Article

Listening to the City, Attentively: A Spatio-Temporal Attention-Boosted Autoencoder for the Short-Term Flow Prediction Problem

Stefano Fiorini *, Michele Ciavotta and Andrea Maurino

Department of Informatics, Systems and Communication, University of Milano-Bicocca, Viale Sarca, 336, 20126 Milan, Italy
* Correspondence: s.fiorini2@campus.unimib.it

Abstract: In recent years, studying and predicting mobility patterns in urban environments has become increasingly important as accurate and timely information on current and future vehicle flows can successfully increase the quality and availability of transportation services (e.g., sharing services). However, predicting the number of incoming and outgoing vehicles for different city areas is challenging due to the nonlinear spatial and temporal dependencies typical of urban mobility patterns. In this work, we propose STREED-Net, a novel autoencoder architecture featuring time-distributed convolutions, cascade hierarchical units and two distinct attention mechanisms (one spatial and one temporal) that effectively captures and exploits complex spatial and temporal patterns in mobility data for short-term flow prediction problem. The results of a thorough experimental analysis using real-life data are reported, indicating that the proposed model improves the state-of-the-art for this task.

1. Introduction

In recent years, academia and industry have devoted much time and energy to the study and creation of models to describe and predict mobility dynamics, or flow prediction in urban areas. This interest is motivated by the need to comprehend displacement dynamics, which are also rapidly changing due to alternative electric and shared public transport systems, to define effective regulatory strategies for human mobility and freight transport in the smart city [1]. This rush to create increasingly accurate predictive models is also motivated by the pursuit of enhancing the quality of services provided to citizens by both private companies, such as shared mobility companies, and public administrations. As an example, private companies offering shared vehicles can benefit from accurate models to estimate demand in order to improve vehicle relocation operations [2]. On the other hand, the public decision maker can rely on real-time flow data and deep learning models to swiftly identify risky traffic conditions [3].

The problem addressed in this work concerns the short-term flow prediction. More precisely, given a tessellation of the area of interest in squared regions, the number of vehicles entering (Inflow) and exiting (Outflow) each region is to be predicted for the next time period. This problem has inherently spatio-temporal characteristics; evidently, the vehicular flow entering (exiting) a region does not only present temporal dependencies (time of day, flow in the previous hours) but also spatial dependencies as it strongly depends on the traffic leaving (entering) adjacent areas. Formally, such considerations relate to two widely recognized properties in the study of displacement dynamics [4], namely temporal and spatial correlations. Mobility data are innately continuous time series, generally not associated with abrupt changes. This means that the displacement dynamics

in periods temporally close share similarities, and this phenomenon is all the more true when the sampling frequency increases. Similarly, since the outflow of an area constitutes the inflow of its neighbors (and vice versa), there is a manifest spatial correlation in traffic dynamics that is widely recognized and exploited in the literature. The most recent research has also shown that some spatial and temporal patterns influence forecasting more than others. This is the case for some *districts* [5], conglomerate areas featuring similar functional characteristics (e.g., residential, commercial and industrial areas), that show correlated traffic patterns and explain much of the city's traffic.

Finally, external factors also have a profound impact on the use of vehicles. For instance, it is well known in the literature that weather conditions and the days of the week (workdays vs. weekend) affect displacement dynamics, especially for lightweight transport means like bikes [4].

The literature already proposes several models to address the traffic flow prediction problem; some use convolutions [4,6,7], and others use a combination of convolutions and LSTM [8,9], just to mention the most common approaches. Nevertheless, many architectural solutions have not been fully investigated, and there is often a lack of clear indication as to which features most impact the performance of a model. For this reasons, in this paper we propose a new model, STREED-Net (Spatio Temporal REsidual Encoder-Decoder Network), the result of a careful literature analysis and experiments, which not only presents a novel architectural structure but consistently outperforms state-of-the-art methods.

Unlike existing models from the literature for this problem, implements an autoencoder architecture featuring time-distributed 2D convolutional layers [10], instead of standard convolutions. The rationale is to effectively process a sequence of frames (for spatial and time coherence) rather than focusing on each frame separately. Although time-distributed convolutions already (partially) capture temporal dependencies, our proposal provides CMU layers [11] between the decoder and encoder in order to strengthen temporal consistency without losing the advantage offered by the use of convolutions in learning spatial patterns. As a matter of fact, compared with the more traditional LSTM [12] or GRU [13], CMU is designed to handle spatial information over time using convolutional layers natively, and it also allows time dependency to be modeled without using the state-to-state transition (used instead by LSTM and GRU). Finally, in the decoder component is in charge of building the prediction from the information processed by the encoder and CMUs stage. The decoder includes two attention mechanisms, namely *Temporal Attention* and *Spatial Attention*, to identify and reinforce the information that predominantly influences the prediction.

The main contributions of this paper can be summarized as follows:

1. We propose a novel autoencoder-based architecture that outperforms the state-of-the-art for the flow prediction problem. To the best of our knowledge, STREED-Net is the first autoencoder architecture that combines the use of time-distributed convolutional blocks with residual connections, a CMUs and two different attention mechanisms. Moreover, unlike other state-of-the-art models [4,6–8], focuses only on recent time dependencies (closeness), that is it only relies on a small number of time periods preceding the one to be predicted. This results in fewer hyperparameters to tuned.
2. The impact of the most important components of the architecture on prediction is assessed through an ablation study and discussed. In particular, the study demonstrates the ability of the two different attention blocks to capture and harness important underlying temporal and spatial information.
3. Finally, this work presents a methodologically sound comparative assessment against the best models from the literature on real-life case studies. The analyses consider different error measures, the number of trainable parameters, and a complexity indicator (number of FLOPs). Results indicate that outperforms the considered state-of-the-art approaches using a relatively reduced number of parameters and FLOPs.

The rest of this paper is organized as follows. In Section 2, the literature on techniques used in flow and traffic prediction is analyzed. Section 3 defines the flow prediction

problem in urban areas, while in Section 4, the core deep learning techniques exploited in this work and the proposed framework are described in detail. In Section 5 data and results of experiments are presented and analyzed. Finally, the conclusions and recommendations for future work are discussed in Section 6.

2. Related Work

Several studies have addressed the problem of predicting vehicle flows in urban environments. This problem has been initially modeled as a time series prediction problem for each city area and approached through classical statistical methods at first, and ANN (e.g., deep learning) later. In particular, different statistical methods have been applied, including autoregressive integrated moving average (ARIMA) [14], Kalman filtering [15], and their variants, as well as other classical approaches such as Bayesian networks [16], Markov chain [17], and SVR models [18]. Other approaches have used k-means clustering, principal component analysis, and self-organizing maps to mine spatio-temporal performance trends [19]. However, classical statistics models show some weaknesses when applied to the flow prediction problem, namely they are unable to capture the spatial dependencies between the various areas because data for each region of the city are considered as independent time series, and ii) they fail to capture the nonlinear relationship between space and time, which is essential for reliable prediction. Further studies overcame these downsides by considering spatial relationships [20] and external factors (e.g., environment and weather conditions [21]) within traditional time-series prediction methods.

ANN have exploited in flow predictions for their capability of capturing the non-linear spatial and temporal relationships within data. Initial works using ANN followed two main approaches. The first one exploits variants of RNN [22] such as (i) LSTM [12] and (ii) GRU [13], whose architectures can effectively capture both the long-term pattern and short-term fluctuation of time series. The second research line applies models based on CNN to identify spatial dependencies in traffic networks, treating dynamic traffic data as a sequence of frames [23]. Noticeably, while 2D convolutions with residual units [4,6], 3D convolutions [7] and a combination of 2D and 3D convolutions [24] are widely used, we are unaware of any model incorporating time-distributed 2D convolutions or implementing an autoencoder-based architecture. Those proposals, furthermore, do not feature specific architectural elements to capture temporal patters.

Spatial and temporal dependencies are intrinsic to traffic data, making it essential to consider both aspects at the same time when predicting mobility dynamics. In this direction, deep learning-based approaches have been recently proposed, which exploit architectures able to capture spatial and temporal patterns. For this reason, the authors in [8,9,25] have combined convolution layers and LSTMs to capture both aspects: compared to models using only convolutions, they try to strengthen the model's ability to identify temporal patterns. However, unlike STREED-Net, these are not autoencoders and do not explore the possibility to employ more advanced solutions beyond LSTMs, like Multiplicative Cascade Units (CMUs). Additionally, although in [9,25] an attention mechanism is used, these are fundamentally different from our attention mechanisms. Lastly, the literature returns a single architecture that simultaneously implements an autoencoder model with inner CMU layers [11]. However, differs from [11] by proposing to use time-distributed 2D convolutions in the encoder stage and by including two attention mechanisms (temporal and spatial) in the decoder stage.

In recent years, with the development of graph convolutional networks [26], which can be used to capture the structural characteristics of the graph network, we are witnessing their use in the field of traffic prediction [27]. Those approaches assume the existence of an origin-destination matrix, which provides details about the connections between different areas. This information, however, is often not available. One of the first graph-based works is [28] where the authors propose DCRNN, it is a model that captures the characteristic space through random walks on the graphs, and the temporal feature through the encoder-decoder architecture, while in [29] they apply the temporal graph convolutional network

(T-GCN) model, which is in combination with the graph convolutional network (GCN) and gated recurrent unit (GRU). In [30] the authors propose a method of forecasting the traffic flow based on dynamic graphs: the traffic network is modeled by dynamic probability graphs. The convolution of the graph is performed on the dynamic graphs to learn the spatial features, which are then combined with the LSTM units to learn the temporal features. Finally, in [31] the authors propose a dynamic perceptual graph neural network model for the temporal and spatial hidden relationships of deep learning segments. Indeed, the proposed model learns potential relationships of temporal features and spatial features. For a comprehensive review, we refer the reader to [32,33], while a comprehensive library providing an open-source implementation of a number of models for traffic problems is presented in [34]. We provide a more detailed description of a selection of the approaches mentioned above in Section 5.

3. Problem Statement

Given a tessellation of the area of interest (henceforth referred to as *city*) in regularly-shaped regions, a set of historical observations regarding trajectories of vehicles within the city and, possibly, other spatial and non-spatial data sources for a reference time horizon T_H of H time points, the *citywide vehicle flow prediction problem* [7] is defined as the problem of minimizing the prediction error for vehicle *Inflow* and *Outflow* at time t' that is the first time point after T_H.

In the literature, there are several definitions of location/region with different granularity and different semantic meaning [35]. However, when it comes of traffic forecasts, the majority of works uses a rectangular tessellation, which maximizes the number of neighboring areas. Similarly, in this study, the geographical space of interest (city) is logically partitioned into a regular grid of size $N \times M$ oriented by longitude and latitude [4]. Each element of the grid is termed *region* and is addressable through a pair of coordinates (n, m) corresponding to the nth row and the mth column of the grid.

The term Inflow (Outflow, respectively) refers to the number of vehicles entering (leaving) a specific region in the considered time unit (Figure 1) [7]. More specifically, the Inflow (Outflow) indicates the number of pedestrians, cars, public transport and sharing vehicles entering (leaving) the region in a certain time period. As shown in Figure 1, by analyzing the movement data of the vehicles, it is possible to obtain the Inflow and Outflow matrix, which encompass the information about displacements between the areas of the city at each time t. More in detail, let $\tau_i = \{s_i^1, s_i^2, \ldots s_i^t\}$ be a trajectory where s_i^t represents the position of vehicle i at time t, and let $\mathcal{T}$ be a collection of trajectories. The Inflow (Outflow, respectively) of a region (n, m) at time t, namely $\iota_{n,m}^t$ ($\omega_{n,m}^t$) can be formally defined as in Equations (1) and (2), respectively.

$$\iota_{n,m}^t = \sum_{\tau_i \in \mathcal{T}} \phi_\iota(\tau_i, t, n, m) \tag{1}$$

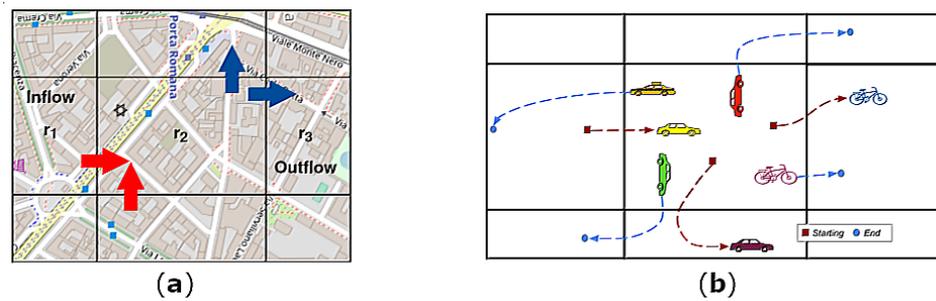$$\omega_{n,m}^t = \sum_{\tau_i \in \mathcal{T}} \phi_\omega(\tau_i, t, n, m) \tag{2}$$

where

$$\phi_\iota(\tau_i, t, n, m) = \begin{cases} 1, & \text{if } s_i^{t-1} \notin (n, m) \wedge s_i^t \in (n, m) \\ 0, & \text{otherwise.} \end{cases}$$

and

$$\phi_\omega(\tau_i, t, n, m) = \begin{cases} 1, & \text{if } s_i^{t-1} \in (n, m) \wedge s_i^t \notin (n, m) \\ 0, & \text{otherwise.} \end{cases}$$

Figure 1. Measurement of flows. (**a**) Inflow and Outflow; (**b**) Measurement of flows.

Finally, the state of the vehicular flow at time t can be represented by a tensor (also referred to as *frame* in what follows) $F_t \in \mathbb{R}^{N \times M \times C}$, where C indicates the number of flow variables considered in the analysis, in this specific case $C = 2$ (Inflow/Outflow), whereas $N \times M$ is the total number of regions in the city. Then, to take into account the temporal dependence, over the time horizon T (divided into H time points), the flow representation is extended to a tensor of four dimensions $F \in \mathbb{R}^{H \times N \times M \times C}$, which represents the main input to our problem. The problem at issue then becomes predicting F_t given a *volume*, that is a sequence of past tensors $\mathcal{V} \subset F$. It is worth noting that the resulting problem shows several similarities with the frame prediction problem [8] since the tensor F can be seen as a four dimensional volume composed of H consecutive images, each of which featuring C channels.

4. STREED-Net

STREED-Net, as shown in Figure 2, is a *Autoencoder* deep learning model that combines time-distributed convolutions and CMU with two different types of *Attentions* (spatial and temporal). This section presents STREED-Net, detailing its components and relations, prefacing it with a brief introduction to the main underpinning concepts, namely the autoencoder architecture and the attention mechanism.

Autoencoder architecture. Given a set of unlabeled training examples $\{x^1, x^2, x^3, ...\}$, where $x^i \in \mathbb{R}^n$, an autoencoder neural network is an unsupervised learning algorithm that applies backpropagation setting the target values to be equal to the inputs $y(i) = x(i)$. It is a neural network that is trained to learn a function $h_{W,b}(x) = \hat{x} \approx x$, where W and b are weights and biases of the ANN, respectively. In other words, an autoencoder is a learned approximation of the identity function, so as to output $\hat{x}$ that is as much as possible similar to x. The overall network can be decomposed into two parts: an encoder function $h = f(x)$, which maps the input vector space onto an internal representation, and a decoder that transforms it back, that is $\hat{x} = g(h)$. This type of architecture has been applied successfully to different difficult tasks, including traffic prediction [11].

Attention mechanism. In DNN Attention Mechanism helps focus on important features of the input, shadowing the others. This paradigm is inspired by the human neuro-visual system, which quickly scans images and identifies sub-areas of interest, optimizing the usage of the limited attention resources [36]. Similarly, the attention mechanism in DNN determines and stresses on the most informative features in the input data that are likely to be most valuable to the current activity.

Recently, attention has been widely applied to different areas of deep learning, such as natural language processing [37], image recognition [38], image captioning [39], image generation [40] and traffic prediction [41].
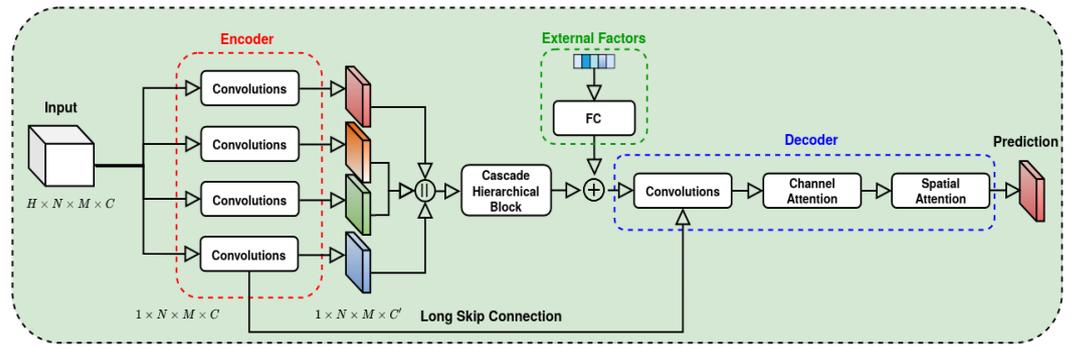
Figure 2. STREED-Net Architecture.

4.1. Encoder

The encoder structure depicted in Figure 3 is the first block of the STREED-Net architecture.

It is composed of an initial convolutional layer, a series of *residual units*, and a final convolution layer. Unlike similar approaches (e.g., STAR [6]), the proposed encoder structure introduces three novel aspects: (i) each layer is time-distributed, meaning that the model learns from a sequence of frames (for time coherence) instead of focusing on each frame singularly. (ii) it applies further convolutions after the residual unit, so that to reduce the frame size and (iii) it applies a BN after each convolution to avoid gradient disappear/explode problems and achieve faster and more efficient reported optimization [42,43].
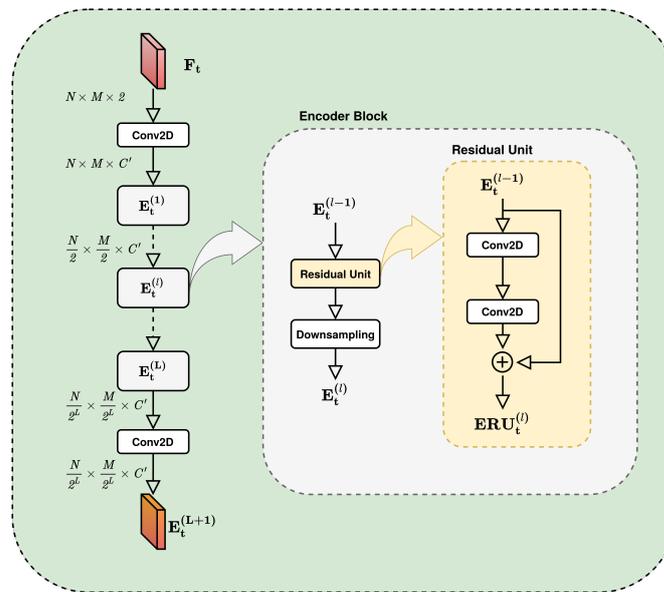

Figure 3. Encoder.

Unlike other works from the literature, where the distant temporal information is also used (from the previous day and previous week), the encoder takes as input a four-dimensional tensor $F \in \mathbb{R}^{H \times N \times M \times 2}$. This tensor is a sequence of consecutive three-dimensional frames conveying flow information of nearby periods (with regard to the prediction time t'). Such a tensor (also referred to as *closeness* in the literature [7]) is obtained by selecting p points preceding the prediction time t', i.e., the sequence $[F_{t'-p}, F_{t'-(p+1)}, ..., F_{t'-1}]$. In this way, STREED-Net can focus on the most recent dynamics only. Each frame in F is processed by the convolution layer to extrapolate spatial information. It is worth noting that in Figure 2, the encoder is represented by a collection of identical blocks in parallel execution on the input frames instead of (as in reality) a single convolution applied sequentially. Such a representation is used to highlight that a time-distributed layer is trained by taking into account all input frames simultaneously. The use

of this approach leads the model to identify temporal (that is, inter-frame) dynamics, rather than looking only to spatial dependencies within each frame.

Each convolutional layer is followed by a ReLU activation function and a *BN* layer. Formally, we have:

$$E_t^{(0)} = BN(ReLU(W_e^{(0)} * F_t + b_e^{(0)}))$$

(3)

where $E_t^{(0)}$ corresponds to the output of the first convolutional layer, F_t is one of p frames in input to the model and $*$ the convolution operator. $W_e^{(0)}$ and $b_e^{(0)}$ are the weights and biases of the respective convolutional operation. Next, **L** encoder blocks (see Figure 3) are placed. Each of these blocks is composed of a residual unit followed by a downsampling layer:

$$E_t^{(l)} = Downsampling(ResUnit(E_t^{(l-1)}))$$

(4)

where $E_t^{(l-1)}$ and $E_t^{(l)}$ correspond respectively to the input and output of the encoder block; l takes values in $\{1, ..., \mathbf{L}\}$. The residual units have been implemented as a sequence of two convolutional layers whose output is eventually summed to the block input. Mathematically, in STREED-Net the residual unit are defined as follows:

$$c_1 = BN(ReLU(W_1^{(l)} * E_t^{(l-1)} + b_1^{(l)}))$$

(5)

$$c_2 = BN(ReLU(W_2^{(l)} * c_1 + b_2^{(l)}))$$

(6)

$$ERU_t^{(l)} = E_t^{(l-1)} + c_2$$

(7)

where $E_t^{(l-1)}$ is the residual unit input and $ERU_t^{(l)}$ (*Encoder Residual Unit*) is used to indicate the result of $ResUnit(E_t^{(l-1)})$. $*$ is the convolution operator, $W_1^{(l)}$, $W_2^{(l)}$ and $b_1^{(l)}$, $b_2^{(l)}$ are the weights and biases of the respective convolutional operations.

For what concerns the *Downsampling*, it has been implemented as:

$$E_t^{(l)} = BN(ReLU(W_{ds}^{(l)} * ERU_t^{(l)} + b_{ds}^{(l)}))$$

(8)

where $W_{ds}^{(l)}, b_{ds}^{(l)}$ and $*$ indicate a convolutional layer with *kernel size* and *stride* parameters set to halve the height and width of the input frame.

The rationale behind the design of this architecture is threefold: (*i*) a deep structure is needed for the model to grasp dependencies not only among neighboring regions but also among distant areas; (*ii*) *Deep* networks are difficult to train as they present both the problem of the explosion or disappearance of the gradient and a greater tendency to overfitting due to the large number of parameters. To try to avoid these obstacles and to make the training model more efficient, we introduced residual units. Finally, (*iii*) the downsampling layers were introduced to ensure translational equivariance [44].

Finally, the encoder structure ends with a closing convolution-ReLU-BN sequence, which has as its main objective to reduce the number of *feature maps*. In this way, the next architectural component (i.e., the Cascading Hierarchical Block) will receive and process a smaller input, reducing the computational cost of the CMU array. The encoder output is:

$$E_t^{(L+1)} = BN(ReLU(W_e^{(L)} * E_t^{(L)} + b_e^{(L)}))$$

(9)

The output of the encoder is a tensor $E^{(L+1)} \in \mathbb{R}^{H \times N/2^{\mathbf{L}} \times M/2^{\mathbf{L}} \times C'}$, where C' is the number of *feature maps* generated by the last convolution of the encoder.

4.2. Cascading Hierarchical Block

A connection section between the encoder and the decoder is provided to handle the temporal relationships among the frames. Unlike what is proposed in other works that combine the use of CNN with the use of RNN such as LSTM [45], STREED-Net

implements a Cascading Hierarchical Block with CMU (CMU) [11], which computes the hidden representation of the current state directly using the input frames of both previous and current time steps, rather than what happens in recurrent networks that model the temporal dependency by a transition from the previous state to the current state. This solution is designed to explicitly model the dependency between different time points by conditioning the current state on the previous state, improving the model accuracy; incidentally, it also reduces training times.

The fundamental constituent of CMU architecture is the MU [46], which is a non-recurrent convolutional structure whose neuron connectivity, except for the lack of residual connections, is quite similar to that of LSTM [47]; the output, however, only depends on the single input frame h. Formally, MU is defined by the following equation set:

$$g_1 = \sigma(W_1 * h + b_1) \tag{10}$$

$$g_2 = \sigma(W_2 * h + b_2) \tag{11}$$

$$g_3 = \sigma(W_3 * h + b_3) \tag{12}$$

$$u = tanh(W_4 * h + b_4) \tag{13}$$

$$MU(h; W) = g_1 \odot tanh(g_2 \odot h + g_3 \odot u) \tag{14}$$

where σ is the sigmoid activation function, $*$ the convolution operator and $\odot$ the element-wise multiplication operator. $W_1 \sim W_4$ and $b_1 \sim b_4$ are the weights and biases of the respective convolutional gates and W denotes all MU parameters.

CMU incorporates three MU. Unlike MU, CMU accepts two consecutive frames as input to model explicitly the temporal dependencies between them. The more recent frame in time is inputted to a MU to capture the spatial information of the current representation. The older frame is instead processed by two MU in sequence to overcome the time gap. The partial outputs are then added together and finally, thanks to two gated structures containing convolutions along with non-linear activation functions, the output of the CMU (X_t^{l+1}) is generated. CMU is described by the following equations:

$$h_1 = MU(MU(E_{(t-1)}^l; W_1); W_1) \tag{15}$$

$$h_2 = MU(E_t^l; W_2) \tag{16}$$

$$h = h_1 + h_2 \tag{17}$$

$$o = \sigma(W_o * h + b_o) \tag{18}$$

$$X_t^{l+1} = o \odot tanh(W_h * h + b_h) \tag{19}$$

where W_1 and W_2 are the parameters of the MU in the left branch and of the MU in the right branch respectively, W_o, W_h, b_o and b_h are the weights and biases of the corresponding convolutional gates. The cascading hierarchical block uses CMUs to process all frames at the same time (see Figure 4):

$$X_{cmu} = CascadeCMU(E^{(L+1)}) \tag{20}$$

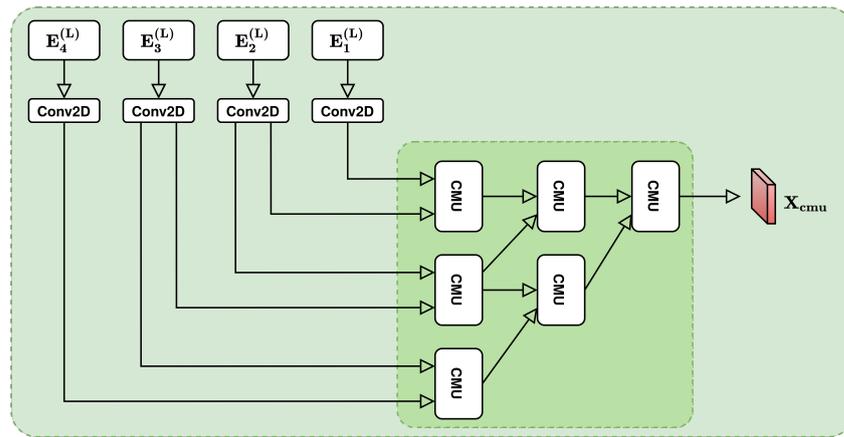where $X_{cmu} \in \mathbb{R}^{N \times M \times C'}$.

Figure 4. Cascading Hierarchical Block.

4.3. External Factors

As mentioned in Section 1, many complex external factors, such as the day of the week, holidays, and weather conditions, affect displacement dynamics. For this reason, following similar approaches from the literature [4,6], features a specific input branch to integrate external information. The input is a one-dimensional vector that contains information that refers to prediction time t'.

Through the use of two fully connected overlapping layers, this information is conveyed, encoded, into the mainstream of the network. The first level is used to embed each sub-factor, while the second reshapes the external factors embedding space to match the size of the *CMU output vector*.

4.4. Decoder

The decoder is the last component of and its task is to generate the flow prediction starting from the latent representation that corresponds to the output of the cascading hierarchical block.

As shown in Figure 5, the decoder takes as input a tensor $z = X_{cmu} + X_{ext}$, where $z \in \mathbb{R}^{N \times M \times C'}$, which is the result of the sum of the outputs of the hierarchical structure and the network dedicated to incorporate external factors. X_{ext} is added at this point of the network to allow the model to use the information extracted from the external factors during the reconstruction phase.
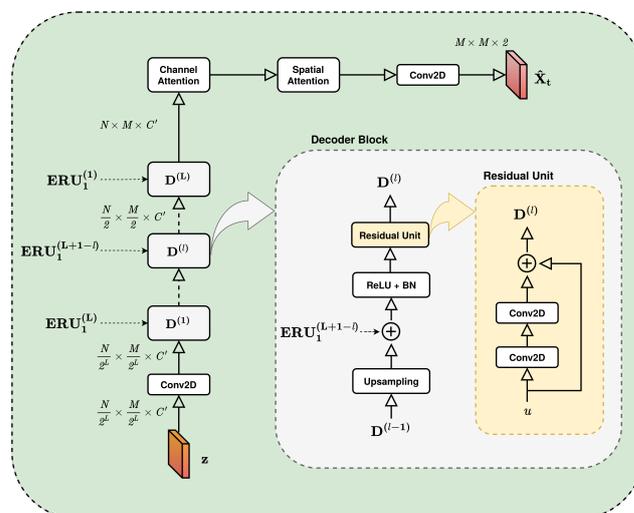

Figure 5. Decoder.

The decoder architecture features a structure that is somehow symmetrical to that of the encoder with an array of residual units preceded and followed by a convolutional layer (Equation (21)).

$$D^{(0)} = BN(ReLU(W_d^{(0)} * z + b_d^{(0)}))$$ (21)

Nevertheless, this symmetry is breached by the presence of two significant differences. The first one is the presence of a long skip connection before every residual unit. The long skip connection is used to improve the accuracy and to recover the fine-grained details from the encoder. Another benefit is a significant speed up in model convergence [48]. The impact of the long skip connection on the prediction quality is assessed and discussed in Section 5.4.

A generic decoding block $D^{(l)}$, $\forall l \in \{1 \ldots L\}$ can be formally defined as the sequential application of the following three operations:

$$sc^{(l)} = Conv2DTranspose(D^{(l-1)}) + ERU_1^{(L+1-l)}$$ (22)

$$U^{(l)} = BN(ReLU(sc^{(l)}))$$ (23)

$$D^{(l)} = ResUnit(U^{(l)})$$ (24)

where $D^{(l-1)}$ corresponds to the input block, *Conv2DTranspose* indicates the transposed convolution operation (also known as deconvolution), which doubles the height and width of the input, and $sc^{(l)}$ (skip connection) is the sum of u with $ERU_1^{(L+1-l)}$, i.e., the output of the remaining encoder unit at level $L + 1 - l$ for the most recent frame. The residual units of the decoder are structured exactly like those of the encoder.

The second difference is the presence of two attention blocks (viz. Channel and Temporal Attention) before the final convolution layer. More details are provided in the following subsections.

Channel Attention

After the convolutional stage of the decoder, a three-dimensional tensor, referred to as $D^{(L)} \in \mathbb{R}^{N \times M \times C'}$, is obtained with the channel size C'. Since the dimension of the channel also includes the temporal aspects compressed by the cascading hierarchical block, the channel attention [49] has been introduced to identify and emphasize the most valuable channels. Figure 6 depicts the inner structure of the *Channel Attention Block*. Given the input tensor, $D^{(L)}$ a channel attention map $A_c \in \mathbb{R}^{1 \times 1 \times C'}$ is created by applying attention block deduction operations on the channels. More precisely, through the operation of global average pooling and global max pooling performed simultaneously, two different feature maps (X^{max} and X^{avg}) of size $1 \times 1 \times C'$ each are spawned. The rationale behind the choice to use both pooling strategies is that the avg pooling (X^{avg}) allows for the computation of spatial statistics [50], whereas max pooling (X^{max}) provides basic translation invariance to the internal representation by observing the maximum presence of different features.
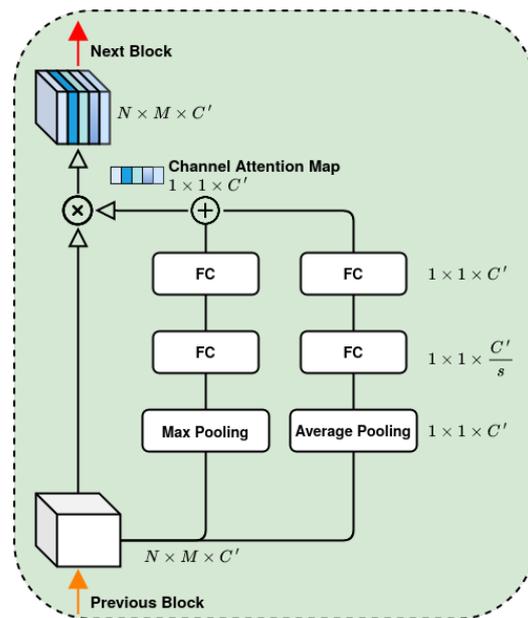
Figure 6. Channel Attention Block.

The two feature maps (X^{max} and X^{avg}) go through two Fully Connected (FC) layers that allow the model to learn (and assess) the importance of each channel. The first layer performs a dimensionality reduction, downsizing the input feature maps to $1 \times 1 \times \frac{C\prime}{s}$, based on the choice of the reduction ratio *s*; the second layer restores the feature maps to their original size. This approach has proven to increase the model efficiency without accuracy reduction [41]. Once these two steps have been completed, the two resulting feature maps are combined into a single tensor through a weighted summation as:

$$A_c = \sigma(\Lambda_1 \otimes FC(FC(X^{max})) + \Gamma_1 \otimes FC(FC(X^{avg}))) = \\ \sigma(\Lambda_1 \otimes W_2(W_0(X^{max})) + \Gamma_1 \otimes W_3(W_1(X^{avg}))) \tag{25}$$

where σ denotes the sigmoid function, $W_0 \in R^{C' \times \frac{C'}{s}}$, $W_1 \in R^{C' \times \frac{C'}{s}}$, $W_2 \in R^{\frac{C'}{s} \times C'}$ and $W_3 \in R^{\frac{C'}{s} \times C'}$ represent the weights of the FC layers, and Λ_1 and Γ_1 are two trainable tensors with the same size as the two feature maps. Λ and γ are set during the training phase and weight the relative importance of each element of the two feature maps. Finally, the process of getting channel attention can be summarized as:

$$D' = A_c \otimes D^{(L)} \tag{26}$$

where D' is the operation output and $\otimes$ denotes the element-wise multiplication.

Unlike its original version [49], the weights of Fully Connected layers are independent of each other, and we add two variables Λ and Γ to enable the network to learn how to best balance the impact of the two branches.

Spatial Attention

Cities are made up of a multitude of different functional areas. Areas have different vehicle concentrations and mobility patterns; thus, the spatial attention mechanism has the task of identifying where are located the most significant areas and scale their contribution to improve the prediction. Figure 7 presents the main internals involved in the calculation of the spatial attention map.
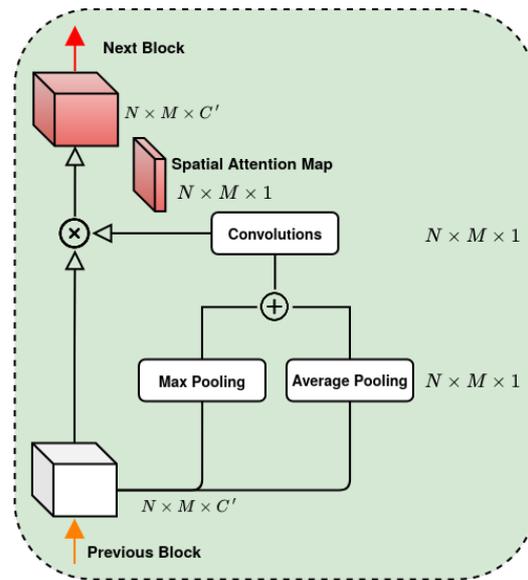
Figure 7. Spatial Attention Block

The spatial attention map $A_s \in \mathbb{R}^{N \times M \times 1}$ can be calculated by applying pooling operations along the axes of the channel to highlight informative regions [51]. Therefore, first the global average pooling (X^{avg}) and global max pooling (X^{max}) operations are applied along the channel axes and, as in the *Channel Attention Block*, two distinct feature maps of size $N \times M \times 1$ are obtained. Instead of simply concatenating these two feature maps as in [49], they are combined by a weighted sum to enhance the network's learning capability. Subsequently, the combined feature map passes through a convolution layer with a filter size of 4×4 and the sigmoid activation function is applied, as reported in Equation (27).

$$A_s = \sigma(f^{4 \times 4}(\Lambda_2 \otimes X^{max} + \Gamma_2 \otimes X^{avg})) \tag{27}$$

where σ denotes the sigmoid function and $f^{4 \times 4}$ represents a convolution operation with the filter size of 4×4. It is worth noting that the filter size depends on the size of the areas that make up the city. For the case studies addressed in this work (see Section 5), which feature rather large regions, the proposed model does not need to focus on large area clusters; therefore, the size of the filter in this work (4×4) is reduced compared to those proposed in [49].

Finally, the process of getting spatial attention can be summarized as:

$$D'' = A_s \otimes D' \tag{28}$$

5. Experimental Analysis

This section reports on an extensive experimental evaluation of the proposed model by comparing it against several reference models (see Section 5.1) using three different performance metrics and three different case studies (detailed in Section 5.2). An ablation study and the analysis of computational complexity complete the section.

5.1. Reference Methods

The proposed model is compared against the following state-of-the-art methods expressly devised to solve the citywide vehicle flow prediction problem [7]:

ST-ResNet [4]: it is one of the first deep learning approaches to traffic prediction. It predicts the flow of crowds in and out each individual region of activity. ST-ResNet uses three residual networks that model the temporal aspects of *proximity*, *period*, and *trend* separately.

MST3D [7]: this model is architecturally similar to ST-ResNet. The three time dependencies and the external factors are independently modeled and dynamically merged by assigning different weights to different branches to obtain the new forecast. Differently from ST-ResNet, MST3D learns to identify space-time correlations using 3D convolutions.

ST-3DNet [24]: the network uses two distinct branches to model the temporal components of *closeness* and *trend*, while the daily period is left out. Both branches start with a series of 3D convolutional layers used to capture the spatio-temporal dependencies among the input frames. In the *closeness* branch, the output of the last convolutional layer is linked to a sequence of residual units to further investigate the spatial dependencies between the frames of the *closeness* period. The most innovative architectural element is the *Recalibration Block*. It is a block inserted at the end of each of the two main branches to explicitly model the contribution that each region makes to the prediction.

3D-CLoST [8]: the model uses sequential 3D convolutions to capture spatio-temporal dependencies. Afterwards, a fully-connected layer encloses the information learned in a one-dimensional vector that is finally passed to an LSTM block. LSTM layers in sequence allow the model to dwell on the temporal dependencies of the input. The output of the LSTM section is added to the output produced by the network for external features. The output is multiplied by a *mask*, which allows the user to introduce domain knowledge: the mask is a matrix with null values in correspondence with the regions of the city that never have Inflow or Outflow values greater than zero (such areas can exist or not depending on the conformation of the city) while it contains 1 in all other locations.

STAR [6]: this approach aims to model temporal dependencies by extracting representative frames of *proximity*, *period* and *trend*. However, unlike other solutions, the structure of the model consists of a single branch: the frames selected for the prediction are concatenated along the axis of the channels to form the main input to the network. In STAR as well, there is a sub-network dedicated to external factors and the output it generates is immediately added to the main network input. Residual learning is used to train the deep network to derive the detailed outcome for the expected scenarios throughout the city.

PredCNN [11]: this network builds on the core idea of recurring models, where previous states in the network have more transition operations than future states. PredCNN employs an autoencoder with CMU, which proved to be a valid alternative to RNN. Unlike the models discussed above, this approach considers only the temporal component of *closeness* but has a relatively complex architecture. The key idea of PredCNN is to sequentially capture spatial and temporal dependencies using CMU blocks.

ACFM [25]: this module is composed of two progressive Convolutional Long Short-Term Memory (ConvLSTM [52]) units connected via a convolutional layer. Specifically, the first ConvLSTM unit takes the sequential flow features as input and generates a hidden state at each time-step, which is further fed into the connected convolutional layer for spatial attention map inference. The second ConvLSTM unit aims at learning the dynamic spatial-temporal representations from the attentionally weighted traffic flow features.

HA: the algorithm generates Inflow and Outflow forecasts by performing the arithmetic average of the corresponding values of the same day of the week at the same time as the instant in time to be predicted. This classical method represents a baseline in our comparative analysis, as it has not been developed specifically for the flow prediction problem.

Excluding MST3D, which has been entirely reimplemented following the indications of the original paper strictly, and PredCNN, whose original code has been completed of some missing parts, for all the other models the implementation released by the original authors has been used. The STREED-Net code, together with all the code realized for this research work, is freely available (https://github.com/UNIMIBInside/Smart-Mobility-Prediction, accessed on 7 October 2022).

We conclude this section by pointing out that, although the literature offers numerous proposals for deep learning models based on graphs with performances often superior to those of convolutional models, for the problem addressed in this paper, preliminary experiments that we conducted with graph-based models did not lead to satisfactory results.

This is due to the nature of the problem considered, whose basic assumption is to be able to observe only the inflow and outflow across all areas of the city. Such a scenario is feasible and more realistic than one in which the trajectory or origin-destination pair of all vehicles is known but makes it impossible to create graphs with nontrivial connections (i.e., not between adjacent areas) for the problem under consideration.

5.2. Case Studies

Three real-life case studies are considered for the experimental analysis, which differ in both the city considered (New York and Beijing) and the type of vehicle considered (bicycle and taxi). This choice allow the models to be assessed on usage patterns that are expected to be significantly distinct. Follows a brief description of the considered case studies:

BikeNYC. In this first case study the behavior of bicycles in New York city is analyzed. The data has been collected by the *NYC Bike system* in 2014, from 1 April to 30 September. Records from the last 10 days form the testing data set, while the rest is used for training. The length of each time period is of 1 h.

TaxiBJ. In the second case study, a fleet of cabs and the city of Beijing are considered. Data have been collected in 4 different time periods: 1 July 2013–30 October 2013, 1 March 2014–30 June 2014, 1 March 2015–30 June 2015, 1 November 2015–15 April 2016. The last four weeks are test data and the others are used for training purposes. The length of each time period is set to 30 min.

TaxiNYC. Finally, a data set containing data from a fleet of taxicabs in New York is considered. Data have been collected from 1 January 2009 to 31 December 2014. The last four weeks are test data and the others are used for training purposes. The length of each time period is set to one hour. This case study has been specifically created to perform a more thorough and sound experimental assessments than those presented in the literature.

The city of New York has been tessellated into 16×8 regions, while the city of Beijing has been divided into 32×32 areas; the discrepancy in the number of regions considered is due to the large difference in extension between the two cities. The Beijing area (16,800 km^2) is 22 times bigger than the New York area (781 km^2).

The Beijing taxi data set (TaxiBJ) and New York Bike data set (BikeNYC) are available via [4]; they are already structured to carry out the experiments reported in this work. As for the TaxiNYC dataset, available for experiments on GitHub (https://github.com/UNIMIBInside/Smart-Mobility-Prediction/tree/master/data/TaxiNYC, accessed 5 October 2022.), it has been expressly built for this work by processing and structuring data available from the *NYC government website* (https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, accessed 20 September 2021.).

A Min-Max normalization has been applied to all data sets to convert traffic values based on the scale $[-1, 1]$. Note, however, that in the experiments a denormalization is applied to the expected values to be used in the evaluation.

In the three experiments, public holidays, metadata (i.e., DayOfWeek, Weekday/Weekend) and weather have been considered as external factors. Specifically, the meteorological information reports the temperature, the wind speed, and the specific atmospheric situation (viz., sun, rain and snow).

5.3. Analysis of Results

This section presents and discusses the results of experiments performed by running STREED-Net and the models presented in Section 5.1 on the three case studies. Moreover, three different evaluation metrics are used in this study to compare the results obtained:

Root Mean Square Error (RMSE), *Mean Absolute Percentage Error* (MAPE) and *Absolute Percentage Error* (APE), which are defined as follows:

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N} \sum_{m=1}^{M} [(\hat{\iota}_{n,m} - \iota_{n,m})^2 + (\hat{\omega}_{n,m} - \omega_{n,m})^2]}{N \times M}} \tag{29}$$

$$MAPE = 100 \cdot \frac{\sum_{n=1}^{N} \sum_{m=1}^{M} \left| \frac{(\hat{\iota}_{n,m} - \iota_{n,m}) + (\hat{\omega}_{n,m} - \omega_{n,m})}{(\hat{\iota}_{n,m} - \iota_{n,m})} \right|}{N \times M} \tag{30}$$

$$APE = 100 \cdot \sum_{n=1}^{N} \sum_{m=1}^{M} \left| \frac{(\hat{\iota}_{n,m} - \iota_{n,m}) + (\hat{\omega}_{n,m} - \omega_{n,m})}{(\hat{\iota}_{n,m} - \iota_{n,m})} \right| \tag{31}$$

where $\hat{\iota}_{n,m}$ and $\hat{\omega}_{n,m}$ are, respectively, the predicted Inflow and Outflow for region (n, m) at time t' and $N \times M$ is the total number of regions in the city.

Finally, it is worth noting that to account for and reduce the inherent stochasticity of learning-based models, each experiment was repeated ten times (replicas) using a different random seed in each replica. Mean and standard deviation are reported for each metric to provide a robust indication of the overall behavior of the compared methods.

5.3.1. BikeNYC

For the BikeNYC case study STREED-Net parameters have been set as follows. The number n of input frames has been set to 4, the number L of encoding and decoding blocks has been set to 2. This decision has been dictated by the size of the grid (16×8): setting L greater than 2 (for example 3) would result in an encoder output tensor of size $4 \times 2 \times 1 \times C$, which would be too small to allow the CMU block to effectively capture the time dependencies in the section located between the encoder and decoder. After some preliminary tests, the number of convolutional filters has been set to 64 in the first layer of the encoder and in the subsequent blocks, while in the last layer it has been set equal to 16. In this way, the dimensionality of the input vector goes from $I \in \mathbb{R}^{4 \times 16 \times 8 \times 2}$ to $O \in \mathbb{R}^{4 \times 4 \times 2 \times 16}$ as the encoder output. Symmetrically, the convolutions within the decoder use 64 filters, except for the final layer which uses only 2 filters to generate the prediction of the Inflow and Outflow channels. The parameters corresponding to the dimensionality of the convolution kernel (kernel size equal to 3, batch size equal to 16 and learning rate equal to 0.0001. The number of epochs is set to 150), to the batch size and to the learning rate have been optimized with the *Bayesian optimization* technique.

As for the models from the literature, they have been arranged and trained following carefully the parameter values and indications reported in the respective publications.

As shown in Table 1, STREED-Net outperforms all other considered approaches for all evaluation metrics. In addition, the small standard deviation values are evidence of the robustness of the proposed approach. Nonetheless, it is worth observing that all learning-based approaches return similar results. We believe this is mainly due to the reduced size of the data set that does not allow the models to be adequately trained. Moreover, the tessellation used in this case study (widely used in the literature), with a small grid of dimensions (16×8), tends to level off the metrics and hinder a more precise performance assessment.

Table 1. Results obtained for the Bike NYC data set.

Model	RMSE	MAPE	APE
HA	6.56	26.46	$4.09 \cdot 10^5$
ST-ResNet	5.01 ± 0.07	21.97 ± 0.26	$3.40 \cdot 10^5 \pm 4.06 \cdot 10^5$
MST3D	4.98 ± 0.05	22.03 ± 0.47	$3.41 \cdot 10^5 \pm 7.26 \cdot 10^5$
3D-CLoST	4.90 ± 0.04	21.38 ± 0.20	$3.31 \cdot 10^5 \pm 3.12 \cdot 10^5$
PredCNN	4.81 ± 0.04	21.38 ± 0.24	$3.31 \cdot 10^5 \pm 3.76 \cdot 10^5$
ST-3DNet	4.75 ± 0.06	21.42 ± 0.28	$3.31 \cdot 10^5 \pm 4.36 \cdot 10^5$
STAR	4.73 ± 0.05	20.97 ± 0.13	$3.24 \cdot 10^5 \pm 2.02 \cdot 10^5$
ACFM	4.68 ± 0.13	20.98 ± 0.68	$3.25 \cdot 10^5 \pm 1.05 \cdot 10^5$
STREED-Net	$\mathbf{4.67 \pm 0.03}$	$\mathbf{20.85 \pm 0.15}$	$\mathbf{3.23 \cdot 10^5 \pm 2.31 \cdot 10^5}$

5.3.2. TaxiBJ

As with the experiment discussed above, for the TaxiBJ case study, the parameters of the models have been set according to the specifications given in the respective publications. In the case of STREED-Net, the hyperparameters are kept unchanged in the two experiments, except for the number L of encoding and decoding blocks, which has been increased to 3 because the grid is larger (32×32) in this experiment and more convolutional layers are needed to map the input tensor of the model. Also for this experiment, the kernel size, batch size, and learning rate parameters have been optimized with *Bayesian optimization* and the best values found were 3, 16, and 0.0001 respectively. The number of epochs has been set at 150. Notice that these values are the same used in the BikeNYC experiment.

As can be seen from Table 2, STREED-Net outperforms all other methods, in particular, reducing MAPE and APE by 2.9%, and 2.8%, respectively, compared with the second-best approach. The difference in performance in favor of the proposed model, in this experiment is more appreciable because the data set used for the training process is more significant but also because the number of regions is higher. This last consideration highlights how the proposed model seems suitable to be applied in real-world scenarios, i.e., where high model accuracy and dense tessellation are required (i.e., the city is partitioned into a large number of small regions).

Table 2. Results obtained for the Taxi Beijing data set.

Model	RMSE	MAPE	APE
HA	40.93	30.96	$6.77 \cdot 10^7$
ST-ResNet	17.56 ± 0.91	15.74 ± 0.94	$3.45 \cdot 10^7 \pm 2.05 \cdot 10^6$
MST3D	21.34 ± 0.55	22.02 ± 1.40	$4.81 \cdot 10^7 \pm 3.03 \cdot 10^5$
3D-CLoST	17.10 ± 0.23	16.22 ± 0.20	$3.55 \cdot 10^7 \pm 4.39 \cdot 10^5$
PredCNN	17.42 ± 0.12	15.69 ± 0.17	$3.43 \cdot 10^7 \pm 3.76 \cdot 10^5$
ST-3DNet	17.29 ± 0.42	15.64 ± 0.52	$3.43 \cdot 10^7 \pm 1.13 \cdot 10^6$
STAR	16.25 ± 0.40	15.40 ± 0.62	$3.38 \cdot 10^7 \pm 1.36 \cdot 10^6$
ACFM	15.67 ± 0.23	15.16 ± 0.33	$3.32 \cdot 10^7 \pm 7.25 \cdot 10^5$
STREED-Net	$\mathbf{15.61 \pm 0.11}$	$\mathbf{14.73 \pm 0.21}$	$\mathbf{3.22 \cdot 10^7 \pm 4.51 \cdot 10^5}$

5.3.3. TaxiNYC

As mentioned earlier, the TaxiNYC case study was created specifically to be able to evaluate the behavior of the proposed model in a wider set of scenarios than the literature. Consequently, in order to make a fair comparison, it was necessary to search for the best configuration of hyperparameters not only for the STREED-Net model but also for all the other approaches considered. The optimized parameters and the relative values used in the training phase are briefly summarized below for each model. Unreported configuration values have been set as for the BikeNYC case study since the two experiments share the same map size (16×8). The parameters for each model are as follows:

- **ST-ResNet*.** Optimized parameters: number of residual units, batch size and learning rate. Optimal values found: 2, 16 and 0.0001.
- **MST3D.** Optimized parameters: batch size and learning rate. Optimal values found: 16 and 0.00034.
- **PredCNN.** Optimized parameters: encoder length, decoder length, number of hidden units, batch size and learning rate. Optimal values found: 2, 3, 64, 16 and 0.0001.
- **ST-3DNet.** Optimized parameters: number of residual units, batch size and learning rate. Best values found: 5, 16 and 0.00095.
- **STAR*.** Optimized parameters: number of remaining units, batch size and learning rate. Optimal values found: 2, 16 and 0.0001.
- **3D-CLoST.** Optimized parameters: number of LSTM layers, number of hidden units in each LSTM layer, batch size, and learning rate. Optimal values found: 2, 500, 16, and 0.00076.
- **ACFM.** Optimized parameter: learning rate. Optimal value found: 0.0003.
- **STREED-Net.** Optimized parameters: kernel size, batch size, and learning rate. Optimal values found: 3, 64 and 0.00086.

It is worth noting that, preliminary experiments showed a convergence issue for the training phase of both STAR and ST-ResNet models. In particular, they were unable to converge for any combination of parameters. This behavior is due to the strong presence of *outliers* and to the concentration of the relevant Inflow and Outflow values in a few central regions of the city. To overcome this issue, Batch Normalization layers have been inserted in the structure of the two models. In particular, Batch Normalization layers have been added after each convolution present in the residual units (a possibility that has already been foreseen in the original implementations) and after the terminal convolution of the networks (an option not considered in the source code provided by the original authors). For this reason, ST-ResNet and STAR are marked with an asterisk in the Table 3, which summarizes the experimental results.

Table 3. Results obtained for the Taxi New York data set.

Model	RMSE	MAPE	APE
HA	164.31	27.19	$7.94 \cdot 10^5$
ST-ResNet*	**35.87 ± 0.60**	22.52 ± 3.43	$6.57 \cdot 10^5 \pm 1.00 \cdot 10^5$
MST3D	48.91 ± 1.98	23.98 ± 1.30	$6,98 \cdot 10^5 \pm 1.34 \cdot 10^4$
3D-CLoST	48.17 ± 3.16	22.18 ± 1.05	$6.48 \cdot 10^5 \pm 3.08 \cdot 10^4$
PredCNN	40.91 ± 0.51	25.65 ± 2.16	$7.49 \cdot 10^5 \pm 6.32 \cdot 10^4$
ST-3DNet	41.62 ± 3.44	25.75 ± 6.11	$7.52 \cdot 10^5 \pm 1.78 \cdot 10^5$
STAR*	36.44 ± 0.88	25.36 ± 5.24	$7.41 \cdot 10^5 \pm 1.53 \cdot 10^5$
ACFM	36.75 ± 0.94	**19.10 ± 1.08**	**$5.58 \cdot 10^5 \pm 2.21 \cdot 10^4$**
STREED-Net	36.22 ± 0.72	20.29 ± 1.48	$5.93 \cdot 10^5 \pm 4.31 \cdot 10^4$

As it can be seen from Table 3, STREED-Net achieved excellent results in this experiment as well, ranked as one of the best models. In particular, as far as the RMSE is concerned, the performances obtained are very close to the best one (achieved by ST-ResNet*, which is considerably different from the original ST-ResNet). As for MAPE and APE values, place our proposal ranks as the second-best approach, closely after ACFM.

5.4. Ablation Study

In this section, an ablation study conducted on STREED-Net is presented, in which variations in the input structure and in the network architecture are analyzed. The study, for reasons of space, refers only to BikeNYC case study and does not involve the full combinatorics of all possible variants of the proposed model but aims to assess the impact on performance metrics of some parameters (namely, the number of input time points *n*) and specific architectural choices (viz., long skip connection, attention blocks, and external

factors input branch), while maintaining all other conditions. More precisely, in what follows, STREED-Net is compared against the 5 different variations described below:

- **STREED-Net_N3**. Same architecture as STREED-Net, but input volumes with 3 frames ($[X_{t-3}, X_{t-2}, X_{t-1}]$).
- **STREED-Net_N5**. Same architecture as STREED-Net, but input volumes with 5 frames ($[X_{t-5}, X_{t-4}, X_{t-3}, X_{t-2}, X_{t-1}]$).
- **STREED-Net_NoLSC**. STREED-Net by removing the *long skip connection* between encoder and decoder.
- **STREED-Net_NoAtt**. STREED-Net without the *attention* blocks.
- **STREED-Net_NoExt**. STREED-Net without the external factors.

Notice that the study does not consider the variations with $n = 1$ and $n = 2$ as such values would not allow the network to capture meaningful temporal patterns between traffic flows.

Table 4 reports the results of the ablation study conducted. Each data point in the table has been obtained performing 10 times the training procedure for each model variation changing the random seed, and evaluating the resulting network on the test set. The mean and standard deviation are reported.

Table 4. Results obtained from ablation studies.

Model	RMSE	MAPE	APE
STREED-Net_N3	4.75 ± 0.04	21.18 ± 0.18	$3.28 \cdot 10^5 \pm 2.73 \cdot 10^3$
STREED-Net_N5	4.74 ± 0.03	21.03 ± 0.22	$3.26 \cdot 10^5 \pm 3.43 \cdot 10^3$
STREED-Net_NoLSC	4.84 ± 0.04	21.53 ± 0.24	$3.33 \cdot 10^5 \pm 3.71 \cdot 10^3$
STREED-Net_NoAtt	4.78 ± 0.04	20.95 ± 0.27	$3.25 \cdot 10^5 \pm 4.20 \cdot 10^3$
STREED-Net_NoExt	4.76 ± 0.04	20.99 ± 0.29	$3.26 \cdot 10^5 \pm 4.55 \cdot 10^3$
STREED-Net	**4.67 ± 0.03**	**20.85 ± 0.15**	$\mathbf{3.23 \cdot 10^5 \pm 2.31 \cdot 10^3}$

The results show that regarding the time horizon, for the BikeNYC case study, $n = 4$ allows the model to obtain better results. This means that, considering the particular setup, for the city of New York 4 hours of data allow to predict more accurately the dynamics of bicycle mobility whereas considering a greater amount of information ($n = 5$) would reduce the accuracy of the network. It is plausible to believe that considering a larger number of temporal instants would lead the network to grow in the number of parameters to be trained and thus require a larger amount of data to identify possible longer-term patterns.

From the architectural point of view, the two components attention block and long skip connection, confirm their importance in improving the performance of the proposed model, accounting for a 2.36% and 3.64% increase in RMSE, respectively. In particular, as regards the attention block, not only STREED-Net reaches lower average error values, but also the standard deviation is reduced, proving that the attention blocks are effective in helping the network single out the most meaningful information and in making the training process more stable. Finally, the experiment shows also a strong impact of the long skip connection mechanism, which, as illustrated in Section 4.4, connects the encoder to the decoder to convey fine-grained details through the network.

5.5. Number of Trainable Parameters and FLOPs

A brief analysis of the number of trainable parameters and the computational complexity (measure in number of FLOPs) of each model for the different case studies is reported in this section. For what concerns the number of trainable parameters, as shown in Table 5, STREED-Net has a generally low number compared to other models as only STAR features lesser parameters to train. Such a reduced number of parameters is due to the fact that the dimensionality of the input is reduced by the encoder downsampling mechanism. The model with the highest number of parameters is 3D-CLoST, which uses both 3D convolutions and LSTM.

Table 5. Number of trainable parameters.

Model	BikeNYC	TaxiNYC	TaxiBJ
ST-ResNet	906,272	458,304	2,696,992
MST3D	668,218	668,378	8,674,370
3D-CLoST	13,099,090	19,477,648	72,046,714
PredCNN	3,967,906	3,967,906	4,827,842
ST-3DNET	540,696	617,586	903,242
STAR	161,052	310,076	476,388
ACFM	182,065	270,581	969,893
STREED-Net	582,673	582,673	765,497

Finally, Table 6 provide the computational complexity of each model in terms of floating point operations (FLOPs) as in [53] for each case study. As can be seen from the results obtained, the model with the higher computational complexity is PredCNN, which is based on CMUs. While, 3DCLoST is the model with the shortest forward and backward times. STREED-Net, instead, has a middle-range computational complexity compared to the other models, despite its autoencoder structure, the use of attention blocks, and CMUs. This occurs because although the number of network parameters is small, the network employs high complexity operators. However, the training and execution times of STREED-Net are compatible with its applicability in full-scale real-world scenarios.

Table 6. Computational complexity (in number of FLOPs).

Model	BikeNYC	TaxiNYC	TaxiBJ
ST-ResNet	230,849,450	115,735,786	5,459,663,018
MST3D	33,042,250	33,042,570	272,483,226
3D-CLoST	29,613,094	9,601,920	338,148,804
PredCNN	1,015,468,288	1,015,468,288	9,883,813,888
ST-3DNET	171,242,496	190,130,922	1,823,295,898
STAR	40,449,706	78,231,530	928,100,922
ACFM	41,687,924	93,643,568	621,498,864
STREED-Net	130,047,738	130,047,738	1,067,063,882

6. Conclusions

Predicting vehicular flow is one of the central topics in the domain of intelligent mobility. It is a challenging task, influenced by several complex factors, such as spatio-temporal dependencies and external factors. In this study, we have developed a new deep learning architecture, based on convolutions and CMU to forecast the Inflow and Outflow in each region of the smart city. A comprehensive experimental campaign has been conducted on three different real-world case studies. The results showed that consistently outperforms state-of-the-art models in predicting dynamics in all the experiments conducted on the three performance metrics considered. This work also reports and analyzes the results from an ablation study and complexity analysis.

For future developments, the possible integration of other external factors, such as the territorial characteristics of each geographical area, should be tested. Moreover, it would be appropriate to increase the granularity of the city tessellation, as well as conduct transfer learning experiments to study the applicability of the proposed model to scenarios with a reduced amount of data available.

Author Contributions: Conceptualization, S.F. and M.C.; methodology, S.F. and M.C.; software, S.F.; validation, S.F. and M.C.; formal analysis, S.F. and M.C.; writing—original draft preparation, S.F., M.C. and A.M.; writing—review and editing, S.F. and M.C.; supervision, A.M.; project administration, A.M. All authors have read and agreed to the published version of the manuscript.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this paper are freely available and can be found at the following link: https://github.com/UNIMIBInside/Smart-Mobility-Prediction/tree/master/data (accessed on 6 October 2022).

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

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial Neural Network
CMU	Cascade Multiplicative Unit
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
HA	Historical Average
LSTM	Long Short-Term Memory
MU	Multiplicative Unit
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SVR	Support Vector Regression
BN	Batch Normalization
DNN	Deep Neural Network

References

1. Zheng, Y.; Capra, L.; Wolfson, O.; Yang, H. Urban computing: Concepts, methodologies, and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2014**, *5*, 1–55. [CrossRef]
2. Tolomei, L.; Fiorini, S.; Ciociola, A.; Vassio, L.; Giordano, D.; Mellia, M. Benefits of Relocation on E-scooter Sharing—A Data-Informed Approach. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 19–22 September 2021; pp. 3170–3175. [CrossRef]
3. Yuan, C.; Li, Y.; Huang, H.; Wang, S.; Sun, Z.; Li, Y. Using traffic flow characteristics to predict real-time conflict risk: A novel method for trajectory data analysis. *Anal. Methods Accid. Res.* **2022**, *35*, 100217. [CrossRef]
4. Zhang, J.; Zheng, Y.; Qi, D.; Li, R.; Yi, X.; Li, T. Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artif. Intell.* **2018**, *259*, 147–166. [CrossRef]
5. Liu, Y.; Lyu, C.; Khadka, A.; Zhang, W.; Liu, Z. Spatio-Temporal Ensemble Method for Car-Hailing Demand Prediction. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 1–6. [CrossRef]
6. Wang, H.; Su, H. STAR: A Concise Deep Learning Framework for Citywide Human Mobility Prediction. In Proceedings of the 2019 20th IEEE International Conference on Mobile Data Management (MDM), Hong Kong, China, 10–13 June 2019; pp. 304–309.
7. Chen, C.; Li, K.; Teo, S.G.; Chen, G.; Zou, X.; Yang, X.; Vijay, R.C.; Feng, J.; Zeng, Z. Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 893–898.
8. Fiorini, S.; Pilotti, G.; Ciavotta, M.; Maurino, A. 3D-CLoST: A CNN-LSTM Approach for Mobility Dynamics Prediction in Smart Cities. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 3180–3189.
9. Yao, H.; Tang, X.; Wei, H.; Zheng, G.; Li, Z. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 5668–5675.
10. Chowanda, A. Spatiotemporal Features Learning from Song for Emotions Recognition with Time Distributed CNN. In Proceedings of the 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI), Jakarta, Indonesia, 28 October 2021; Volume 1, pp. 407–412.
11. Xu, Z.; Wang, Y.; Long, M.; Wang, J.; Kliss, M. PredCNN: Predictive Learning with Cascade Convolutions. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; pp. 2940–2947.
12. Yu, R.; Li, Y.; Shahabi, C.; Demiryurek, U.; Liu, Y. Deep learning: A generic approach for extreme condition traffic forecasting. In Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, TX, USA, 27–29 April 2017; pp. 777–785.
13. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734.

14. Moayedi, H.Z.; Masnadi-Shirazi, M. Arima model for network traffic prediction and anomaly detection. In Proceedings of the 2008 International Symposium on Information Technology, Kuala Lumpur, Malaysia, 26–28 August 2008; Volume 4, pp. 1–6.

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

16. Sun, S.; Zhang, C.; Yu, G. A Bayesian network approach to traffic flow forecasting. *IEEE Trans. Intell. Transp. Syst.* **2006**, *7*, 124–132. [CrossRef]

17. Qi, Y.; Ishak, S. A Hidden Markov Model for short term prediction of traffic conditions on freeways. *Transp. Res. Part C Emerg. Technol.* **2014**, *43*, 95–111. [CrossRef]

18. Wu, C.H.; Ho, J.M.; Lee, D.T. Travel-time prediction with support vector regression. *IEEE Trans. Intell. Transp. Syst.* **2004**, *5*, 276–281. [CrossRef]

19. Asif, M.T.; Dauwels, J.; Goh, C.Y.; Oran, A.; Fathi, E.; Xu, M.; Dhanya, M.M.; Mitrovic, N.; Jaillet, P. Spatiotemporal patterns in large-scale traffic speed prediction. *IEEE Trans. Intell. Transp. Syst.* **2013**, *15*, 794–804. [CrossRef]

20. Tong, Y.; Chen, Y.; Zhou, Z.; Chen, L.; Wang, J.; Yang, Q.; Ye, J.; Lv, W. The Simpler the Better: A Unified Approach to Predicting Original Taxi Demands Based on Large-Scale Online Platforms. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17), Halifax, NS, Canada, 13–17 August 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1653–1662.

21. Qian, X.; Ukkusuri, S.V. Spatial variation of the urban taxi ridership using GPS data. *Appl. Geogr.* **2015**, *59*, 31–42. [CrossRef]

22. Azzouni, A.; Pujolle, G. A long short-term memory recurrent neural network framework for network traffic matrix prediction. *arXiv* **2017**, arXiv:1705.05690

23. Ma, X.; Dai, Z.; He, 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]

24. Guo, S.; Lin, Y.; Li, S.; Chen, Z.; Wan, H. Deep spatial–temporal 3D convolutional neural networks for traffic data forecasting. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3913–3926. [CrossRef]

25. Liu, L.; Zhang, R.; Peng, J.; Li, G.; Du, B.; Lin, L. Attentive crowd flow machines. In Proceedings of the 26th ACM International Conference on Multimedia, Lisbon, Portugal, 22–26 October 2018; pp. 1553–1561.

26. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907

27. Jiang, W.; Luo, J. Graph Neural Network for Traffic Forecasting: A Survey. *arXiv* **2021**, arXiv:2101.11174.

28. Li, Y.; Yu, R.; Shahabi, C.; Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv* **2017**, arXiv:1707.01926.

29. Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; Li, H. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 3848–3858. [CrossRef]

30. Peng, H.; Du, B.; Liu, M.; Liu, M.; Ji, S.; Wang, S.; Zhang, X.; He, L. Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning. *Inf. Sci.* **2021**, *578*, 401–416. [CrossRef]

31. Li, Y.; Zhao, W.; Fan, H. A Spatio-Temporal Graph Neural Network Approach for Traffic Flow Prediction. *Mathematics* **2022**, *10*, 1754. [CrossRef]

32. Lee, K.; Eo, M.; Jung, E.; Yoon, Y.; Rhee, W. Short-Term Traffic Prediction With Deep Neural Networks: A Survey. *IEEE Access* **2021**, *9*, 54739–54756. [CrossRef]

33. Yin, X.; Wu, G.; Wei, J.; Shen, Y.; Qi, H.; Yin, B. Deep Learning on Traffic Prediction: Methods, Analysis, and Future Directions. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 4927–4943. [CrossRef]

34. Wang, J.; Jiang, J.; Jiang, W.; Li, C.; Zhao, W.X. LibCity: An Open Library for Traffic Prediction. In Proceedings of the 29th International Conference on Advances in Geographic Information Systems, Beijing, China, 2–5 November 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 145–148. [CrossRef]

35. Kemp, K.; Sean, C.A.; Ola, A.; Jochen, A.; Carl, A.; Brandon, B.; David, A.B.; Barry, B.; Scott, B.; Daniel, G.B.; et al. *Encyclopedia of Geographic Information Science*; Sage: Thousand Oaks, CA, USA, 2008.

36. Ungerleider, S.K.; Ungerleider, L.G. Mechanisms of visual attention in the human cortex. *Annu. Rev. Neurosci.* **2000**, *23*, 315–341. [CrossRef] [PubMed]

37. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.

38. Zheng, H.; Fu, J.; Mei, T.; Luo, J. Learning multi-attention convolutional neural network for fine-grained image recognition. In Proceedings of the IEEE International Conference on Computer Vision, Waikoloa, HI, USA, 9–13 December 2017; pp. 5209–5217.

39. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2048–2057.

40. Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D.; Wierstra, D. Draw: A recurrent neural network for image generation. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1462–1471.

41. Liu, Y.; Liu, Z.; Lyu, C.; Ye, J. Attention-Based Deep Ensemble Net for Large-Scale Online Taxi-Hailing Demand Prediction. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 4798–4807. [CrossRef]

42. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:1502.03167.

43. Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. How Does Batch Normalization Help Optimization? In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2018; Volume 31, pp. 2483–2493.

44. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press Cambridge: Cambrisge, MA, USA, 2016; Volume 1.

45. Ranjan, N.; Bhandari, S.; Zhao, H.; Kim, H.; Khan, P. City-Wide Traffic Congestion Prediction based on CNN, LSTM and Transpose CNN. *IEEE Access* **2020**, *8*, 81606–81620. [CrossRef]

46. Kalchbrenner, N.; Oord, A.; Simonyan, K.; Danihelka, I.; Vinyals, O.; Graves, A.; Kavukcuoglu, K. Video pixel networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 1771–1779.

47. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]

48. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17), San Francisco, CA, USA, 4–9 February 2017; AAAI Press: Palo Alto, CA, USA, 2017; pp. 4278–4284.

49. Woo, S.; Park, J.; Lee, J.Y.; So Kweon, I. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.

50. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.

51. Komodakis, N.; Zagoruyko, S. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In Proceedings of the ICLR, Toulon, France, 24–26 April 2017.

52. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015.

53. Bianco, S.; Cadene, R.; Celona, L.; Napoletano, P. Benchmark Analysis of Representative Deep Neural Network Architectures. *IEEE Access* **2018**, *6*, 64270–64277. [CrossRef]