CartaGenie: Context-Driven Synthesis of City-Scale Mobile Network Traffic Snapshots

Kai Xu*,†, Rajkarn Singh*,†, Hakan Bilen†, Marco Fiore‡, Mahesh K. Marina†, Yue Wang◇

**denotes equal contributions*

The University of Edinburgh†, IMDEA Networks Institute‡, Samsung◇

ABSTRACT

Mobile network traffic data offers unprecedented opportunities for innovative and transformative studies within and beyond mobile networking. However, progress is hindered by a limited access to the real-world mobile network data that is needed to develop and dependably test network data-driven solutions. As a contribution to overcome this barrier, we propose CartaGenie, a generator of realistic snapshots of mobile traffic. Taking a deep generative modeling approach and through a tailored conditional generator design, CartaGenie can synthesize high fidelity, artefact-free, city-level spatial traffic using only contextual information about the target geographical region that is easily found in public repositories. Experiments with real-world traffic measurement data collected in multiple metropolitan areas show that CartaGenie can produce dependable network traffic loads for areas where no prior traffic information is available, significantly outperforming a comprehensive set of state-of-the-art benchmarks. Moreover, tests with practical case studies demonstrate that the synthetic data generated by CartaGenie is as good as real data in supporting diverse data-driven applications.

1 INTRODUCTION

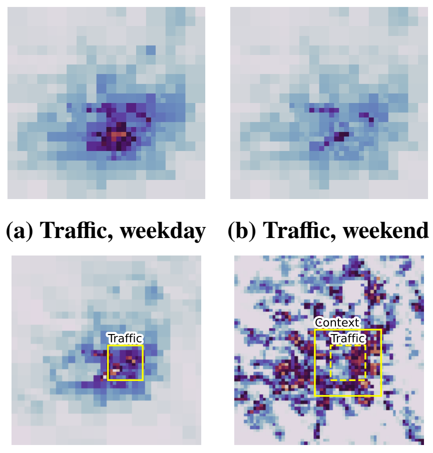
Mobile devices have reached very high penetrations, and the coverage of wireless networks is nearly ubiquitous. As a result, the network traffic generated by such devices over space and time, and the associated consumption of mobile services, offer an unprecedented source of information on individual- and population-level patterns of activities and movement [17, 49, 58]. Unsurprisingly, this potential has led to many diverse applications, including: modelling human mobility and migration patterns [11, 13, 33, 34]; inferring commuting patterns, crowding and inequities [18, 21, 41, 65]; monitoring demographic patterns [19, 22, 24, 70]; detecting land use and its dynamics [28]; planning transportation systems and urban space [55]; and monitoring road traffic [36, 56]. Equally, there are numerous applications enabled by mobile network traffic data that broadly aim at optimizing and automating mobile network operations, *e.g.*, by obtaining insights for future infrastructure deployment [20], improving energy efficiency of mobile network infrastructure [67], or optimizing next-generation mobile network paradigms [14].

Despite the breadth and importance of these applications, access to mobile network traffic data is very limited. When access is granted, this happens under restrictive non-disclosure agreements (NDAs) with the data owners, which limit the analyses to narrowly defined scopes and prevent further circulation of the data. This is also the case for aggregated datasets reporting the total traffic generated by all users associated to a same base station, which enable many of the aforementioned applications. Even though such data is de-personalized (*e.g.*, in the sense of the European GDPR [2]) from being geographically aggregated data over a large numbers of users, its sharing is still curbed by mobile operator concerns on potential leakage of commercially sensitive information, which ultimately hinders data-driven research and innovation.

In this paper, we look at synthetic data generation as an effective way to lower the access barrier to mobile traffic data. In particular, we aim at synthetically generating realistic (and open) *snapshots* of mobile traffic over any target region from contextual information that is easily obtained via public repositories, such as demographic and land use characteristics.

Attaining the above goal however poses significant challenges. First, mobile traffic data exhibits hard-to-model characteristics such as spatial correlations and skewness [61, 63] and traffic pattern fluctuations over time [43], as illustrated in Figures 1a and 1b. Second, the available contextual attributes (interchangeably referred to as 'conditions' in the rest of the paper) can only partially explain the traffic in the corresponding region, which is also driven by additional unobserved latent factors. Third, the dependence of mobile traffic on conditions is not deterministic, but inherently stochastic. Fourth, regions of interest for traffic generation (*e.g.*, cities) typically have different spatial dimensions, which gives rise to the challenge of synthesizing varied size traffic snapshots.

To solve the problem at hand, we present CartaGenie (§2 and §3), a novel method rooted in deep generative modelling



(a) Traffic, weekday (b) Traffic, weekend

(c) Traffic patch (d) Context patch

Figure 1: (a), (b) Spatiotemporal variation of mobile network traffic in City A from our dataset. (c) Example of a mobile traffic patch, for which the generation process is handled independently. (d) The outer square outlines the wider context patch for the traffic patch in (c). Here only one context attribute (population density) is shown.

for high-fidelity and generalizable synthesis of city-scale mobile traffic snapshots from contextual input. CartaGenie casts spatial traffic synthesis as a strongly conditioned generation problem, and solves it via a tailored deep convolutional neural network (CNN) architecture design. Specifically, CartaGenie incorporates several innovations to address the aforementioned challenges, as follows:

- The CNN design underlying CartaGenie naturally models spatial characteristics of mobile traffic while capturing temporal variations is facilitated through 'time period' as an attribute in the context input.
- The inherent stochasticity of traffic and the latent factors that affect mobile demands beyond the observable context are jointly modeled via a *latent variable* as an additional input to CartaGenie generator. Furthermore, CartaGenie employs a specialized FiLM *conditioning layer* [53] to ensure that the aforementioned stochasticity is captured.
- The synthesis of traffic for regions of any geographic size is made possible by traffic generation at smaller sized *patch* level (*e.g.*, urban blocks rather than whole cities), as illustrated in Figure 1c. To ensure that no artefacts are created when sewing up the separately generated traffic patches into a region-wide traffic snapshot, each traffic patch is associated with a sufficiently wider context patch, as illustrated in Figure 1d.

We extensively evaluate CartaGenie to assess the fidelity of the synthetic traffic, its generalizability, and its utility for different downstream tasks. Our experiments are based on a multi-city real-world traffic measurement dataset from a major European operator, augmented with contextual attributes from public sources (§4). Our results (§5 and §6) show that:

- CartaGenie provides significantly superior city-level fidelity compared to a range of state-of-the-art benchmarks, including image generation in computer vision [35] and the only existing but rudimentary mobile traffic generation approach in [20].
- CartaGenie can effectively generalize by synthesizing traffic snapshots for new unseen cities that closely mimic the corresponding real data, solely based on contextual inputs.
- Two different case studies demonstrate that the use of synthetic traffic data with CartaGenie provides nearly indistinguishable results from those derived with real data, thereby proving the effectiveness of CartaGenie as an enabler of data-driven applications within and beyond mobile networking.

From a wider perspective, our work falls in the growing body of work on synthetic data generation in various domains, including computer vision, health, government, and art [8, 9, 12, 51, 62]. Our experience from this work however highlights the strong influence that the generation task and domain have on designing effective methods. In the networking domain, synthetic data generation is at its infancy. Early works have focused on time series [42], or fail to faithfully capture spatiotemporal variations in mobile traffic patterns [20]. We make the first steps towards the generation of dependable and accessible spatial mobile traffic, which can unlock data-driven solutions in mobile networking and beyond. Upon publication of this work, we intend to make generated datasets with CartaGenie publicly available.

2 OVERVIEW

2.1 Problem Statement

Let $\mathcal{X} = \{X^1, X^2, \dots, X^N\}$ be a real-world mobile network traffic measurement dataset that contains N sets of observations of mobile traffic maps, such that each set is collected in a different geographical region. For the sake of concreteness, in this work we will consider whole cities as the regions of interest, and use the terms region and city interchangeably; however, the approach we propose is general, and applies to other definitions of region. The data for each city $n \in \{1, \dots, N\}$ includes observations over a span of time T^n, hence $X^n = \{\boldsymbol{x}_1^n, \boldsymbol{x}_2^n, \dots, \boldsymbol{x}_{T^n}^n\}$. In particular, we represent every mobile traffic observation $\boldsymbol{x}_t^n \in \Re^{H_x^n \times W_x^n}$ as a single channel image, whose pixels map to the spatial units over which network traffic is recorded. Then, each pixel value corresponds to the network traffic load recorded at that geographic location; and, H_x^n and W_x^n are the height and width of the observations in pixels, respectively, which may differ among cities.

Each observation $\boldsymbol{x}_t^n$ is associated with a set of A auxiliary observations (or *conditions*), *i.e.*, attributes that may partly explain the volume of traffic generated by mobile users (*e.g.*,

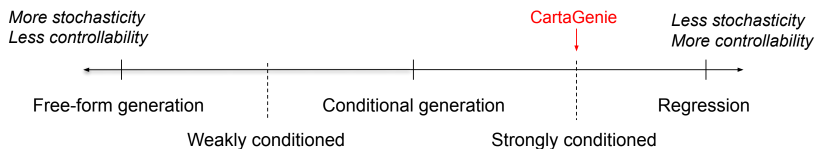


Figure 2: Spectrum of controllable data synthesis methods, and positioning of CartaGenie in this spectrum.

population distribution in the region, land use characteristics, or presence of points of interest). We denote the set of auxiliary observations for each city n as its *context*, and represent it as the set $C^n = \left\{\boldsymbol{c}_1^n, \boldsymbol{c}_2^n, \dots, \boldsymbol{c}_{T^n}^n\right\}$. Here, $\boldsymbol{c}_t^n = (\boldsymbol{v}^n, w_t)$, where $\boldsymbol{v}^n \in \Re^{A \times H_c^n \times W_c^n}$ is a multi-channel image with one channel per attribute, and H_c^n and W_c^n are the height and width of the context in pixels, respectively; whereas, w_t is a time context attribute that captures the time dimension t of the context $\boldsymbol{c}_t^n$ as a discrete variable. Note that the auxiliary observations alone are typically insufficient to explain the corresponding network traffic, also because we constrain them to be easily available from public repositories. For example, there can be multiple plausible traffic loads that have the same auxiliary observations, which we illustrate in the case of our reference datasets in §4.2.2.

Our goal is to *generate* synthetic network traffic data $\boldsymbol{s}_t^m \in \Re^{H_s^m \times W_s^m}$ for a previously *unseen* region m at a specified time t and given auxiliary observations $\boldsymbol{c}_t^m$, in a way that the synthetic samples $\boldsymbol{s}_t^m$ exhibit similar characteristics as the real training data $\mathcal{X}$ and are compatible with the provided $\boldsymbol{c}_t^m$.

2.2 Requirements

Solving the problem set out in §2.1 above entails learning to generate realistic large city-scale traffic data with significant variability from context attributes and limited amount of training data. Two requirements need to be implicitly met by the solution. The first is *generalization*: by enforcing that the synthesis targets new regions (cities in our case), for which no traffic information is available, we inherently require that the data synthesis process generalizes to areas different from those seen during model training. It is thus essential that the synthesized data is not simply a copy of the existing real data, rather it is novel and realistically adapted to the target region. The second requirement is *controllability*, *i.e.*, the ability to control the generation process. Specifically, the method should allow users to modify the generated network traffic by varying certain properties of the target region (*e.g.*, the population density, or the distribution of business areas).

From a data generation perspective, the degree of required controllability can be categorized as in Figure 2. On the right-end of the spectrum, generation is completely controllable, and the real data $\mathcal{X}$ (used for training) can be fully explained by the conditions C such that data synthesis is completely deterministic: synthesizing $\boldsymbol{s}$ using $\boldsymbol{c}$ is then essentially a classical regression problem. We experimentally show in §5 that a synthesis based on a full controllability assumption performs poorly in our case, as it cannot capture the complex and partly stochastic nature of city scale mobile traffic patterns. On the left end, the assumption is that there are no conditions to control the synthesis, and the generation aims at recreating real data distributions from a noise input. Popular methods to address this problem include Generative Adversarial Networks (GANs) [26] and Variational Autoencoders (VAEs) [40, 59] that are purely stochastically driven. In §5, we show that neglecting conditions also yields poor results, as it discards valuable information and lacks controllability.

2.3 Proposed Solution: CartaGenie

Referring to Figure 2, our approach to controllability and stochasticity falls in between the two extremes, in the domain of *conditional generation*. Here, available conditions can only partially explain the data to be generated. However, there are unexplained parts, either due to missing conditions or stochasticity, that we want our synthesizer to model. While conditional variants of GANs and VAEs [38, 47] are the standard tools to model unexplained latent variables, we propose an original and ultimately more effective design, outlined next.

Our proposed method CartaGenie is a conditional deep generator based on a tailored convolutional neural network (CNN) design that can naturally capture spatial correlations in traffic. To model unobserved conditions and stochasticity, we introduce a noise (or latent) variable $z \in \Re^{D_z}$ as an extra input to the generator, along with the context $\boldsymbol{c}$, the latter for controlling the generated output. Note that $\boldsymbol{c}$ includes the time context attribute, which allows generating traffic snapshots for various time periods with different traffic patterns (*e.g.*, compare Figures 1a and 1b). The generator $\mathcal{G}_\theta : \boldsymbol{c}, z \rightarrow \boldsymbol{x}$ is modelled as a CNN with weights θ. This model is trained with available real traffic data $\mathcal{X}$ and the corresponding set of conditions C. It can then take conditions for a target region along with latent variable z as input, and generate a synthetic mobile traffic map sample for that region.

Figure 3 shows a schematic of CartaGenie conditional generator architecture. An important remark is that $\boldsymbol{c}$ is a combination of spatial ($\boldsymbol{v}$) and non-spatial (w) data, whereas z is also non-spatial. In CartaGenie, the non-spatial inputs z and w are processed via a specialized FiLM conditioning layer [53], as highlighted in Figure 3. This design avoids the risks of a naive conditioning on the latent variable (*i.e.*, simply concatenating $\boldsymbol{c}$ and z), which is known to cause weak conditioning that

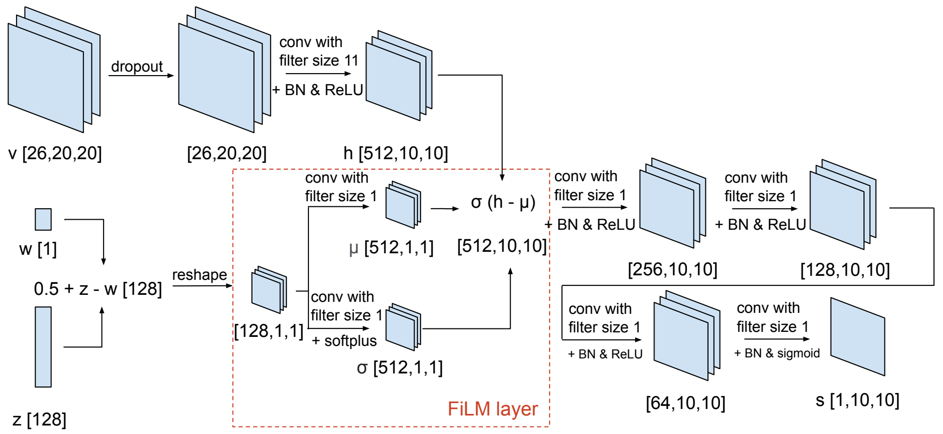


Figure 3: Schematic of CartaGenie neural network architecture. Annotations show tensor sizes and activations.

can lead to the neural network completely ignoring stochasticity [35, 45]. Instead, the specialized intermediate conditioning layer ensures that the latent variable is duly accounted for in the generation process. Also, it provides an elegant way of integrating the time context as a noise modulator, as detailed in the next section. The result of the separate convolutions along spatial and non-spatial inputs are then merged via an affine transformation into a hidden representation, which is then processed by stacked convolution layers with size-1 kernels (i.e. convolutions that process each pixel independently), and batch normalization (BN) layers in between, to produce the final sample $\boldsymbol{s}$.

To allow generation of mobile traffic snapshots for different cities varying in their spatial dimensions through a single generator model, the generation in CartaGenie is done at a smaller *patch* level. Doing so has the additional advantages in terms of efficient training and avoiding overfitting, as elaborated in the next section. But care needs to be taken to avoid artefacts when sewing up generated patches to make up the city level traffic map. To this end, CartaGenie takes a conditionally independent pixel level generation and ensures for each traffic patch the corresponding context patch is sufficiently wider, as illustrated in Figure 1d; artefacts-free generation then naturally results from the Markov blanket property [10].

The learning of the above outlined model is done by minimizing the expected reconstruction error in the synthesized traffic maps relative to the corresponding real ones. This reconstruction error is based on L1 distance (sum of absolute differences between individual pixel values) between real and synthesized traffic maps. The model optimization is performed via stochastic gradient descent (using mini-batches over different city, time and patches). Note the the real traffic data for a few cities (along with corresponding context information) is only used for training the CartaGenie model to learn the weights. At generation time, the model solely relies on provided inputs on context for the target city and latent variable, as reflected by Figure 3.

3 DETAILED DESIGN

CartaGenie is effectively a conditional neural sampler $\mathcal{G}_\theta : \boldsymbol{c}, z \to \boldsymbol{s}$, *i.e.*, a neural network that learns a mapping from conditions $\boldsymbol{c}$ and a random, noise vector z to produce a realistic mobile traffic map sample $\boldsymbol{s}$. We now detail each component of the CartaGenie architecture depicted in Figure 3.

3.1 Latent Variable Modeling and Handling Time Input via the FiLM Layer

In designing our conditional generator, we face two issues. The first issue arises from how the latent variable (noise vector) z is dealt with. If we simply concatenate it (after reshaping) to the conditions $\boldsymbol{c}$, the generator tends to ignore the noise vector during training and leads to deterministic outputs given $\boldsymbol{c}$. Related work also found this issue [35, 45], and proposals to instead use dropout layers as a source of stochasticity were found not effective either [60]. In CartaGenie, we address this issue by using the feature-wise linear modulation (FiLM) conditioning layer [53]. When adapted to our case, the FiLM layer takes two inputs $\boldsymbol{h} \in \Re^{N_h \times W_h \times H_h}$ and $z \in \Re^{N_z \times 1 \times 1}$, where $\boldsymbol{h}$ is an intermediate hidden representation of the spatial context $\boldsymbol{v}$ that is encoded with our model. It first uses a layer of neural network that transforms z to $\boldsymbol{\mu} \in \Re^{N_h \times 1 \times 1}$ (without activation) and $\boldsymbol{\sigma} \in \Re^{N_h \times 1 \times 1}$ (with the softplus $f(x) = \log(1 + \exp(x))$ activation to ensure $\boldsymbol{\sigma}$ is positive). Then, an affine transformation is performed to $\boldsymbol{h}$ as $\boldsymbol{h} \leftarrow \boldsymbol{\sigma}(\boldsymbol{h} - \boldsymbol{\mu})$. Notice that all individual pixels for each channel (corresponding to each spatial context attribute) are transformed using the same shift and scale, which means our z effectively models some global variation. Unlike a naive approach of concatenating $\boldsymbol{c}$ and z, this specific layer for conditional modelling has two benefits: (1)

there is no need to manually convert z, which is a non-spatial variable, into a spatial one in order to perform channel-wise concatenation; (2) this prevents the convolution layers from simply ignoring the noise vector, which is a common issue when naive concatenation is performed.

The second issue is that we not only have spatial conditions $\boldsymbol{v}$ but also time as a condition w, which is non-spatial and is thus the same for all pixels of the traffic and spatial context. A naive way to address this is, again, to reshape the non-spatial condition w into a spatial one, by creating a map of pixels with value w and size matching to that of $\boldsymbol{v}$, and concatenating it to $\boldsymbol{v}$. We aim at a better solution to this issue, and use the non-spatial condition to modulate the latent variable z into a W-modal distribution, where W denotes the number of values that w can take. While in general w can take any number of discrete values, for the sake of concreteness, we consider the case where w is a binary variable (*e.g.*, working day or weekend day). Hence we use the transformation $z \leftarrow 0.5 + z - w$. When z is a *unimodal* distribution, this effectively makes the input to the generator a *bi-modal* distribution so that the model can learn different behavior depending on which of the two values w takes (*e.g.*, weekday traffic pattern vs. weekend traffic pattern). Note that this does not limit the data generation capability of our model.

3.2 Artefact-Free Patch-Level Traffic Snapshot Generation

In CartaGenie, we perform model training and traffic generation at smaller patch level. For training, to create the samples, we divide the $H_x^n \times W_x^n$ pixels of the traffic data $\boldsymbol{x}_t^n$ of each city n in the training data into smaller *traffic patches* $\boldsymbol{x}_t^{n,l}$, $l \in \{1, \ldots, L\}$. Figure 1c illustrates one sample patch in one of the cities from our dataset. This approach has several advantages. On the one hand, different cities have diverse geographical span, hence their traffic maps have varied spatial dimensions. Using fixed smaller sized traffic patches as we do allows using the same generator model architecture regardless of the city dimensions considered for training or generation. It also allows using diverse traffic patches from different snapshots together to enable a more efficient training via stochastic gradient optimization. Moreover, different local sub-regions of a same city share similar patterns between traffic and spatial conditions: training at the level of patches can be then seen as a form of *weight-sharing* – a type of *regularization* technique – to enforce the model to learn the actual casual relationship between $\boldsymbol{c}$ and $\boldsymbol{x}$ instead of memorizing the mapping. In our experiments, we set $\boldsymbol{x}_t^{n,l} \in \Re^{N_x \times N_x}$ with $N_x = 10$, *i.e.*, a 0.25×0.25 km^2 area covered by each patch.

When considering a traffic patch $\boldsymbol{x}_t^{n,l}$, only the portion of the city-wide spatial context $\boldsymbol{v}^n$ that is in the geographical proximity of the patch stays relevant to the learning process. We thus associate to each $\boldsymbol{x}_t^{n,l}$ a trimmed spatial context (which we call context patch) $\boldsymbol{v}^{n,l}$ (identical across all spatial context attributes) that includes a margin around the traffic patch, as exemplified in Figure 1d. The margin is required to ensure that the context surrounding pixels at the border of $\boldsymbol{x}_t^{n,l}$ is properly considered during the learning process, as discussed further below. Clearly, the number of context patches is the same as that of traffic patches, *i.e.*, L, and we denote by $\boldsymbol{c}_t^{n,l} = (\boldsymbol{v}^{n,l}, w_t)$ the complete spatiotemporal context corresponding to $\boldsymbol{x}_t^{n,l}$. In our experiments, we set $\boldsymbol{v}^{n,l} \in \Re^{N_c \times N_c}$ with N_c=20. This is to ensure artefact free generated traffic snapshots as discussed below and also empirically supported by the parametric analysis in §F.2. Consistently with the patch level real training data, CartaGenie outputs synthetic data for each traffic patch in the target city map, *i.e.*, $s_t^{n,l} \in \Re^{10\times 10}$.

Conditionally independent pixel level generation to avoid artefacts. While patch-level synthetic traffic data can be accurate locally, it has a potential downside when used to recreate city-level traffic maps, as artefacts may appear at the boundary of patches that are separately generated. To address this concern, we take a *conditionally independent pixel-level generation* approach. This may seem counter-intuitive given that we expect pixels (especially the nearby ones) in a generated traffic map to be correlated. Our approach, however, is in fact consistent with this expectation and is rooted in the property of Markov blanket [10], which in our setting translates to: if the context patch corresponding to a traffic patch is large enough to cover the Markov blanket of any pair of nearby traffic pixels, they would be *conditionally independent* even when the marginal distribution of these traffic pixels may be correlated. To see this, we need to zoom in and look at the pixel level of data. Denoting individual pixels in $\boldsymbol{x}$ and $\boldsymbol{v}$ as $p_{i,j}^{\boldsymbol{x}}$ and $p_{i,j}^{\boldsymbol{v}}$, where i, j are spatial indices, at a specified time t (which we omit for simplicity), we assume the underlying generative process for each $p_{i,j}^{\boldsymbol{x}}$ is $\{v_{i',j'}\}_{(i',j')\in\mathcal{B}(i,j)}, z \longrightarrow p_{i,j}^{\boldsymbol{x}}$, where $\mathcal{B}(i,j)$ are the collection of indices for condition (spatial context) pixels that actually affect $p_{i,j}^{\boldsymbol{x}}$. In other words, we assume that only a local region of condition pixels have effect on traffic pixels, a reasonable assumption for our task. Given this, our approach essentially means the following: so long as the size of the context $\mathcal{B}(i,j)$ around the pixel $p_{i,j}^{\boldsymbol{x}}$ is large enough to cover its Markov blanket, it can be generated independently from the other pixels.

To implement our above described conditionally independent pixel generation idea, we need to ensure that: (1) for each traffic patch, a corresponding but larger context patch is used; and (2) the neural network generates each pixel independently given the context patch. (1) ensures that the traffic data is faithfully modeled by the generative process; and (2) *guarantees* that there would be no artefacts by design when generating city-level traffic maps by sewing together different generated local patches of the city. Therefore, in our architecture, we have the first convolution layer to map the context into the same shape of traffic patches by using a kernel size of $N_c - N_x + 1 = 11$. After the following FiLM layer, all convolutions layers have a kernel size of 1, which independently process the intermediate "hidden images" and output the synthetic traffic map.

3.3 Model Training and Optimization

Learning of CartaGenie is done by minimizing its expected reconstruction error as measured by L1 loss:

$$\mathcal{L}(\theta) = \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{l=1}^{L} \mathbb{E}_{z_t^{n,l} \sim \mathcal{N}(0,1)} \left\| \mathcal{G}_\theta(\boldsymbol{c}_t^{n,l}, z_t^{n,l}) - \boldsymbol{x}_t^{n,l} \right\|_1 \quad (1)$$

Here N, T and L, respectively, represent the number of cities, days and patches in the real data used for training. We use L1 loss as it has been successfully adopted in the literature [60] and also experimentally found it to better than other alternatives including L2 and scaled L1/L2. Also note that minimizing expected reconstruction error based on L1 loss corresponds to a maximum likelihood estimation of θ with a Gaussian prior on z and Laplace likelihood, which is detailed further in §A.

We *normalize* the traffic in the training data to fall between 0 and 1, as detailed in §B.2. Moreover, to better utilize the neural network capacity, we design it to output the generated traffic in log-space and then transform it back to the original domain, as detailed in §B.3.

To train our model, we use mini-batch stochastic gradient descent by the ADAM optimizer [39] with a learning rate of 0.0001 and β_1 of 0.5 for 25 epochs; other parameters are set as PyTorch's defaults. We use mini-batches over N, T and L to estimate (1) via Monte Carlo, which is a common practice in neural network training.

We take the latent vector dimension $D_z = 128$ for z. In terms of number of channels, $N_c \longrightarrow 8F \longrightarrow 4F \longrightarrow 2F \longrightarrow F \longrightarrow 1$ where the base number of features are $F = 64$. For the FiLM layer that processes z, it is $N_z \longrightarrow 16F$, which is twice as $8F$. All intermediate activations that process $\boldsymbol{v}$ are ReLU [50], following a batch normalization [32] layer, and the final activation is Sigmoid. To reduce over-fitting to the conditions, we add a channel-wise dropout layer [64] to the input conditions with a rate of 0.02. We report how the above hyper-parameter settings are selected in §F.5.

4 EVALUATION METHODOLOGY

4.1 Goals

With CartaGenie we aim to generate synthetic network traffic map data that (*i*) has high fidelity, *i.e.*, closely mimics the real-world network traffic distribution; (*ii*) generalizes well to unseen geographic regions; and (*iii*) can be put to use for practical networking and beyond networking applications. Our evaluations aim at assessing these requirements, and rely on real-world datasets (§4.2), suitable quantitative and qualitative performance metrics (§4.3), and a comprehensive set of benchmarks for comparative analysis (§4.4).

We first assess the fidelity and generalizability of CartaGenie by assessing its performance in synthesizing traffic in previously unseen cities (§5.1). We then compare such performance against those of benchmarks that include state-of-the-art approaches (§5.2). Finally, we demonstrate the utility of CartaGenie in two application use cases that rely on mobile network traffic snapshots, *i.e.*, network edge datacenter planning and understanding urban population dynamics (§6). Additional results from a thorough ablation study that prove the effectiveness of the design and parameterization choices underlying CartaGenie are presented in §F.

4.2 Data

4.2.1 Mobile Traffic Dataset. Our real-world mobile traffic dataset comes from a major mobile network operator in Europe. The dataset covers 4 major cities in one country (anonymously indicated as Cities A, B, C, and D, respectively, in the rest of the paper) and consists of mobile traffic load measurements for 47 days. Each city is divided into grid cells, each of size 250×250 m^2. The total volume of mobile data traffic measured in each cell is aggregated over 24-hour intervals. Different cities have different surface areas, resulting in 80×80 cells for both City A and City B, 128×96 cells for City C and 96×80 cells for City D.

4.2.2 Datasets for Conditional Attributes. We condition the synthetic traffic generation on attributes that provide contextual information about the region of interest. We specifically select attributes that are commonly available from public sources, so as to make the approach as reusable as possible.

Population. This attribute represents the number of residents in each grid cell, as reported in national census data. When comparing the heatmap of this attribute in Figure 4 with Figure 1a, we note that the population is correlated with traffic in non-obvious ways. For instance, a positive correlation exists in rural areas, whereas towns generate higher traffic volumes than the surrounding countryside; however, the correlation turns negative in the business-oriented city center where residents are infrequent, but work or tourism activities actually produce the highest traffic volumes [46].

Land use. These attributes represent the incidence of specific types of land use in each grid cell. Land uses capture the utilization of the territory, and highlight zones of the target region with different purposes. We extract information on the land use of four cities from the Copernicus Urban Atlas 2012 repository [1], and transform each land use type to an attribute

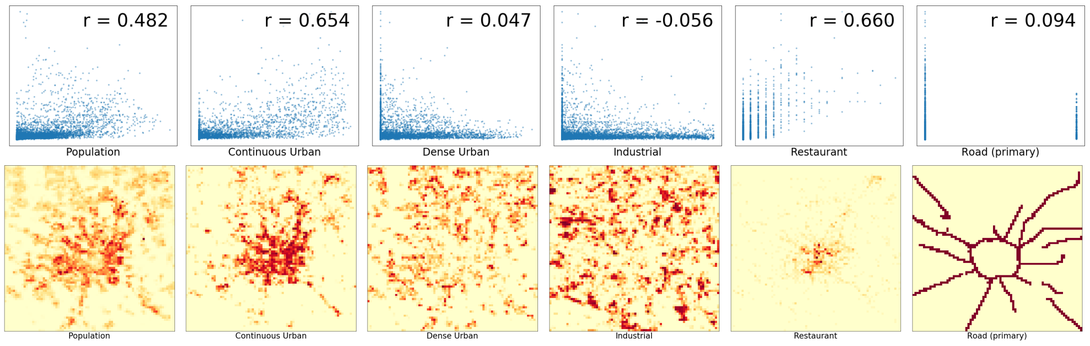


Figure 4: Scatterplots (top row): mobile traffic volumes (y-axis) versus attribute values (x-axis); each plot refers to one conditional attribute, and reports samples for all City A grid cells as well as the resulting correlation coefficient. Heatmaps (bottom row): geographical distribution of values of conditional attributes, providing a visual comparison against the mobile traffic observed in City A (Figure 1a). A complete set of results for all conditions is in §C.

map where the value associated to a grid cell represents the fraction of the cell covered by that land use type. Copernicus specifies a large number of land use types, many of which yield no relevance for mobile traffic. In order to avoid feeding CartaGenie with uninformative input, we filter out land uses with near-zero Pearson's correlation coefficient with respect to the mobile traffic, while aggregating others that are semantically similar. Finally, we retain 11 land use types: (*i*) Continuous Urban Fabric, (*ii*) Discontinuous Dense Urban Fabric, (*iii*) Discontinuous Medium Density Urban Fabric, (*iv*) Discontinuous Low Density Urban Fabric, (*v*) Discontinuous Very Low Density Urban Fabric, (*vi*) Isolated Structures, (*vii*) Green Urban Areas, (*viii*) Industrial and Commercial Areas, (*ix*) Air and Sea Ports (*x*) Leisure Facilities, and (*xi*) Non-urban Barren Lands. Figure 4 shows attribute maps for a subset of these land use types for City A.

Points of Interest (PoIs). PoIs indicate the presence of specific categories of landmarks, and are complementary to population and land use. We obtain PoI data from the OpenStreetMap (OSM) initiative using the Overpass API [6], and transform each PoI category into an attribute map by counting the number of PoIs in the target category that are located in every grid cell. Similarly to what we did for land uses, we filter out the tens of insignificant PoI categories in the original data using a correlation analysis with traffic. This results in retaining 14 PoI categories: (*i*) Tourism, (*ii*) Cafe, (*iii*) Parking, (*iv*) Restaurant, (*v*) Post Office and Police Station, (*vi*) Traffic signals, (*vii*) Office, (*viii*) Public transport, (*ix*) Shop, (*x*) Secondary roads, (*xi*) Primary roads, (*xii*) Motorways, and (*xiii*) Railway stations, and (*xiv*) Tram stops. Figure 4 illustrates attribute maps for two selected PoIs for City A.

Time. Mobile data traffic is inherently time-varying, and different geographical distributions of the demand for mobile services can be observed at different times. This suggests conditioning the generation of synthetic data on a time attribute. As we consider daily traffic aggregates in this work, we only introduce a simple time condition, which differentiates working days (Monday to Friday) from weekend days (Saturday and Sunday). As detailed in §3.1, the binary time context is used to modulate the latent variable into a bi-modal multivariate Gaussian input. We remark, however, that CartaGenie can potentially work at finer temporal scales as it faithfully captures variation over time. One can imagine to additionally model the latent space as a time series on top of CartaGenie, thereby making it a spatiotemporal model. We leave a thorough investigation and evaluation of time conditioning to future work.

As described above, CartaGenie uses a total of 26 different spatial conditions and 1 condition for time. An important remark is that none of the attributes is correlated in a straightforward way with the mobile traffic, as exemplified by Figure 4 and extensively presented in §C. This suggests that no naive statistical model can be easily derived from any of these individual attributes nor even an elaborate regression model combining these attributes to produce a reliable traffic estimate from the attributes, which justifies the sophisticated and tailored approach we take in designing CartaGenie.

4.3 Evaluation Criteria

4.3.1 Quantitative snapshot quality metrics. Since we represent traffic snapshots as images, image quality measures are an obvious choice for evaluating the fidelity of our synthetic traffic data. Specifically, we use two well established image quality assessment measures: Peak Signal-To-Noise Ratio (PSNR) and Structural Similarity Index Metric (SSIM). The metrics $PSNR(x, s)$ and $SSIM(x, s)$ measure the similarity between a pair of real and synthetic traffic images x and s; as they are very commonly adopted, we omit their formal definitions here, and provide them in §D.

In fact, in our setup, we have *sets* of traffic snapshots for both real (*e.g.*, from multiple days) and synthetic (*e.g.*, from multiple runs of CartaGenie) data. Hence, x and s are samples of two distributions $p(x)$ and $p(s)$, whose similarity is the significant target for our evaluation. To this end, we forward the comparison problem to the optimal transport (OT) regime [54] by using PSNR and SSIM as cost functions. §D.3 provides formal definitions for the resulting optimally-transported OT-SSIM and OT-PSNR metrics we employ to measure the fidelity of synthetic traffic with respect to the real-world data.

4.3.2 *Qualitative criteria.*

We also provide visualizations and statistics of mobile traffic as a more intuitive way to assess the fidelity of the synthetic data. In addition to maps of synthetic traffic, these qualitative metrics include (a) traffic histograms at pixel level; (b) time series of daily total traffic; and, (c) histograms of total traffic at day level.

4.4 Benchmarks

We consider a range benchmarks for comparative assessment of CartaGenie, as outlined below.

Regression. A naive approach to solve the traffic snapshot generation task is to train a model that takes conditions c as input and predicts x. Common choices for such a regression task are multi-layer perception (*MLP*) neural networks and convolution neural networks (*CNN*). These methods can be viewed as sophisticated variants of state-of-the-art mobile traffic generation approaches [20]. However, neither of them can model stochasticity, thus we expect them not to be able to characterize important features of the data. MLP regression in comparison with CNN regression also serves to show that convolutions provide better modeling of spatial correlations. Formally, we implement MLP regression by flattening the spatial condition $\boldsymbol{v}$ and concatenating it with the non-spatial condition w, which is then fed to a multi-layer perception of shape $(26 \times 20 \times 20 + 1) \longrightarrow 8192 \longrightarrow 4096 \longrightarrow 2048 \longrightarrow 10 \times 10$ with ReLU activations in between. For CNN regression, w is first mapped to the same spatial shape as $\boldsymbol{v}$ (with a single channel) by a linear layer, after which they are concatenated along the channel dimension. The result is then fed to a multi-layer convolution architecture of channels $(26 + 1) \longrightarrow 512 \longrightarrow 256 \longrightarrow 128 \longrightarrow 64 \longrightarrow 1$ with ReLU activations in between. The filter sizes of the each layer are 5, 4, 3, 2, 1 in order.

Generative adversarial networks (GANs). We also evaluate GANs on the traffic map generation task. In comparison to the regression baseline, GANs model stochasticity by randomly sampling latent vectors [26]. However, they aim at non-conditional generation, hence their outputs cannot be conditioned on a context $\boldsymbol{c}$. GANs let us show that conditions are indeed required to synthesize high-fidelity traffic maps. In our experiments, we use the standard DCGAN architecture [45] modified to an image size of 10×10.

Conditional GAN and Conditional VAE. We also consider the conditional variant of GANs [47] as well as conditional VAEs [38] among our benchmarks. As these two methods can be easily added to CartaGenie by adapting the training process and modifying the loss function, as elaborated in §B.1, we consider them as part of the ablation study in §F.1.

Image-to-Image Translation. As we represent $\boldsymbol{c}$ and $\boldsymbol{x}$ as images with different number of channels, we use the powerful *Pix2Pix* image-to-image translation framework [35], which has been successfully used in computer vision tasks that are similar to our traffic generation problem. These models are both conditional and stochastic, and provide a deep network architecture tailored for image-to-image translation. Specifically, we use the U-Net architecture [60] without one hidden layer in the encoder and decoder to adapt it to 10×10 images.

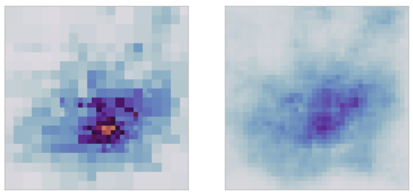
5 RESULTS

Across this section we use a multi-city setup for evaluation as the default setting. Specifically, we use 3 cities for training and 1 for testing, and repeat this for 4 times with a different test city each time while using the data of remaining cities for training, giving 4 results. When quantitatively comparing different models or variants, we report the averaged results out of 4 results; we report both the training and testing performance, which respectively show how good the model fits the data, and how good the model generalizes to unseen cities.

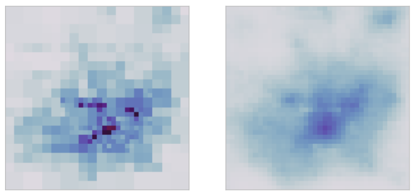
5.1 Generalization to unseen cities

We first show how our model performs on each city separately when training on the rest of cities. Figure 5 shows the qualitative results for all four cities. For each city there are three sub-figures. We show the average behavior across weekdays and weekends in the first two sub-figures. Our model successfully captures the overall spatial pattern as well as the weekday versus weekend temporal pattern. We also provide the total traffic per day over the entire city as a time series plot for each city (last sub-figure) to show additional temporal variations. From the result for City A and City B, the model successfully captures the weekday or weekend pattern with additional daily variations, while the generalization is not as good for City C because there is an additional long-term trend. For City D, the pattern of total traffic is not obvious even in the real data.

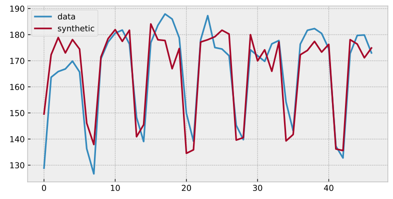
We summarize the corresponding quantitative results in terms of OT-PSNR and OT-SSIM for the above qualitative results in Table 1. As we can see, all but City A attain appealing results: consistently over the desired levels – 25 in terms of OT-PSNR and around 0.8 for OT-SSIM. The numbers for City A are relatively low in comparison, compared to other cities. This is because City A has more extreme traffic patterns. Specifically, the high traffic level in the hotspot area in City A is much larger than others. For this reason, the model fails to capture or generate it as such pattern has not been seen in other cities during training; we also provide an in-depth analysis on



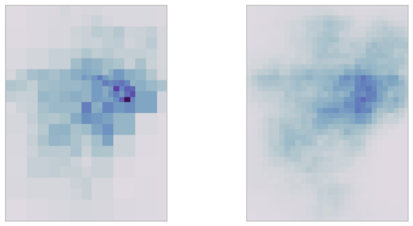
(a) Weekday averaged



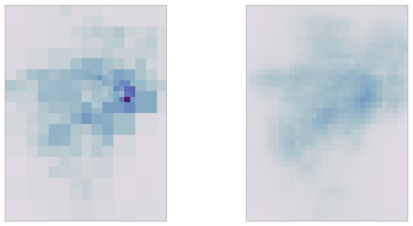
(b) Weekend averaged



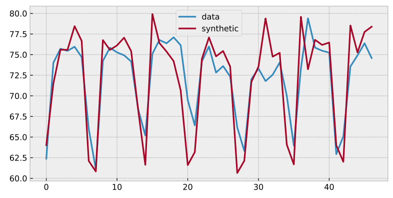
(c) Total traffic



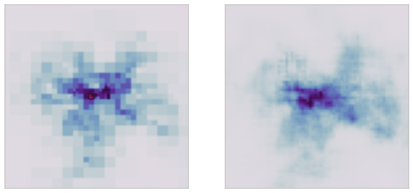
(d) Weekday averaged



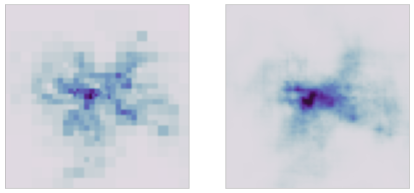
(e) Weekend averaged



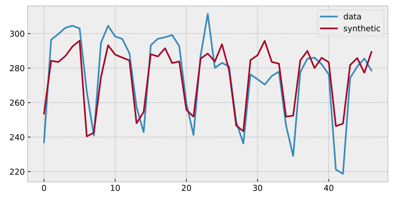
(f) Total traffic



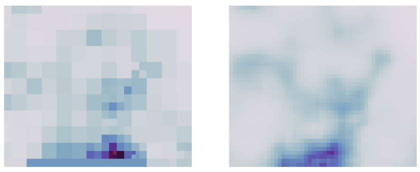
(g) Weekday averaged



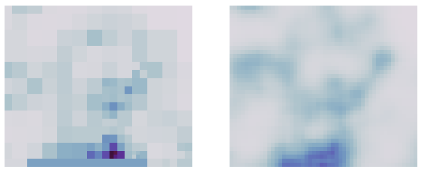
(h) Weekend averaged



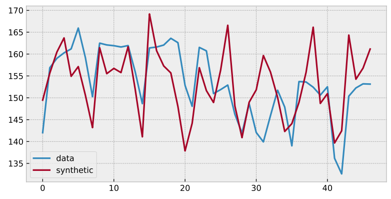
(i) Total traffic



(j) Weekday averaged



(k) Weekend averaged



(l) Total traffic

Figure 5: Generalization results for all cities; each row is for a different city – City A (top row), City B (second row), City C (third row) and City D (bottom row). For each pair of city maps, real traffic is on the left, and synthetic on the right.

	Condition	OT-PSNR↑	OT-SSIM↑
City A	Weekday	24.73	0.651
	Weekend	28.28	0.721
City B	Weekday	32.00	0.797
	Weekend	32.52	0.819
City C	Weekday	28.49	0.783
	Weekend	29.31	0.801
City D	Weekday	29.68	0.806
	Weekend	30.41	0.809

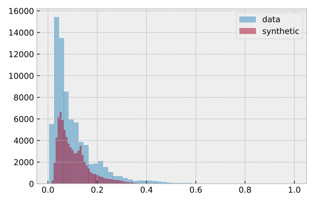
Table 1: Quantitative evaluation results of CartaGenie on the four available cities.

Metric	Condition	Type	GAN	MLP	CNN	Pix2Pix	Ours
OT-PSNR↑	Weekday	Train	20.99	27.61	32.88	31.04	32.41
		Test	21.14	25.63	28.61	26.02	28.72
	Weekend	Train	22.61	28.09	33.13	31.21	32.38
		Test	21.81	27.30	30.05	27.43	30.05
OT-SSIM↑	Weekday	Train	0.433	0.718	0.849	0.820	0.855
		Test	0.435	0.617	0.757	0.613	0.761
	Weekend	Train	0.448	0.713	0.853	0.805	0.838
		Test	0.386	0.621	0.784	0.616	0.786

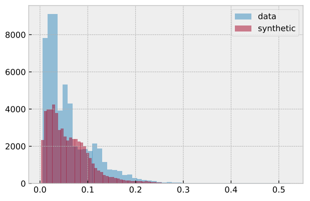
Table 2: Quantitative comparisons between CartaGenie and all baselines. Numbers are average of 4 runs using 3 cities for training and 1 for testing.

this training-testing domain shift in §E. Also note that the performance for weekends is consistently better than that of weekdays, likely because there are less variations during weekends.

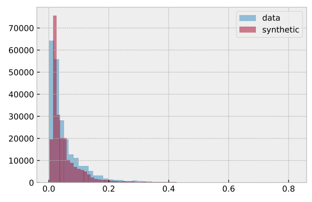
We now take a closer look at (mis-)matches in the distribution between real and synthetic traffic. Figure 6 shows histograms of traffic at pixel level: although the overall density pattern is captured by the model, it is still different from the real data, especially for City A and City B. Again, it might be due to the domain shift between training and testing data and might disappear if more diverse training data is available. Figure 7 portrays histograms of total traffic in time: CartaGenie successfully captures that there are two modes in total traffic



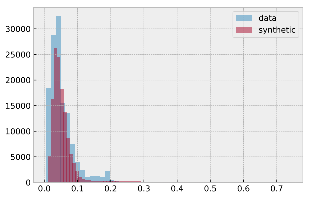
(a) City A



(b) City B

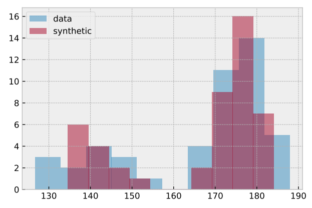


(c) City C

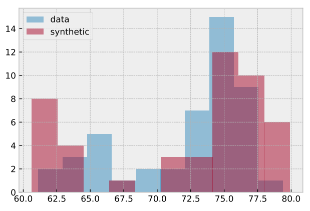


(d) City D

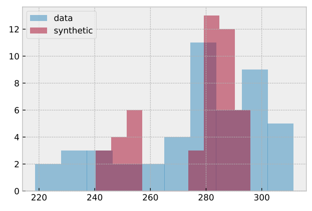
Figure 6: Traffic histogram at pixel level with CartaGenie.



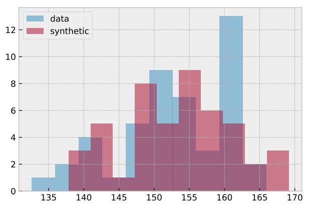
(a) City A



(b) City B



(c) City C



(d) City D

Figure 7: Histogram of total traffic with CartaGenie.

corresponding to weekday and weekend patterns. The variation of each mode in the synthetic data is also close to that of real data, except for City C.

5.2 Comparison against baselines

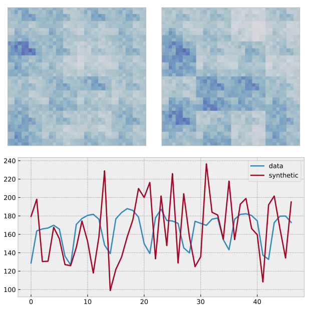
We now compare the generalization ability to unseen cities against the four benchmarks, namely MLP regression, CNN regression, (unconditional) GAN, and Pix2Pix.

5.2.1 Qualitative comparisons. Figure 8, 9, 10 show the similar set of qualitative results to the previous section when using City A as the held-out (*i.e.*, test) city[1]. As it can be seen from Figure 8, the generated city maps from GAN, MLP regression and Pix2Pix are visually degraded compared to ours in the previous section. For GAN, as the model is an unconditional generator, we do not expect it to be able to generate the desired city map. For MLP regression, the use of MLP fails to capture spatial correlation well. The bad performance of Pix2Pix might be surprising as it is the state-of-the-art image translation model. However, as previously discussed, our tasks differs from image-to-image translation as the conditions serve a different role in the data generation process. Especially for pixels around the edges, for which the context is actually missing in image-to-image translation framework. Lastly, the relatively good visual performance of CNN regression is not surprising as the CNN regression based architecture mainly differs from CartaGenie in two ways: (i) it lacks the latent variable z to model stochasticity; (ii) it does not obey the requirement for conditionally independent generation (as discussed in §3.2). For (i), we will see next that this approach fails to altogether capture the variation in the data completely. For (ii), we can see that there are visible artefacts of vertical and horizontal mosaic in the city map.

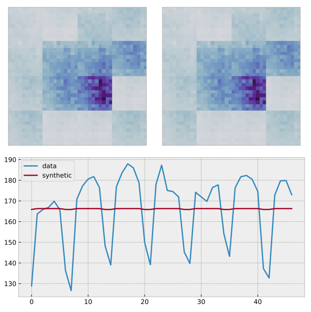
We now look at Figure 9 and Figure 10 which indicate how different models capture the distribution aspects of the data. Despite the poor visual results in Figure 8, GAN performs reasonably well in capturing the data distribution, especially in Figure 9. MLP regression and CNN regression also capture spatial distribution to some extent – especially for CNN regression, the performance is close to ours. However, as shown by Figure 10, they fail to capture variation in time, as by definition these approaches do not consider modeling inherent stochasticity. Finally, although Pix2Pix takes the conditional GAN approach which ideally should also model the distribution, the use of dropout as a source of stochasticity leads to degenerated performance in terms of variation in time, which has been also pointed out by previous studies [35, 45]. Although this is not a major issue in the context of image translation, correctly modelling stochasticity is important in our setting. Crucially, correctly modeling variation caused by time paves the way for CartaGenie to be used as a component in a spatiotemporal model by additionally modeling the dynamics of z to control the dynamics of $\boldsymbol{x}$; as previously mentioned, we will explore this aspect in future work. Pix2Pix results additionally suffer from severe edge effects (Figure 8) and also fail to capture the traffic distribution characteristics (Figure 10).

5.2.2 Quantitative comparisons. We summarize the performance of CartaGenie and all baseline methods in terms of OT-PSNR and OT-SSIM in Table 2. Overall, CartaGenie outperforms all the benchmarks by a clear margin. The only exception is CNN regression, which performs similarly to CartaGenie but completely fails to model the required stochasticity in data as reflected by Figure 10c.

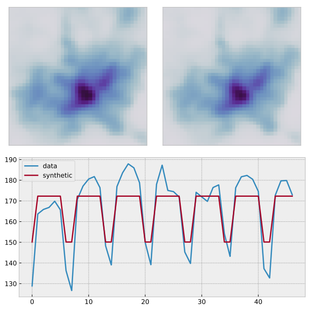
[1]We omit results for other cities as they yield similar observations in terms of qualitative differences with and among benchmark approaches.



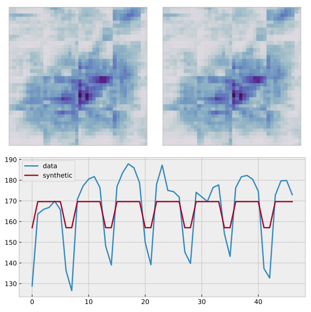
(a) GAN



(b) MLP regression

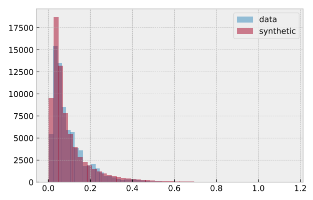


(c) CNN regression

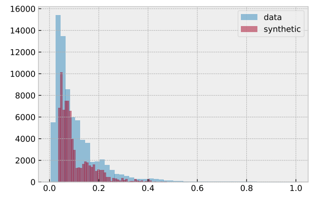


(d) Pix2Pix

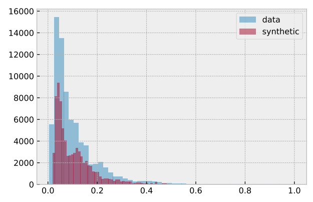
Figure 8: Baselines: generalization results on City A. For each sub-figure, from top to bottom are the averaged synthetic city map for weekday (left) and weekend (right) and the time series plot of daily total traffic.



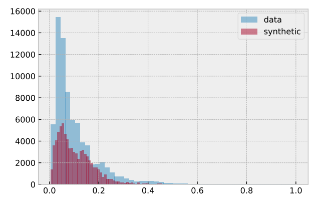
(a) GAN



(b) MLP regression

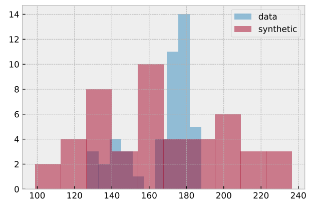


(c) CNN regression

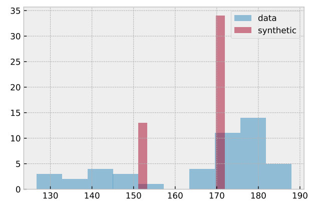


(d) Pix2Pix

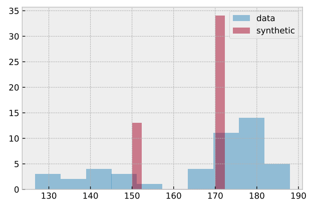
Figure 9: Baselines: traffic histogram at pixel level.



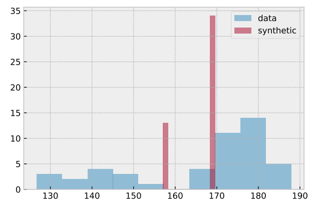
(a) GAN



(b) MLP regression



(c) CNN regression



(d) Pix2Pix

Figure 10: Baselines: histogram of total traffic.

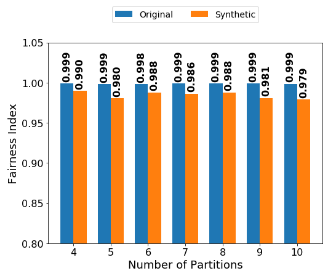
6 USE CASES

In order to assess the quality of the synthetic datasets generated by CartaGenie, we employ them to support two practical use cases in mobile networking and demography that can take advantage of city-scale traffic map information. Specifically, we consider (*i*) the planning of edge datacenters serving the radio access network of a mobile telco operator (§6.1), and (*ii*) the estimation of dynamic presence in urban areas (§6.2). In both cases, we use CartaGenie-generated traffic for previously unseen cities to inform models of edge datacenter planning and dynamic presence; we then compare the results against those obtained by feeding the same models with the actual traffic recorded by the operator in the target cities.

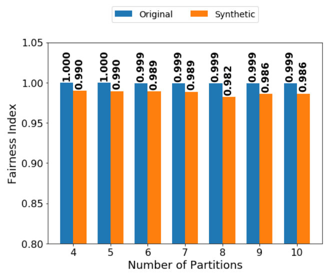
6.1 Network edge datacenter planning

Recent trends in networking are increasingly fostering the softwarization and virtualization of the infrastructure, so as to improve its programmability and flexibility. At the radio access of mobile networks, such a tendency has led to the emergence of functional split paradigms, where data communication tasks traditionally performed in hardware at base stations are moved to a reduced number of edge datacenters located in proximity of the base stations they serve [25, 31].

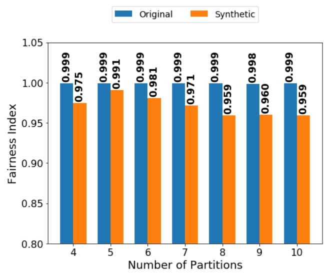
The planning of edge datacenter infrastructure needs to be driven by the knowledge of the typical spatial distribution of traffic demands. Specifically, we consider a recent model for the placement of datacenters that aims at minimizing the transmission latency (*i.e.*, geographical distance), while balancing the load across facilities [44]. As per this model, each spatial area (*e.g.*, base station coverage zone, antenna sector, or, as in our case, tile of a geographical tessellation corresponding to a pixel) is to be associated to one of a operator-defined number of datacenters. It addresses the problem by representing the system as a graph where vertices map onto pixels $p_{i,j} \in \mathcal{P}$, each characterized by its expected daily traffic $x_{i,j}$; edges $e_{(i,j)\to(i',j')} \in \mathcal{E}$ link pixels $p_{i,j}$ and $p_{i',j'}$ only if they are adjacent in space. Then, it partitions the graph so that the total traffic generated by the cells in each partition $d \in \mathcal{D}$ is balanced. This results in a set of geographical regions (each a union of the spatial areas of the cells in a same partition) that can be associated to one edge datacenter: intuitively, the



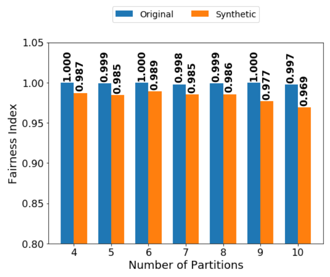
(a) City A



(b) City B



(c) City C



(d) City D

Figure 11: Jain's Fairness Index of load across different edge datacenters obtained using CartaGenie generated synthetic traffic relative to using actual traffic, for a varying number of edge datacenters.

baricenters of the cells in each cluster can be mapped to the ideal positions where to deploy the datacenters.

Formally, the model defines graph partitioning as an Integer Linear Programming (ILP) problem:

$$\min \sum_{e_{(i,j)\to(i',j')}\in\mathcal{E}} \phi(e_{(i,j)\to(i',j')})$$
$$\text{s.t. } (1-\epsilon) \le \frac{\sum_{p_{i,j}\in\mathcal{P}} \psi(p_{i,j},d)\cdot \boldsymbol{x}_{i,j}}{\frac{1}{|\mathcal{D}|}\sum_{p_{i,j}\in\mathcal{P}} \boldsymbol{x}_{i,j}} \le (1+\epsilon), \quad \forall d\in\mathcal{D}$$
$$\sum_{d\in\mathcal{D}} \psi(p_{i,j},d) = 1, \quad \forall p_{i,j}\in\mathcal{P} \tag{2}$$

Here, $\phi(e_{(i,j)\to(i',j')})$ and $\psi(p_{i,j},d)$ are decision variables set to one if edge $e_{(i,j)\to(i',j')}$ is cut by the partition, and if pixel $p_{i,j}$ is associated with datacenter d, respectively; they are zero otherwise. This means that $\forall e_{(i,j)\to(i',j')}\in\mathcal{E}$, $\forall d\in\mathcal{D}$, $\phi(e_{(i,j)\to(i',j')})$ takes a value one only if $e_{(i,j)\to(i',j')}$ links cells in partitions that are associated to different datacenters.

We assess the quality of the synthetic traffic generated by CartaGenie to solve the problem of network edge datacenter planning in (2). Specifically, we first feed the synthetic traffic data to the ILP problem, and solve it: this returns cell graph partitions (hence datacenter locations) based on CartaGenie output. Then, we enforce the CartaGenie-driven partitions on the real-world traffic, and measure how balanced demands are across datacenters, using Jain's fairness index [57]. Finally, we compare the result with the fairness index obtained when the planning problem in (2) is solved using the actual traffic data directly. Ultimately, this evaluation approach lets us investigate how accurate a planning based on the synthetic traffic would be, with respect to the ground-truth reference.

Figure 11 summarizes the outcome of the test. Each plot refers to a different city, under a varying number of 4 to 10 edge datacenters – a reasonable range for the target cities. Performance is consistently good across all settings: CartaGenie-driven synthetic traffic allows planning the datacenter deployment in a way that the loss of fairness (in the Jain's index acceptation) is in the 1-3% range compared to using real data.

	Condition	OT-PSNR↑	OT-SSIM↑
City A	Weekday	26.19	0.664
	Weekend	27.72	0.720
City B	Weekday	31.80	0.777
	Weekend	32.00	0.804
City C	Weekday	27.80	0.781
	Weekend	30.53	0.804
City D	Weekday	29.62	0.801
	Weekend	30.23	0.803

Table 3: Difference between population estimates computed using original versus synthetic traffic.

6.2 Dynamic presence estimation

The estimation of human presence in metropolitan areas is a challenging open problem in geo-informatics, which aims at capturing fluctuations in the distribution of people occurring at timescales of minutes, hours or days[2]. In this context, models have been proposed that leverage mobile traffic as a proxy for user presence, and map spatiotemporal variations in the network activity onto population density dynamics.

We consider a recent multivariate regression model that links mobile network traffic and activity levels to dynamic population density [37]. Formally, the model is defined as $\boldsymbol{p} = \exp(k_1\lambda+k_2)\cdot\boldsymbol{x}^{k_3\lambda+k_4}$, where $\boldsymbol{p}$ is the population computed as a power law of the mobile traffic volume $\boldsymbol{x}$, λ is the activity level computed as the mean number of network events (*e.g.*, established calls) per subscriber, while k_1, k_2, k_3, and k_4 are constant model parameters.

In our case, $\boldsymbol{x}$ is the total traffic observed in a single day, which allows inferring the average daily distribution of people in the target urban region. We set λ (the average activity level observed during a 24-hour period) to 0.5, and all constants to the typical values indicated in [37]. Note that the returned $\boldsymbol{p}$ is inherently different from inhabitant distributions captured,

[2]It is worth noting that dynamic presence is a very different concept from the static population density information that is available from census data, and which we use as a contextual attribute input to CartaGenie. The latter only capture home locations, whereas the former tries to model time-varying distributions induced by the daily mobility of individuals.

e.g., by census: indeed, $\boldsymbol{p}$ accounts for the locations and activities of people during the whole day, which results in a very diverse presence.

In order to assess the quality of CartaGenie for this use case, we generate daily population estimates separately based on (*i*) the synthetic traffic generated by CartaGenie and (*ii*) the actual traffic recorded by the operator. We then compare the resulting population densities in terms of OT-SSIM and OT-PSNR. Performance figures are summarized in Table 3, when the mobile traffic in each reference city (in the columns) is generated with CartaGenie trained on the other urban regions. The results are very good overall, with OT-PSNR well above 26 in all cases, and OT-SSIM typically around 0.8.

7 RELATED WORK

Network traffic generation is a very different problem depending on the dimension and scale it is tackled at. In its most traditional form, traffic generation aims at creating packet-level workloads for which several tools exist and are commonly used in research and engineering. Some of these tools simply generate traffic patterns as per specified input (*e.g.*, iperf [3], Ostinato [7]). Other tools such as D-ITG [16] additionally employ probabilistic models to emulate the load of different applications; specifically, distributions are used to represent the statistical features of the flows (*e.g.*, size and inter-arrival time of packets), with parameter values typically justified using some real-world measurements. Such approaches are also adopted by traffic generators embedded in popular network simulators, such as ns-3 [4] or OMNeT++ [5]. The above outlined form of traffic generation does not have a spatial dimension as such and so accounting for spatial correlations does not arise. Temporal correlations are also mostly overlooked. Our focus, however, is on a different and macroscopic scale, as we are concerned with mobile network data traffic aggregated over multiple users and flows. Here, the challenge is to be able to model the spatial and temporal fluctuations of large volumes of traffic, rather than packet-level dynamics of individual flows.

Di Francesco et al. [20], in what is the only existing work related to mobile network traffic generation to the best of our knowledge, assemble a cellular dataset for a given region by integrating multiple sources of data: (i) census data to infer subscriber locations; (ii) base station locations and radio propagation models to associate subscribers with base stations; (iii) estimated data demand per subscriber. It is this last source of data that links to our work. In [20], this is done by focusing solely on the busy-hour period and simply modeling per-subscriber demand across a given region (e.g., urban) from the real operator traffic dataset as a probability distribution (specifically, log-normal); this distribution is then "independently" sampled for the created dataset to estimate the per-subscriber data demand at any given location. In doing so, it implicitly makes an assumption that the peak traffic generated by a mobile subscriber corresponds to their residential location, which is seemingly incorrect especially in urban regions. In sharp contrast, CartaGenie is a neural network based data-driven method, that does not make any such assumptions, preserves correlations and finer variations in traffic over space and also models changes in traffic patterns over time. Moreover, unlike [20], we validate the fidelity of CartaGenie against ground-truth information and also demonstrate it to be capable of generating realistic synthetic traffic over previously unseen city-scale regions purely based on publicly available contextual input.

DoppelGANger [42] is a recent work that shares the same high-level goal as ours, that is to reduce the barrier to data-driven research in networking and beyond via synthetic data generation. DoppelGANger is however only broadly related in that it exclusively focuses on time-series data (e.g., web article views over time, network monitoring data over time). On the other hand, the spatial dimension features prominently in our work as we aim at generating city-scale mobile network data traffic for a given period of time, which requires modeling inherent and complex context-dependent correlations in mobile traffic across space and time. There is a further difference between DoppelGANger and our CartaGenie method with respect to their underlying model. DoppelGANger employs a standard conditional GAN, based on Recurrent Neural Network (RNN) with batched time-series generation to capture temporal correlations. In contrast, we find that GANs provide little benefit for our strongly conditioned generation case so we design a tailored model architecture in CartaGenie.

Beyond the networking domain, our mobile traffic map generation problem is broadly related to data generation problems in two other domains – image synthesis in computer vision and road traffic generation in the transportation domain. Pix2Pix [35] is a representative example in the former case which takes a conditional GAN approach for image generation based on U-Net architecture [60]. As we show in our evaluation results, Pix2Pix approach has two key limitations when applied to the mobile traffic map generation problem compared to our well tailored CartaGenie approach: (i) from using dropout for stochasticity, it fails to model variation in the data; (ii) it introduces artefacts when sewing a city traffic map from generated traffic patches. For the case of road traffic generation, TrafficGAN [71] is a representative example. Like Pix2Pix, TrafficGAN also takes a conditional GAN approach but assumes knowledge of traffic correlations among roads in the target region for (road) traffic generation. Such an assumption is strong and unrealistic assumption in our setting. In contrast, our CartaGenie method does not make any assumption about the nature or knowledge of (mobile) traffic in the target region and solely relies on easily available contextual input.

8 CONCLUSIONS

We have presented CartaGenie, a novel method rooted in deep generative modeling, for synthesizing high fidelity city-scale mobile network traffic snapshots. At its heart, CartaGenie is a conditional generator with tailored convolutional neural network model that brings together several innovations to faithfully capture complex spatiotemporal variations inherent to mobile traffic and to generate artefact free traffic maps for varied sized regions for a given time. CartaGenie requires only publicly available contextual information of a region (*e.g.*, city) such as population density in order to generate realistic mobile traffic map for that region. Our extensive evaluations based on real-world mobile traffic measurement data show that CartaGenie is not only able to generate city-scale mobile traffic snapshot data that is superior in fidelity to a range of baselines including the state-of-the-art but also it generalizes well in synthesizing mobile traffic maps for unseen cities. The above strengths in turn allow synthetic data generated with CartaGenie to be used for data-driven applications in mobile networking and beyond in a way that is indistinguishable from using actual data, as we demonstrate through two use cases.

REFERENCES

[1] [n. d.]. Copernicus Urban Atlas. ([n. d.]). https://land.copernicus.eu/local/urban-atlas/urban-atlas-2012.

[2] [n. d.]. General Data Protection Regulation (GDPR). ([n. d.]). https://gdpr-info.eu/.

[3] [n. d.]. iPerf3: A TCP, UDP, and SCTP network bandwidth measurement tool. ([n. d.]). https://iperf.fr/.

[4] [n. d.]. Network Simulator (NS-3). ([n. d.]). https://www.nsnam.org/.

[5] [n. d.]. OMNeT++: Discrete Event Simulator. ([n. d.]). https://omnetpp.org/.

[6] [n. d.]. OpenStreetMap Overpass API. ([n. d.]). http://overpass-turbo.eu/.

[7] [n. d.]. Ostinato Packet Generator. ([n. d.]). https://ostinato.org/.

[8] [n. d.]. Synthetic data for public good. ([n. d.]). https://datasciencecampus.ons.gov.uk/projects/synthetic-data-for-public-good/.

[9] N. Z. Abay et al. 2018. Privacy Preserving Synthetic Data Release Using Deep Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

[10] D. Barber. 2012. *Bayesian reasoning and machine learning*. Cambridge University Press.

[11] H. Barbosa et al. 2018. Human mobility: Models and applications. *Physics Reports* 734 (2018), 1 – 74. http://www.sciencedirect.com/science/article/pii/S037015731830022X

[12] B. K. Beaulieu-Jones, Z. S. Wu, C. Williams, R. Lee, S. P. Bhavnani, J. B. Byrd, and C. S. Greene. 2018. Privacy-preserving generative deep neural networks support clinical data sharing. *bioRxiv* (2018). https://doi.org/10.1101/159756

[13] R. Becker et al. 2013. Human Mobility Characterization from Cellular Network Data. *Commun. ACM* 56, 1 (Jan. 2013), 74–82. https://doi.org/10.1145/2398356.2398375

[14] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez. 2020. DeepCog: Optimizing Resource Provisioning in Network Slicing With AI-Based Capacity Forecasting. *IEEE Journal on Selected Areas in Communications* 38, 2 (2020), 361–376.

[15] C. M Bishop. 2006. *Pattern recognition and machine learning*. springer.

[16] A. Botta, A. Dainotti, and A. Pescapè. 2012. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks* 56, 15 (2012), 3531–3547.

[17] F. Calabrese, L. Ferrari, and V. D. Blondel. 2014. Urban Sensing Using Mobile Phone Network Data: A Survey of Research. *Comput. Surveys* 47, 2 (2014), 20. https://doi.org/10.1145/2655691

[18] S. Chang, E. Pierson, P.W. Koh, et al. 2020. Mobility network models of COVID-19 explain inequities and inform reopening. *Nature* (2020). https://doi.org/10.1038/s41586-020-2923-3

[19] P. Deville, C. Linard, S. Martin, M. Gilbert, F. R. Stevens, A. E. Gaughan, V. D. Blondel, and A. J. Tatem. 2014. Dynamic population mapping using mobile phone data. *Proceedings of the National Academy of Sciences* 111, 45 (2014), 15888–15893. https://doi.org/10.1073/pnas.1408439111

[20] P. Di Francesco, F. Malandrino, and L. A. DaSilva. 2018. Assembling and Using a Cellular Dataset for Mobile Network Analysis and Planning. *IEEE Transactions on Big Data* 4, 4 (2018), 614–620.

[21] Y. Dong et al. 2015. Inferring Unusual Crowd Events from Mobile Phone Call Detail Records. In *Machine Learning and Knowledge Discovery in Databases*. 474–492.

[22] R.W. Douglass, D.A. Meyer, M. Ram, D. Rideout, and Song D. 2015. High resolution population estimates from telecommunications data. *EPJ Data Science* 4, 4 (2015).

[23] A. M. Eskicioglu and P. S. Fisher. 1995. Image quality measures and their performance. *IEEE Transactions on Communications* 43, 12 (1995), 2959–2965.

[24] Z. Fang, F. Zhang, L. Yin, and D. Zhang. 2018. MultiCell: Urban Population Modeling Based on Multiple Cellphone Networks. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3 (Sept. 2018). https://doi.org/10.1145/3264916

[25] A. Garcia-Saavedra, G. Iosifidis, X. Costa-Perez, and D. J. Leith. 2018. Joint Optimization of Edge Computing Architectures and Radio Access Networks. *IEEE Journal on Selected Areas in Communications* 36, 11 (2018), 2433–2443. https://doi.org/10.1109/JSAC.2018.2874142

[26] I. Goodfellow et al. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[27] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[28] S. Grauwin, S. Sobolevsky, S. Moritz, I. Gódor, and C. Ratti. 2015. Towards a Comparative Science of Cities: Using Mobile Traffic Records in New York, London, and Hong Kong. In *Computational Approaches for Urban Environments*, M. Helbich, J. Jokar Arsanjani, and Leitner M. (Eds.). Vol. 13. Springer.

[29] M. Gutmann and A. Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 297–304.

[30] A. Horé and D. Ziou. 2010. Image Quality Metrics: PSNR vs. SSIM. In *International Conference on Pattern Recognition*. 2366–2369.

[31] C. I, C. Rowell, S. Han, Z. Xu, G. Li, and Z. Pan. 2014. Toward green and soft: a 5G perspective. *IEEE Communications Magazine* 52 (February 2014).

[32] S. Ioffe and C. Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[33] S. Isaacman et al. 2012. Human Mobility Modeling at Metropolitan Scales. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*. 239–252. https://doi.org/10.1145/2307636.2307659

[34] S. Isaacman, V. Frias-Martinez, and E. Frias-Martinez. 2018. Modeling Human Migration Patterns during Drought Conditions in La

Guajira, Colombia. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS '18)*. https://doi.org/10.1145/3209811.3209861

[35] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[36] A. Janecek, D. Valerio, K. A. Hummel, F. Ricciato, and H. Hlavacs. 2015. The Cellular Network as a Sensor: From Mobile Phone Data to Real-Time Road Traffic Monitoring. *IEEE Transactions on Intelligent Transportation Systems* 16, 5 (2015), 2551–2572. https://doi.org/10.1109/TITS.2015.2413215

[37] G. Khodabandelou, V. Gauthier, M. Fiore, and M. A. El-Yacoubi. 2019. Estimation of Static and Dynamic Urban Populations with Mobile Network Metadata. *IEEE Transactions on Mobile Computing* 18, 9 (2019), 2034–2047. https://doi.org/10.1109/TMC.2018.2871156

[38] D. P. Kingma et al. 2014. Semi-supervised learning with deep generative models. *Advances in neural information processing systems* 27 (2014), 3581–3589.

[39] D. P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[40] D. P. Kingma and M. Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[41] Kevin S. Kung, Kael Greco, Stanislav Sobolevsky, and Carlo Ratti. 2014. Exploring Universal Patterns in Human Home-Work Commuting from Mobile Phone Data. *PLOS ONE* 9 (06 2014), 1–15. https://doi.org/10.1371/journal.pone.0096180

[42] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar. 2020. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In *ACM Internet Measurement Conference, IMC '20*. 464–483. https://arxiv.org/abs/1909.13403

[43] C. Marquez et al. 2017. Not All Apps Are Created Equal: Analysis of Spatiotemporal Heterogeneity in Nationwide Mobile Service Usage *(CoNEXT '17)*. 180–186. https://doi.org/10.1145/3143361.3143369

[44] C. Marquez et al. 2018. How Should I Slice My Network? A Multi-Service Empirical Evaluation of Resource Sharing Efficiency. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. 191–206. https://doi.org/10.1145/3241539.3241567

[45] M. Mathieu, C. Couprie, and Y. LeCun. 2015. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440* (2015).

[46] M. Michalopoulou, J. Riihijärvi, and P. Mähönen. 2010. Studying the Relationships between Spatial Structures of Wireless Networks and Population Densities. In *IEEE Global Telecommunications Conference (GLOBECOM)*. 1–6.

[47] M. Mirza and S. Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[48] K. P. Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

[49] D. Naboulsi, M. Fiore, S. Ribot, and R. Stanica. 2016. Large-Scale Mobile Traffic Analysis: A Survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 124–161.

[50] V. Nair and G. E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

[51] Beata Nowok, Gillian Raab, and Chris Dibben. 2016. synthpop: Bespoke Creation of Synthetic Data in R. *Journal of Statistical Software, Articles* 74, 11 (2016), 1–26. https://doi.org/10.18637/jss.v074.i11

[52] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das. 2011. Understanding traffic dynamics in cellular data networks. In *Proceedings IEEE INFOCOM*. 882–890.

[53] E. Perez et al. 2017. FiLM: Visual Reasoning with a General Conditioning Layer. *arXiv:1709.07871 [cs, stat]* (Dec. 2017). arXiv:cs, stat/1709.07871

[54] G. Peyré and M. Cuturi. 2020. Computational Optimal Transport. (2020). arXiv:stat.ML/1803.00567

[55] F. Pinelli, R. Nair, F. Calabrese, M. Berlingerio, G. Di Lorenzo, and M. L. Sbodio. 2016. Data-Driven Transit Network Design From Mobile Phone Trajectories. *IEEE Transactions on Intelligent Transportation Systems* 17, 6 (2016), 1724–1733. https://doi.org/10.1109/TITS.2015.2496783

[56] Z. Qin et al. 2018. EXIMIUS: A Measurement Framework for Explicit and Implicit Urban Traffic Sensing. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys '18)*. 1–14. https://doi.org/10.1145/3274783.3274850

[57] A. Durresi R. Jain and G. Babic. 1999. Throughput fairness index: An explanation. *ATM Forum Document Number: ATM Forum/990045* (Feb. 1999).

[58] J. Reades, F. Calabrese, A. Sevtsuk, and C. Ratti. 2007. Cellular Census: Explorations in Urban Data Collection. *IEEE Pervasive Computing* 6, 3 (2007), 30–38. https://doi.org/10.1109/MPRV.2007.53

[59] D. J. Rezende, S. Mohamed, and D. Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).

[60] O. Ronneberger, P. Fischer, and T. Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[61] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang. 2012. Characterizing geospatial dynamics of application usage in a 3G cellular data network. In *Proceedings of IEEE INFOCOM*. 1341–1349.

[62] E. Shein. 2018. The State of Fakery. *Commun. ACM* 61, 3 (Feb. 2018), 21–23. https://doi.org/10.1145/3178125

[63] R. Singh et al. 2019. Urban Vibes and Rural Charms: Analysis of Geographic Diversity in Mobile Service Usage at National Scale. In *The World Wide Web Conference*. 1724–1734.

[64] N. Srivastava et al. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[65] I. Ucar, M. Gramaglia, M. Fiore, Z. Smoreda, and E. Moro. 2017. Netflix or Youtube? Regional income patterns of mobile service consumption. In *NetMob*.

[66] M. Uehara et al. 2016. Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920* (2016).

[67] G. Vallero, D. Renga, M. Meo, and M. A. Marsan. 2019. Greener RAN Operation Through Machine Learning. *IEEE Transactions on Network and Service Management* 16, 3 (2019), 896–908.

[68] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[69] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

[70] F. Xu, P. Zhang, and Y. Li. 2016. Context-Aware Real-Time Population Estimation for Metropolis. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*. 1064–1075. https://doi.org/10.1145/2971648.2971673

[71] Y. Zhang, Y. Li, X. Zhou, X. Kong, and J. Luo. 2019. TrafficGAN: Off-Deployment Traffic Estimation with Traffic Generative Adversarial Networks. In *2019 IEEE International Conference on Data Mining (ICDM)*. 1474–1479. https://doi.org/10.1109/ICDM.2019.00193

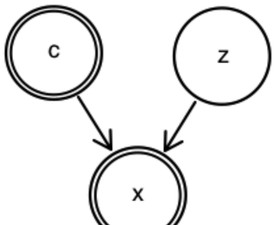


Figure 12: Plate diagram of the latent variable model underlying CartaGenie. Double circles are for observed variables and single circle represents unobserved, latent variables.

A PROBABILISTIC INTERPRETATION OF CARTAGENIE

The underlying latent variable model behind $\mathcal{G}_\theta$ is $p_\theta(\boldsymbol{x} \mid \boldsymbol{c}) = \int p_\theta(\boldsymbol{x} \mid \boldsymbol{c}, z)p(z)\mathrm{d}z$, where θ is the parameters of the conditional distribution. This generative process of this conditional distribution is implemented by our network $\mathcal{G}_\theta$. Note that we make the assumption that z and $\boldsymbol{c}$ are conditionally independent, *i.e.*, $p(z \mid \boldsymbol{c}) = p(z)$ (Figure 12). This assumption makes sense as we intend to use z to model additional variations that are not included in $\boldsymbol{c}$. Note that once conditioned on observations, z and $\boldsymbol{c}$ become dependent.

To learn θ, we perform maximum likelihood estimation (MLE) to maximize the conditional marginal log-likelihood:

$$\log p_\theta(\mathcal{X} \mid C) = \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{l=1}^{L} \log \mathbb{E}_{z_t^{n,l} \sim p(z)} \left[p_\theta(\boldsymbol{x}_t^{n,l} \mid \boldsymbol{c}_t^{n,l}, z_t^{n,l}) \right] \tag{3}$$

In order to obtain an unbiased gradient via Monte Carlo estimation using mini-batches, we optimize the following lower-bound obtained by Jensen's inequality[3] as our objective:

$$\begin{aligned}\mathcal{L}(\theta) &= \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{l=1}^{L} \mathbb{E}_{z_t^{n,l} \sim p(z)} \left[\log p_\theta(\boldsymbol{x}_t^{n,l} \mid \boldsymbol{c}_t^{n,l}, z_t^{n,l}) \right] \\ &\le \log p_\theta(\mathcal{X} \mid C).\end{aligned} \tag{4}$$

It can be shown that the bound is tight if and only if θ is the maximizer of (3) [15, 48].

Now we state the choice of $p(z)$ and $p_\theta(\boldsymbol{x} \mid \boldsymbol{c}, z)$ to fully specify our latent variable model. For $p(z)$, we use a multivariate standard normal $\mathcal{N}(\mathbf{0}, \mathbf{1})$. More precisely, *each dimension* of our D_z-dimensional latent, uninterpretable multivariate random variable z is independent, identically distributed as $\mathcal{N}(0, 1)$. For $p_\theta(\boldsymbol{x} \mid \boldsymbol{c}, z)$, we use a Laplace distribution with unit variance and mean parameterized by our generative network, i.e., $p_\theta(\boldsymbol{x} \mid \boldsymbol{c}, z) = \mathcal{L}\text{aplace}(\boldsymbol{x}; \mathcal{G}_\theta(\boldsymbol{c}, z), \mathbf{1})$. The rationale is that the log-likelihood of a Laplace distribution is a scaled version of L1 norm $\|\mathcal{G}_\theta(\boldsymbol{c}, z) - \boldsymbol{x}\|_1$, which has been successfully adopted in the literature [60].

[3]We obtain the inequality $\mathbb{E}_{z_t^{n,l} \sim p(z)} \left[\log p_\theta(\boldsymbol{x}_t^{n,l} \mid \boldsymbol{c}_t^{n}, z_t^{n,l}) \right] \le \log \mathbb{E}_{z_t^{n,l} \sim p(z)} \left[p_\theta(\boldsymbol{x}_t^{n,l} \mid \boldsymbol{c}_t^{n,l}, z_t^{n,l}) \right]$ by the concavity of the logarithm function, and plug the expression into (3).

B ADDITIONAL DESIGN DISCUSSION AND DETAILS

B.1 Design choices

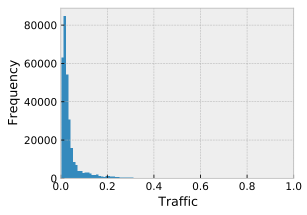
Classic strategies for the design of the model that are different from that adopted by CartaGenie may seem reasonable. Below, we further justify our choices.

Why not VAEs. It might seem appealing to add an additional encoder to our model, forming a VAE [38, 40, 59]. This turns out to be unnecessary in our strongly-conditioned case. For free-form generation or weakly-conditioned generation, the encoder serves the role of providing training signal to the decoder network when the stochasticity from the prior is too large; when stochasticity is large, estimating the expectation via samples from the prior can be ineffective. In our setup, using samples from the Gaussian prior to estimate the expectation is sufficient and we empirically found no benefit from introducing an additional encoder. Empirical study on the effect of variational training in §F supports our claim.

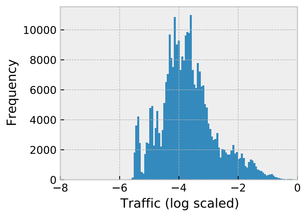
Why not adversarial training (GANs). It might also seem sensible to add an additional discriminator to our model, following the (conditional) GAN framework [26, 47]. Again, this is not necessary in our case. For free-form or weakly-conditioned generation, GANs have two benefits in helping with training: (a) the discriminator relaxes the choice of explicit loss function, which can be hard to design, especially for natural images; and (b) the discriminator provides training signal to the generator even from unpaired synthetic and real data via the noise-contrastive estimation principle [29] or equivalently, the density ratio estimation perspective [66]. In our strongly-conditioned case, we found that the effect of (a) is limited. The use of the L1 loss already leads to satisfactory results and adding an extra discriminator does not help; nor does completely replacing the explicit loss with the adversarial loss. Since an explicit loss works well in our setup, we do not need training with un-paired examples via a discriminator, and the benefit of (b) is also negligible. Empirical study on the effect of variational approach and adversarial training in §F supports our claim.

B.2 0-1 normalization of traffic

In order to help neural network training, the traffic values in $\boldsymbol{x}^n$ pixels need to be normalized so that they are approximately between 0 and 1. To this end, we divide the values by $x_{\max}$, *i.e.*,



(a) Original space



(b) Logarithmic space

Figure 13: Histogram of (0,1)-normalized per-pixel traffic data in a sample city scenario.

the *global* per-pixel maximum traffic volume value observed in the training data.

The normalized data still includes a large number of null values, for pixels where no traffic is recorded at a specific time t. This is a result of the high spatial skewness that is known to characterize mobile traffic [52, 61]. In our reference scenarios, the traffic values recorded in the pixels of $\boldsymbol{x}^n$ of a given city n are typically log-Gaussian distributed, as exemplified in Figure 13a. In order to make the data well-behaved in the log-space so that the neural network can use its full capacity, we shift it by a small value ϵ to avoid negative infinite minimum values upon log-transformation.

Finally, the applied normalization and transformation is:

$$\boldsymbol{x} \leftarrow \epsilon + (1-\epsilon)\,\frac{\boldsymbol{x}}{\boldsymbol{x}_{\max}},$$

which results in $\boldsymbol{x} \in [\epsilon, 1]$ and $\log(\boldsymbol{x}) \in [\log(\epsilon), 0]$. In all our experiments, we use the maximum traffic value (*across all four cities in our multi-city dataset*) as $\boldsymbol{x}_{\max}$. For ϵ, we set it to $\exp(-8)$ throughout, so that the mode of data distribution in the log-domain in desirably around the middle of the data range, as shown in Figure 13b, which is the normalized and transformed version of Figure 13a.

Our results show that we can reverse this pre-processing through a single city-level rescaling post-processing step that can effectively match the city-level synthetic traffic in unseen cities. This indicates that our model actually captures the relationship between $\boldsymbol{c}$ and $\boldsymbol{x}$, up to a multiplier.

B.3 Output parameterization of the neural network

As the traffic is log-Gaussian distributed (see Figure 13a, for example), the neural network capacity can be better utilized by directly outputting traffic in log-space, and then transform it to the original domain. To do so, a straightforward approach is to restrict the neural network output to $[-8, 0]$ (the range of traffic data in the log-space) using a sigmoid function σ with an affine transformation. In a case where the maximum traffic value (after scale and log-transformation) is 0, the network might find it hard to produce values that are close if we naively map the entire sigmoid output to $[-8, 0]$, since $\lim_{x\to\infty} \sigma(x) = 1$. To address this, we use a transformation which ensures that a network output of $[\sigma(-4), \sigma(4)]$ that can then be mapped to $[-8, 0]$ in the log space and equivalently $[0, 1]$ in the original space. In other words, the network only needs to output between -4 and 4 to cover the traffic values.

Such transformation is usually described by a pair of link and inverse link functions. The link function g and the corresponding inverse link function g^{-1} that serve our goal are:

$$\begin{aligned} g(x) &= (\sigma(4) - \sigma(-4))(\log(x)/8 + 1) + \sigma(-4), \\ g^{-1}(y) &= \exp(8((y - \sigma(-4))/(\sigma(4) - \sigma(-4)) - 1)) \end{aligned} \tag{5}$$

where $y_{\min} =, y_{\max} =$ Denote our neural network with a final sigmoid activation as f_θ with output in the range of $[\sigma(-4), \sigma(4)]$. We apply the inverse link function as follows to get the output in the range of $[-8, 0]$.

Then reversing the transformation and normalization as described in Appendix B.2 results in a output traffic map from our generator with pixel values in the original domain.

C COMPLETE SET OF CONDITIONAL ATTRIBUTES

Figure 14 provides a complete view of the mobile network traffic correlation scatterplots and spatial heatmaps of the attributes considered in our study.

D DETAILS ON COMPUTING EVALUATION METRICS

D.1 PSNR

PSNR is among the simplest and most widely used image quality metrics. It computes variations at each pixel independently and takes full reference (original) image into consideration while reporting the final value [23]. PSNR is therefore highly interpretable, however it does not model HVS as it lacks the ability to assess spatial correlation in image pixels. PSNR is preferred over other measures like Mean Squared Error (MSE) because it is scale invariant as it normalizes the difference between pixels with the dynamic range of the images/traffic heatmaps.

Let x be the original image and s be the synthetic image. PSNR is inversely proportional to MSE and is computed as,

$$PSNR(x, s) = 10 \log_{10}\left(\frac{K^2}{MSE}\right),$$

where $MSE(x, s) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{i,j} - s_{i,j})^2$ and K is the maximum pixel value (max traffic volume in our case) observed in x.

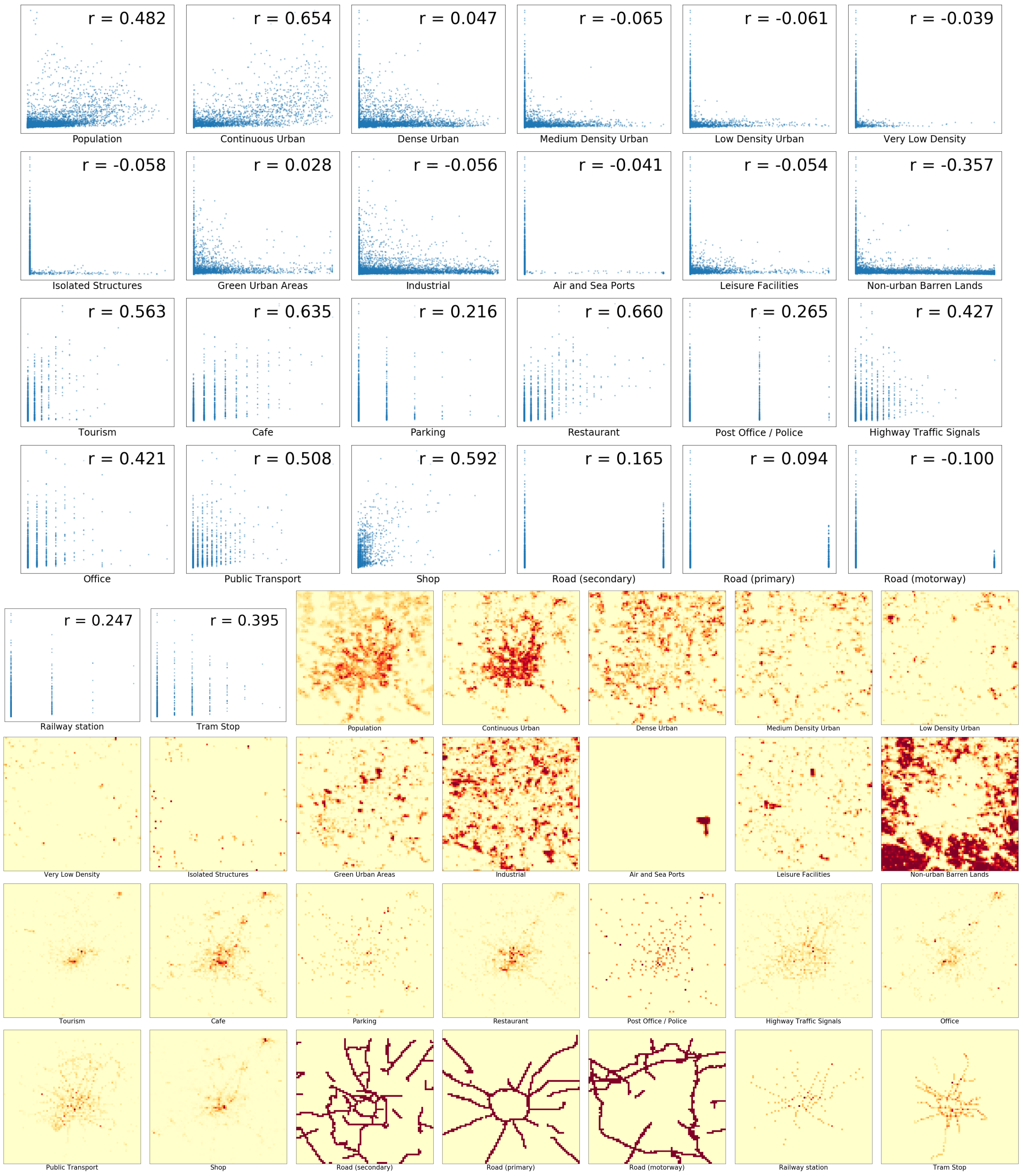


Figure 14: Scatterplots showing correlations with mobile network traffic, and heatmaps showing geographical distributions, for all conditional attributes in City A.

D.2 SSIM

We also use the SSIM [69], a classical image fidelity metric, to assess the quality of the traffic heatmaps generated by CartaGenie. SSIM is used along with a traditional metric (like MSE or PSNR), since it does not only compare individual pixels (*i.e.*, grid cells) but also takes into account the whole spatial construct of the generated images (traffic heatmaps in our case) [30].

SSIM is computed as,

$$SSIM(x, s) = \frac{(2\mu_x\mu_s + c_1)(2\sigma_{xs} + c_2)}{(\mu_x^2 + \mu_s^2 + c_1)(\sigma_x^2 + \sigma_s^2 + c_2)}$$

where μ_x (μ_s) is the average traffic among all cells of original image x (synthetic image s); σ_x^2 (σ_s^2) is the variance of x (s), σ_{xs} is the covariance between x and s. $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are the constants where $k_1 = 0.01$ and $k_2 = 0.03$ by default and L is the dynamic range of network traffic observed across the cells. The value of SSIM lies within $[0, 1]$ such that SSIM is 1 when $x = s$.

D.3 OT-SSIM and OT-PSNR

PSNR and SSIM cannot be *directly* used to measure the distance or discrepancy between the corresponding distributions ($p(x)$ and $p(s)$) given only finite samples ($\{x^t\}_{t=1}^T$ and $\{s^t\}_{t=1}^T$). As a solution, we forward this problem to the optimal transport (OT) regime [54] by using PSNR and SSIM as cost functions. The metric *OT-c* we would use, where c is either PSNR or SSIM, is obtained by solving the following optimal transport problem:

$$\text{OT-}c(\{x^t\}_{t=1}^T, \{s^t\}_{t=1}^T) = \max_{\Gamma} \sum_{i=1}^{T} \sum_{j=1}^{T} \Gamma_{i,j} c(x^i, s^j)$$

$$\text{subject to} \quad \Gamma_{i,j} > 0, \sum_{j'=1}^{T} \Gamma_{i,j'} = 1/T, \sum_{i'=1}^{T} \Gamma_{i',j} = 1/T$$

where $c(x, s)$ is either $PSNR(x, s)$ or $SSIM(x, s)$, leading to the corresponding OT-PSNR and OT-SSIM metrics. Intuitively, these OT variants find a pairing matrix (represented by the binary 0-1 matrix Γ) between two sets for which the sum of PSNR or SSIM is maximized. Such formulation is widely used to characterize the distance between two distributions given finite samples [54].

Finally, the formulation of OT-PSNR and OT-SSIM above is only defined over unconditional distributions ($p(x)$ and $p(s)$) given samples ($\{x^t\}_{t=1}^T$ and $\{s^t\}_{t=1}^T$). Although there exist variants of such metrics for the corresponding conditional distributions ($p(x \mid \mathbf{c})$ and $p(s \mid \mathbf{c})$) given a set of finite samples with different conditions ($\{x^t, \mathbf{c}_1^t\}_{t=1}^T$ and $\{s^t, \mathbf{c}_2^t\}_{t=1}^T$), to avoid extra complexity, we simply report OT-PSNR and OT-SSIM for the same set of conditions *separately*. For each city, the spatial conditions are static thus the only variant in conditions is time, or more precisely in our setup, the binary variable which indicates weekday or weekend. For this reason, we report OT-PSNR and OT-SSIM for weekdays and weekends separately for each city.

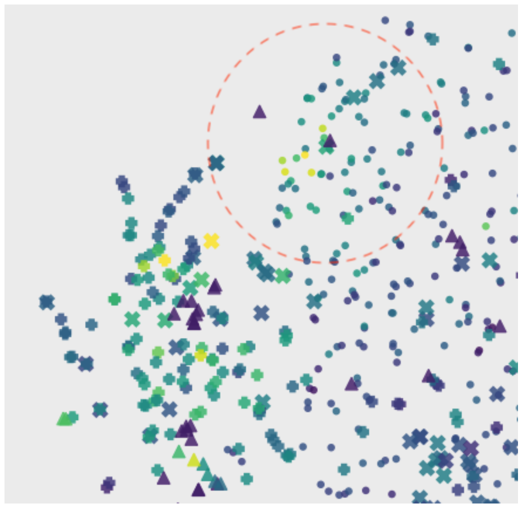


Figure 15: High-traffic region of Figure 16. Labels are consistent with Figure 16: dots for City A, crossings for City B, plus sign for City C and triangle for City D.

E DOMAIN-SHIFT ANALYSIS OVER THE FOUR CITIES

As discussed in § 5.1, CartaGenie seems unable to generate the central hotspot region correctly for City A, when trained on the rest of the cities. We now analyze the underlying root cause. Our hypothesis is that this is in fact expected given the training data used as there is a domain-shift present between training and testing. We now take a visualization approach to confirm this. Specifically, we apply t-SNE [68] to project the context patches of all cities into 2D and make a scatter plot of all points by marking them by their corresponding cities and coloring them via their corresponding central traffic values. Such a t-SNE visualization is shown in Figure 16. In t-SNE plots, close points are neighbors in their original dimension. In our setting, they are similar context patches. This plot can then help us see if there are any context patches that are similar but their corresponding traffic value are dissimilar, especially for those with high-volume traffic.

Firstly, from Figure 16 we can see that the data is homogeneous, except for a few outliers, i.e., points closer to each other but having different colors (traffic values). This means that the conditional distribution $p(\boldsymbol{x} \mid \mathbf{c}, z)$ we want to model is in general consistent among four cities in our dataset. This explains the overall outstanding generalization performance of CartaGenie on the leave-one-city-out evaluations in § 5.1.

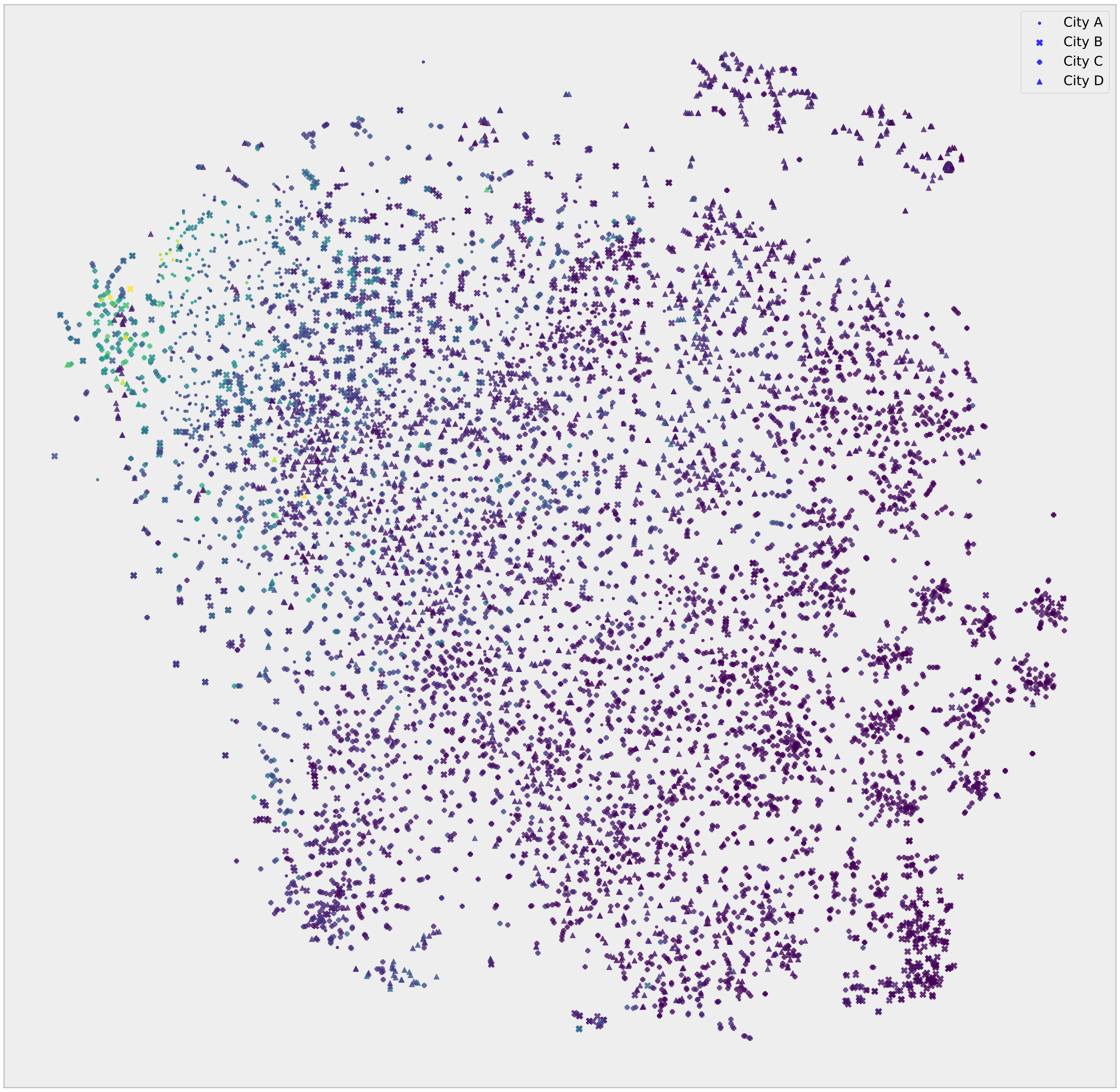


Figure 16: *t*-SNE visualization for four cities studied. Points are colored by the corresponding central traffic values – darker the color smaller (close to 0) the traffic value, and brighter color reflects higher traffic volume.

We now zoom into the region where high-traffic points are located, as shown by Figure 15. Here we can see roughly two regions of hot-spots, between which the one where points from City A are located is annotated by the red dashed circle. Within it, most of the points are from City A (as dots) and only three points are from other cities: 1 from City B (crossing) and 2 from City D (triangles). The colors of those 3 points are not consistent with those from City A here, especially the triangle points. However, they are actually the training data used to train CartaGenie when City A is left out as test city for evaluation. In other words, the training data (City B-D) indeed contains a different pattern (in terms of dependence of traffic on context information) than City A for the hot-spot region in City A: the training data shows a lower traffic volume. This is exactly what we see from the result presented in § 5.1.

In conclusion, the fact CartaGenie has difficulty generating the hot-spot region of City A is rooted in the data and is not a problem of CartaGenie itself. In fact, we argue that any pure data-driven method cannot avoid this problem unless extra information is given, which is out of the scope of this paper.

F ABLATION STUDY

F.1 All you need is explicit loss

As discussed earlier, in our task of strongly-conditioned generation for traffic maps, it turns out that some recent developments in the weakly-conditioned generation are not helpful. Specifically, we found that neither adding an adversarial loss nor using a variational formulation, which are necessary in the context of free-form of weekly-conditioned generation, are needed to generate faithful traffic maps in our setting. We now provide evidence to support our claim by (i) adding an adversarial loss provided by an additional discriminator; and (ii) introducing a variational posterior by an additional encoder for z. The discriminator used for (i) contains two separate networks with the DCGAN architecture that process $\boldsymbol{s}$ and $\boldsymbol{c}$ to some latent embeddings. The two embeddings are then concatenated and passed to a MLP classifier with 2 hidden layers. This specific architecture is required to make the adversarial training successful; simply concatenating $\boldsymbol{s}$ and $\boldsymbol{c}$ via channels does not work. The encoder used for (ii) has a similar architecture to the discriminator of GAN except for the last layer. In the case of GAN, the last layer has 1-dimensional output with sigmoid for classification, whereas with variational encoder the last layer has twice the dimension of latent space. To verify that our discriminator is working properly (which might not always be true depending on network architecture as we explained), we also provide results for which the explicit loss is removed. Table 4 summarizes the train and test performance with these different settings. From the table we can see all training combinations have similar testing performance. As the benefit of using (additional) adversarial loss or introducing a variational posterior for z is not apparent, in CartaGenie, we choose the simplest version – explicit loss only – as it does not require any additional discriminator or encoder.

F.2 The effect of different context sizes

As discussed in §3.2, our specific conditionally-independent instantiation of the generator relies on the assumption that the conditions are large enough to cover the Markov blanket of neighboring pixels. In this section, we empirically study the effect of the condition size on the model performance. To do so, we vary the size of condition ($W_{\boldsymbol{c}} = H_{\boldsymbol{c}}$) within $[15, 20, 25, 30]$ and report the quantitative metrics on the training and testing set in Fig. 17. As it can be seen, using a large context improves training performance, which is as we expect. However, such increasing trend does not transfer to testing performance after $W_{\boldsymbol{c}} = H_{\boldsymbol{c}} = 20$ and even makes it worse beyond that point. This is because when a larger context is given, the model can potentially memorize the mapping from $\boldsymbol{c} \rightarrow \boldsymbol{x}$ (up to some stochasticity). In other words, there is a trade-off between "better" conditioning and the risk of over-fitting. Therefore, to avoid such over-fitting, we take a context with dimension size of 20 in our model. It is worth to mention that a small context size as 15 seems to be strong baseline already: in this case, each pixel of traffic has surrounding context of at least 5×5. This corresponds to 1.25×1.25 km^2, which is reasonably large towards modelling network traffic.

F.3 The necessity of modelling z correctly

In this section we take a closer look of the effect of z in our generator. To recall, CartaGenie uses a FiLM to inject z in the generative process. We consider three alternatives: (1) not modelling z, (2) concatenation of z with $\boldsymbol{c}$, (3) instead using dropout as source of stochasticity to compare with injecting z via FiLM as we do in CartaGenie. Figure 19 illustrates the impact of these modelling differences on the histogram of total traffic in City A. In this figure, City A, City C and City D are used for training and therefore what we plot is on the training set. This is intentional as we are examining the modelling capability instead of assessing generalization, which is a different concern. As expected, Figure 19a shows no variation at all, as the generator is in fact deterministic as we saw previously with CNN Regression (Figure 10c). Figure 19b shows some variation when concatenating z with $\boldsymbol{c}$. The model tends to ignore z as it is from the input directly. In fact, in our early experiment of a very deep architecture, z tends to be ignored completely, yielding similar plot as Figure 19a. Similarly, using dropout as the source of stochasticity fails to model the overall distribution in training; the total traffic plot (Figure 10d) for Pix2Pix, which uses dropout as the source of stochasticity, also shows its poor performance at testing time. Finally, the use of FiLM yeilds successful modelling of variation, shown by Figure 19d.

F.4 Performance under different amount of data

We have also evaluated the effect of training data size and found that CartaGenie's performance improves with increase in training data.

As CartaGenie is a data-driven method, it is worth understanding how the model performs with different amounts of data. To do so, we train the model with all choices of 1, 2 and 3 cities and test it on the rest of cities. We report the average performance of all runs with the same number of training cities on the corresponding training and testing set, shown

Metric	Condition	Type	Explicit	Explicit + Adv.	Explicit + Var.	Adversarial
OT-PSNR↑	Weekday	Train	32.54	29.51	31.41	29.71
		Test	28.93	28.30	28.44	28.59
	Weekend	Train	32.52	30.23	31.30	30.66
		Test	30.05	29.37	29.72	29.77
OT-SSIM↑	Weekday	Train	0.856	0.776	0.841	0.779
		Test	0.765	0.750	0.760	0.758
	Weekend	Train	0.838	0.780	0.839	0.801
		Test	0.775	0.767	0.765	0.764

Table 4: Performance with optional adversarial loss and variational training. Explicit refers to the Laplace likelihood we use by default in CartaGenie. "Adv." is short for "Adversarial" and "Var." is short for "Variational". Note that Explict + Var. is just a normal conditional VAE.

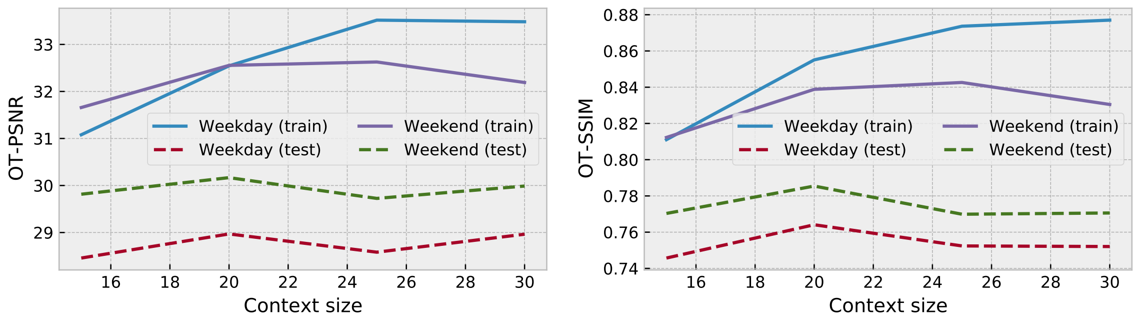


Figure 17: Change of performance by varying W_c and H_c

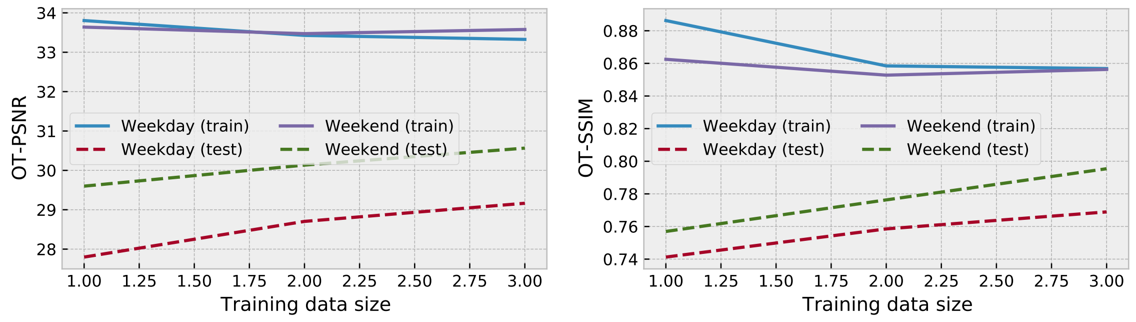
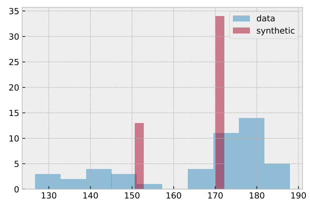


Figure 18: Impact on performance with increasing amount of training data.

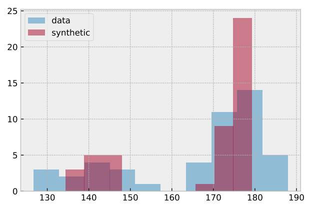
in Figure 18. As it can be seen, there is a clear trend on the benefit of using more data on testing performance, closing the gap between training and testing performance. This is not surprising as our model is based on the maximum likelihood estimation of the underlying probabilistic model, whose asymptotic property guarantees the performance would improve and converge as the data goes to infinity. Importantly, there is no indication of convergence within $1 - 3$ cities, meaning that the performance of our model would be further improve given more data. This is inline with most of the recent successes of deep learning methods which require a large amount of data [27].

F.5 Hyper-parameter tuning

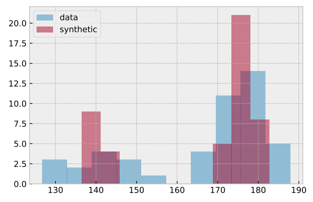
Here we study how the performance of our model changes with the settings of different hyper-parameters. We study the effect of changing the following parameters one at a time while fixing the rest of them to the default settings: (a) number of base features F; (b) size of latent space; (c) batch size; (d) learning rate; and (e) dropout rate. Recall from §3.3 that the default settings for these parameters are: $F = 64$, $N_z = 128$, a batch size of 64, a learning rate of 1×10^{-4} and a dropout rate of 0.02. The change of average training and testing performance is given in Figure 20. For (a), (c) and (e), we simply picked the one that gives overall good performance on training set. For (b) and (d), the trend on training set is monotonically decreasing



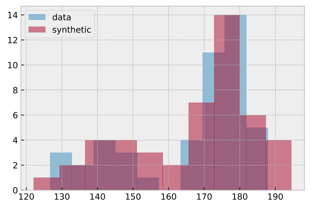
(a) Not modelling z



(b) Concatenating z & c



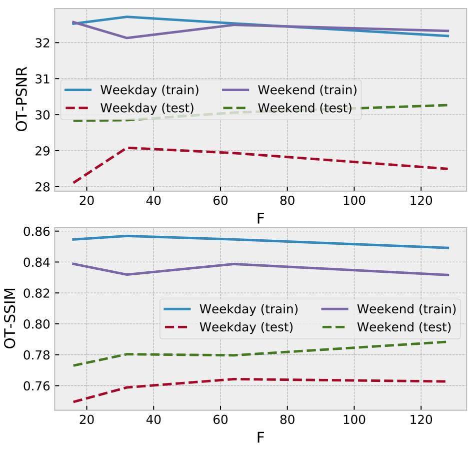
(c) Dropout



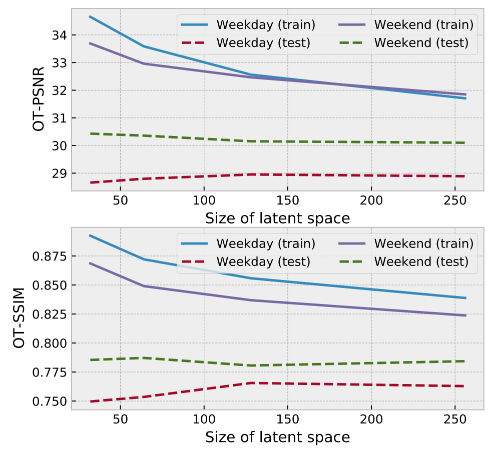
(d) Injecting z via FiLM

Figure 19: Comparison of different latent variable modelling approaches.

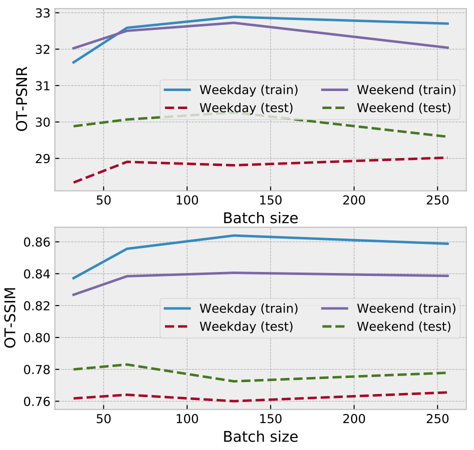
as the PSNR and SSIM metrics do not reflect their effect, e.g. they ignore the importance of stochasticity to be modelled (as also observed earlier in section 5), we instead use typical values, which are $N_z = 128$ and a learning rate of 1×10^{-4}. Note that although some heuristics are used to select hyper-parameters, the actual testing performance is quite insensitive to the changes of all hyper-parameters used here, suggesting the robustness of our model.



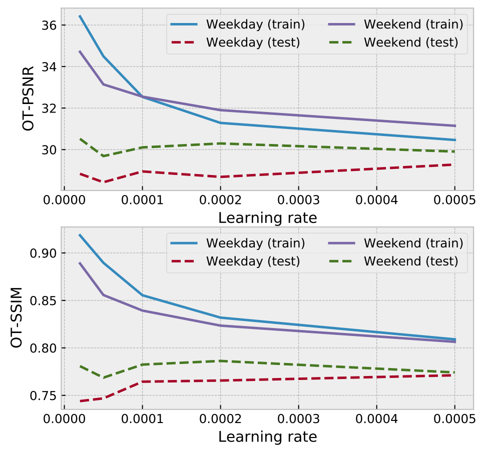
(a) Number of features



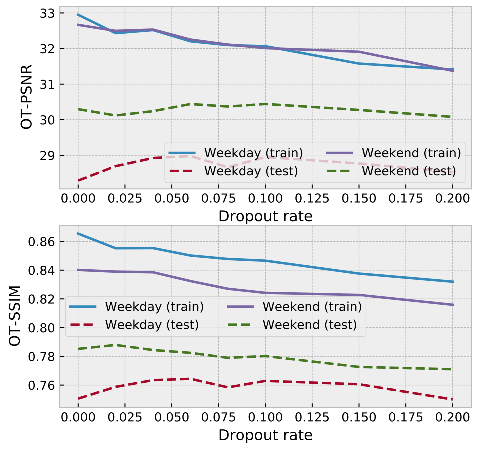
(b) Size of latent space



(c) Batch size



(d) Learning rate



(e) Dropout rate

Figure 20: Effect of various hyper-parameter values on training and testing performance measured via OT-PSNR and OT-SSIM. Numbers are average of 4 runs using each city as the held-out one.