Traffic Anomaly Detection Via Conditional Normalizing Flow

Zhuangwei Kang*, Ayan Mukhopadhyay*, Aniruddha Gokhale*, Shijie Wen†, Abhishek Dubey*
* Institute for Software Integrated Systems, Vanderbilt University
† Cisco Systems, Inc.

***Abstract*—Traffic congestion anomaly detection is of paramount importance in intelligent traffic systems. The goals of transportation agencies are two-fold: to monitor the general traffic conditions in the area of interest and to locate road segments under abnormal congestion states. Modeling congestion patterns can achieve these goals for citywide roadways, which amounts to learning the distribution of multivariate time series (MTS). However, existing works are either not scalable or unable to capture the spatial-temporal information in MTS simultaneously. To this end, we propose a principled and comprehensive framework consisting of a data-driven generative approach that can perform tractable density estimation for detecting traffic anomalies. Our approach first clusters segments in the feature space and then uses conditional normalizing flow to identify anomalous temporal snapshots at the cluster level in an unsupervised setting. Then, we identify anomalies at the segment level by using a kernel density estimator on the anomalous cluster. Extensive experiments on synthetic datasets show that our approach significantly outperforms several state-of-the-art congestion anomaly detection and diagnosis methods in terms of Recall and F1-Score. We also use the generative model to sample labeled data, which can train classifiers in a supervised setting, alleviating the lack of labeled data for anomaly detection in sparse settings.**

***Index Terms*—Anomaly Detection, Traffic Congestion, Flow Model**

I. Introduction

Transportation Management Centers are critical for managing the surface road network. Monitoring is critical in practice; delays accrued during the monitoring phase delay response and resolution [1]. Frequently, secondary crashes and long-clearance times lead to additional congestion on critical arterial road segments. To improve the real-time monitoring of extensive road networks, transportation agencies are increasing the available sensing modalities, often in smart corridors. However, this drastic increase in the number of sensors raises an essential question from an operational perspective—how can transportation agencies monitor thousands of sensors in (near) real-time to detect incidents of interest? Our conversations with local transportation agencies revealed that this monitoring is largely performed manually, an infeasible strategy in the long run. One approach to enable transportation agencies to utilize an extensive array of sensors is to detect potentially anomalous patterns in real-time using the data generated by the sensors; then, human experts (or potentially decision-theoretic approaches [2]) can narrow their focus on the anomalies and take necessary operational actions.

Challenges The problem of monitoring large sensor streams to detect incidents of interest is basically an unsupervised anomaly detection problem in multivariate time series (MTS); traffic data pertaining to road segments are collected across time and span several dimensions. While anomaly detection has been traditionally done for various traffic condition variables with techniques such as CUSUM [3], K-nearest Neighbor [4], Isolation Forest [5], and forecasting models (e.g., ARIMA [6]), deep neural networks (DNN) have gradually become the state-of-the-art due to the remarkable capability of modeling high-dimensional MTS data. However, despite the universal approximation power of DNN on learning unknown data distributions, performing anomaly detection on MTS is still challenging. For example, many DNN-based approaches either rely on an uncontaminated training dataset to learn the normal traffic patterns (semi-supervised) or reframe the detection task as a classification task using a fully-labeled traffic mobility dataset (supervised).

Such approaches, however, are not practical in general. Moreover, such data often do not account for a large number of incidents, such as phantom traffic jams, slowdowns, and weather hazards. Further, even if labels are available, such classifiers and approaches identify point anomalies where the observation is clearly far away from what is expected globally. However, transportation networks often suffer from contextual anomalies. Consider that an observed value of congestion on a road segment (at a given time) can be an anomaly if it deviates significantly from expected historic behavior, which might be caused by severe weather, big events, road construction, etc. These challenges require the design of an *unsupervised* detector that can generalize decisions based on multi-dimensional probability distributions learned over both the spatial and temporal aspects of the traffic time series.

In addition, traditional anomaly detection techniques focus on maximizing the accuracy of detection. However, in the specific use case of transportation centers, the goal of such a detector is to ensure that the search space for monitoring is shrunk for domain experts. As a result, in practice, the detector must demonstrate high recall with relatively low precision, i.e., false negatives are more costly than false positives because missing the alerts usually leads to late emergency response, congestion cascades, or even chain collisions. Finally, an additional challenge is proactive model improvement; agencies must ensure that the learned model used to detect anomalies is improved proactively to detect

potentially unseen anomalies.

Contributions This paper systematically addresses these challenges by developing a traffic anomaly detection framework based on conditional normalizing flow, a probabilistic generative model that can tractably perform density estimation and sampling in extremely high dimensional spaces [7]. Through this approach, we can model the multimodal distributions of traffic data. In particular, we propose a principled MTS anomaly detection and diagnosis model for traffic data that comprises an LSTM-Encoder-Decoder (LSTM-EncDec) model and a Normalizing Flow architecture [7], specifically a RealNVP [8] flow. The former makes sequence-to-sequence forecasting with a sliding-window scheme to extract internal spatial-temporal information from ground-truth data. The flow model is used to model complex data distribution in the high-dimensional transit data. Specifically, it performs conditional density estimation using the outputs of the forecasting model. To ensure tractability of our approach, we divide the road network of a city into clusters, and perform anomaly detection at the granularity of clusters. Then, we use a simpler density estimator based on a kernel density function to identify anomalies at the granularity of road segments.

We compare our approach with existing state-of-the-art baselines using traffic data collected from the City of Nashville, Tennessee. Experimental results show that our approach has superior performance and sensitivity on anomaly detection in traffic networks.

II. Related Work

Existing research on anomaly detection for surface transportation systems can be broadly classified into three classes: *reconstruction-based*, *prediction-based*, and *density-based* approaches. We review the principle and weaknesses of each strategy and eventually propose CondRealNVP which makes up for these deficiencies by combining the ideas of prediction-based and density-based approaches.

Reconstruction-based approaches leverage the notion that normal samples can be better reconstructed from a latent space than anomalies. AutoEncoder is the foundation of this class [9]. Hu et.al [10] combined AutoEncoder with graph convolutional networks to detect unexpected travel time in a set of directed weighted graphs. Madarash et.al [11] used LSTM-predicted maneuver labels to reduce false alarms when using LSTM AutoEncoder (LSTM-AE) to detect anomalous driving modality. These studies are limited to training the detectors using unpolluted data. Under an unsupervised setting, *Contextual AutoEncoder* [12] extends the regular LSTM-AE to multiple decoders. However, a common issue with AE-based methods is that the L2 optimization objective enforces models to learn a generic summarization of underlying regularities of ground-truth data, even for outliers, leading to severe over-fitting [13]. Variational AutoEncoder employs an additional Kullback-Leibler divergence loss term to alleviate this problem. It has been combined with LSTM for MTS anomaly detection [14], [15].

Prediction models rely on the fact that normal samples are more predictable than anomalies. One common implementation of this class is stacked LSTM models [16]–[18]. The performance of bidirectional LSTM (BiLSTM) in freeway traffic forecasting was investigated in [19]. Basak et al. [20] analyzed the cascade effects of traffic congestion using a citywide ensemble of intersection level connected LSTM models. The major drawback of these attempts is that the forecasting accuracy is likely to be affected by anomalies when training models with polluted datasets [21], leading to unreliable anomaly detection results.

Density-based approaches detect anomalies based on the principle that the density around a normal sample is similar to that around its neighbors. Chiang et al. [22] designed a two-step congestion cascaded identification strategy: (1) use a kernel density function to compute anomaly score for road segments; (2) form up congested cascades by unifying attribute coherence and spatial-temporal closeness of detected congested segments. Dias et al. [23] employed RealNVP and masked autoregressive flow for trajectory anomaly detection. Their experimental results show that flow models outperform classical density-based methods. Although density-based detectors do not need labeled training data [24], they only focus on the underlying data distribution, therefore, cannot capture the sequential correlation in time series.

III. Background

A. Normalizing Flow

Normalizing flows define a series of bijective transformations that can transform the probability density $p_X(x)$ of a random variable $X \in \mathbb{R}^D$ to a well-known base distribution $p_Z(z)$ defined by a random variable $Z \in \mathbb{R}^D$ [25]. The random variable Z is chosen such that it has an explicit probability density function. The problem of training the normalizing flow is to learn an invertible transformation, f such that $z = f(x)$ and $x = f^{-1}(z)$. The transformation is a sequence of bijective functions composed together, i.e., $f = (f_1.f_2 \cdots)$. Once learned, the forward mapping, $X \to Z$ can be used for density estimation and the inverse mapping $Z \to X$ can be used for sampling (synthetic data generation). This mapping presents a key advantage that enables exact density estimation without loss of dimensional information, making it suitable for anomaly detection. In particular, the marginal likelihood $p_X(x)$ can be expressed as:

$$p_X(x) = p_Z(f(x)) \left| \det \left(\frac{\partial f(x)}{\partial x} \right) \right| \tag{1}$$

where $p_Z(f(x))$ is density of x under the base distribution p_Z and $\det \left(\frac{\partial f(x)}{\partial x} \right)$ is the determinant of the Jacobian of f. The main challenges of modeling arbitrary distributions using normalizing flow lie in designing the compositional and invertible transformation f. Further, the choice of the architectures are restricted by the need for the efficient computation of the determinant of the Jacobian matrix.

B. RealNVP

One of the recent innovations in normalizing flow is the use of the real-valued non-volume preserving transformations [8] as the function f. Effectively, RealNVP is a set of affine

coupling layers, one of the possible bijective transformations that can be used to design the composition f.

To explain this further, consider the example of a single layer transformation (several such layers are composed in practice) Y that maps X to Z. RealNVP transformation Y partitions X into two disjoint groups, where the first d dimensions remain unchanged while the latter part, i.e., from the $d+1$-th to the D-th dimension, undergoes an affine transformation. Formally,

$$\begin{aligned} y^{1:d} &= x^{1:d} \\ y^{d+1:D} &= x^{d+1:D} \odot \exp(s_{net}(x^{1:d})) + t_{net}(x^{1:d}) \end{aligned} \tag{2}$$

s_{net} and t_{net} indicate a "scale" and a "translation" function respectively and $\odot$ stands for element-wise product. The representation power of RealNVP depends on s_{net} and t_{net}, which can be any arbitrarily complex function (often a neural network architecture). Note that because the first d dimensions remain unchanged during transformation, to make the flow model capture the full picture of input space, RealNVP swaps active and inactive dimensions in an alternating manner. A convenient way to realize this is to multiply the D dimensional inputs and outputs with a binary mask vector.

RealNVP guarantees its computation of Jacobian function is efficient because the Jacobian is a block-triangular matrix, where elements on the diagonal are an identity matrix and a diagonal matrix whose diagonal elements correspond to the vector $\exp\left(s_{net}(x^{1:d})\right)$. Therefore, the determinant of Jacobian, which simplifies to $\exp(\sum_j (s_{net}(x^{1:d})_j))$ can be efficiently computed. If the flow is implemented using K such layers, which is required to ensure better learning, the probability density of a given sample x can be calculated as follows:

$$log(p_X(x)) = \log(p_Z(z)) + \sum_{k=1}^{K} \log\left(\left|\exp(\sum_j (s_{net}^k(y_{k-1}^{1:d}))_j)\right|\right) \tag{3}$$

where the first term denotes the likelihood of z (transformed from x) on the base distribution, and the second term represents the accumulated changes while transforming x to z. Thus, the training objective of RealNVP is to find the right set of hyperparameters of the s_{net} and t_{net} that maximize the overall likelihood of the observed data, which can be denoted as $\theta^* = \arg\max_{\theta\in\Theta} \frac{1}{|\mathcal{D}|} \sum_{x\in\mathcal{D}} \log p_X(x;\theta)$, where $\mathcal{D}$ is the observed data and θ denotes parameters of s_{net} and t_{net} functions.

IV. Methodology

Let S denote the set of all road segments under consideration. Consider an arbitrary segment $S_i \in S$ on which (near) real-time speed is monitored continuously; we assume that the estimated harmonic mean speed on segment S_i is computed and stored at discrete times $t \in \{1, 2, ..., T\}$. We denote this observation at time t by v_i^t. The free-flow speed $\hat{v}_i$, an intrinsic property of segment S_i, is calculated based upon the 85th-percentile of the observed speeds on the segment S_i for all time periods [26]–[28]. The historical average speed, denoted by $\bar{v}_i^t$, signifies the regular traffic condition on S_i, which is calculated by taking the harmonic average of speeds on S_i for each hour of day and for each day of the week. Then, the congestion rate is defined as

$$x_i^t = \frac{\bar{v}_i^t - v_i^t}{\hat{v}_i} \tag{4}$$

The congestion rate of N roadway segments can be modeled as an N-dimensional time series of length T, denoted by X, i.e., $X = \{x^1, x^2, \dots x^T\}$, where $x^t \in \mathbb{R}^N$ is an N dimensional vector representing a measurement at time t. The congestion observation from the ith segment at time t is x_i^t. At an arbitrary time t, x^t therefore denotes a snapshot of congestion at all roadway segments. Each time step can have additional features associated with it, e.g., day of the week and hour of the day. We denote such features for the t-th time step be λ^t.

The primary goal of our framework is to detect points in time at which anomalous congestion may occur. With the obtained detection results, the secondary target is to recognize the roadway segments most likely to have caused the abnormal observation at each timestamp. Figures 1 and 2 together demonstrate a four step method we propose for fulfilling these targets: (1) time series clustering based on similarity measures; (2) unsupervised anomaly detection based on conditional RealNVP; (3) anomaly diagnosis at the road segment granularity based on non-parametric kernel density estimation; (4) auxiliary supervised anomaly detection based on multi-layer perceptron.

A. Time Series Clustering

In practice, X might be composed of thousands of dimensions with heterogeneous temporal patterns, semantic meanings, or underlying dependencies. It is computationally difficult to learn patterns or explicit probability distributions for extremely high-dimensional data. One way to alleviate this challenge is by identifying dimensions that are related in the feature space. To tackle this, we perform data-driven clustering to partition the given time series into separate groups based on similarity (where similarity is based on an appropriate distance in the feature space, e.g., the $\ell 1$ norm). This step facilitates anomaly detection and diagnosis in two aspects. First, it ensures that learning the probability distribution over the input time series is tractable. Second, similarity in the feature space naturally associates semantic meaning to the clusters; e.g., we observe that different clusters correspond (roughly) to different types of roads such as highways and on-ramps. Learning an explicit distribution for a particular cluster therefore enables us to learn a distribution of traffic in a particular type of roadway.

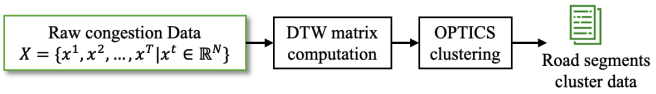


Fig. 1: Grouping the road segments into clusters.

Traditional clustering methods cannot be applied directly to time-series settings due to the temporal nature of the data.

A general idea of time series clustering, as shown in Figure 1, is to first convert temporal data to a *flat* representation by computing a similarity or distance matrix; then, a standard clustering algorithm (e.g., KMeans [29], DBSCAN [30]) can be used to partition the flat representation. We leverage the commonly-used "Dynamic Time Warping" (DTW) [31] distance to measure pair-wise similarities between time series. Particularly, consider arbitrary road segments S_i and S_j. The DTW distance between the segments (in the feature space defined by congestion) can be calculated as the squared root of the sum of squared distances between every element in $x_i^t \ \forall t \in \{1, \ldots, T\}$ and its nearest point in $x_j^t \ \forall t \in \{1, \ldots, T\}$. Intuitively, the distance reflects how similar was *each* congestion value observed in segment S_i to *any* congestion value observed in segment S_j, and then aggregates the similarities. Given the similarity measures, we use the OPTICS algorithm [32] for clustering. The OPTICS algorithm is density-based, which does not require the number of clusters as a prerequisite. Given the clusters, we perform anomaly detection in each cluster independently.

B. Timestamp-level Anomaly Detection

We can perform density estimation on the raw data using RealNVP directly; recall that the normalizing flow approach allows us to perform tractable density estimation. However, our initial experiments proved otherwise as the flow model failed to capture contextual anomalies. This observation is not surprising; the transformations in RealNVP operate along the feature dimensions but discard the temporal correlation in the data. In contrast, recurrent neural networks with gated memory such as LSTM (long short-term memory networks) [33] have been proven to be powerful tools for modeling sequential data. This inspires us to explore the possibility of capturing "point" and "contextual" anomalies simultaneously by aggregating an LSTM-based Encoder-Decoder and a normalizing flow model. As LSTM requires three-dimensional inputs (the batch size, the number of time steps, and the number of features), we use overlapping sliding windows $x^{\{1:\tau\}}$ of length τ as inputs, where each window is further divided into a context window $x^{\{1:t_0-1\}}$ and a prediction window $x^{\{t_0:\tau\}}$ (see figure 2). We explain the functioning of the LSTM below.

1) LSTM Encoder-Decoder: We use an LSTM-Encoder-Decoder structure that defines two separate components: an encoder and a decoder, each of which is composed of a stack of LSTM layers. During inference, the encoder first converts data $x^{\{1:t_0-1\}}$ into a single fixed-length representation vector, $f_{\theta_{enc}} : x^{\{1:t_0-1\}} \rightarrow e$, given by the last hidden states of LSTM, that contains all the information needed for the input of a subsequent decoder. The encoder vector e is then repeated $\tau - t_0 + 1$ times and used to initialize the internal states of decoder LSTM cells. The decoder then generates the ultimate hidden states of the target sequence $x^{t_0:\tau}$ in an autoregressive manner, i.e., $f_{\theta_{dec}} : e \rightarrow h^{t_0:\tau}$.

With respect to anomaly detection, the autoregressive scheme enables RNN to propagate and leverage historical information. Also, the encoder-decoder structure prevents

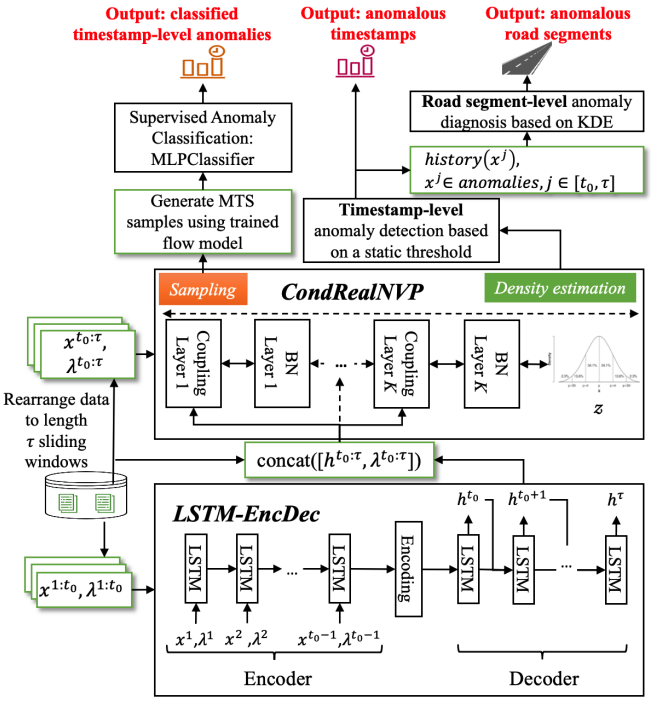


Fig. 2: Anomaly detection, roadway segment-level anomaly diagnosis, and supervised anomaly classification.

out-of-distribution data from being constructed from compressed historical information, which in turn ensures that the density is learned without too many outliers. Deterministic encoder-decoder models use mean squared error as anomaly score to measure the deviation between observations and predictions. However, this score may result in sub-optimal anomaly detection decisions due to two reasons. First, noise in data and randomness from the model's parameters may interfere with the training procedures. Second, the decisions are concluded from a fixed-length of historical data, therefore lacking a global perspective. We bypass this issue by integrating the encoder-decoder structure with the normalizing flow (described below) and training the overall architecture by maximizing the log likelihood function (which is inherently probabilistic).

Consider an observation x^t where $t \in \{t_0 : \tau\}$. Let the output of the last layer of the decoder for x^t be denoted by h^t. Intuitively, given h^t, which implicitly contains information summarized from previous time steps, our ultimate goal is to estimate the likelihood of x^t in the entire input space $\mathcal{X}$, i.e., $p(x^t|h^t)$. A low value implies the observation is either rare in the input space or deviates from contextual behavior. Our implementation of the LSTM Encoder-Decoder model is shown in figure 2. We include temporal features such as week-of-year, day-of-week, and hour-of-day with the encoder's input to facilitate learning the seasonality and trend patterns of time series. Then, the likelihood of the observations from t_0 to τ can be represented as:

$$p(x^{t_0:\tau} \mid x^{1:t_0-1}, \lambda^{1:t_0-1}, \theta_{enc}, \theta_{dec}) = \prod_{t=t_0}^{\tau} p(x^t|h^t, \lambda^t, \theta_{dec}) \tag{5}$$

where h^t denotes the LSTM hidden at x^t that is autoregressively derived from the previous step. Next, we explain how to compute the density function mentioned in equation 5 using a conditional RealNVP flow model.

2) Conditional RealNVP: Note that while the LSTM requires a three-dimensional input, such an input cannot directly fit into the flow model. As a result, we begin by flattening the time dimension. Recall that our goal is to learn a set of bijective functions that enable transformation between a simple distribution and the real-world data distribution, as mentioned in section III-B. We use a multivariate Gaussian distribution with a diagonal covariance matrix as the base distribution, which is a common choice for normalizing flow [25]. Let y^t denote an arbitrary latent representation learned as part of the transformation. RealNVP partitions a given x into two disjoint groups, one of which is unchanged and mapped to the $1:d$ dimensions, while the other part of x undergoes a transformation and is mapped to $d+1:D$ dimensions (see equation 2 in section III). To model the conditional distribution shown in equation 5, we concatenate h^t and λ^t with the unchanged part of y^t, forming the inputs of *st-networks* in each coupling layer (see figure 2). During the transformations, we use binary mask vectors to extract the changed and unchanged dimensions in y^t, where the unchanged dimensions are multiplied by ones and the other dimensions are multiplied by zeros. Note that the outputs of *st-networks* preserve the dimensions of $d+1:D$ using the inverse mask vector so that we can compute corresponding values smoothly.

We stack K coupling layers to ensure the flow models can perform adequate changeovers when modeling complicated distributions, corresponding to the second term in equation 3. We also place a bijective batch normalization (BN) layer after every coupling layer. Our design is motivated by prior work by Dinh et.al [8], who use BN layers to stabilize the training process. As the BN layer is essentially a linear function, it is invertible and the computation of the Jacobian is efficient.

3) Training and Inference: The flow and encoder-decoder models are trained together via minimizing the below loss function with the Adam [34] optimizer. Given a batch of sliding windows B, according to the optimization objective 3 and equation 5, the loss function is parameterized as:

$$\mathcal{L} = -\frac{1}{|B| \cdot (\tau - t_0 + 1)} \sum_{x^{t_0:\tau} \in B} \sum_{t=t_0}^{\tau} \log p_X(x^t | h^t, \lambda^t; \theta) \quad (6)$$

where θ denotes all trainable parameters in the workflow.

During inference, the procedure of anomaly detection is straightforward and computationally efficient after training. Given a sample x^t, we use the trained network to perform density estimation, and flag the point as an anomaly based on a exogenous threshold ϵ.

C. Segment-level Anomaly Diagnosis

We now have a general architecture that can detect anomalies from real-time congestion data. However, note that an anomalous data point x^t (say) consists of N dimensions, where each dimension corresponds to a road segment. This detection does not fully solve our problem; recall that our goal is to enable TMC operators focus their attention (e.g., secondary inspection of cameras and resource allocation) to a small subset of segments. However, N can still be large in practice. Now, given an anomalous time vector x^t, we describe how to diagnose an anomaly down to the granularity of an individual segment.

Consider $x^t =< x_1, x_2, ..., x_n >$ is a detected anomalous vector, we investigate the data distribution at time t by gathering historical data at the same period of $[t - \frac{\sigma}{2}, t + \frac{\sigma}{2}]$, where σ denotes a configurable window size. We assume the collected data have similar patterns as time t, thus forming a dataset for density estimation. Next, we train a density estimation model with a Gaussian kernel for each time series, then determine the density threshold using a split validation dataset that is not seen during training.

D. Supervised Anomaly Classification

Given the normalizing flow model (that can perform exact density estimation and efficient sampling) and the LSTM-EncDec model (that can capture temporal correlations), we can generate labeled synthetic data to train supervised classifiers for anomaly detection. The procedure of generating MTS sequences are as follows: we first provide a warm-up sequence (an initial context window) as the input of the encoder to produce the decoder's initial hidden states. Anomalies and normal samples are then sampled from a standard normal distribution and then transformed to the output space (with decoder hidden states as conditional inputs). Generated samples are reused as inputs of the next iteration until the desired time series length is reached. The samples can then be used to train a classifier. We use a multi-layer perceptron classifier in our analysis.

V. Experimental Evaluation

A. Data

1) Congestion Rate data: We use an INRIX[1] traffic mobility data collected for one year (2019) from the city of Nashville, Tennessee. The details are summarized in Table I. Specifically, this dataset contains estimated "real-time" harmonic mean flow speeds, free-flow (reference) speeds, and historical average speeds of 364 interstate road segments with a five-minute frequency. The congestion rate measurements are derived based on the equation 4. We impute missing values at a specific road segment by interpolating observations from nearby segments. If nearby segments also contain missing values, we impute by using historical averages.

Property	Values
# roadway segments	364
# records/segments	104832
collection period	2019-01-01 00:00 –2019-12-30 23:55
frequency	5 minutes

TABLE I: Details of the traffic dataset collected from Nashville TN

[1] https://inrix.com/

2) Synthetic Testing dataset: Given that the traffic data is unlabeled, we evaluate the proposed method using a synthetic dataset generated from ground-truth data between October and December 2019. First, we model the ground-truth MTS as a multivariate Gaussian distribution, whose parameters are learned from empirical observations. We sample from the multivariate Gaussian. Then, we randomly inject "point" and "contextual" anomalies at a fraction (α) of half-hour length time slices and in a fraction (β) of road segments respectively, where α and β are hyper-parameters that control the temporal and spatial distribution of anomalies. The motivation of injecting anomalies in a temporal manner is that traffic congestions are not instantaneous events in practice. Point anomalies are created by perturbing the values obtained from the first step by a factor drawn from a uniform distribution $\mathcal{U}(-g, +g)$, where g denotes the magnitude of congestion rate of the day. Contextual anomalies are introduced by flipping the time slices that have minimum and maximum hourly average values.

B. Baselines

In terms of MTS anomaly detection, we compare our approach against prediction-based (e.g., AR, DeepLog-LSTM) and reconstruction-based (e.g., AE, VAE, EncDec-AD) anomaly detectors. These models are briefly described as the following: i) AR [35] trains a linear auto-regression model for individual time series and compute the anomaly score by averaging the prediction errors across all time series; ii) AE [36] is an autoencoder model using MLP for encoder and decoder; iii) VAE is a variational autoencoder with MLP encoder and decoder; iv) EncDec-AD [9] replaces MLP layers in AE with LSTM layers; v) DeepLog-LSTM [17] employs a stacked LSTM model for MTS anomaly detection, whose anomaly score is the single-step prediction error. As the baselines do not support anomaly diagnosis at the segment level, we adopt the same KDE-based method as in the proposed method.

C. Model Configurations

The encoder and decoder consist of 2 LSTM layers. For clusters A-D and G, encoder LSTM layers consist of 128 and 64 hidden units (and the opposite for the decoder) [37]. The hidden size is changed to 64 and 32 for cluster E-F and H. The flow model consists of 10 interleaving bijection and Batch Normalization layers. The *st-network* of every bijection layer is formed with 2 MLP layers (128 hidden dimensions for clusters A-D, G, and 32 for clusters E-F, H), where the s_{net} is activated with the Tanh function, and the t_{net} uses the ReLU function [37]. The model is trained for a maximum of 300 epochs with a batch size of 64. Out of the training set, 30% is kept out as validation set for early stopping. All experiments are run on a single Nvidia TITAN X GPU (12GB) and the code implementations are based on the Tensorflow Keras library version 2.4.0 and Tensorflow Probability 0.11.0.

D. Experiment Setup

1) Clustering: The high dimensionality of the MTS make computing the DTW distance matrix (see section IV-A) time-consuming. Therefore, we sample 100 segments uniformly at random, then use one week of data from the segments to generate the cluster prototype. Then, we calculate the DTW distances between the remaining segments and centroids of initialized clusters and merge them into the nearest cluster.

2) Anomaly detection and diagnosis on synthetic testing dataset: Our model is trained with the congestion data from Jan-2019 to Sep-2019 and evaluated on synthetic data of the remaining months. We empirically configure the sliding window size to 72 (6 hours) and the moving step length to 12 (1 hour). Point and contextual anomalies are detected together for all experiments. We evaluate the anomaly detection performance from two perspectives: effectiveness (based on temporal parameter (α) and sensitivity (based on spatial parameter (β). Effectiveness measures whether anomalies can be found in the case of high imbalance between anomalous and normal data. Sensitivity, on the other hand, evaluates situations in which only a portion of road segments are under anomalous congestion at a specific time, which challenges our approach to capture anomalies with high sensitivity. For each α and β pair, we generate the synthetic test set five times for all clusters and calculate the thorough average model performance.

3) Supervised Classification: For each individual cluster, we generate five one-month long datasets. Normal and anomolous data are sampled from the overall flow architecture. Then for each cluster, we train an MLP classifier with the binary cross-entropy loss and Adam optimizer. Trained MLP classifiers are evaluated on synthetic datasets using the area under the curve (AUC) score metric.

E. Results and Discussion

Clustering The MTS clustering step groups the 364 road segments into eight clusters that include 55, 67, 68, 79, 10, 12, 62, and 11 road segments, respectively. We name the clusters using the letters *A-H*. Empirical results show that roadways in the same cluster usually have similar functions or properties. For instance, Cluster A mainly covers Exit road segments. Cluster B involves the highways (e.g., I-65, I-40) that connect Nashville towards neighboring cities. Cluster C consists of road segments around on-ramps. Experimental results for individual clusters can be found in our Github repository.

Anomaly Detection First, we compare CondRealNVP with baseline methods for anomaly detection from aspects of effectiveness and sensitivity. The former is achieved by configuring the fraction of abnormal time slices α to 5%, 3%, and 1%, and the latter is by setting the fraction of anomalous road segment β to 100%, 50%, and 25%. We conducted controlled experiments and fixed β to 50% when testing the effectiveness and configure α as 5% for the sensitivity test. These settings ensure the sparsity of irregular traffic congestion in temporal and spatial. The effectiveness test results shown in Table II reflect that CondRealNVP consistently outperforms

other methods, with average improvements of 0.203–0.335 and 0.154–0.212 in terms of average Recall and F1-Score. There is a clear trend that the Recall and F1-Score degrade as the decreasing α; however, our approach is relatively more robust and guarantee acceptable performance even in the case of $\alpha = 1\%$. We also observe that CondRealNVP identifies anomalous congestion situation more sensitively than baselines in the cases of a small portion of road segments are congested (β is small). Except the situation where all road segments in a cluster suffer heavy congestion in a specific time slice ($\beta = 100\%$), we observe that the *CondRealNVP* model comprehensively outperforms the other approaches.

metrics	Recall			F1-Score		
anomaly rate α	5%	3%	1%	5%	3%	1%
AR	0.376	0.343	0.265	0.446	0.401	0.292
AE	0.576	0.504	0.359	0.475	0.421	0.295
VAE	0.574	0.518	0.362	0.490	0.435	0.307
EncDec-AD	0.529	0.472	0.358	0.466	0.401	0.272
DeepLog-LSTM	0.411	0.345	0.278	0.439	0.374	0.246
CondRealNVP	**0.752**	**0.710**	**0.604**	**0.632**	**0.583**	**0.480**

TABLE II: Avg. Recall and F1-score of the effectiveness test across 8 clusters under different anomaly rates ($\beta = 50\%$). Best results are presented in bold.

metrics	Recall			F1-Score		
road segments β	100%	50%	25%	100%	50%	25%
AR	0.482	0.376	0.275	**0.580**	0.446	0.241
AE	0.446	0.576	0.504	0.516	0.475	0.369
VAE	0.459	0.574	0.486	0.541	0.490	0.370
EncDec-AD	0.468	0.529	0.462	0.519	0.466	0.342
DeepLog-LSTM	0.429	0.411	0.359	0.530	0.439	0.288
CondRealNVP	**0.530**	**0.752**	**0.648**	0.575	**0.632**	**0.484**

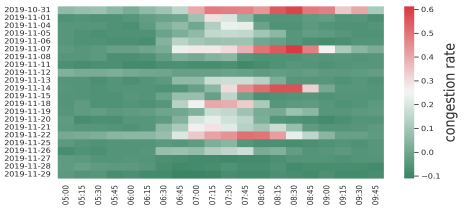
TABLE III: Avg. Recall and F1-Score of sensitivity test under different ratios of anomalous road segments ($\alpha = 5\%$).

Segment-Level Detection Given cluster level anomaly detection, we now evaluate the accuracy of performing segment level detection. We present the experimental results in Table IV. It can be seen that the overall trend coincides with what we observed in the sensitivity test. Specifically, CondRealNVP obtains 0.102–0.184 and 0.031–0.078 improvement regarding Recall and F1-Score compared with baseline methods. Finally, recall that anomaly detection for traffic centers is intended in near real-time. On average, for each cluster, inference (including the time taken to train KDE models) takes around 34 milliseconds, which is an acceptable latency in practice.

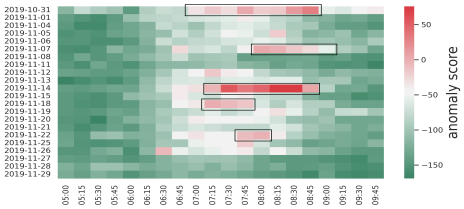
metrics	Recall			F1-Score		
road segments β	100%	50%	25%	100%	50%	25%
AR	0.459	0.238	0.126	0.570	0.341	0.178
AE	0.401	0.322	0.210	0.491	0.345	0.221
VAE	0.422	0.333	0.211	0.517	0.352	0.218
EncDec-AD	0.427	0.300	0.192	0.507	0.341	0.211
DeepLog-LSTM	0.395	0.242	0.156	0.514	0.333	0.199
CondRealNVP	**0.520**	**0.508**	**0.282**	**0.572**	**0.420**	**0.245**

TABLE IV: Avg. Recall and F1-Score for segment-level anomaly diagnosis with $\alpha = 5\%$.

Visualization of real-world traffic anomaly We also show a case study on real-world data to evaluate our approach. We use real congestion data from cluster G for weekdays between Oct. 31st and Nov. 29th in Q4 2019, which is not seen during training. Figure 3 shows (a) the 85th percentile congestion rate (average in 15 minutes) for all segments in the cluster and (b) the corresponding average anomaly score assigned by CondRealNVP. We select the period between 5 AM and 10 AM, which generally covers rush hours of the roadway (e.g., interstates 65) in Cluster G. The 85th percentile congestion intensity indicates the extent of congestion in the cluster, implying that only 15% road segments are under heavier congestion states than presented. It can be seen that anomalous congestion occurred at 10/31, 11/07, 14, 18, and 22 with an apparent cascading pattern. As expected, our approach successfully discovers the peak hours and assigns notable anomaly scores from early stages.



(a) The 85th percentile congestion rate (average in 15 minutes)



(b) Avg. anomaly scores per 15 minutes

Fig. 3: Visualization of real-world traffic anomalies and anomaly scores. Boxes highlight the most noticeable periods that are likely under abnormal congestion. Heatmaps show the results of Cluster G from 5 AM to 10 AM (rush hours) for 22 working days. Our method sensitively captures the time periods when recurring congestion occurred.

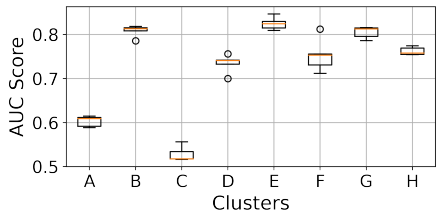


Fig. 4: Average AUC score for the MLPClassifiers on 5 synthetic testing datasets. Boxplot shows the performance variation of MLPClassifiers trained on 5 datasets drawn from CondRealNVP.

Supervised Classification Finally, we evaluate the efficacy of learning a classifiers in a supervised setting using samples drawn from CondRealNVP. The classifiers are evaluated on five synthetic testing datasets, as we conducted in the previous section, with $\alpha = 5\%$ and $\beta = 50\%$. We report the average AUC score in Figure 4. One can see that classifiers have acceptable discrimination capability (AUC score ≥ 0.7) in 6 of the 8 clusters. The relatively lower AUC scores in clusters A (off-ramp/Exit segments) and C (on-ramp segments) are probably due to the extremely low volume of abnormal congestion data in such clusters.

VI. Conclusion

In this paper, we present an end-to-end framework to address the problem of citywide traffic congestion anomaly detection and real-time anomaly diagnosis. Road segments congestion rate is formulated as multivariate time series. In the framework, we identify clusters of segments and use a conditional normalizing flow model for every cluster that combines an LSTM Encoder-Decoder network with a RealNVP model for density-based anomaly detection. Then, we perform KDE-based real-time anomaly diagnosis to locate anomalous road segments in the anomalous clusters. Extensive experiments conducted on synthetic datasets manifest that our approach significantly outperforms several state-of-art methods for both anomaly detection and anomaly diagnosis. The proposed approach also demonstrates reliable performance in detecting real-world traffic congestion. In the future, we plan to (1) integrate geographical information during the segments clustering process; (2) integrate attention mechanism with the LSTM networks to differentiate the importance of road segments when deriving anomaly score at a particular time; and (3) explore other normalizing flow models.

References

[1] Y. Senarath, A. Mukhopadhyay, S. Vazirizade, H. Purohit, S. Nannapaneni, and A. Dubey, "Practitioner-centric approach for early incident detection using crowdsourced data for emergency services," in *IEEE International Conference on Data Mining*, 2021, pp. 1318–1323.

[2] G. Pettet, H. Baxter, S. M. Vazirizade, H. Purohit, M. Ma, A. Mukhopadhyay, and A. Dubey, "Designing decision support systems for emergency response: Challenges and opportunities," *arXiv preprint arXiv:2202.11268*, 2022.

[3] M. Wilbur, A. Dubey, B. Leão, and S. Bhattacharjee, "A decentralized approach for real time anomaly detection in transportation networks," in *IEEE International Conference on Smart Computing*, 2019, pp. 274–282.

[4] F. Harrou, A. Zeroual, and Y. Sun, "Traffic congestion monitoring using an improved knn strategy," *Measurement*, vol. 156, p. 107534, 2020.

[5] P. Mercader and J. Haddad, "Automatic incident detection on freeways based on bluetooth traffic monitoring," *Accident Analysis & Prevention*, vol. 146, p. 105703, 2020.

[6] H. Z. Moayedi and M. Masnadi-Shirazi, "Arima model for network traffic prediction and anomaly detection," in *2008 International Symposium on Information Technology*, vol. 4, 2008, pp. 1–6.

[7] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*, 2015, pp. 1530–1538.

[8] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[9] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[10] Y. Hu, A. Qu, and D. Work, "Graph convolutional networks for traffic anomaly," *arXiv preprint arXiv:2012.13637*, 2020.

[11] C. Madarash-Hill and J. Hill, "Enhancing access to ieee conference proceedings: a case study in the application of ieee xplore full text and table of contents enhancements," *Science & Technology Libraries*, vol. 24, no. 3-4, pp. 389–399, 2004.

[12] A. Chevrot, A. Vernotte, and B. Legeard, "Cae: Contextual auto-encoder for multivariate time-series anomaly detection in air transportation," *Computers & Security*, p. 102652, 2022.

[13] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[14] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.

[15] S. Lin, R. Clark, R. Birke, S. Schönborn, N. Trigoni, and S. Roberts, "Anomaly detection for time series using vae-lstm hybrid model," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 4322–4326.

[16] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89, 2015, pp. 89–94.

[17] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.

[18] D. Nam, R. Lavanya, R. Jayakrishnan, I. Yang, and W. H. Jeon, "A deep learning approach for estimating traffic density using data obtained from connected and autonomous probes," *Sensors*, vol. 20, no. 17, p. 4824, 2020.

[19] R. L. Abduljabbar, H. Dia, and P.-W. Tsai, "Development and evaluation of bidirectional lstm freeway traffic forecasting models using simulation data," *Scientific reports*, vol. 11, no. 1, pp. 1–16, 2021.

[20] S. Basak, A. Dubey, and L. Bruno, "Analyzing the cascading effect of traffic congestion using lstm networks," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 2144–2153.

[21] W. Wu, L. He, W. Lin, Y. Su, Y. Cui, C. Maple, and S. A. Jarvis, "Developing an unsupervised real-time anomaly detection scheme for time series with multi-seasonality," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[22] M.-F. Chiang, E.-P. Lim, W.-C. Lee, and A. T. Kwee, "Btci: A new framework for identifying congestion cascades using bus trajectory data," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 1133–1142.

[23] M. L. Dias, C. L. C. Mattos, T. L. da Silva, J. A. F. de Macedo, and W. C. Silva, "Anomaly detection in trajectory data with normalizing flows," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[24] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[25] I. Kobyzev, S. Prince, and M. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[26] K. K. Dixon, C.-H. Wu, W. Sarasua, and J. Daniel, "Posted and free-flow speeds for rural multilane highways in georgia," *Journal of Transportation Engineering*, vol. 125, no. 6, pp. 487–494, 1999.

[27] H. Park, A. Haghani, and M. Hamedi, "Quantifying non-recurring congestion impact on secondary incidents using probe vehicle data," Tech. Rep., 2013.

[28] A. Pankaj, "A comprehensive study on the estimation of freeway travel time index and the effect of traffic data quality," Ph.D. dissertation, The University of Wisconsin-Milwaukee, 2019.

[29] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[30] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[31] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, USA:, 1994, pp. 359–370.

[32] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] K.-H. Lai, D. Zha, G. Wang, J. Xu, Y. Zhao, D. Kumar, Y. Chen, P. Zumkhawaka, M. Wan, D. Martinez *et al.*, "Tods: An automated time series outlier detection system," *arXiv preprint arXiv:2009.09822*, 2020.

[36] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2002, pp. 170–180.

[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.