordingly, plan a demand-side management-based policy toward mitigating the congestion. There are various traffic congestion measures available; evaluating and finding the appropriate congestion measures is crucial. For instance, implementing policies such as congestion pricing, restrictions for heavy transport vehicles (like cargo and delivery trucks), or vehicle diversion to a congestion-free road can neutralize the network congestion during rush hour. The CIPNet predicts the grid-based traffic congestion index from the chronological sequence of the traffic data, which quantifies congestion levels and shows which particular region in the urban area will be affected by congestion in the future. Based on the prediction map, the traffic management agency can systematically plan and respond to the change by implementing demand-side policies, and individual drivers can plan their travel route ahead of time by circumventing the congested region. The prediction map can also be valuable while planning for traffic management during special events [18]. As both strategies are based on profound knowledge of traffic flow patterns, they can be used by policy-makers and transportation experts as part of a decision-making system to manage traffic congestion proactively. In the medium to long run, these approaches will reduce or eliminate congestion in the highly affected region by ensuring even distribution of traffic across the transportation network. The contribution of this paper can be summarized as follows:

- We develop an efficient city-wide traffic congestion pattern algorithm based on Image Processing. The algorithm generates the map which shows the parts of the road network suffering from high reoccurring congestion.
- We develop a grid-based congestion index prediction algorithm, based on the convolution autoencoder, which learns the spatial and temporal relationship from the sequence of historical images to predict a city-wide grid-based traffic congestion index.

- Both models can be generalized to the large-scale traffic analysis problem. Because the TCPIP algorithm is based on simple arithmetic operations, and the CIPNet is based on convolutional and pooling layers.
- Our extensive experiments on the Seoul city transportation network demonstrate the efficiency and effectiveness of the proposed approaches.

The remainder of the paper is organized as follows: Section 2 presents related literature on traffic congestion patterns and traffic forecasting. Section 3 presents traffic congestion data acquisition and preprocessing, traffic congestion pattern analysis methodology, and network congestion index prediction (problem statement and explanation of our deep neural network architecture for traffic congestion index prediction). Section 4 presents the data source statistics, results and findings for congestion analysis (congestion index, stochastic congestion map, and reoccurring congestion patterns), and traffic forecasting (metrics used to test the effectiveness of the proposed model, detailed explanations of model construction, and performance comparison of the proposed model with the forecasting models such as Convolutional Recurrent Autoencoder and ConvLSTM). Finally, Section 5 presents the discussion and conclusion of our work and indicates the future direction of this study.

2. Related Work

Undoubtedly, traffic jams are the most frustrating experience that a traveler can face in daily life. The impact of traffic congestion is particularly severe in developing or underdeveloped countries [8]. In this section, related work on congestion pattern analysis and congestion prediction is presented.

2.1. Traffic Congestion Pattern Analysis

Reoccurring congested roads are responsible for a significant amount of traffic congestion. To address this problem, static congestion clusters were constructed in [19] based on data from Floating Car (FC). It was constructed iteratively by comparing whether pairs of roads were in the same congestion state for at least a certain threshold (minimum congestion duration). In [20], the traffic intensity pattern was analyzed for different cases (all day, the effect of marketplaces, and day-wise traffic variations) based on data from Global Positioning System (GPS). In [21], the author uses Jaccard similarity to identify recurrent congestion that occurs on different days but at approximately the same location and time of day based on data from Floating Car. In [22], the author uses a sum of congested link lengths to identify relieving or worsening congestion based on data from historical congestion patterns. In [21], typical congestion patterns are identified using the cluster method based on the traffic performance index derived from data collected by Floating Car.

Our model is superior to the existing model in [20–23] because these models do not consider city-wide traffic analysis. The literature in [14] does consider the generation of city-wide static congestion clusters, but the algorithm is computationally intensive as it iteratively compares between each pair of roads to generate clusters, compared to our model which is based on a simple arithmetic operation. Moreover, the data pre-processing in the proposed model is simpler compared to [21], which uses a data cube with a different granularity of space and time that increases the dimension to find recurrent congestion on a single road link. Our proposed model does not use database search, which saves much time in operation compared to the literature [22,23], which first searches for similar traffic patterns to approximate the traffic state.

2.2. Traffic Congestion Predcition

Research on traffic forecasting has a rich history, starting from data-driven approaches that mainly focus on the development of statistical and mathematical models such as historical average [24], autoregressive integrated moving average model (ARIMA) [25], to parametric approaches such as K-nearest neighbor (KNN) [26], Support Vector Machine (SVM) [27], Bayesian network (BN) [28], and so on. All of the above methods requires ex-

tensive prior knowledge of the domain and feature engineering. Recently, the Deep Neural Network (DNN) model has become popular in various domains due to its ability to handle multidimensional data without feature engineering, its potential of non-linear learning ability, and availability of cheap and high computational power. Some of the well-known works based on the deep learning model are autonomous systems [29], medical diagnosis [30], transportation systems [31,32], radio resource management and caching [33–35], etc. Therefore, researchers have focused on the deep learning-based algorithm and remarkable results has been obtained based on Long Short-Term Memory (LSTM) [36,37], Convolutional Neural Network (CNN) [38,39], Autoencoder-LSTM [40], etc.

Most of the research work in transportation focuses on predicting traffic parameters such as traffic speed, traffic flow or traffic volume, etc., on a single road or few major roads in a region. Some of the recent work in large-scale traffic prediction; in [41,42] the researcher studies on the urban traffic signal problem and improves the traffic flow by synchronizing the traffic signals at the intersections. In [38], the researcher used data from Floating Car of some major roads to generate a two-dimensional time-space matrix and used a CNN model to predict the short-term (10 and 20 min) speed of road traffic. In [39], the researcher used the trajectories of taxis GPS, to generate a traffic flow volume matrix to predict the short-term (15 min) traffic flow volume. The time interval between samples is 15 min (sparse) unrealistic for real traffic flow volume. In [43], the researcher proposed an autoencoder-based DNN model to predict the traffic congestion of the inter-regional transportation network. Since the model is based on a fully connected architecture, the model becomes very computationally intensive for a large-scale traffic prediction problem. In [44], the researcher used the ConvLSTM model to predict the short-term traffic flow, but the training time is very high due to the large resolution of input image for the LSTM layer. To overcome the problem of trainability in [43] and time complexity in [44], we had proposed a novel hybrid DNN based on Convolutional Recurrent Autoencoder in our previous work [32]. PredNet, where first, a convolution encoder encodes the image sequence into a low-resolution feature map, followed by an LSTM, which trains on a low-resolution matrix. Finally, the convolutional decoder decodes the low resolution back to the original image dimension. Convolutional Autoencoder in [32] solves the trainability problem from [43] while decreasing the resolution for the LSTM model, solving the time complexity problem from [42]. Although the model in [32] outperforms the state-of-the-art neural network in [43] and [44], there is still room for improvement. Due to the large amount of computational power required, the PredNet model was trained with a batch size of one, which usually results in noisy estimation of the gradient (high variance). In this work, we propose a new architecture (CIPNet) by removing the LSTM layers between the convolutional encoder block and the decoder block in PredNet, which reduces the computational cost and allows the model to train with a higher batch size while achieving better performance by estimating the gradient accurately. It also solves the problems of trainability and time complexity for large scale traffic prediction.

3. Methodology

In this section, we first describe our approach to data acquisition, preprocessing, and database design from raw snapshots of the online traffic congestion map. Then, we propose city-wide (i) congestion pattern analysis based on image processing and (ii) grid-based congestion index prediction. Figure 1 shows the complete methodology for traffic congestion analysis.

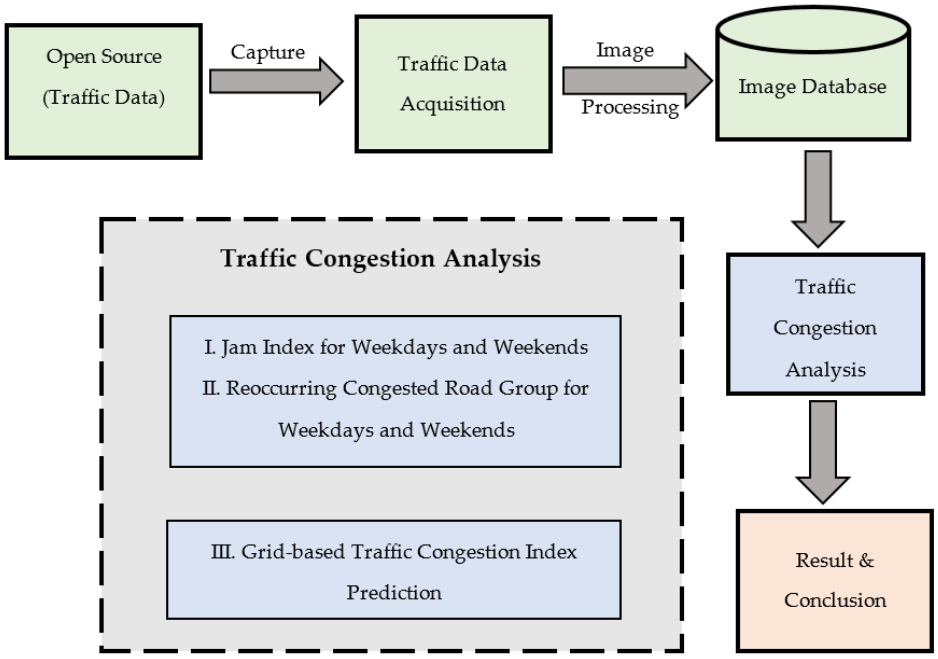


Figure 1. A general block diagram showing the methodology of the research paper.

3.1. Traffic Data

Most previous research uses data from a fixed sensor installed on each road or from a fleet of vehicles (Vehicular ad hoc networks (VANETs), Floating Car) operating on each road. In [19–21], each floating car reports its time and location information via GPS or cellular phone, from which traffic parameters such as speed and congestion are calculated. Each vehicle in Vehicular ad hoc networks (VANETs) in [45,46] acts as a data collection device and transmits the information using vehicle-to-vehicle communication and vehicle-to-fixed roadside infrastructure communication. Each Inductive loop in [47] and traffic camera in [48] collect information when vehicles pass over or through sensor's area. The number of operating devices for city-wide data collection should be very large, so the deployment, operation, and maintenance costs are very high. Moreover, all these infrastructures need to be deployed in each new city to establish traffic analysis. Recently, there has been an upsurge of web-based online public services worldwide. Online portals such as Google Traffic [49], Bing Map [50], Seoul Transportation Operation and Information Service (TOPIS) [51], Baidu Map [52], DOTD [53], Sigalert [54], Live Traffic NSW [55], Wisconsin Traffic [56], and I-Traffic [57] provide accurate real-time traffic information such as the congestion level of each road and the average speed of the road section. These online web services use multiple sources of data collection such as inductive loop, crowdsourcing, etc. to provide accurate real-time data for the entire city.

3.1.1. Traffic Data Acquisition

For data collection, we build a web crawling program that takes a snapshot of the website. To develop the web crawler program, it is essential to know exactly how the user browses a website, from the web address to the specific location on the web page where the information is located. Selenium IDE is the most widely used web crawling tool that controls and automates browsers like Chrome and Firefox through an API called Web Driver. Based on the knowledge of the website roadmap and Selenium IDE, the web crawling programming takes a snapshot of the raw traffic image as shown in Figure 2a. The image consists of road networks with three different congestion levels: congestion is represented by a red color, slow by a yellow color, and free by a green color. Algorithm 1 provides a step-by-step process for the traffic data acquisition method.

Algorithm 1 Traffic Data Acquisition

1. **Input:** Online Traffic Website (URL), Total number of collection days (D)
2. **Require:** Web browser (chrome), Selenium IDE, compatible web driver
3. **Output:** Raw snapshot of traffic Image
4. **for** all available days **d (0** $\leq d \leq D$**) do**
5. driver = webdriver.chrome() #Load web driver
6. driver.get(URL) # Load website
7. driver.fullscreen_window()
8. sleep(10) # Wait for website to load
9. **for** zoom in range (z = 2) **do**
10. driver.find_element_by_xpath('button path').click()
11. sleep(2)
12. **end**
13. driver.get_screenshot_as_file('save location with date and time as filename')
14. driver.close()
15. sleep(286)
16. **end**

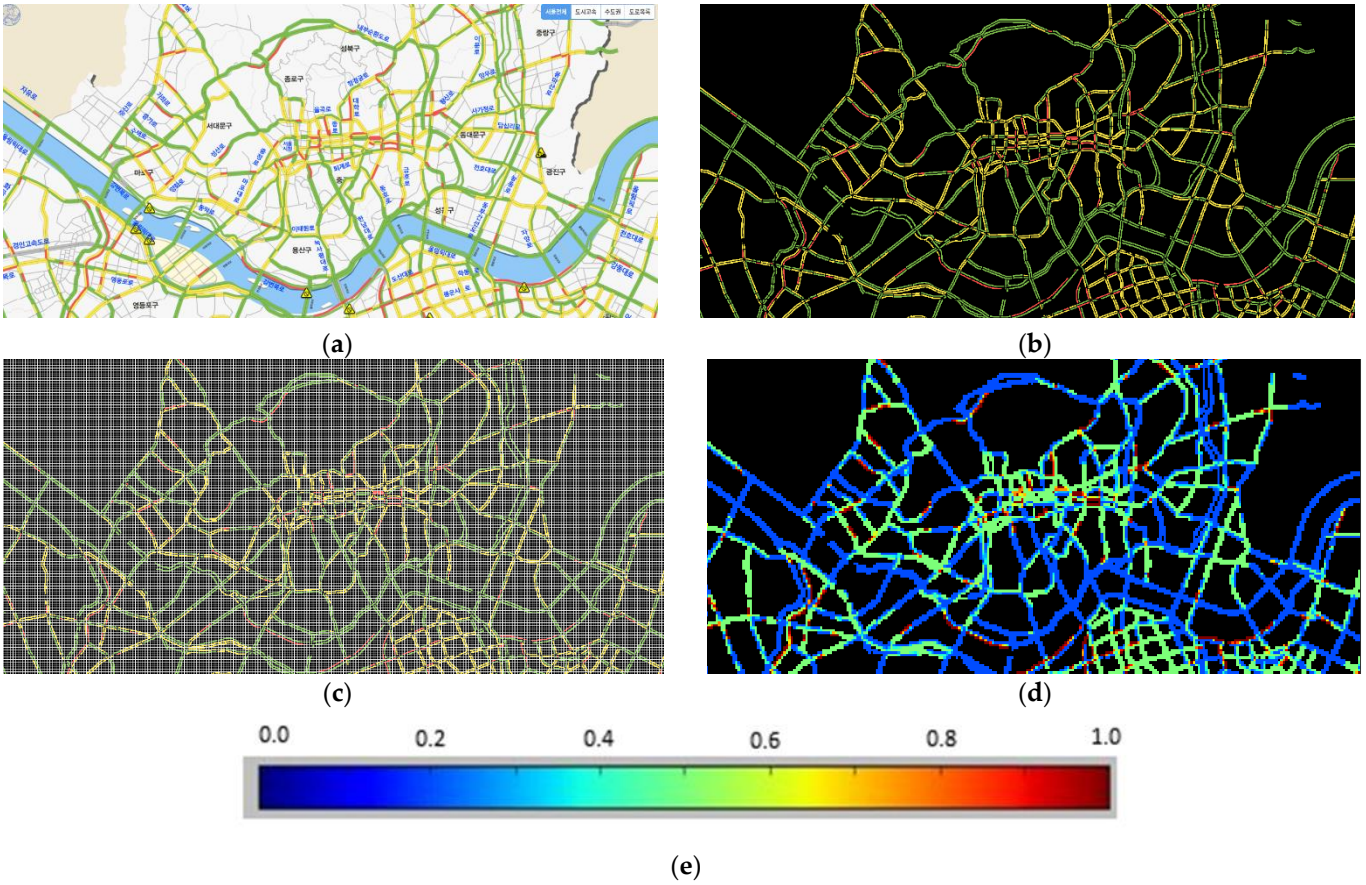


Figure 2. Image Data Preprocessing. (**a**) A snapshot of traffic congestion data from TOPIS website. (**b**) A sample of the Image dataset after road network extraction based on Algorithm 1, where the black color represents the background. (**c**) The road network segmented into 200 × 200 m^2 grids. (**d**) The congestion index of each grid. (**e**) Congestion Index distribution color scale.

Traffic website and number of days for data collection are the inputs; web browser (google chrome), selenium IDE, and web driver are program requirements. We loop the whole process until the completion condition (i.e., data collected for all mentioned number of days) is reached, as in line 4. We load the compatible web driver of the web browser

to use Selenium, as in line 5. Then we open the website using the 'get' command and use the 'fullscreen_window' command to load the website in full-screen mode and wait for a few seconds for the website to load completely, as in lines 5 to 8. Then we use the 'find_element_by_xpath' command to click on the website screen. In our case, we click on the zoom button link to zoom in on the website, waiting a few seconds between each click until the website is fully loaded, as in lines 9 to 12. Then we use the 'get_screenshot_as_file' command to capture the raw image of the traffic and save the data with the date and time as the file name, as in line 13. Finally, we close the web driver and sleep until the next screenshot time, as in lines 14 and 15.

3.1.2. Road Network Extraction

Digital Image is a collection of pixels, where each pixel in a color image takes on three values (red, green, and blue), whereas, in a binary or grayscale image it takes on only one value. Each pixel value is between [0, 255]. The congestion levels has unique color composition with lower and upper bounds ([75, 80, 230], [77, 100, 255]), ([75, 217, 230], [78, 238, 255]), and ([75, 190, 120], [124, 202, 160]) for red, yellow and green colors, respectively. To extract the road network with congestion information from the image, the Pseudo-algorithm is presented in Algorithm 2. The input to the algorithm is a raw snapshot image (r), the requirement of the algorithm is upper bound (U) and lower bound (L) for each congestion level (congestion, slow and free), and the output of the algorithm is only the road network image. Line 4 returns a loop for all the raw images in the database. For each input image, line 5 provides a loop to compute the mask images for all the congestion levels. Line 6 performs the pixel-wise comparison of the input image with the boundary condition for each congestion level to update the mask image for each congestion level. Here, (m_c) is the mask image for congestion level c, where $c \in [jam,\ slow,$ and $free]$, (i, j) is the pixel position, $r(i, j)$ is the pixel value of the input image, and U_c and L_c are the upper and lower boundary conditions for each congestion level. The mask image pixel $m_c(i, j)$ is updated to '1' if the corresponding pixel value in the raw image is within the boundary conditions, otherwise updated by '0'. In line 8, all the mask images $(m_{jam}, m_{slow}$ and $m_{free})$ are summed element-wise to produce a single mask image (m) containing a pixel value of '1' for the road network and '0' for the non-road network or background. In line 9, the 'bitwise and' operation (&) is performed between the final mask image (m) and the original input raw image (r) to generate the image with only road network, as shown in Figure 2b. Finally, in Line 10, the output matrix is saved as an image file. The whole process from line 5 to 10 is executed in a loop until the condition in line 4 is satisfied.

Algorithm 2 Road Network Extraction

1. **Input:** Image Data (r)
2. **Require:** Lower (L) and Upper (U) boundary value for each congestion level
3. **Output:** Image with Road network only
4. **for** each image in database **do**
5. | **for** each congestion level c in congestion list [jam, slow, and free] **do**
6. | | $m_c(i,j) = \begin{cases} 1 & L_c \le r(i,j) \le U_c, \\ 0 & other\ wise \end{cases} \qquad c \in [\mathrm{jam}, \mathrm{slow}, \mathrm{free}]$
7. | end
8. | $m = m_{jam} + m_{slow} + m_{free}$
9. | output $= r \,\&\, m$
10. | Save **output** matrix as image
11. **end**

3.1.3. Grid Representation

Deep Neural Network models become more complex and require more computation power, memory, and time to train the model as the input image resolution increases. Instead

of using original resolution image for the DNN model, we segmented the image into non-overlapping grids of 5 × 5 pixels [58], as shown in Figure 2c. Each grid corresponds to a 200 × 200 m^2 region, reducing the resolution by a factor of 25 and reducing the complexity of the DNN model. We represent congestion statistics of each grid by congestion index rather than congestion level, as shown in Figure 2d. The grid congestion index is calculated based on Equation (1), which consider the length of road for each congestion level in the grid.

$$Congestion\ Index = \frac{0.2 \times l_f\ + 0.5 \times l_s + 1.0 \times l_j}{l_f + l_s + l_j} \times 100 \quad (1)$$

Here, l_f, l_s, and l_j are the length, and the factor 0.2, 0.5, and 1.0 are the weights of the free, slow and jam congestion level of the road network. The weight are chosen based on logical considerations, and a similar approach is also discussed in the literature [58]. In general, only roads with jam and slow congestion states are considered in the calculation of the congestion index. Since the slow-congestion state has less impact on traffic mobility than the congested condition when calculating the congestion index, the weight for the slow-congestion state is kept lower than that for the congestion condition. To distinguish congestion-free roads from the background and show the impact of congestion-free roads on the entire network. In addition to congested and slow roads, we also considered congestion-free roads in the calculation of the network congestion index.

Table 1 shows the threshold condition for each grid to be categorized into Jam, Slow, and Free states. For a grid to be categorized as jam congestion level, it should have at least 50% congested roads, as indicated in grid state number (G.S.N) 2 and 3. Similarly, for a grid to be categorized as slow congestion level, it should have at least 50% roads with a slow-congestion state and congested road length should not exceed 50%, as shown in G.S.N 4 and 5. Similarly, the grid to be categorized as a free congestion level, the length of congested and slow congested road in the grid should not exceed 50%, as shown in G.S.N 6. Without considering the road length of free congestion level, the grid with (50% jam + 50% free) congestion condition as in G.S.N 3 and the grid with 100% slow congestion state as in G.S.N 4 has an equal congestion index of 50. However, the impact of G.S.N 3 should be higher compared to G.S.N 4 because there are roads with congestion condition in G.S.N 3. In order to remove such occlusion, in this work, we have considered the free congestion level road with a weighting factor of 0.2 while calculating the grid congestion index. With the inclusion of free congested roads, Table 1 shows the upper and lower limits for jam, slow, and free congestion levels for grid [100, 60], [60, 35], and for free [35, 20] with better edge resolution. The weights for jam should always be the highest, followed by the slow state, and the free state should have the lowest weight compared to the other two.

Table 1. Grid Congestion Index boundary value for categorizing congestion level. The boundary value is marked in bold.

G.S.N	Grid Status	Grid C.L	Congestion Index	Congestion Index [1]
1.	100% Jam	Jam	1.0 × 100 = **100**	1.0 × 100 = 100
2.	50% Jam + 50% Slow	Jam	1.0 × 50 + 0.5 × 50 = 75	1.0 × 50 + 0.5 × 50 = 75
3.	50% Jam + 50% Free	Jam	1.0 × 50 + 0.2 × 50 = **60**	1.0 × 50 + 0.0 × 50 = 50
4.	100% Slow	Slow	0.5 × 100 = 50	0.5 × 100 = 50
5.	50% Slow + 50% Free	Slow	0.5 × 50 + 0.2 × 50 = **35**	0.5 × 50 + 0.0 × 50 = 25
6.	100% Free	Free	0.2 × 100 = **20**	0.0 × 100 = 0
7.	No Road (Background)	X	0	0

G.S.N: Grid State Number; Grid C.L: Grid Congestion Level; Congestion Index [1]: with factor of 0.0 for l_f in numerator of Equation (1).

3.2. Traffic Congestion Analysis

In recent years, numerous research has been conducted in the field of transportation and traffic management towards traffic forecasting [59–61], prediction [32,38,39,44], and navigation planning. However, little to no priority has been given to research on the

root cause of traffic congestion, i.e., reoccurring congestion. In this section, we propose a city-wide reoccurring traffic patterns based on the Image Processing.

Algorithm 3 Reoccurring Congested Road Pattern

1. **Input:** Jam mask image, m_{jam} (from Algorithm 1)
2. **Output:** Reoccurring Congested Road Pattern
3. count = 0
4. **for** each day data in Jam database **do**
5. **for** each mask image in each day Jam database **do**
6. U_h ⟵ add Jam mask image
7. count = count + 1
8. **if** count is equal to 12 **do**
9. JR_h ⟵ subtract U_h matrix by 6
10. Convert negative pixel values in JR_h to 0
11. Save JR_h
12. Reset count to 0
13. **end**
14. **end**
15. Add all the JR_h of one day to get most Jam route of entire day (MCRR)
16. **end**
17. **for** each MCRR in database **do**
18. Convert MCRR >0 = 1
19. RCRP ⟵ Add each MCRR
20. **end**
21. RCRP ⟵ Subtract RCRP by the half the number of observation day

Congested roads are very crucial, especially in a busy area, where these roads can temporarily create bottlenecks in the transport network and lead all adjacent road sections into a state of congestion. This effect is even more serious when some road networks are frequently affected. Finding and solving the congestion problem from the reoccurring congested roads can significantly reduce the congestion level in the city. Generally, reoccurring congestion road patterns (RCRP) are different on weekdays, weekends, holidays, and in different weather conditions. In this paper, the algorithm for finding crucial reoccurring congestion roads in the city is presented by Algorithm 3. The input to the algorithm is the jam mask image from Algorithm 2, and the output of the algorithm is reoccurring congested road pattern. First, we calculate the most crucial reoccurring route (MCRR) of the whole day, as seen in lines 3 to 16, and then we generate reoccurring congestion road patterns as shown in lines 17 to 21. For MCRR generation, in the first step, we create the most frequent reoccurring jam road during one hour JR_h as in lines 5 to 13, and then add all hourly patterns to develop the MCRR. In line 3, we initialize a counter to keep track of hourly data. We add a loop the get each day jam masked data from database as in line 4, and then add another loop to get individual mask images of a particular day as in line 5. In line 6, an element-wise addition operation adds the jam images and simultaneously increments the counter value in line 7. Line 8 checks if all data of the respective hour has been processed, if so, an hourly jam pattern is created by subtracting the hourly matrix by a factor of 6 (30 min, i.e. the resulting matrix experiences the jammed state for at least 30 min) as in line 9. We convert the negative pixel of the JR_h matrix to 0, store the hourly pattern, and reset the counter for the next hour's analysis, as in lines 10 to 12. After calculating the most frequent hourly congestion of a whole day, all hourly pattern images are summed element-wise to generate MCRR. For the RCRP pattern, all MCRRs are converted to binary by replacing pixel values greater than zero with 1, as in line 17. Then, all binary MCRRs are summed element-wise as in line 19. Finally, each element of the RCRP matrix is subtracted by half the number of MCRR observation days to produce an RCRP pattern as in line 21.

Identifying Crucial Reoccurring Congested Road

Let $m_{jam}(t)$ be the jammed masked image of the raw snapshot image $r(t)$ and let the time interval between two consecutive mask images be 5 min, i.e., 12 mask images in one hour. From Algorithm 2, we know that $m_j(t)$ is an array that has value '1' for jam road pixels and '0' for everything else (road network with slow and free congestion and background). The road network experiencing congestion in an hour 'h' represented by U_h is given by Equation (2). The pixel value in U_h ranges from '0' representing, no congestion on the road, or in the background to '12' representing, there is congestion during the entire hour.

$$U_h = \sum_{t=0}^{11} m_{jam}(\mathrm{t}) \tag{2}$$

In this paper, we only consider the road segments that have a jammed state for at least 30 min in an hour. Therefore, subtracting the value 6 from U_h pixel by pixel yields a new matrix containing only the road network suffering from congestion for at least 30 min in hour 'h' called 'hourly jam route' (JR_h), given by Equation (3). All negative pixel values of the array JR_h are replaced by zero and all positive pixel values are replaced by 1, as shown in Equation (4).

$$JR_h = U_h - 6 \tag{3}$$

$$JR_h(i,j) = \begin{cases} 0 & if\ (jr_h)_{ij} < 0 \\ 1 & if\ (jr_h)_{ij} > 0 \end{cases} \tag{4}$$

Here, $(jr_h)_{ij}$ represents the i^{th} row and j^{th} column element of the matrix JR_h. The union of all hourly jam route (JR_h) in a day gives the reoccurring congested roads. These road clusters are the maximum crucial reoccurring road (MCRR) of a day, and for day 'd' it is given by Equation (5). The value in the MCRR matrix ranges from '0' representing no congestion, or background to '24' representing the road at jammed state for more than 30 min in an hour and has a repeating pattern for 24 h.

$$MCRR_d = \sum_{h=0}^{23} \mathrm{JR}_h^d \tag{5}$$

The MCRR pattern varies widely or repeats on some particular days. Since, traffic congestion depends on various factors such as weekday, weekend, holiday, or weather conditions, in this paper we will only analyze MCRR on a weekday to derive a maximum reoccurring congested road pattern (RCRP). First, the MCRR matrix is converted into a binary matrix according to Equation (6), and then the summation of all binary MCRR for weekdays, as shown in Equation (7), yields the RCRP matrix.

$$MCRR(i,j) = \begin{cases} 1 & if\ (mcrr)_{ij} > 0 \\ 0 & otherwise \end{cases} \tag{6}$$

$$RCRP = \sum_{d=0}^{n} MCRR_d \tag{7}$$

Here, $(mcrr)_{ij}$ represents the i^{th} row and j^{th} column element in matrix MCRR. The value of the RCRP matrix ranges from '0' representing no congestion or background, to 'n' representing road at jam state for more than 30 min in an hour, with the trend repeating for any number of the hours in a day, but at least once in 'n' days. The number of reoccurring congested road groups from Equation (7) is too large. To reduce the number of the reoccurring jammed road groups, we consider only the road groups where the pattern repeats for at least half of the number of observation days $(n/2)$. We can achieve the result by subtracting the RCRP matrix pixel-wise by '$n/2$', as shown in Equation (8). The resulting

matrix removes less frequently congested roads, and this matrix represents the city-wide traffic congestion pattern.

$$RCRP = RCRP - n/2 \tag{8}$$

3.3. Traffic Congestion Index Prediction

In this section, we first explain the problem statement and then discuss on the architecture for predicting the city-wide grid traffic congestion index using the time-series sequence of historical data.

3.3.1. Problem Statement

Let $N \in \{1, 2, \ldots, n\}$ be the chronological order of n image data in a database collected at sampling rate t. We design a deep neural network (f) that takes past p images as the input sequence (X) to predict the short-term congestion level of the traffic network (Y) at the prediction horizon of k. Table 2 shows the input sequence and its corresponding prediction output for the 1st and i^{th} time-series input to the network.

Table 2. Input sequence to the model and its corresponding prediction output.

Input Sequence	Prediction Output
$X^1_{pt} = \{\ x_1, x_2, \ldots,\ x_{p-2}, x_{p-1},\ x_p\}$	$Y^1_{pt} = \{x_{p+k}\}$
$X^i_{(i+p)t} = \{\ x_i, x_{i+1}, \ldots, x_{i+p-1},\ x_{i+p}\}$	$Y^i_{(i+p)t} = \{x_{(i+p)+k}\}$

Here, $X_{\blacksquare}$ represent the time-series sequence number, $X_{\blacksquare}$ represent the time at which sequence was feed to the network, $\{X, Y\} \subset N$, and $\{p, k\} \in \mathbb{N}$. For the i^{th} time-series input the model f can be defined as

$$Y^i_{(i+p)t} = f\left(X^i_{(i+p)t}, \theta\right) \tag{9}$$

Here, θ are the model parameters. We divide our database N into 3 parts training, validation, and testing. We generate m sets of historical time series data from the training dataset, $X_{train} = \left\{X^1, X^2, X^3, \ldots, X^m\right\}$, such that the corresponding forecast data point associated with X_{train} given by, $Y_{train} = \left\{x_{p+k}, x_{(1+p)+k}, x_{(2+p)+k} \ldots, x_{(m+p)+k}\right\}$ also lies in the training dataset. Therefore, we can train our neural network based on supervised learning.

3.3.2. Model Architecture

The proposed congestion index prediction network is based on a convolutional auto-encoder named CIPNet. The schematic is shown in Figure 3. The proposed model consists of two blocks (i) convolutional encoder and (ii) convolutional decoder. The convolutional encoder performs feature extraction from the input image and is mainly consists of convolutional layers and downsampling layers. The convolutional layer is the central building block of a CNN model. It contains a set of learnable filters that have a small receptive field but span the entire depth of the input volume. Each filter is convoluted with the input volume to compute an activation map. The output of the convolution layer is obtained by stacking the activation maps of all the filters. Since the filter has a small receptive field, the number of the parameters to be learned is significantly reduced compared to the full connection model, and a network can grow deeper with fewer parameters. Moreover, the small receptive field also allows sharing weights and exploits spatial correlation. Between two successive convolutional layers, the pooling layer is usually placed, whose task is to achieve shift invariance by gradually reducing the spatial dimension of the feature map while preserving the essential information. In this paper, the downsampling layer is performed by convolution operation with filter size and strides of 2×2. Since the input to the model is a time-series of historical images, each layer in the convolutional

encoder network is enclosed by the Time Distribution layer. The stack of convolutional layers and downsampling layers converts the original image into the latent space. On the other hand, the convolutional decoder network performs the image reconstruction from the latent space back to the original resolution. It consists of convolutional layers and upsampling layers. The upsampling operation is similar to the downsampling layer operation, where instead of halving the spatial dimension, the size of the previous layer is doubled. The upsampling layers are typically used for the generative model. Here, the upsampling operation is performed by the Convolutional Transpose layer, with a filter size and strides of 2 × 2. The ReLU activation function is applied to all layers, followed by batch normalization and dropout. The architecture includes a skip connection, i.e., a connection from the initial layers in the convolutional encoder to the later layers in the convolutional decoder, allowing the gradient to flow directly through the skip connection back from later layers to initial layers, solving the vanishing gradient problem in a deep neural network. The details of the model architecture are presented in Section 4.3.2.

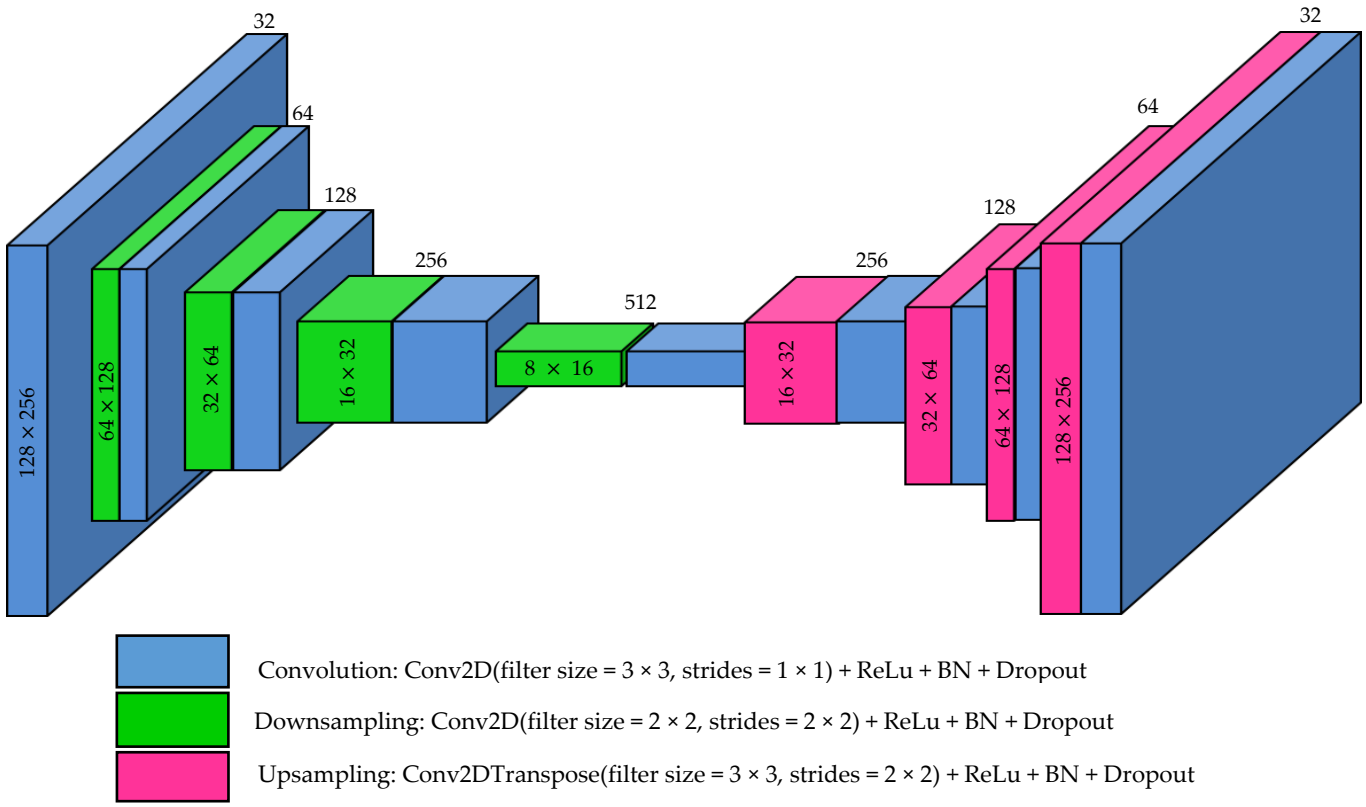


Figure 3. CIPNet Model Architecture. The model contains two part convolutional encoder and convolutional decoder. The color code represent the type of convolutional operation.

3.3.3. Training Process of CIPNet

Algorithm 4 summarizes the training process of CIPNet. At first, we preprocess the raw snapshot image to construct the training dataset. Then, we create the sequence of training samples and train the model via gradient-descent backpropagation and an Adam optimization algorithm as in lines 3 to 10. For a given temporal input sequences X_t and the target value Y_t for an arbitrary time interval t with prediction horizon k. During the training process, we initialize the model parameter θ with a 'he uniform' distribution with default values as in line 11. Then, we randomly select the batch of training instances S_b from the set S as in line 13 and repeat the process until predefined stopping criteria are satisfied as in lines 12 to 15. After the completion of the iteration, an optimal set of parameter θ representing the prediction model f is generated as in line 16.

Algorithm 4 Training process for CIPNet

Input: Grid traffic congestion index matrices: $\{\boldsymbol{x_t}|\boldsymbol{t} = \boldsymbol{0}, \boldsymbol{2}, \boldsymbol{3}, \ldots, \boldsymbol{n-1}\}$
sequence of input image: p, prediction horizon: k
Output: CIPNet model f
$\boldsymbol{S} \leftarrow \emptyset$
for all available time intervals $\boldsymbol{t}\ (\boldsymbol{1} \leq \boldsymbol{t} \leq \boldsymbol{n})$ **do**
  for all time-series sequence $\boldsymbol{i}, (\boldsymbol{0} \leq \boldsymbol{i} \leq \boldsymbol{n-p-k})$ **do**
    $\boldsymbol{X^i_{(i+p)t}} = \{\boldsymbol{x_i}, \boldsymbol{x_{i+1}}, \ldots, \boldsymbol{x_{i+p-1}}, \boldsymbol{x_{i+p}}\}$
    $\boldsymbol{Y^i_{(i+p)t}} = \{\boldsymbol{x_{(i+p)+k}}\}$
    put a training instance $\left(X^i_{(i+p)t}, Y^i_{(i+p)t}\right)$ into S
  end
end
initialize model parameters θ
do
  randomly select a batch of training instances S_b from S
  find θ by minimizing the objective $\mathcal{L}(\theta) = \left\|Y_t - \widehat{Y_t}\right\|_2^2$ with S_b
while the stopping criteria is not satisfied
Output the learned CIPNet model f

4. Experiment and Result Analysis

In this section, we evaluate the proposed (i) traffic congestion analysis based on the Image processing (TCPIP) algorithm and (ii) grid congestion index prediction network based on a deep neural network (CIPNet). Both algorithms are evaluated on the same dataset.

4.1. Data Source

In this study, we choose Seoul, the capital of South Korea, as a case study. Figure 2a shows the raw snapshot of the spatial transportation network image taken from the TOPIS traffic web page. Each image has a resolution of 1366 × 694 pixels and covers an area of 54.64 km × 27.1 km (scale 1 cm = 1.3 km). In this work, we collected the traffic congestion data for the entire day (24 hours) from 1 August 2020 to 31 October 2020, a total of 92 days, with an interval of 5 min (288 samples per day). Out of 92 days, we remove the data from some particular day due to missing data caused by an error or failure of the web crawling program. Furthermore, we divide the database into weekdays and weekends. We analyze the traffic congestion patterns for both weekdays and weekends. On the other hand, for grid congestion index prediction, we experimented only with weekday's data. For the congestion index prediction algorithm, we divide the weekday database (07:00 to 13:00) into three parts, namely: training, validation, and testing. The samples from 1st August to 30th September are used as training data, samples from 1st November to 10th November are used as validation data, and the sample from 11th November to 31st November is used to test the predictive ability of the CIPNet. Sections 3.1.2 and 3.1.3 describe the preprocessing of the data.

4.2. Traffic Congestion Analysis

In this subsection, we first analyze the city-wide traffic congestion index, which provides an initial overview of the congestion state. Second, we generate the stochastic congestion map, which exhibits the likelihood of the road network is congestion at a given time. Finally, we construct the city-wide reoccurring congestion road pattern, which provides information about the road priority where traffic management agencies can apply measures such as traffic diversion, selective entry, entry pricing, or road expansion to reduce congestion.

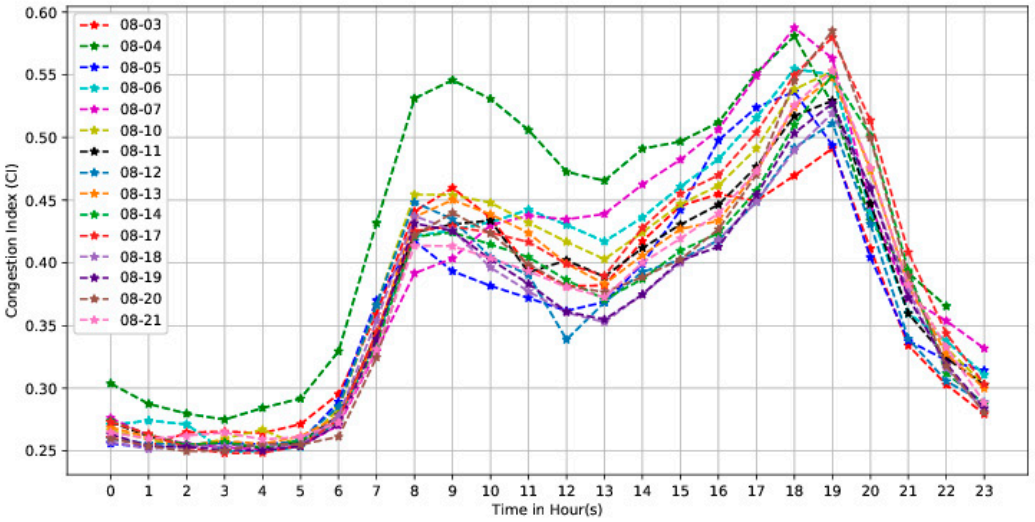
4.2.1. Jam Index Distribution

The congestion of the city can be analyzed quantitatively using the value of the jam index from Equation (1). Here, the value of jam index ranges from 20% when all the roads in network are in the free-state (normally value should be zero, but to differentiate the effects of congestion levels, we considered the congestion-free road length for the calculation of the congestion index, see Section 3.1.3) to 100% when all the roads are in the congested state.

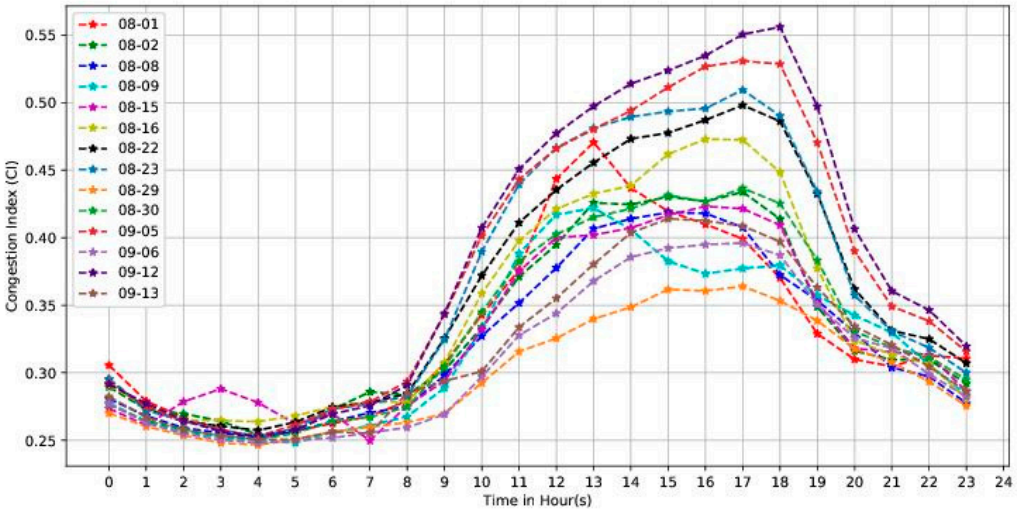
Figure 4a is the graph of the congestion index distribution for weekdays and shows similar trends for all working days. The congestion index increases rapidly from (25–30%) at 06:00 to about (40–45%) at 09:00 and remains in the same range until 15:00, then further increases to (50–60%) between 18:00 and 20:00, and gradually decreases thereafter. On weekdays, the morning peak hour begins at 08:00, where congestion is relatively low compared to the evening peak hour, which occurs between 17:00 and 21:00. Based on the congestion index boundary condition from Table 1, the city of Seoul experiences a free congestion condition in the early morning until 07:00 and between 23:00 and 24:00, while it approaching a city-wide congestion condition between 18:00 and 19:00 and has sloe congestion condition during the rest of the period. Similarly, Figure 4b shows the distribution of the jam index for weekends. The congestion index value does not follow similar trends as in the weekday analysis. Nevertheless, there is a sharp increase in the congestion index from (20–35%) in the morning until 09:00 to (35–55%) between 09:00 and 21:00 with the maximum between 15:00 and 19:00. From Table 1, it is observed that the city experiences congestion free condition in the early morning till 09:00 and late evening between 21:00 and 24:00, and slow congestion condition during the rest of the day. The analysis of congestion index distribution on weekdays and weekends shows that road traffic in Seoul city does not operate at optimal capacity, and large congestion occurs during peak hours. Consequently, a study of reoccurring traffic patterns and congestion predictions is needed to evaluate and solve the city-wide congestion problem.

4.2.2. Stochastic Congestion Maps

In this subsection, we construct the city-wide stochastic congestion map based on weekday data to analyze the traffic statistics and congestion patterns. The stochastic congestion map represents the likelihood of the occurrence of a congestion in a road network based on the observation of many days [60,61]. To construct the stochastic congestion map, we first, extract jam road segments from the database based on Algorithm 2 and group the image based on their timestamp, multiple of 3 hours, i.e., (0–3, 3–6, 6–9, ... 18–21, and 21–24). Then, all the images in their respective groups are added element-wise and finally each group are divided by the total number of the images in their respective group to obtain the stochastic congestion map. Figure 3 shows the dynamics of congestion during weekdays.



(**a**)



(**b**)

Figure 4. Road traffic jam index distribution graph. (**a**) The congestion index of the city during weekdays. (**b**) The congestion index of the city during weekends.

In Figure 5, the stochastic map for (0–3) and (3–6) hours is not included because they have a very-low recurrent congestion probability similar to that of the stochastic pattern for (21–24) hours, and the reoccurring congestion pattern of other time-period can be clearly seen. During the early morning, (6–9) hours, most of the road network in the central of Seoul is free of congestion, whereas a moderate level of reoccurring congestion (40–60%) to a high probability of reoccurring congestion is observed on the other roads, especially in the outskirts of the city. This pattern is true for Seoul, as many people commute from the outer suburbs to work in central Seoul. During (9–12) hours, in addition to the periphery, central Seoul also experiences a moderate to high probability of reoccurrence of congestion. After the end of the morning rush hour, the probability of congestion of most roads in the periphery decreases, while most of roads in the central region continue to experience a high probability of reoccurring congestion. Similarly, during (15–18) hours, the probability of reoccurring congestion rises for all road networks, especially in the central region. During (18–21) hours, almost all road networks show high probability of

reoccurring congestion, the maximum during the entire day. After 21:00 h, the likelihood of reoccurring congestion subsides to minimum except for few roads. The stochastic congestion map shows the reoccurring congestion probability, which is time dependent and changes frequently. Hence, for the traffic authority to focus on the most vulnerable road segment, in this paper we generate reoccurring traffic congestion pattern explained in Section 3.2.

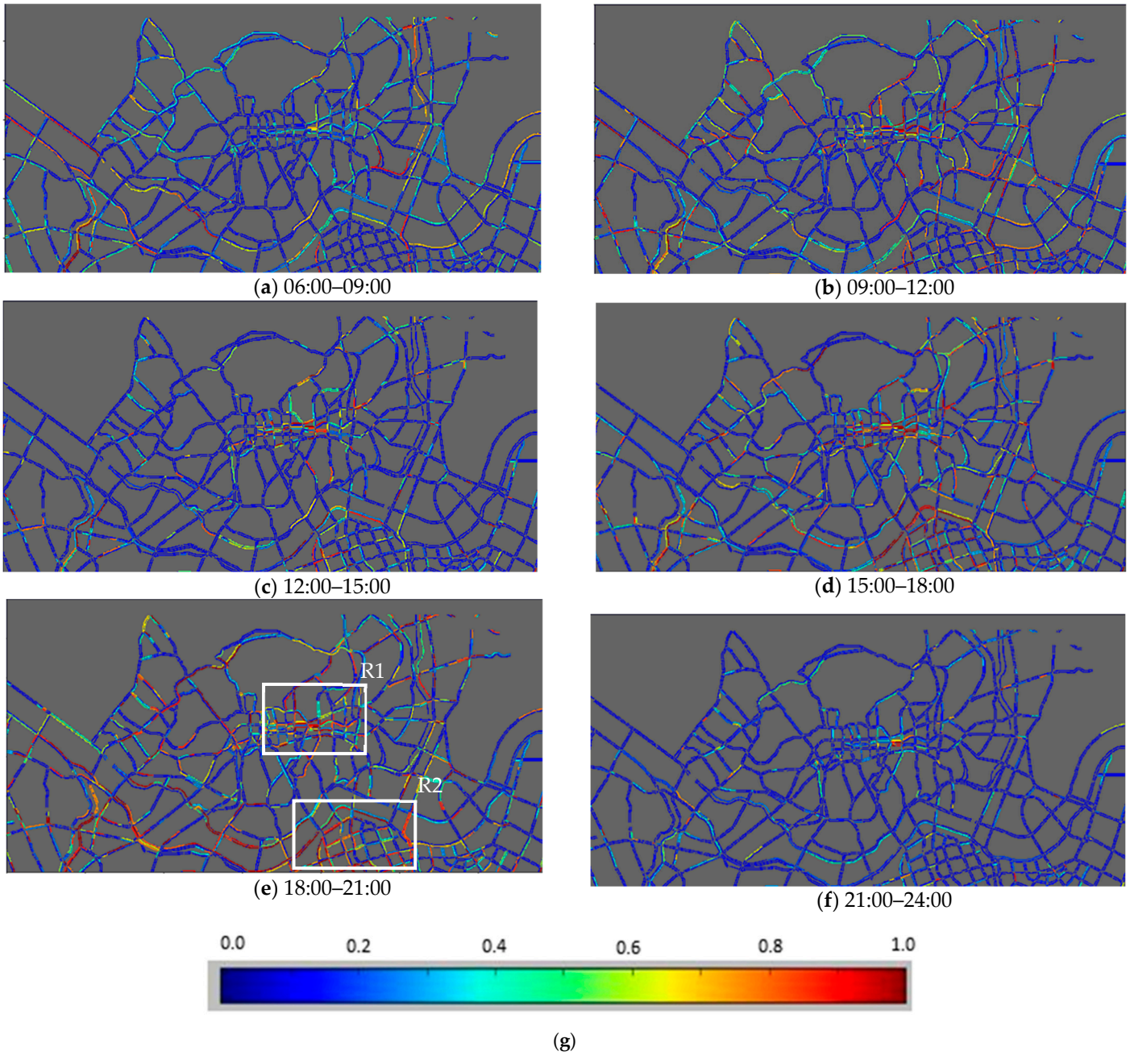
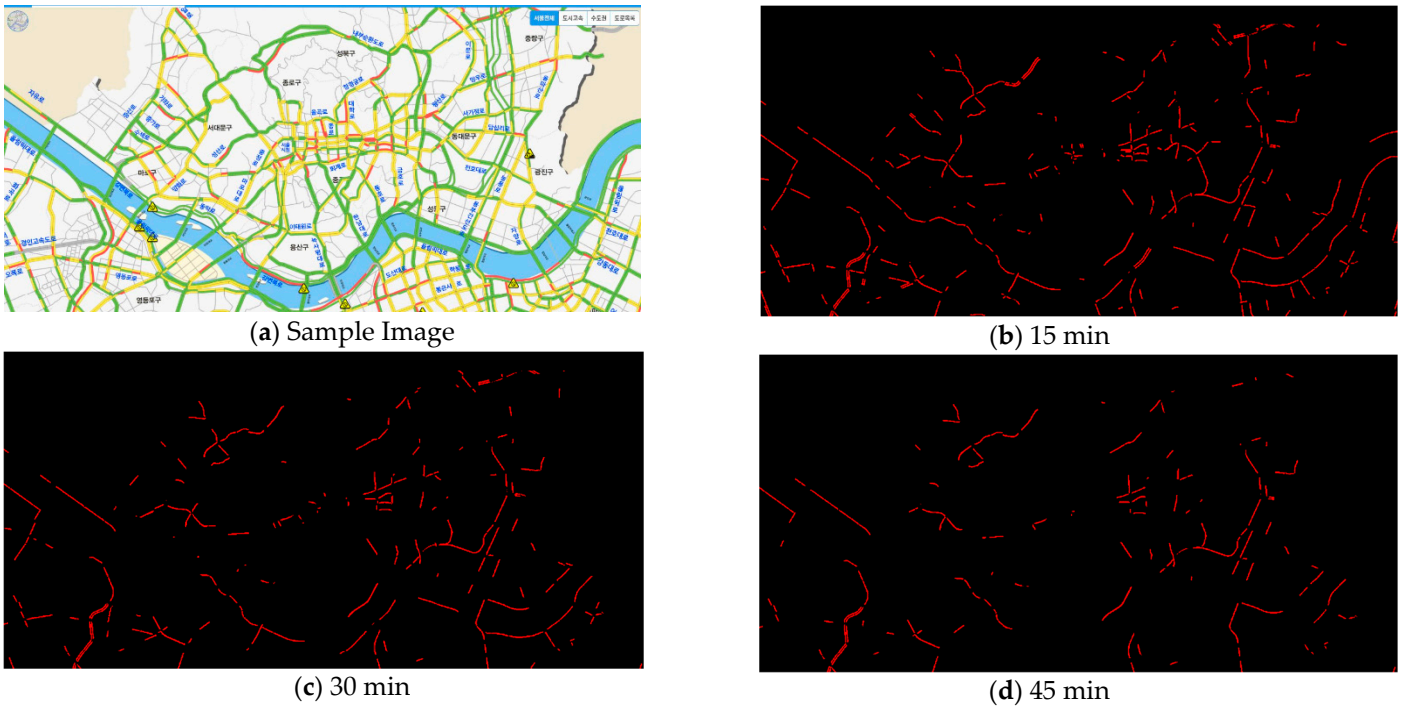


Figure 5. Weekdays Stochastic Congestion Maps 06:00–24:00. The likelihood of reoccurring congestion in the road network, in Figure (**a**–**f**). The likelihood color scale for the occurrence of traffic congestion, in Figure (**g**).

4.2.3. Reoccurring Traffic Congestion Pattern

In this subsection, we construct the reoccurring traffic congestion pattern based on Algorithm 3 presented in Section 3.2. Figure 5 shows the reoccurring traffic congestion pattern under different conditions. Figure 6a is an example of a raw image extracted from the website showing the spatial structure of the road. Figure 6b–d shows the reoccurring

congestion patter of the city based on the road network affected by congestion for at least 15, 30, and 45 min in an hour in a day and repeating the same pattern for at least half of the observation day. The frequently reoccurring congested roads are shown in 'red' color, and all other non-congested roads with background are shown in black color. In Figure 6b, the number of the reoccurring roads is immense and can be seen in almost every part of the city. Thus, the 15 min congestion statistic is not an optimal criterion for designing the reoccurring congestion pattern. Instead, 30 and 45 min congestion patterns give the road network with high priority. Therefore, to alleviate the city-wide congestion problems in the road network, the traffic management authority can use the congestion pattern from Figure 6c,d to investigate and solve the congestion cause on each road. Solving the congestion problem of congestion-prone roads can significantly reduce congestion in the city.



(**a**) Sample Image (**b**) 15 min

(**c**) 30 min (**d**) 45 min

Figure 6. Reoccurring traffic congestion pattern from Seoul City on weekdays. (**a**) The sample of a captured raw image from TOPIS traffic web service. Reoccurring traffic congestion pattern based on road network affected by congestion for at least (**b**) 15 min, (**c**) 30 min, and (**d**) 45 min in an hour.

4.3. Grid Congestion Index Prediction

In this subsection, we first present the performance comparison model and metric to verify the effectiveness and superiority of the proposed architecture, then we present in-depth experiments with the proposed model, and finally the results and analysis between the proposed and comparison models.

4.3.1. Comparison Model and Metrics

To compare the performance of the proposed Convolutional Auto-encoder-based neural network, two state-of-the-art deep learning neural networks, namely: ConvLSTM [44] and PredNet [32] were selected. ConvLSTM is similar to LSTM, where the internal matrix multiplication is replaced by a convolutional operation, allowing both spatial and temporal features to be learned. PredNet is a hybrid deep neural network formed by adding an LSTM layer between the convolutional encoder and the convolutional decoder in the convolutional Autoencoder. The model learns both the spatial and temporal relationship of the input sequence. The baseline performance measure is the performance level that the

dataset is currently at, and this value helps determine if the models performance is really improving or in other words, it helps determine if the model is learning. We performed the congestion prediction for a prediction horizon of 10, 20, 30, 45, and 60 min. In this paper, we present the performance result based on the mean absolute error (MAE), mean square error (MSE), and root mean square error (RMSE). Equations (10) and (11) define MAE and MAE.

$$MAE = \frac{1}{R}\frac{1}{C}\sum_{i=0}^{R-1}\sum_{j=0}^{C-1}\left|Y(i,j) - Y'(i,j)\right| \tag{10}$$

$$MSE = \frac{1}{R}\frac{1}{C}\sum_{i=0}^{R-1}\sum_{j=0}^{C-1}\left(Y(i,j) - Y'(i,j)\right)^2 \tag{11}$$

4.3.2. CIPNet Implementation

In this section, we explain the details of the CIPNet model architecture. As shown in Figure 3, the proposed model has 17 layers, nine convolutional layers, four downsampling layers, and four upsampling layers. The input to the model is a sequence of 12 historical images with the dimension of 128 × 256 and a channel depth of 1. The output of the trained model is a predicted image with a time horizon of (10, 20, 30, 45, and 60) minutes and have same image dimension as the input image. The sequence of input images is fed to a 2D convolutional layer which has 32 filters of size 3 × 3, a stride of 1 × 1 with zero paddings, a kernel weight initialization based on 'he uniform' and 'ReLU' as activation function. This layer convolves the input image without reducing the spatial dimension of the inputs. The convolutional layer is followed by a downsampling layer performed by a 2D convolutional layer consisting of 64 filters of size 2 × 2 and the stride of 2 × 2 without zero padding. The convolution layer and the downsampling layer are stacked one after another. The convolutional layer extracts the feature without decreasing the spatial size. The downsampling layer, on the other hand, halves the spatial size in both directions while doubling the number of filters compared to previous layer. The input dimension of 12 × 128 × 256 × 1 is encoded into 12 × 8 × 16 × 1 by the convolutional encoder. The combination of upsampling layers and convolutional layer regenerates the latent image representation to its original dimension. The upsampling layer doubles the spatial dimension while reducing the filter size to its half, formed by a Convolutional 2D Transpose layer with a filter size of 2 × 2 and strides of 2 × 2 without zero padding. All layers have a dropout of 0.1 and batch normalization layers.

Since CIPNet is a modified version of PredNet [32], we adopt some basic implementation parameters, such as the number of past image sequences at 12 and skip connections between the initial layers and later layers. We compare the performance of the proposed model, CIPNet, with ConvLSTM [44] and PredNet [32]. We implemented the ConvLSTM with a six-layer filter configuration of [48, 36, 24, 12, 6, and 1], with a filter size of 3 × 3 and strides of 1 × 1 with zero padding. Each layer is trained with the activation function ReLU, followed by the dropout layer with a value of 0.1 and a batch normalization. The input for the ConvLSTM models has the same input as that of the proposed model. For PredNet, we adopted the configuration of Convolutional Encoder with filter number [32, 64, 96, 128, 160, 192, 8] followed by 4 LSTM layers with hidden units 672 and then convolutional decoder with filter size [192, 160, 128, 96, 64, 32]. Each layer has ReLU activation followed by a dropout layer of 0.1 and batch normalization layers.

4.3.3. Prediction Result and Analysis

Figure 7 shows the detailed MSE result of all the prediction models for prediction horizons of 10 min on 12 October 2020, in a period from 09:00 to 12:50. The proposed network, CIPNet, has the lowest mean squared error compared to PredNet, ConvLSTM and Baseline. The proposed model achieves significantly lower MSE than Baseline and ConvLSTM and slightly better than PredNet. Out of 48 instances, CIPNet achieves lower prediction error 41 times, PredNet achieves lower prediction error five times and for the

other two times, both CIPNet and PredNet perform equally well. Table 3 shows the hourly average MSE and MAE values for all prediction models and baseline performance for 10 and 60 min prediction horizons on 12, 13, and 14 November 2020 from 09:00 to 13:00. The result shows that the proposed model has the lowest average MSE and MAE error on all three days compared to all other models. The PredNet is close to the performance of the CIPNet for the 10 min horizon. However, for the 60 min prediction, the performance of the proposed model, CIPNet, is similar to PredNet, while the performance of ConvLSTM is poor. Figure 8a,b shows the end–end result of CIPNet for the grid congestion index forecast with the prediction horizon of 10 min on 1 September 2020 at 12:00 and 18:00, respectively. Figure 8 shows that the CIPNet has a forecasting capability for a fine level of granularity and it also shows that it is visually intuitive for policy-makers to plan ahead. Figure 9 shows the average MSE value for all the prediction models on an entire testing dataset for the prediction horizon of 10, 20, 30, 45, and 60 min. The chart shows that the MSE of the proposed model is the lowest for the prediction horizon up to 30 min after which the performance is either lower or similar to PredNet. The plot also shows that all the three prediction models are adaptive as their MSE loss is lower than the baseline performance.

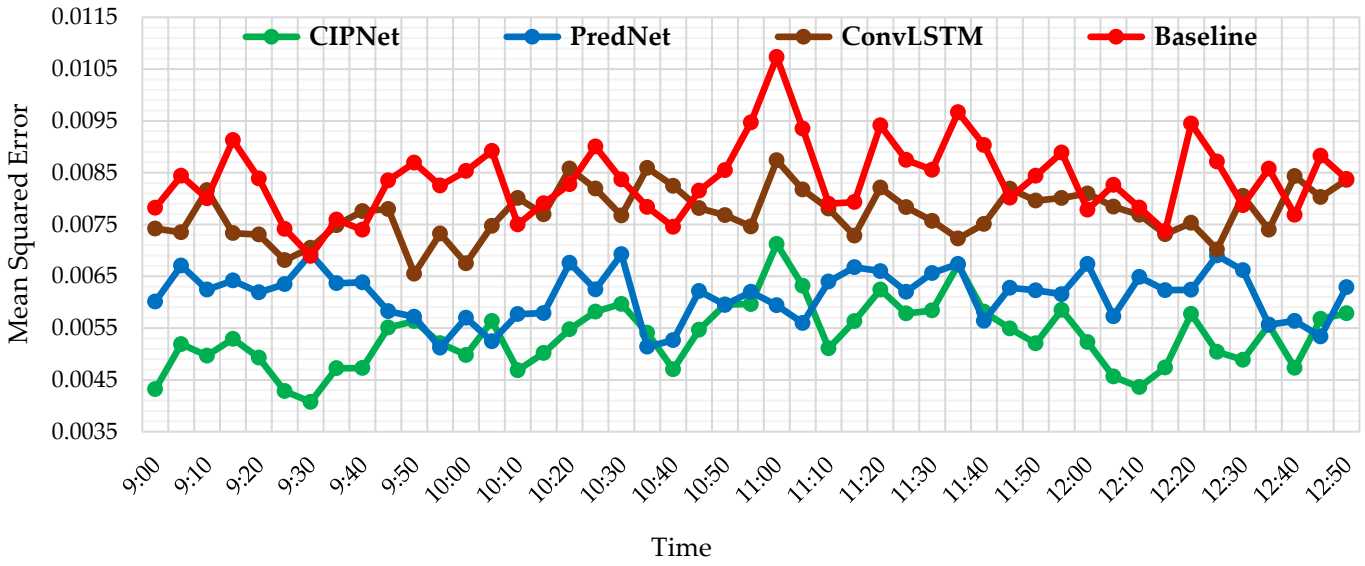
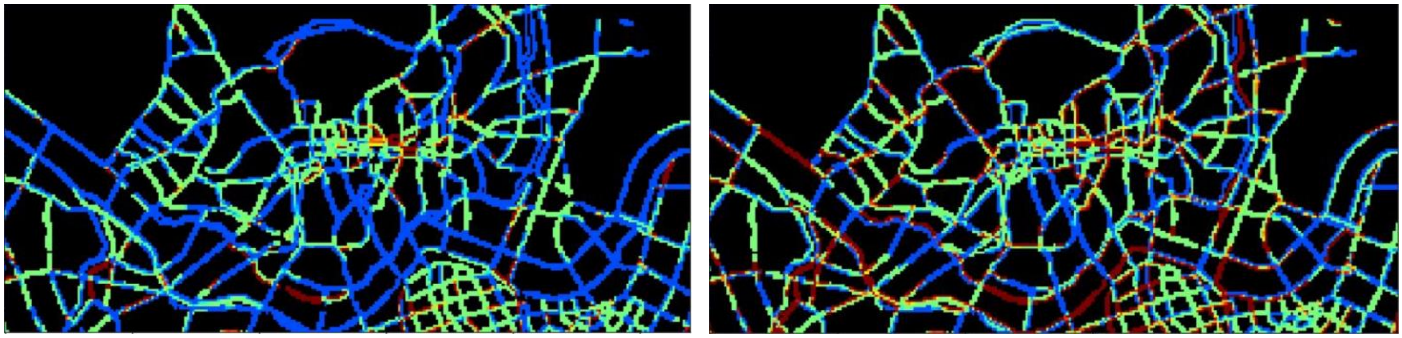


Figure 7. Mean Squared Error Comparison of the Grid Congestion Index Prediction model for a forecast horizon of 10 min on 12 October 2020.

Table 3. Mean Squared Error and Mean Absolute Error of all the models at prediction horizons of 10 and 60 min. Best result are marked in bold.

Date & Time	10 min								60 min							
	MSE				MAE				MSE				MAE			
	C.Net	P.N	C.L	B.L	C.Net	P.N	C.L	B.L	C.Net	P.N	C.L	B.L	C.Net	P.N	C.L	B.L
09-10	**0.0049**	0.0053	0.0062	0.0080	**0.0173**	0.0192	0.0207	0.0214	0.0076	**0.0073**	**0.0065**	0.0130	0.0215	**0.0198**	0.0255	0.0311
10-11	**0.0053**	0.0061	0.0069	0.0083	**0.0182**	0.0190	0.0206	0.0215	0.0089	**0.0077**	0.0097	0.0144	0.0221	**0.0210**	0.0270	0.0355
11-12	**0.0059**	0.0062	0.0063	0.0089	**0.0195**	0.0205	0.0209	0.0229	**0.0082**	0.0084	0.0101	0.0135	0.0271	**0.0239**	0.0256	0.0299
12-13	**0.0051**	0.0065	0.0072	0.0083	**0.0130**	0.0175	0.0210	0.0221	**0.0079**	0.0080	0.0088	0.0132	**0.0192**	0.0222	0.0266	0.0278
09-10	**0.0045**	0.0055	0.0051	0.0088	**0.0189**	0.0194	0.0195	0.0243	0.0077	**0.0065**	0.0091	0.0213	**0.0187**	0.0212	0.0256	0.0310
10-11	**0.0047**	0.0046	0.0047	0.0095	0.0170	**0.0165**	0.0206	0.0212	0.0074	**0.0064**	0.0094	0.0221	**0.0198**	**0.0198**	0.0282	0.0323
11-12	0.0058	**0.0053**	0.0075	0.0078	0.0197	**0.0187**	0.0231	0.0250	**0.0081**	0.0084	0.0099	0.0217	0.0210	**0.0199**	0.0211	0.0356
12-13	**0.0050**	0.0057	0.0061	0.0072	**0.0176**	0.0192	0.0218	0.0212	**0.0065**	0.0075	0.0074	0.0193	**0.0201**	**0.0210**	0.0210	0.0289
09-10	**0.0058**	0.0063	0.0079	0.0081	0.0244	**0.0198**	0.0220	0.0225	0.0068	0.0070	**0.0063**	0.0227	0.0278	**0.0213**	0.0217	**0.0212**
10-11	**0.0045**	0.0052	0.0052	0.0077	**0.0165**	0.0195	0.0214	0.0208	0.0066	**0.0065**	0.0087	0.0229	**0.0222**	0.0227	0.0229	0.0231
11-12	**0.0048**	0.0055	0.0057	0.0080	**0.0172**	0.0189	0.0211	0.0216	0.0072	**0.0069**	0.0089	0.0210	0.0176	**0.0165**	0.0278	0.0278
12-13	**0.0050**	0.0059	0.0062	0.0083	**0.0177**	0.0186	0.0192	0.0209	**0.0076**	**0.0080**	0.0107	0.0224	0.0198	**0.0177**	0.0223	0.0288
Avg.	**0.0051**	0.0057	0.0063	0.0082	**0.0181**	0.0189	0.0210	0.0221	0.0075	**0.0074**	0.0088	0.0189	0.0214	**0.0206**	0.0246	0.0294

C.Net: CIPNet, P.N: PredNet, C.L: ConvLSTM, B.L: Baseline.



(**a**) 12:00 (**b**) 18:00

Figure 8. Grid Congestion Index Prediction on 1 September 2020 for a prediction horizons of 10 min. (**a**) Prediction result at 12:00. (**b**) Prediction result at 18:00 h.

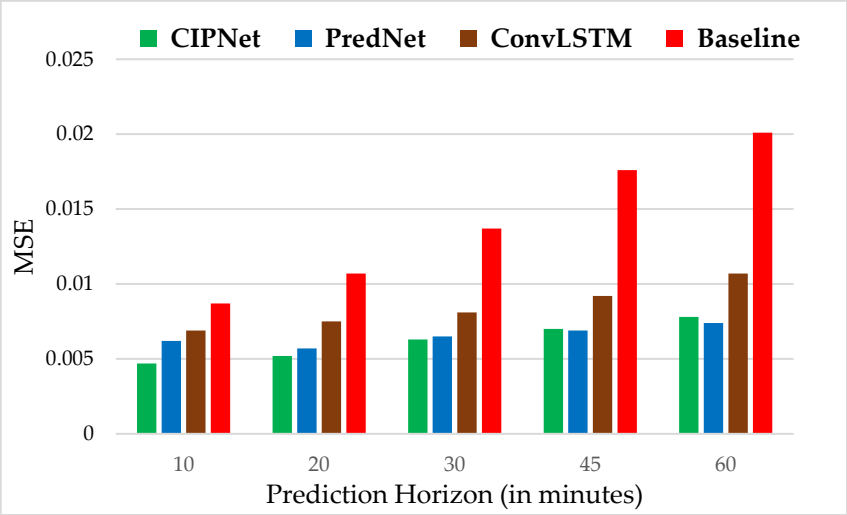


Figure 9. Average MSE for all the prediction model for a prediction horizons of 10, 20, 30, 45, and 60 min.

4.4. Computaional Complexity

The computational complexity of the algorithm depends solely on its logical structure and input size. Table 4 shows the time complexity of the TCPIP algorithm for different input image resolutions. For a one-day analysis, the TCPIP algorithm takes 288 images, or 12 images per hour. As shown in Table 4, image resolution number (I.N) 7, with the image resolution of 1 × 1, requires 12.98 milliseconds (ms) to process a single image. In other words, with virtually no input, the logical and conditional instructions of the TCPIP algorithm require 12.98 ms to process a single loop (i.e., base complexity). Moreover, with the increase of input resolution, the algorithm requires an additional time for processing, the value of which depends on the image size. For small image resolutions as in I.N 5 and I.N 6, the algorithm requires infinitesimal additional processing time. However, with further increase in input resolution as in I.N 1 to 4, there is a significant increase in additional computation time. For the 1366 × 694 resolution image used in this study, the computation time increases by about 4.4 ms per image and 1264.3 ms per daily analysis compared to the base time complexity. The TCPIP algorithm takes about 5 seconds (s) to generate the congestion pattern for a single day of data. Once the algorithm has generated the daily analysis, the result can be saved and reused in future pattern generation. Initially, the time complexity of the algorithm is high as it has to analyze a large number of days, but once the initial analysis is completed. The model takes only about 5 s to generate a

new pattern, and therefore we can say that the TCPIP algorithm is suitable for a real-world application.

Table 4. TCPIP Model Comparison based on Computation Time.

I.N	Image Resolution	Computation Time per Image (in Millisecond)	Computation Time per Day (in Millisecond)
1.	1366 × 694	17.37	5003.7
2.	1024 × 512	16.23	4673.9
3.	512 × 256	14.09	4059.6
4.	256 × 128	13.30	3829.5
5.	128 × 64	13.08	3767.6
6.	64 × 32	13.01	3747.2
7.	1 × 1	12.98	3739.4

I.N: Image Resolution Number.

For deep neural network algorithms, the model complexity depends on the number of trainable parameters and connection types between layers. The model with convolution layers requires few trainable parameters due to local connectivity and weight sharing features compared to layers with full-connection. Table 5 shows the comparison of model complexity in terms of trainable parameters. As shown in Table 5, the trainable parameter of CIPNet for the input resolution of (12, 128, 256, 1 is 6.1 million, which is approximately four times lower compared to PredNet. Table 6 shows the performance comparison of the CIPNet model in terms of computation time for different prediction horizons. After training, the CIPNet took 29 ms to predict a sample of grid congestion index. Therefore, the proposed CIPNet is the most efficient and best suited for real-world applications. Table 6 shows that the proposed CIPNet architecture requires the least training time for all three prediction horizons, although the number of epochs at which the model reaches saturation is large. It is about 3 and 15 times faster than PredNet and ConvLSTM, respectively, in terms of learning capability. In Table 6, it appears that the training time required for model saturation decreases with the prediction horizon, but this is not the case. Since, the same dataset is used to generate training samples (i.e., input sample (X) and output labels (Y)) for all prediction horizons, as the prediction horizon increases, the number of training samples decreases, and so does the training time.

Table 5. Prediction Model Comparison based on the number of Trainable Model Parameters.

Models	Description (No. of Layers)	Input Parameters	Trainable Model Parameters
CIPNet	Convolutional Autoencoder (17)	(12, 128, 256, 1)	6,102,113
PredNet [32]	Convolutional Recurrent Autoencoder (22)	(12, 128, 256, 1)	28,440,185
ConvLSTM [44]	ConvLSTM (6)	(12, 128, 256, 1)	264,160

Table 6. Prediction Model Performance Comparison based on Training Time.

Prediction Horizon	Model Variables	CIPNet	PredNet [32]	ConvLSTM [44]
10 min	Epochs Training Time (s)	40 2760	27 8046	23 41,055
30 min	Epochs Training Time (s)	39 2535	25 6971	25 41,827
60 min	Epochs Training Time (s)	41 2598	23 6414	22 32,792

All the mentioned models were trained on a real-world traffic image (Seoul) captured from the TOPIS website containing congestion levels. The models were implemented on an Ubuntu 18.04.4 machine with NVIDIA TITAN Xp Graphics cards. The performance of the TCPIP algorithm does not depend on the graphics card. The predictive models were trained using an Adaptive Moment Estimation (ADAM) optimizer, with a learning rate of 1e-3, a learning decay rate of 0.9, and a variable moving average of 0.999, and a batch size of 16 for CIPNet and 2 for ConvLSTM and PredNet. We trained all the prediction models based on the mean square error loss function. The prediction models were developed on Python 3.8 platforms using the Tensor flow and Keras deep learning libraries and the OpenCV library for an image processing task.

4.5. Discussion

One of the challenges for transportation decision-makers is to locate the congested road links, routes, or zones in the transportation network that have negative impacts on the overall network and develop strategies for sustainable urban mobility, especially considering the limited budget and resources [62]. The congestion pattern analysis results from Section 4.2 and the grid congestion-index prediction results from Section 4.3 are efficient algorithms with low computational complexity, as discussed in Section 4.4. Based on the paper finding and congestion demand management tools, policy makers, transportation experts or researchers can efficiently negate the congestion problem. In this section, we present some practical aspects of decision making using congestion pricing and selective access. Congestion pricing is a popular demand management tool that determines the price charged to consumers for the actual congestion costs caused by using the network [63]. The pricing strategy is based on either static or dynamic traffic flow and can be link-based, path-based, zone-based, etc.

For the practical aspects of the research, including the view of the decision maker, in this subsection, we select and analyze two high density zones shown in Figure 5e, R1 and R2, represented by white rectangular boxes, for the pattern analysis result. Here, the zone 'R1' is the center Seoul Region, while the zone 'R2' is a small region from Gangnam area in Seoul City. The stochastic congestion map in Figure 5 shows that the likelihood of congestion in both zones is very low until 09:00, with the slight congestion in the afternoon until 15:00, which becomes stronger between 15:00 and 21:00 and has a very low probability after that. For the other parts, except for the region inside the boxes, the likelihood is high only between 18:00 and 21:00. In the morning hours until 09:00, the demand management procedures such as traffic diversion or the selective entry need to be applied only on some roads on the outskirt of the city. From 09:00 to 18:00, strategies such as traffic diversion or dynamic congestion pricing can be an effective method. From 18:00 to 21:00, almost all road networks within the boxes are congested. The demand management measures such as the alternative routing are not possible in this scenario. However, by applying strategies such as congestion pricing for small vehicles and banning for heavy vehicles from entering the zone, the policy maker can prevent the situation from worsening. Similarly, for the same timeline from 18:00 to 21:00, the policy makers can choose for road-based congestion pricing for the region outside the boxes. The traffic redirecting strategy helps to distribute congestion from highly congested roads to congestion-free roads. Congestion pricing discourages travelers from taking private trips or making the unwanted trips through congestion zones, and encourages them to use cheaper and more efficient modes of transportation such as carpooling, public transportation, etc.

Based on the real-time grid congestion-index prediction result, policy makers can plan and apply urban mobility measures more effectively than the hourly stochastic congestion results. Figure 8a,b show the prediction result at noon and 6 p.m., on 1 September 2020, respectively. At noon, the grids mostly have congestion index values below 0.5, and very few grids have congestion index values above 0.8. Based on the result, policy makers can target the grids that are approaching a jammed condition and take various measures, such as imposing vehicle restrictions on grids with a high congestion index value, diverting

vehicles to grids with a lower congestion index, or apply dynamic congestion pricing method based on the future value of the grid congestion index value. All demand management methods such as vehicle restrictions, diversions, and congestion pricing can be applied when most girds have a low congestion index. However, in a situation where the majority of girds have high congestion index, as shown in Figure 8b, only dynamic congestion pricing can be an effective method to ensure sustainable mobility without inducing congestion due to diversions. Moreover, short to long term congestion prediction helps traffic management agencies in tactical planning for some special events. Predicting the congestion condition of roads that have reoccurring congestion value of 0.7 or more is not important whereas predicting the congestion for the network with a probability of about 0.5 is of great importance.

5. Conclusions

In this article, we first propose an inexpensive and general approach for data collection using Selenium from an open-source online TOPIS website and create two datasets, (i) city road network with congestion levels only and (ii) 5×5 pixel grid-based traffic congestion index. Then we developed two algorithms, (i) traffic congestion pattern analysis based on image processing, TCPIP, which performs simple arithmetic operations on the historical traffic images to generate the city-wide reoccurring congestion road pattern, as in Section 3.2. Moreover, (ii) grid traffic congestion-index prediction, CIPNet, based on the Convolutional Autoencoder architecture that learns a feature representation and temporal correlations from the historical grid congestion index data to predict the future city-wide grid congestion index, as in Section 3.3. As can be seen in Section 4.2.1, the traffic congestion index of the city is abnormally high, particularly during the morning and evening peak hours, which indicates that the traffic management authority should take measures to reduce the congestion. Similarly, Section 4.2.2 presents the stochastic congestion map, which shows the likelihood of the road network is in a congested state at a given time. The traffic management authority can use the congestion pattern from Section 4.2.3 to investigate the cause of congestion on each road and solve it to significantly reduce the congestion in the city. Similarly, Section 4.3 analyzes the result of the predictive models. The analysis shows that for short-term prediction, 10 to 30 min, CIPNet shows remarkable performance compared to ConvLSTM and PredNet in terms of prediction performance. For long-term prediction, 60 min, PredNet performs better as it consists of LSTM layers which have better long-term learning ability. In Section 4.4, an in-depth study on computational complexity is presented; the study shows that both models are feasible for particle applications in terms of resource utilization and computation time. Finally, Section 4.5 discusses the decision maker's point of view on reducing congestion through demand response.

This research proposes inexpensive data collection methods whose sources are readily available for the cities around the world through an open-source website such as Google Traffic Map, TOPIS, and Bing maps. Both proposed models are computationally efficient in terms of resource consumption, processing time and output performance and can be generalized to the problem of large-scale traffic analysis due to their unique capabilities, as TCPIP algorithm consists of simple arithmetic operations and CIPNet consists of convolution and pooling layers. Therefore, this work has the potential for more cost-effective and efficient methods that can provide a basis for analyzing the congestion pattern and predicting grid congestion index of any city, provided there is an online web service that provides the traffic information. In addition, both the model can also be used in other areas of traffic management, such as analyzing or predicting traffic volume, speed, or occupancy provided the input to the model is an image.

Although the research shows encouraging results in pattern analysis and predictive performance, there is still room for improvement in the computational efficiency of the model. In the current form of the two datasets, since most of the values do not contain roads, computational resources are wasted on background learning. In future research,

we will try to exclude background information from the datasets to further improve the efficiency of the models. We will also try to analyze demand management measures for congestion control through a more detailed study. Moreover, this study only focuses on using congestion data for traffic analysis. In future research, we can incorporate external factors such as weather information, real-time construction, and accident information to improve the model performance.

Author Contributions: Conceptualization, N.R. and H.K.; methodology, N.R. and H.K.; software, N.R.; validation, N.R., S.B., Y.-S.H., and H.K.; formal analysis, N.R. and P.K.; investigation, N.R.; resources, H.K. and Y.-S.H.; data curation, N.R. and S.B.; writing—original draft preparation, N.R.; writing—review and editing, N.R., S.B., P.K., Y.-S.H., and H.K.; visualization, N.R., H.K., and S.B.; supervision, H.K.; project administration, H.K.; funding acquisition, H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the data used for the traffic congestion analysis was capture from the open-source traffic web service. They provide accurate traffic information about the city. The readers can access the data by capturing an image from the web service. The link to these web service is mentioned in reference [49–57].

Acknowledgments: This work was supported by Post-Doctoral Research Program of Incheon National University 2017.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Onyeneke, C.; Eguzouwa, C.; Mutabazi, C. Modeling the Effects of Traffic Congestion on Economic Activities-Accidents, Fatalities and Casualties. *Biomed. Stat. Inform.* **2018**, *3*, 7–14. [CrossRef]
2. Wang, C.; Quddus, M.A.; Ison, S.G. Impact of traffic congestion on road accidents: A spatial analysis of the M25 motorway in England. *Accid. Anal. Prev.* **2009**, *41*, 798–808. [CrossRef] [PubMed]
3. Hao, P.; Wang, C.; Wu, G.; Boriboonsomsin, K.; Barth, M. Evaluating the environmental impact of traffic congestion based on sparse mobile crowd-sourced data. In Proceedings of the 2017 IEEE Conference on Technologies for Sustainability (SusTech), Phoenix, AZ, USA, 12–14 November 2017; pp. 1–6.
4. Ye, S. Research on Urban Road Traffic Congestion Charging Based on Sustainable Development. *Phys. Procedia* **2012**, *24*, 1567–1572. [CrossRef]
5. Ukpata, J.O.; Etika, A.A. Traffic Congestion in Major Cities of Nigeria. *Int. J. Eng. Technol.* **2012**, *2*, 1343–1438.
6. Russo, F.; Rindone, C. Planning in road evacuation: Classification of exogenous activities. *Wit Trans. Built Environ.* **2011**, *116*, 639–651.
7. Chung, Y.; Recker, W.W. A methodological approach for estimating temporal and spatial extent of delays caused by freeway accidents. *IEEE Trans. Intell. Transp. Syst.* **2012**, *13*, 1454–1461. [CrossRef]
8. Chung, Y. Identification of Critical Factors for Non-Recurrent Congestion Induced by Urban Freeway Crashes and Its Mitigating Strategies. *Sustainability* **2017**, *9*, 2331. [CrossRef]
9. Chung, Y. Assessment of non-recurrent traffic congestion caused by freeway work zones and its statistical analysis with unobserved heterogeneity. *Transp. Policy* **2011**, *18*, 587–594. [CrossRef]
10. Sun, F.; Dubey, A.; White, J. DxNAT—Deep neural networks for explaining non-recurring traffic congestion. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 2141–2150.
11. Afrin, T.; Yodo, N. A Survey of Road Traffic Congestion Measures towards a Sustainable and Resilient Transportation System. *Sustainability* **2020**, *12*, 4660. [CrossRef]
12. Croce, A.I.; Musolino, M.; Rindone, C.; Vitetta, A. Sustainable mobility and energy resources: A quantitative assessment of transport services with electrical vehicles. *Renew. Sustain.* **2019**, *113*, 109236. [CrossRef]
13. Nugmanova, A.; Arndt, W.-H.; Hossain, M.A.; Kim, J.R. Effectiveness of Ring Roads in Reducing Traffic Congestion in Cities for Long Run: Big Almaty Ring Road Case Study. *Sustainability* **2019**, *11*, 4973. [CrossRef]
14. Triantis, K.; Sarangi, S.; Teodorović, D.; Razzolini, L. Traffic congestion mitigation: Combining engineering and economic perspectives. *Transp. Plan. Technol.* **2011**, *34*, 637–645. [CrossRef]
15. Saha, R.; Tariq, M.T.; Hadi, M. Deep Learning Approach for Predictive Analytics to Support Diversion during Freeway Incidents. *Transp. Res. Rec.* **2020**, *2647*, 480–492. [CrossRef]

16. Nellore, K.; Hancke, G.P. A survey on urban traffic management system using wireless sensor networks. *Sensors* **2016**, *16*, 157. [CrossRef] [PubMed]
17. Gallo, M.; Marinelli, M. Sustainable Mobility: A Review of Possible Actions and Policies. *Sustainability* **2020**, *12*, 7499. [CrossRef]
18. Feng, X.; Saito, M.; Liu, Y. Improve urban passenger transport management by rationally forecasting traffic congestion probability. *Int. J. Prod. Res.* **2016**, *54*, 3465–3474. [CrossRef]
19. Rempe, F.; Huber, G.; Bogenberger, K. Spatio-Temporal Congestion Patterns in Urban Traffic Networks. *Transp. Res. Procedia* **2016**, *15*, 513–524. [CrossRef]
20. Rahaman, M.M.; Shuvo, M.M.M.; Zaber, M.I.; Ali, A.A. Traffic Pattern Analysis from GPS Data: A Case Study of Dhaka City. In Proceedings of the 2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 16–17 March 2018; pp. 1–6.
21. Xu, L.; Yue, Y.; Li, Q. Identifying Urban Traffic Congestion Pattern from Historical Floating Car Data. *Procedia Soc. Behav. Sci.* **2013**, *96*, 2084–2095. [CrossRef]
22. Lee, H.; Hong, B.; Jeong, D.; Lee, J. An Algorithm for Identifying the Change of the Current Traffic Congestion Using Historical Traffic Congestion Patterns. *KIISE Trans. Comput. Pract.* **2015**, *21*, 19–28. [CrossRef]
23. Wen, H.; Sun, J.; Zhang, X. Study on Traffic Congestion Patterns of Large City in China Taking Beijing as an Example. *Procedia Soc. Behav. Sci.* **2014**, *138*, 482–491. [CrossRef]
24. Smith, B.L.; Demetsky, M.J. Traffic Flow Forecasting: Comparison of Modeling Approaches. *J. Transp. Eng.* **1997**, *123*, 261–266. [CrossRef]
25. Kumar, S.V.; Vanajakshi, L. Short-term Traffic Flow Prediction using Seasonal ARIMA Model with Limited Input Data. *Eur. Transp. Res. Rev.* **2015**, *7*, 21. [CrossRef]
26. Zhang, L.; Liu, Q.; Yang, W.; Wei, N.Q.; Dong, D. An Improved K-Nearest Neighbor Model for Short-Term Flow Prediction. *Procedia Soc. Behav. Sci.* **2013**, *96*, 653–662. [CrossRef]
27. Castro-Neto, M.; Jeong, Y.; Jeong, M.; Han, L. AADT Prediction using Support Vector Regression with Data-Dependent Parameters. *Expert Syst. Appl.* **2009**, *36*, 2979–2989. [CrossRef]
28. Sun, S.; Zhang, C.; Yu, G. A Bayesian Network Approach to Traffic Flow Forecasting. *IEEE Trans. Intell. Transp. Syst.* **2006**, *7*, 124–132. [CrossRef]
29. Fan, P.; Guo, J.; Zhao, H.; Wijnands, J.S.; Wang, Y. Car-Following Modeling Incorporating Driving Memory Based on Autoencoder and Long Short-Term Memory Neural Networks. *Sustainability* **2019**, *11*, 6755. [CrossRef]
30. Liu, X.; Song, L.; Liu, S.; Zhang, Y. A Review of Deep-Learning-Based Medical Image Segmentation Methods. *Sustainability* **2021**, *13*, 1224. [CrossRef]
31. Ranjan, N.; Bhandari, S.; Zhao, H.P.; Kim, H. Neural Network Learning-based Traffic Jam Predicition Technique. In Proceedings of the 2019 Fall Conference of the Institute of Electronics and Information Engineers, Gangneung, Korea, 22–23 November 2019; pp. 951–954.
32. Ranjan, N.; Bhandari, S.; Zhao, H.P.; Kim, H.; Khan, P. City-Wide Traffic Congestion Prediction Based on CNN, LSTM and Transpose CNN. *IEEE Access* **2020**, *8*, 81606–81620. [CrossRef]
33. Bhandari, S.; Kim, H.; Ranjan, N.; Zhao, H.P.; Khan, P. Optimal Cache Resource Allocation Based on Deep Neural Networks for Fog Radio Access Networks. *J. Internet Technol.* **2020**, *21*, 967–975.
34. Bhandari, S.; Ranjan, N.; Zhao, H.P.; Kim, H. Artificial Intelligence Enabled Fog Radio Access Networks: A Case Study. In Proceedings of the 2019 Fall Conference of the Institute of Electronics and Information Engineers, Gangneung, Korea, 22–23 November 2019; pp. 993–997.
35. Bhandari, S.; Ranjan, N.; Khan, P.; Kim, H.; Hong, Y.-S. Deep Learning-Based Content Caching in the Fog Access Points. *Electronics* **2021**, *10*, 512. [CrossRef]
36. Ma, X.; Tao, Z.; Wang, Y.; Yu, H.; Wang, Y. Long Short-Term Memory Neural Network for Traffic Speed Prediction using Remote Microwave Sensor Data. *Transp. Res. Part C Emerg. Technol.* **2015**, *54*, 187–197. [CrossRef]
37. Chen, Y.Y.; Lv, Y.; Li, Z.; Wang, F. Long Short-Term Memory Model for Traffic Congestion Prediction with Online open Data. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janerio, Brazil, 1–4 November 2016; pp. 132–137.
38. Ma, X.; Dai, Z.; Ma, J.; Wang, Y.; Wang, Y. Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction. *Sensors* **2017**, *17*, 818. [CrossRef] [PubMed]
39. Sun, S.; Wu, H.; Xiang, L. City-Wide Traffic Flow Forecasting Using a Deep Convolutional Neural Network. *Sensors* **2020**, *20*, 421. [CrossRef]
40. Wei, W.; Wu, H.; Ma, H. An Autoencoder and LSTM-based Flow Prediction Method. *Sensors* **2019**, *19*, 2946. [CrossRef] [PubMed]
41. Dotoli, M.; Fanti, M.P.; Meloni, C. Coordination and real time optimization of signal timing plans for urban traffic control. In Proceedings of the 2004 IEEE International Conference on Networking, Sensing and Control, Taipei, Taiwan, 21–23 March 2004; pp. 1069–1074.
42. Dotoli, M.; Fanti, M.P.; Meloni, C. Real time optimization of traffic signal control: Application to coordinated intersections. In Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics, Taipei, Taiwan, 8 October 2003; pp. 3288–3295.

43. Zhang, S.; Yao, Y.; Hu, J.; Zhao, Y.; Li, S.; Hu, J. Deep auto encoder Neural networks for short term traffic congestion prediction of Transportation Networks. *Sensors* **2019**, *19*, 2229. [CrossRef] [PubMed]
44. Liu, Y.; Zheng, H.; Feng, X.; Chen, Z. Short-Term Traffic Flow Prediction with Conv-LSTM. In Proceedings of the 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP), Nanjing, China, 11–13 October 2017; pp. 1–6.
45. Rui, L.; Zhang, Y.; Huang, H.; Qiu, X. A New Traffic Congestion Detection and Quantification Method Based on Comprehensive Fuzzy Assessment in VANET. *KSII Trans. Internet Inf. Syst.* **2018**, *12*, 41–60.
46. Heba, E.S.; Ayman, E.S. Survey of Traffic Congestion Detection using VANET. *Commun. Appl. Electron.* **2015**, *1*, 14–20.
47. Lu, X.; Varaiya, P.; Horowitz, R.; Guo, Z.; Palen, J. Estimating Traffic Speed with Single Inductive Loop Event Data. *Transp. Res. Rec. J. Transp. Res. Board* **2012**, *2308*, 157–166. [CrossRef]
48. Padmavathi, S.; Naveen, C.R.; Kumari, V.A. Vision based Vehicle Counting for Traffic Congestion Analysis during Night Time. *Indian J. Sci. Technol.* **2016**, *9*, 1–6. [CrossRef]
49. Google Maps. Available online: https://www.google.com/maps/place/Delhi,+India/@28.6471948,76.9531797,11z/data=!3m1!4b1!4m5!3m4!1s0$\times$390cfd5b347eb62d:0$\times$37205b715389640!8m2!3d28.7040592!4d77.1024902 (accessed on 4 September 2019).
50. Bing Maps. Available online: https://www.bing.com/maps/traffic (accessed on 5 September 2019).
51. Seoul Transport Operation & Information Service Center. Available online: https://topis.seoul.go.kr/prdc/openPrdcMap.do (accessed on 5 September 2019).
52. Baidu Maps. Available online: https://map.baidu.com/@13036895.494262943,4748316.384998233,11.52z/maplayer%3Dtrafficrealtime (accessed on 10 September 2019).
53. DOTD "Louisiana Department of Transportation & Development". Available online: https://www.511la.org/#:Alerts (accessed on 10 May 2020).
54. Sigalert "Los Angeles Traffic Report". Available online: https://www.sigalert.com/?lat=33.984259&lon=-118.223015&z=2 (accessed on 20 January 2020).
55. Live Traffic NSW. Available online: https://www.livetraffic.com/ (accessed on 10 May 2019).
56. 511 Wisconsin. Available online: https://511wi.gov/ (accessed on 10 May 2019).
57. I-Traffic. Available online: https://i-traffic.co.za/region/KwaZulu (accessed on 10 August 2020).
58. Ranjan, N.; Bhandari, S.; Zhao, H.P.; Kim, H. Analysis of Correlation between Regional Traffic Congestion Index and Population density. In Proceedings of the 2019 Fall Conference of the Institute of Electronics and Information Engineers, Gangneung, Korea, 22–23 November 2019; pp. 955–958.
59. Lana, I.; DelSer, J.; Velez, M.; Vlahogianni, E.I. Road Traffic Forecasting: Recent Advances and New Challenges. *IEEE Intell. Transp. Syst. Mag.* **2018**, *10*, 93–109. [CrossRef]
60. He, Z.; Zheng, L.; Chen, P.; Guan, W. Mapping to cells: A simple Method to Extract Traffic Dynamics from Probe Vehicle Data. *Comput. Aided Civ. Infrastruct. Eng.* **2017**, *32*, 252–267. [CrossRef]
61. Ban, X.J.; Chu, L.; Benouar, H. Bottleneck Identification and Calibration for Corridor Management Planning. *Transp. Res. Rec. J. Transp. Res. Board* **2008**, *1999*, 40–53. [CrossRef]
62. Mishra, S.; Kumar, A.; Golias, M.M.; Welch, T.; Taghizad, H.; Haque, K. Transportation Investment Decision Making for Medium to Large Transportation Networks. *Transp. Dev. Econ.* **2016**, *2*, 18. [CrossRef]
63. Abou-Zeid, M.; Chabini, I. Methods for Congestion Pricing in Dynamic Traffic Networks. In Proceedings of the 10th IFAC Symposium on Control in Transportation Systems, Tokyo, Japan, 4–6 August 2003; pp. 299–304.