Vehicular Traffic Density Forecasting through the Eyes of Traffic Cameras; a Spatio-Temporal Machine Learning Study

Roozbeh Ketabi
University of Florida
roozbeh@ufl.edu

Mimonah Al Qathrady
University of Florida
mimonah@ufl.edu

Babak Alipour
University of Florida
babak.ap@ufl.edu

Ahmed Helmy
University of Florida
helmy@ufl.edu

ABSTRACT

Forecasting vehicular mobility and density is essential to a wide array of mobile applications, including VANETs, crowd-sourcing, participatory sensing, network provisioning, and shared transportation. Forecasting is intrinsically complex and scarcity and lack-of-scale of vehicular mobility data is adding to the challenge. In this paper, relying on traffic cameras as the main data acquisition tool and the traffic densities extracted from the images, we explore trends pertaining to density data for the purposes of temporal and spatial forecasting. We investigate the promise of deep learning by conducting a comparative analysis of conventional (seasonal) models, and multiple variants of recurrent neural models, based on 40 day-long traffic density data from 58 cameras in London. Our findings show a dramatic reduction in forecast error using deep learning, where the best seasonal model gets 0.0176 mean squared error, and our proposed neural model achieves 0.0067 (62% less error). This is 10.5% in percentage error, down from 19.3%. We also design an end-to-end multivariate architecture that forecasts all the cameras which achieves 0.0125 error (14.5% in percentage error), but is trained in half the time needed to train 58 cameras individually. Finally, to forecast locations without explicit monitoring, we build on these insights and investigate spatial relationships between cameras. We introduce a spatial forecast model similar to the multivariate model. This results in an average reconstruction error of 0.0169 when every camera is reconstructed based on only one camera, and goes down to 0.0125 when 8 cameras are used to predict others (on par with that of the multivariate model with all 58 cameras as input). Moreover, a set of 23 cameras is found that can forecast the other cameras with an error of 0.0086. These results provide great promise for prediction in future vehicular-based networks and services.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Supervised learning*; • **Applied computing** → **Forecasting**; *Transportation*; • **Networks** → *Mobile networks*;

KEYWORDS

spatiotemporal modeling, city traffic modeling, deep learning, LSTM, forecasting

ACM Reference Format:
Roozbeh Ketabi, Mimonah Al Qathrady, Babak Alipour, and Ahmed Helmy. 2019. Vehicular Traffic Density Forecasting through the Eyes of Traffic Cameras; a Spatio-Temporal Machine Learning Study. In *9th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications (DIVANet '19), November 25–29, 2019, Miami Beach, FL, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3345838.3356002

DIVANet '19, November 25–29, 2019, Miami Beach, FL, USA

ACM ISBN 978-1-4503-6907-7/19/11…$15.00
https://doi.org/10.1145/3345838.3356002

1 INTRODUCTION

The problem of forecasting (prediction) traffic density has proven to be a challenging one due to its underlying temporal dynamics and irregularities in seasonality. Spatial scale of such predictions (all across the city) and capturing the spatial relations also add to the challenge. Moreover, in terms of vehicular traffic data, although there has been a lot of efforts in the transportation community, the data is generally not publicly accessible. This lack of suitable data at an expansive scale in time and space causes difficulty in design and evaluation of predictive models. To overcome these challenges, in this work, driven by data from publicly available sources, we analyze and model the behavior of traffic from the viewpoints of traffic cameras. We also conduct a systematic set of evaluations of forecast schemes, ranging from traditional seasonal-based models (e.g., *ARIMA*) to deep learning (*LSTM*-based) forecast.

Several practical formulations of the forecasting problem are considered in this study. An important point in formulating the problem, is how far into the future to predict, denoted as the forecast horizon. Another consideration is the temporal granularity. Going too fine may result in a lot of noisy data, limiting the predictability. Going too coarse may result in quantitatively good performance, but at the cost of losing potentially useful information. In this work, we choose 30 minutes as the forecast horizon (it gives enough time for traffic to change across the city, and is in line with previous studies [6, 18]). The 'history window' considered by the model to make a prediction is usually a parameter selected through model optimization and training, or a hyper-parameter (selected through grid search or advanced algorithms for fine-tuning).

Traffic ***density*** is a particularly useful measure as it can be used (directly or indirectly) for many applications. Instances of such applications include urban planning (e.g., detecting bottlenecks in road networks), VANET (e.g., higher connectivity or infrastructure placement in more dense areas), and participatory sensing (e.g., directing sensors to less populated areas to maintain coverage fidelity). We build our analytic framework on density extracts from millions of images by dozens of traffic cameras. Thakur et al. [18] introduced a dataset of vehicular densities spanning several cities

around the world for a period of over a month, based on public traffic camera images. We build on these image density extracts by creating a data-frame with {camera_id, time, location, and vehicular density}. The density estimate is based on the foreground pixel density and background subtraction algorithm [17]. The number of background pixels naturally changes per camera. But, for the same camera, higher foreground pixel density results in higher traffic, which facilitates traffic analysis at camera locations from a timeseries point of view. To compare various models, density value for forecasting goals has been min-max scaled to the range [0,1] globally. Hence, two different cameras with the same density, may not necessarily have the same number of cars nor occupation.

For the purposes of this study, we focus on data from the city of London. First, we clean the data, then apply a variety of forecast models. We sanitize and setup the dataset, by identifying and selecting 58 cameras (out of ≈180) that have contiguous data for 40 days from 9:30 am to 6:30 pm; the first 75% of which is training data, and the rest is used for evaluation and testing. The model, at a high level, maps the density value of one or multiple timeseries into the next expected value. In that sense, it is referred to as *nowcasting* in some studies [23]. To evaluate the models, first we establish a baseline using seasonal timeseries models, and then improve upon it by modeling the problem using ideas from machine learning and proposing customized deep learning architectures. Then, we investigate the power of deep learning in predicting (or reconstructing) the value of a location (camera) by using the history data only from other cameras (i.e., the model will not see the history of the predicted location/camera). To our knowledge, this paper is one of the first (if not the first) to study this problem.

In short, the questions investigated can be summarized as follows: 1- How predictable is the traffic density (from the viewpoint of the cameras)? 2- Is it possible to accurately model the city traffic in one end-to-end model? 3- Is spatial locality expressed through nearby cameras informative for better forecasts? 4- Is it possible to reduce the number of observing cameras and use the spatial relations of the cameras to still forecast the other (missing) cameras?

After discussing the related work in Section 2, the rest of the paper is organized as follows: First, in Section 3, the dataset is discussed in details and the models are described starting with seasonal forecasting methods. Then a recurrent neural model is presented as an alternative to the seasonal models. Next, we attempt to forecast the city traffic for all the cameras at once in a multivariate model. Building on insights from the previous models, we then explore a hybrid approach where nearby cameras are considered together to improve the forecast. Finally, in Section 4, the problem of spatial forecasting (or interpolation) is explored in which a subset of the cameras are missing from the training entirely. We conclude with a discussion and remarks in Sections 5 and 6.

2 RELATED WORK

We discuss the related work in two parts. First, we focus on the dataset, from conception to analysis. Second, we explore previous attempts at forecasting especially in traffic domain, reviewing recent advances and applications of machine and deep learning.

The framework to acquire traffic camera images, processing densities and establishing the dataset has been introduced by Thakur et al. [18]. Estimates of traffic density are extracted using a background subtraction algorithm [17] and studied from spatial and temporal aspects including observance of self similarity [19] and causality [6]. In particular, Fay et al. [6] have studied the causality of data from traffic cameras on other cameras using Granger Networks and have spatiotemporally modeled the city's traffic by the means of a Gaussian Process. They then presented forecasts for traffic based on vector auto-regressive models, showcased on a real-world dataset of Sydney (part of the global traffic camera dataset). Their forecast horizon is 30 minutes but they follow a finer temporal scale where they predict 6 steps in the future (each step spanning 5 minutes). They achieve a mean-square-error (*MSE*) in range $[0.03, 0.09]$. Even though it is not directly comparable with our results because of differences in problem formulation, we expect errors to increase with finer temporal granularity and longer forecast horizons. We aim for improved accuracy, and lower errors.

Various applications have benefited from advancements of machine/deep learning. Zhang et al. [25] have proposed a trust based deep reinforcement learning framework for VANETs. Tian et al. [20] have studied application of ML models for detection of pedestrian and vehicles in images used for intelligent transportation systems. Numerous studies have attempted modeling of traffic patterns, which cannot be comprehensively covered here, a summary of which is presented. Kamarianakis and Prastacos [12] surveys the pre-deep learning era, focusing on spatial time-series modeling using techniques such as Space-Time *ARIMA* and the Bayesian Vector Autoregressive (*BVAR*). Hamed et al. [9] shows a classic example of *ARIMA* modeling for traffic volume in urban settings. Wang et al. [22] have proposed a spatiotemporal convolutional-*LSTM* hybrid network that has been able to outperform temporal only or spatial only networks on travel demand prediction. Zang et al. [24] have studied traffic speed modeling by the use of a *ConvLSTM* (introduced in [23] for precipitation forecasting) and CNNs hybrid model. Jin et al. [11] model city-wide short-term crowd flows with a deep learning approach that combines *LSTM* and *CNN* to capture spatiotemporal patterns. Chakraborty et al. [4] have investigated the application of the YOLO deep learning model on traffic congestion detection using camera images. Alipour et al. [2] have investigated Markov Chains, LSTMs and CNNs for next location prediction tasks. Polson and Sokolov [16] utilize and tune a deep multi-layer perceptron model for short term traffic flow prediction. Tian et al. [21] combines the power of *LSTMs* and temporal smoothing techniques to infer lost data and learn prediction residuals.

There are three major contributions of this study that differentiate it from the literature: 1. Data-driven study powered by planet-scale data. Traffic camera's data, specifically London's, has not been studied for forecasting purposes using deep learning, to the best of our knowledge. 2. The idea of formulating [geo]spatial locality explicitly as features of a deep learning model that leads to our best performer (incorporating the idea from vector and simultaneous autoregressive model into a deep model). 3. We take forecasting one step further by studying the spatial relations between traffic observed at different parts of the city showing how traffic can be predictable from other cameras across the city.

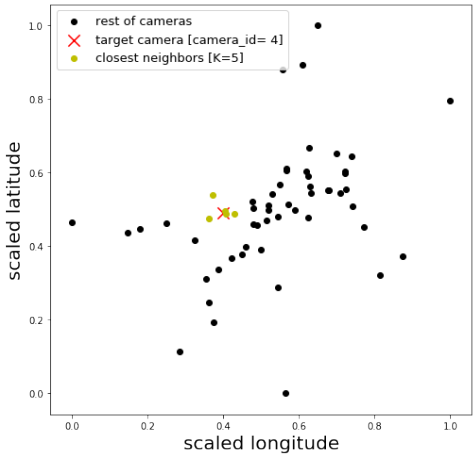


Figure 1: Distribution of the cameras across space on scaled map. Camera 4 and the closest five cameras to it shown.

3 METHODOLOGY AND FORECAST MODELS

This section details the *data* and *forecast models*. The ***data*** used is processed from London's traffic camera images. As previously stated, the original dataset (introduced in [18]), provides us with a measure of density (achieved through a background subtraction algorithm) over 42 days in 2010, in few-minutes granularity for hundreds to thousands of cameras covering large-scale urban areas across the globe. Metadata of the cameras, is joined with the measurement data and then passed through the data cleaning pipeline. We focus on London's data as a first step. Cleaned version of data used for this study consists of 58 cameras over a span of 40 days, with contiguous data from 9:30am to 6:30pm in 30 minutes average aggregates, with at most 2 missing values which are linearly interpolated. The values are min-max scaled globally to [0,1] range.

Figure 1 shows the location of cameras on a scaled longitude-latitude plane with a target camera (camera 4) and its neighbors marked. To get a feel of how the timeseries look like, Figure 2 visualizes camera 4 and its 4 nearby cameras. There are observable hints of periodicity (a spectrum analysis performed for seasonal models suggests seasonality of 19 records which corresponds to a day as the most dominant cycle). Also, it is worth noting that the timeseries do not express strong trends (seem to have stationary means). In addition, there seem to be similarities between them, pointing at the potential of using other cameras (especially cameras nearby) to improve forecasting. Cross-correlation of three closest cameras to camera 4 is plotted in Figure 3. Cross-correlation is calculated as the correlation coefficient of one versus the time-lagged version of the other timeseries. In our trials (including auto-correlation of difference between pairs of timeseries and cross-correlation for each pair), the closest camera seems to have greater correlation (not shown due to space constraints). However, no other significant patterns are observed in the data.

The remainder of this section shall detail the ***forecast models***. We first explore *univariate* models, where the model can be seen as a function on tensors of the shape $(1, timesteps)$ where 1 is the density value of a specific camera, into (1) which represent the

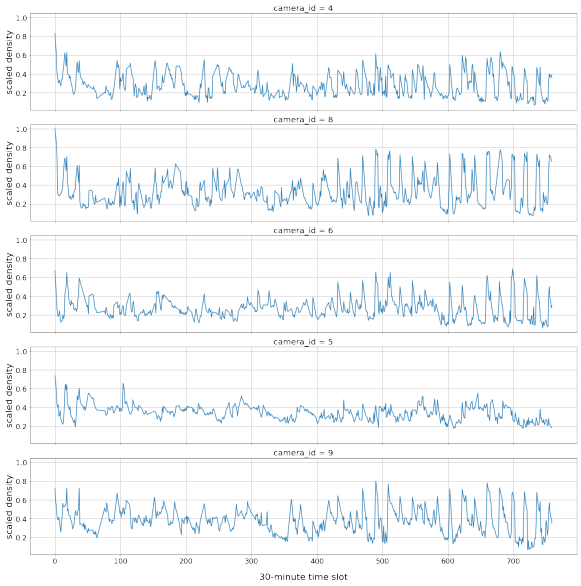


Figure 2: Sample of timeseries for camera 4 and nearby.

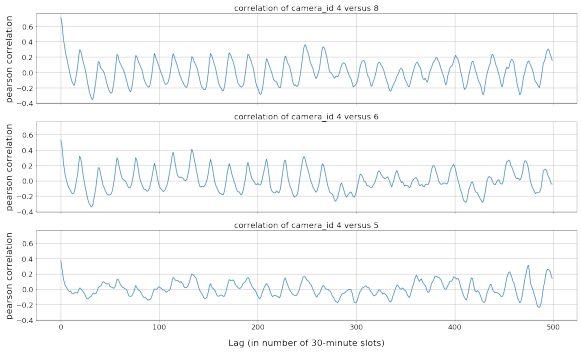


Figure 3: Cross Correlation functions of three closest cameras to camera 4 shows periodic correlation between them suggesting spatial relations may help explain some of variability of the model. X axis is the number of lags in 30 min increments.

very next value for the density of that camera. Then we explore the *multivariate* model, of shape $(58, timesteps) \rightarrow (58)$ where 58 represent all the cameras. In other words, all cameras are used as input over time and the next value of them all is predicted by the model (a snapshot of the city's traffic as seen by the cameras). Then a *hybrid* approach is suggested where for a target camera we consider its neighbors in the input but only forecast that target camera (i.e. $(K, timesteps) \rightarrow (1)$). As mentioned earlier in the introduction, all or parts of the first 75% of the timeseries are used for training and the remaining 25% is kept for evaluation. Mean Square Error (*MSE*) is used for both training and evaluation. In evaluating neural models, the actual input is fed to the trained model (and not the predicted values). Mean Absolute Percentage Error (*MAPE*) and Mean Exponential Error (*MEE*, an exaggerated version of *MSE*) were also studied as training loss functions and found not

leading to meaningful improvement of *MSE* for evaluation. The evaluation metric of *MSE* is calculated over all the predicted values. Mathematically, if y_i is the value at timestep i, and y'_i is the predicted value at that time and N is the length of the predicted period:

$$MSE = \frac{\sum_{i=0}^{N}(y_i - y'_i)^2}{N}$$

After discussing the models, the results are presented in Table 1. In addition, we compare the evaluation performance of the models by the means of their empirical cumulative density using percentage error and coefficient of determination (R^2) in Figure 8. percentage error is defined as the mean absolute error normalized over range of each camera (since the density value is scaled to [0,1] range globally). All the metrics agree on the performance of the predictors.

3.1 Univariate Seasonal Models

To establish a baseline, we look into four seasonal models: seasonal naive, *ARIMA* (Auto-Regressive Integrated Moving Average), *TBATS* (Trigonometric seasonal, Box-Cox transformation, *ARMA* residuals, Trend and Seasonality), and Holt-Winters (additive). These models have been previously studied and summarized in [1]. To keep the article focused, an intuitive description of the models is given and details of mathematics of the models are left out (more details can be found in [7]). In the *seasonal naive* model, the predicted value is equal to the last value from the same season. In this study, a power spectrum analysis of 30 minutes time slots for the mean scaled density of all cameras reveals one day as the dominant cycle. As a result, one day is considered as the season in the model. The *ARIMA model* describes the autocorrelations in data. A seasonal *ARIMA* model is formed by including additional seasonal terms in the *ARIMA* model, and can be written as $ARIMA(p,d,q)(P,D,Q)_m$, where m= number of periods per season. The uppercase notation is used for the seasonal parts of the model, and the lowercase notation for the non-seasonal parts of the model. The seasonal part of the model consists of terms that are very similar to the non-seasonal components of the model, but they involve backshift of the seasonal period. For example, d is the order of first differencing, and D is the order of seasonal differencing. Autoregressive $AR(p)$ implies current values depend on its p-previous values. Moving average $MA(q)$ means the current deviation from the mean depends on q-previous deviations, where q is the order of *MA* process. We used auto.arima model [8], which return the best *ARIMA* model according to Akaike information criterion (*AIC*), but it is not necessarily the best in terms of prediction error (as *AIC* is a model selection criterion penalizing more complex models). The order of differencing d is based on the *KPSS* test and order of seasonal differencing is based on *OCSB* test. The *TBATS model* is used to model series exhibiting multiple complex seasonalities [5]. It uses a combination of *Fourier* terms with an exponential smoothing state space model and a Box-Cox transformation. The *Holt-Winters seasonal model* comprises the forecast equation and three smoothing equations for level ℓ_t, trend b_t, and seasonal component denoted by s_t, with smoothing parameters α, β^* and γ. In this study, we use the additive seasonal method since the seasonal variation is roughly constant. In this method, the seasonal component is expressed in absolute terms in the scale of the observed series, and in the level equation, the series is seasonally adjusted by

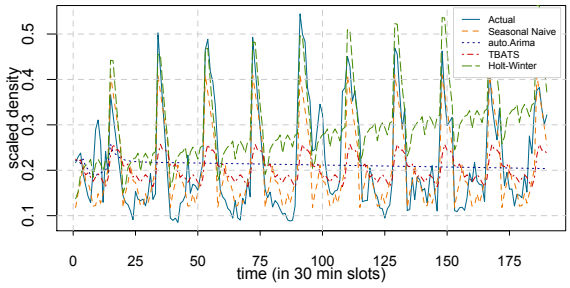


Figure 4: Forecast versus actual for seasonal prediction models camera-48. Horizontal axis represents time in 30 minute bins, with 0 being at the start of evaluation set (the last quarter of the data).

subtracting the seasonal component. More detail about this model can be found in [7]. The models are trained for each camera using their scaled density for 30 days. Then, the models are made to predict the next 10 days. Statistical prediction errors distributions are shown in figure 7, with the *TBATS* model outperforming other seasonal models.

The model's prediction accuracy varies between cameras, where it provides a better result with cameras that have seasonal patterns such as in Figure 4, but predict with lesser accuracy in Camera4 (camera with id 4 in the dataset), as in Figure 9.

3.2 Univariate Recurrent Neural Model

Recurrent Neural Networks (*RNN*) have been proposed and studied for a long time but they have attracted a lot of attention only recently. In particular, Long Short Term Memory (*LSTM*) cells have been very powerful at capturing temporal patterns (even those that have occurred in the distant past) [10]. These cells are the primary recurrent unit in our models. Important parameters of such a model include timesteps (how many timesteps to consider for each training samples), number of units in a cell, and regularization and dropoff rates (used to prevent over-fitting). Note that training samples are overlapping (generated by a rolling window over time on the cameras) and our LSTMs used maintain state across different batches, so that they would see the whole timeseries after one epoch (one pass over the training set). The model is simply a sequential model where a *LSTM* cell is fed into a tiny perceptron and is trained against mean squared error as the loss function. We observe that varying the batch size of the training data, yields no noticeable gains in loss value or training speed (likely due to the size of the data). Hence, a batch size of 1 is used. We also explore adding more parameters to the model (via adding layers or more units), but no significant gains are obtained. Thus, we adopt a simple model consisting of one unidirectional *LSTM* cell (and one perceptron to project to density value). For our neural network models, we use Keras and Tensorflow platform. Hyperparameter tuning is done via Microsoft Neural Network Intelligence (*NNI*). Adam optimizer [15] is used for the training. The training is done on CPU (where we observed speed ups in range 2x to 6x compared to GPU). This might be partially influenced by the sequential nature of the *RNNs*, batch size of 1 and relatively small size of the training set (GPUs

outshine CPUs when large matrix computations can be done in parallel). The average *MSE* achieved by this model is ≈ 0.0073 [1].

3.3 Multivariate Recurrent Neural Model

In the multivariate *RNN* model, we forecast a snapshot of city's traffic all at once. We conjecture that utilizing more cameras into a multivariate model allows for learning patterns between the cameras, and in turn help with the overall performance (as measured by evaluation *MSE*). The model's architecture is visualized in Figure 5.

After running hyperparameter tuning on *NNI* using the *tree-structured Parzen estimator* (*TPE*) algorithm [3] for 72 hours (1000s of trials), on a very large nested parameter search space, the winner state employs: **i-** 320 rectified linear units in a time-distributed manner, applied across each time slice of input for all the studied cameras, **ii-** 160 units in each direction of the Bidirectional *LSTM* cell (stacking *LSTMs* resulted in missing timeseries details and noticeably lower performance), and **iii-** a three-layer perceptron as output projection part of the model with sigmoidal activation functions (2 hidden layers). Dropout and L2 regularization are used to prevent over-fitting. This resulted in 672,918 trainable parameters for the model that can be trained in ≈25 minutes (on a hexa-core 8th-Gen Intel-i7 laptop CPU) before the early-stopping criteria of 'no improvement of 0.0001 for 10 epochs' is met. The *training loss function* here is the same *MSE*. Note that because of the nowcasting nature of the model, for each training example, it predicts a vector of values (of length 58) and thus the value of the loss is the average of the squared errors of all of the cameras at that point (averaged over time, for each epoch). This is different than the way *MSE* is used in the evaluation. For the evaluation, for each camera, each timeseries' squared error is averaged first, and then those values are averaged to obtain the final metric. This is done to be consistent with per-camera models. With all these settings, the model could achieve an evaluation *MSE* of 0.0125. This model on its own does not perform better than univariate models as it results in generally smoother functions not capturing the spikes, but it has a noticeably shorter training time. These insights also pave the way for the hybrid model and spatial forecasting.

3.4 Hybrid Model

The univariate model is trained against a loss function based on the *average* of all cameras' errors. In other words, it aims at minimizing the average, which results in overall smoother figures comparing to when each camera/location model is trained individually. To allow for this per-location (per-camera) optimization, we build a ***hybrid*** model that targets each camera individually, but incorporates the nearby cameras' data as input features to the model (in some sense the model becomes aware of its spatial locality) resulting in a $(K, timesteps) \rightarrow (1)$ model, where $K \in \{1, \ldots, 58\}$ represents the number of cameras to consider together. The next question then becomes, what is the optimal K, i.e., how many neighboring cameras should be considered together for? To answer this, along with the best possible performance for this model, we parameterize the model on K and run hyperparameter tuning on it.

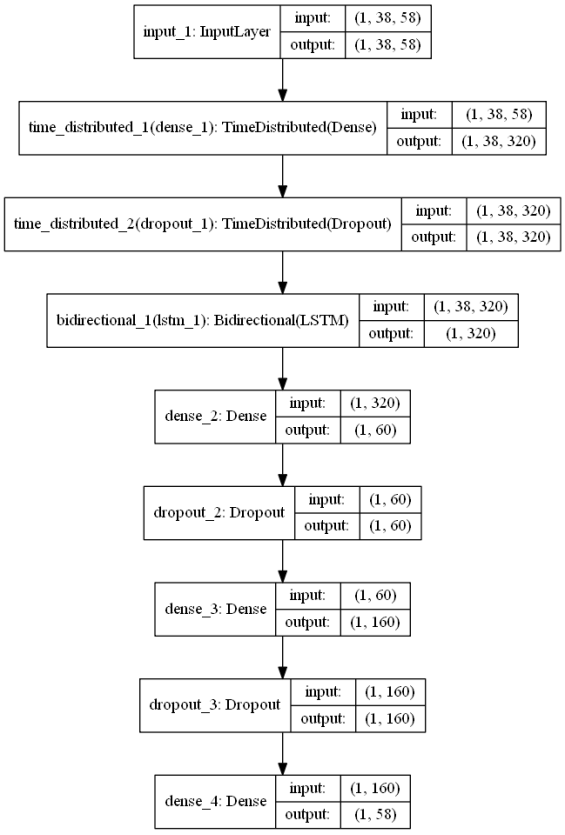


Figure 5: Architecture of the multivariate forecast model. TimeDistributed applies a mapping (Dense layer) to each time slice of the input tensor (across all cameras). A multilayer perceptron with 2 hidden layers is applied to the projections of *LSTM*, mapping it to the desired prediction.

For illustration purposes, without loss of generality, we show the tuning results for one camera (camera 4) in Figure 6. As observed, we find that shorter timesteps are preferred for each training sample, even though we use stateful *LSTMs* that remember the patterns across batches over longer periods of time. Also, fewer number of neighbors are preferred over more cameras, suggesting the spatial locality plays a role in this (as larger values of K implies cameras that are further away). The set of best parameters extracted is applied for all the cameras [2]. The training time varies per camera, resulting in a total time of ≈45 minutes.

Comparing the three presented deep-learning models on camera 4, the multivariate model achieves 0.0093, univariate model achieves 0.0075, and finally, the hybrid model results in *MSE* of 0.0059 (which is over 21% improvement, compared to the univariate model). A sample of forecasts is plotted in Figure 9 [3]. The average error for all the cameras using this hybrid model is 0.0067 (versus 0.0073 for the univariate model).

[1] This may not be the best achievable performance as each camera's model was not individually tuned and rather the same hyper-parameters as the hybrid model were used to be comparable.

[2] Hyperparameter tuning for each camera is a time-consuming task as it involves hundreds to thousands of trials.

[3] More figures are available at https://github.com/rzbhk/eyes_of_traffic_cams

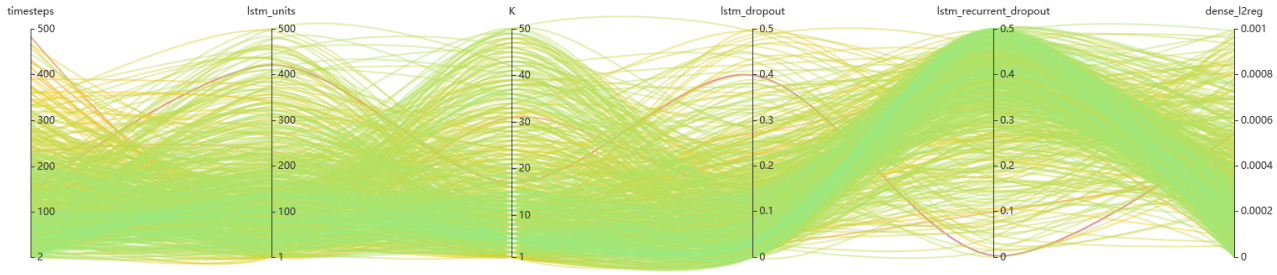


Figure 6: Results of hyper-parameter tuning for hybrid model for Camera4 (brighter greens are lower errors). There is an interesting trend in general: Fewer timesteps (for short term dependencies, though the stateful *LSTM* will remember these dependencies for longer periods of time) with reasonable number of *LSTM* units (adding more doesn't help), and fewer neighbors (K=5) result in better performance (lower errors).

Table 1: Forecast errors (*MSE*) of the various studied models. Lower is better.

	AVG	STD	MEDIAN	MIN	MAX
Hybrid. *RNN*	0.0067	0.0036	0.0063	0.0018	0.0184
Univar. *RNN*	0.0073	0.0037	0.0073	0.0020	0.0177
Multivar. *RNN*	0.0125	0.0062	0.0115	0.0027	0.0328
TBATS	0.0176	0.0092	0.0162	0.0036	0.0464
Season. ARIMA	0.0243	0.0140	0.0235	0.0039	0.0621
Season. Naive	0.0313	0.0202	0.0257	0.0044	0.0965
HoltWinter	0.0341	0.0251	0.0250	0.0049	0.1355

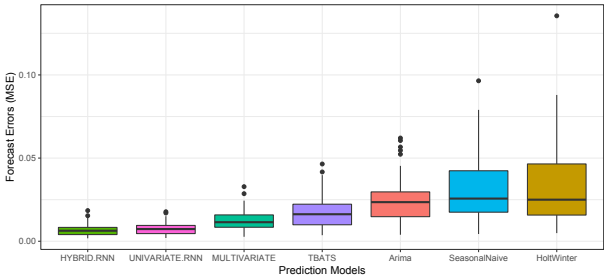
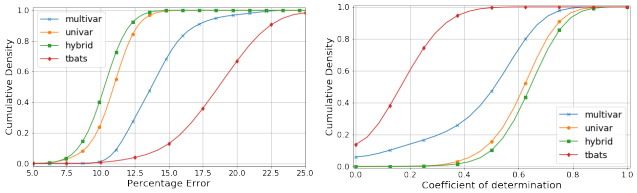


Figure 7: Summary of forecast errors (*MSE*) of all models for all cameras.



(a) (b)

Figure 8: ECDF of percentage error (a) and coefficient of determination R^2 (b) for the models. The more a curve is to left is better for (a) and the more to the right is better for (b). The best performer (hybrid) has an error of ≈10%.

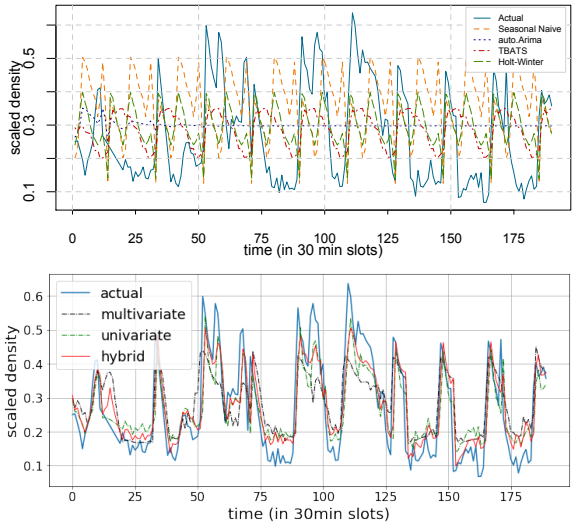


Figure 9: Forecast versus actual for seasonal models (top) and neural models (bottom) for Camera4. Horizontal axis represents time in 30 minute bins, with 0 being at the start of evaluation set (the last quarter of the data). The hybrid model seems to better capture rise and falls.

4 SPATIAL FORECAST

Spatial interpolation is particularly useful in understanding the patterns of traffic across the city with limited vantage points.

Fay et al. [6] have studied the relations between the cameras by the use of Granger Causality test (in a causality network). It was found that, simply put, consistent patterns of correlation exists between nearby cameras (possibly due to the effects of congestion propagation). We plotted the cross-correlation and autocorrelation of mutual differences between the cameras (in a similar manner to Figure 3; not shown due to its size) and observed repeating patterns of correlation between many of the cameras (even over the larger distances). This suggests that some of the observed spatial correlations, over long-time windows, are spurious as they all appear to depend on temporal structures (i.e., diurnal and daily schedules).

Table 2: Spatial forecast average and best errors out of 10 random trials with varying number of input cameras (N).

N	Average Error	Min Error
1	0.0169	0.0125
2	0.0138	0.0114
4	0.0132	0.0113
8	0.0125	0.0114
16	0.0115	0.0102
32	0.0116	0.0096

Building on these findings, we investigate a multivariate model $(N, timesteps) \rightarrow (M)$ where N denotes the number of input cameras, and M denotes the number of cameras (or locations) to predict. The model architecture resembles that of the hybrid model (Section 3.4) but with multiple output targets fed into an average loss function (similar to the multivariate model, in section 3.3). In this experiment, the two sets of input and output cameras are exclusive, i.e., to predict the traffic density at CameraX's location, no information about CameraX values is seen explicitly by the model.

For each input size N, a random choice of size N from all the 58 cameras is drawn (this is repeated over 10 trials). All the remaining cameras are considered in the output (target) camera set. The results are presented in Table 2. With only **one** camera as input, *we can reconstruct the future of the target cameras* (the rest of 57 cameras) with an average error (*MSE*) of 0.0169, while with a proper choice of input camera, errors as low as 0.0125 are possible. This is *as good as the multivariate RNN model where all the cameras are fed in as inputs*. With increase in the number of cameras in the input set, the error seems to drop. However, this may be an artifact of the number of target cameras going down; harder-to-forecast cameras may have been included in the input set and hence excluded from the output. In all cases the gap between the *min* and the *mean* error suggests that the choice of input cameras matters for this purpose. That is, to select good vantage points, the ones which result in a better reconstruction of other timeseries are better candidates. Figure 10 presents a sample of forecast vs. actual for Camera4 (always guaranteed to be in the output and excluded from the input). Reconstructed signals (through forecast) closely follow the original trends. With increase in number of input cameras, the forecast becomes better in capturing the periods when density stays low or high. Comparing to regular forecasts we observe more noise in reconstructed timeseries in general.

As a final investigation, we seek to find the optimal size and the set of input cameras to forecast other cameras' traffic with least error. To achieve this, we formulate the problem as a hyper-parameter optimization, in which the size of input and seed of the random number generators (*RNG*) are considered among the parameters. The number of input cameras is upper-bounded by half the total number of cameras, so at least one half of cameras are being used as targets. The results suggest a set of size 23 can reconstruct the remaining cameras (i.e., 35) with the average *MSE* of 0.0086. The location of the selected cameras is shown on the scaled map (lon-lat) plot in Figure 11.

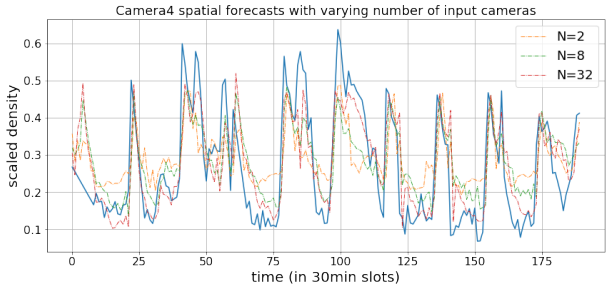


Figure 10: Forecast versus actual (shown in solid blue) for Camera4 versus N=2, 8, 32 number of input cameras. Note this camera is always excluded from the input set.

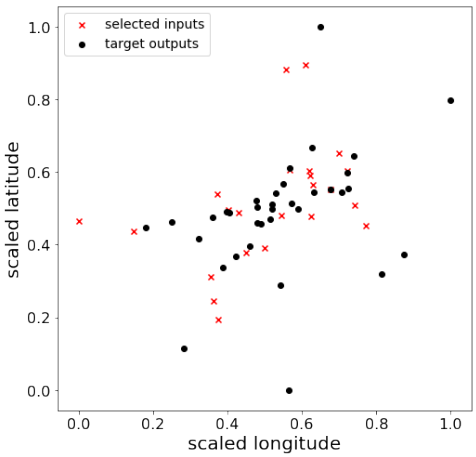


Figure 11: Selected input cameras can reconstruct the remaining cameras (in black) with minimal error (found by hyper-param tuning).

5 DISCUSSION

So far, we have explored the applications of deep learning particularly recurrent neural networks for traffic density prediction. These findings have potential use-cases in various fields including transportation, urban planning, communication infrastructure, resource provisioning and vehicular networking. A direct application of our traffic forecasting involves training these models every month with new data (which takes less than an hour), and online prediction of traffic and congestion for the next 30-minute time windows. Real-time data feeds need to provide the models with most recent actual values, as was done in our study. Below is a list of our ongoing and planned efforts:

- A more sophisticated multivariate model with spatial attention as latent features shall be evaluated. The applicability of more recent alternatives to RNNs such as attention based models (i.e. Transformer) also remains to be explored. The performance goal, pictorially explained, is to push the blue (multivar) curve on Figure 8a towards the left.

- Forecast of traffic fed into city-wide simulations at a microscopic level [13], enables us to study further dynamics of the road, and applications such as shared transportation [14].
- A more comprehensive study of the spatial aspect shall be undertaken, potentially by comparing and contrasting to Gaussian Processes and boosting methods (e.g., XGBoost).
- An effort is planned to alleviate the effects of the image processing limitations. As mentioned earlier, the density number is not easily comparable from camera to camera or city to city. We shall investigate extracting car counts from images using recent deep learning models and establish a car count per time dataset.
- A recent effort is under way to collect recent images of traffic cameras, e.g., from London. We plan to repeat the study on the newer data to fortify the findings. The newer data has a wider time-span enabling study of longer temporal effects and more thorough cross validation schemes, and higher camera density, enabling finer spatial analysis studies.
- An extension is planned to take steps beyond nowcasting and try to forecast longer term futures.
- An extension is planned to study several other cities from the dataset (Washington DC., Seattle, Sydney, Toronto, etc.).

6 CONCLUSIONS

In this paper we investigated various aspects of spatio-temporal traffic forecasting based on processed densities from 58 traffic cameras spanning 40 days. We studied the traffic patterns from temporal (forecast) and spatial (interpolation) aspects. We summarize our findings and contributions as:

- We investigate seasonal and neural models for traffic density forecasting from individual cameras point of view. LSTMs in particular are able to capture short term relationships that happen across long periods of time (hence the name) and are capable of modeling temporal phenomena that does not necessarily exhibit strong seasonal patterns. We found that deep learning models outperform seasonal (conventional) models in all cases (11.1% average percentage error vs. 19.3%).
- We propose an end to end neural architecture which can model the snapshot of traffic (all the cameras) with reasonable performance (14.5% percentage error) that trains in half the time of training 58 individual regressors based on hybrid model.
- Borrowing from ideas of simultaneous timeseries, we incorporate nearby cameras as input features of the deep learning model resulting in $\approx$ 8% reduction in MSE (0.0067 vs 0.0073) and 0.6% reduction in percentage error (at 10.5%).
- Finally we explore forecasting vehicular density without explicit historic information about traffic at camera locations. We showed that it is possible to predict the rest of the city's traffic, with limited number of vantage points (23 cameras can reconstruct the rest with only 0.0086 error). This finding provides great promise and interest to further advance the multivariate model to automatically extract relevant spatial information.

ACKNOWLEDGMENTS

This project was partially supported by NSF grant 1320694, UF Informatics Institute and Najran University, Saudi Arabia.

REFERENCES

[1] Mimonah Al Qathrady and Ahmed Helmy. 2017. Improving BLE Distance Estimation and Classification Using TX Power and Machine Learning: A Comparative Analysis. In *Proceedings of MSWiM'17*. ACM, Miami, FL, USA, 79–83.
[2] Babak Alipour, Leonardo Tonetto, Roozbeh Ketabi, Aaron Yi Ding, Jörg Ott, and Ahmed Helmy. 2019. Where Are You Going Next? A Practical Multi-dimensional Look at Mobility Prediction. *Proceedings of MSWiM* (2019).
[3] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Proceedings of NIPS'11*. 2546–2554.
[4] Pranamesh Chakraborty, Yaw Okyere Adu-Gyamfi, Subhadipto Poddar, Vesal Ahsani, Anuj Sharma, and Soumik Sarkar. 2018. Traffic Congestion Detection from Camera Images using Deep Convolution Neural Networks. *Transportation Research Record* 2672, 45 (2018), 222–231. https://doi.org/10.1177/0361198118777631
[5] Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. 2011. Forecasting time series with complex seasonal patterns using exponential smoothing. *J. Amer. Statist. Assoc.* 106, 496 (2011), 1513–1527.
[6] Damien Fay, Gautam S Thakur, Pan Hui, and Ahmed Helmy. 2013. Knowledge Discovery and Causality in Urban City Traffic: A study using Planet Scale Vehicular Imagery Data. In *Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Computational Transportation Science*. ACM, 67.
[7] Rob J H. and George A. 2014. *Forecasting: principles and practice*. OTexts.
[8] Rob J H., Yeasmin K., et al. 2007. *Automatic time series for forecasting: the forecast package for R*. Number 6/07. Monash University.
[9] Mohammad M. Hamed, Hashem R. Al-Masaeid, and Zahi M. Bani Said. 1995. Short-Term Prediction of Traffic Volume in Urban Arterials. *Journal of Transportation Engineering* 121, 3 (1995), 249–254.
[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[11] Wenwei Jin, Youfang Lin, Zhihao Wu, and Huaiyu Wan. 2018. Spatio-Temporal Recurrent Convolutional Networks for Citywide Short-term Crowd Flows Prediction. In *Proceedings of ICCDA'18*. ACM, New York, NY, USA, 28–35.
[12] Yiannis Kamarianakis and Poulicos Prastacos. 2006. *Spatial Time-Series Modeling: A review of the proposed methodologies*. Working Papers 0604. University of Crete, Department of Economics. https://ideas.repec.org/p/crt/wpaper/0604.html
[13] R. Ketabi, B. Alipour, and A. Helmy. 2017. En route: Towards vehicular mobility scenario generation at scale. In *2017 IEEE Conference on Computer Communications Workshops*. 839–844. https://doi.org/10.1109/INFCOMW.2017.8116485
[14] Roozbeh Ketabi, Babak Alipour, and Ahmed Helmy. 2018. Playing with Matches; Vehicular Mobility through Analysis of Trip Similarity and Matching. *Proceedings of SIGSPATIAL'18* (2018).
[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[16] Nicholas G. Polson and Vadim O. Sokolov. 2017. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies* 79 (2017), 1 – 17. https://doi.org/10.1016/j.trc.2017.02.024
[17] Gautam S Thakur, Mohsen Ali, Pan Hui, and Ahmed Helmy. 2012. Comparing background subtraction algorithms and method of car counting. *arXiv preprint arXiv:1202.0549* (2012).
[18] Gautam S. Thakur, Pan Hui, and Ahmed Helmy. 2012. A Framework for Realistic Vehicular Network Modeling Using Planet-scale Public Webcams. In *Proceedings of the 4th ACM International Workshop on Hot Topics in Planet-scale Measurement (HotPlanet '12)*. ACM, New York, NY, USA, 3–8.
[19] Gautam S Thakur, Pan Hui, and Ahmed Helmy. 2013. On the existence of self-similarity in large-scale vehicular networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*. IEEE, 1756–1761.
[20] Daxin Tian, Chuang Zhang, Xuting Duan, Jianshan Zhou, Zhengguo Sheng, and Victor Leung. 2017. The Cooperative Vehicle Infrastructure System Based on Machine Vision. In *Proceedings of DIVANet'17*. ACM, New York, NY, USA, 85–89. https://doi.org/10.1145/3132340.3132347
[21] Yan Tian, Kaili Zhang, Jianyuan Li, Xianxuan Lin, and Bailin Yang. 2018. LSTM-based traffic flow prediction with missing data. *Neurocomputing* 318 (2018), 297 – 305. https://doi.org/10.1016/j.neucom.2018.08.067
[22] D. Wang, Y. Yang, and S. Ning. 2018. DeepSTCL: A Deep Spatio-temporal ConvLSTM for Travel Demand Prediction. In *2018 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN.2018.8489530
[23] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proceedings of NIPS'15*. 802–810.
[24] D. Zang, J. Ling, Z. Wei, K. Tang, and J. Cheng. 2018. Long-Term Traffic Speed Prediction Based on Multiscale Spatio-Temporal Feature Learning Network. *IEEE Transactions on Intelligent Transportation Systems* (2018), 1–10. https://doi.org/10.1109/TITS.2018.2878068
[25] Dajun Zhang, F. Richard Yu, Ruizhe Yang, and Helen Tang. 2018. A Deep Reinforcement Learning-based Trust Management Scheme for Software-defined Vehicular Networks. In *Proceedings of DIVANet'18*. ACM, New York, NY, USA, 1–7. https://doi.org/10.1145/3272036.3272037