RESEARCH-ARTICLE

Mobile User Traffic Generation Via Multi-Scale Hierarchical GAN

TONG LI, Tsinghua University, Beijing, China

SHUODI HUI, Tsinghua University, Beijing, China

SHIYUAN ZHANG, Tsinghua University, Beijing, China

HUANDONG WANG, Tsinghua University, Beijing, China

YUHENG ZHANG, Tsinghua University, Beijing, China

PAN HUI, Hong Kong University of Science and Technology, Hong Kong, Hong Kong

View all

Published: 26 July 2024
Online AM: 10 May 2024
Accepted: 08 May 2024
Revised: 15 February 2024
Received: 05 January 2023

Mobile User Traffic Generation Via Multi-Scale Hierarchical GAN

TONG LI, SHUODI HUI, SHIYUAN ZHANG, HUANDONG WANG, and YUHENG ZHANG, Tsinghua University, Beijing, China
PAN HUI, Hong Kong University of Science and Technology, Hong Kong, China
DEPENG JIN and YONG LI, Tsinghua University, Beijing, China

Mobile user traffic facilitates diverse applications, including network planning and optimization, whereas large-scale mobile user traffic is hardly available due to privacy concerns. One alternative solution is to generate mobile user traffic data for downstream applications. However, existing generation models cannot simulate the multi-scale temporal dynamics in mobile user traffic on individual and aggregate levels. In this work, we propose a multi-scale hierarchical generative adversarial network (MSH-GAN) containing multiple generators and a multi-class discriminator. Specifically, the mobile traffic usage behavior exhibits a mixture of multiple behavior patterns, which are called micro-scale behavior patterns and are modeled by different pattern generators in our model. Moreover, the traffic usage behavior of different users exhibits strong clustering characteristics, with the co-existence of users with similar and different traffic usage behaviors. Thus, we model each cluster of users as a class in the discriminator's output, referred to as macro-scale user clusters. Then, the gap between micro-scale behavior patterns and macro-scale user clusters is bridged by introducing the switch mode generators, which describe the traffic usage behavior in switching between different patterns. All users share the pattern generators. In contrast, the switch mode generators are only shared by a specific cluster of users, which models the multi-scale hierarchical structure of the traffic usage behavior of massive users. Finally, we urge MSH-GAN to learn the multi-scale temporal dynamics via a combined loss function, including adversarial loss, clustering loss, aggregated loss, and regularity terms. Extensive experiment results demonstrate that MSH-GAN outperforms state-of-art baselines by at least 118.17% in critical data fidelity and usability metrics. Moreover, observations show that MSH-GAN can simulate traffic patterns and pattern switch behaviors.

CCS Concepts: • **Networks** → **Network simulations**; • **Computing methodologies** → **Modeling and simulation**; • **Information systems** → **Spatial-temporal systems**;

Additional Key Words and Phrases: Mobile user traffic, generation, GAN, clustering

Tong Li and Shuodi Hui contributed equally.

This research has been supported in part by the National Natural Science Foundation of China under Grant U20B2060, in part by the National Natural Science Foundation of China under Grant 62171260, and in part by the Institute for Guo Qiang, Tsinghua University under Grant 2023GQS0002.

Authors' Contact Information: Tong Li (Corresponding author), Tsinghua University, Beijing, China; e-mail: tongli@mail.tsinghua.edu.cn; Shuodi Hui, Tsinghua University, Beijing, China; e-mail: hsd17@mails.tsinghua.edu.cn; Shiyuan Zhang, Tsinghua University, Beijing, China; e-mail: zhangshi22@mails.tsinghua.edu.cn; Huandong Wang, Tsinghua University, Beijing, China; e-mail: wanghuandong@tsinghua.edu.cn; Yuheng Zhang, Tsinghua University, Beijing, China; e-mail: z-yh18@mails.tsinghua.edu.cn; Pan Hui, Hong Kong University of Science and Technology, Hong Kong, China; e-mail: panhui@cse.ust.hk; Depeng Jin, Tsinghua University, Beijing, China; e-mail: jindp@tsinghua.edu.cn; Yong Li (Corresponding author), Tsinghua University, Beijing, China; e-mail: liyong07@tsinghua.edu.cn.

ACM 1556-472X/2024/7-ART189
https://doi.org/10.1145/3664655

ACM Reference format:
Tong Li, Shuodi Hui, Shiyuan Zhang, Huandong Wang, Yuheng Zhang, Pan Hui, Depeng Jin, and Yong Li. 2024. Mobile User Traffic Generation Via Multi-Scale Hierarchical GAN. *ACM Trans. Knowl. Discov. Data.* 18, 8, Article 189 (July 2024), 19 pages.
https://doi.org/10.1145/3664655

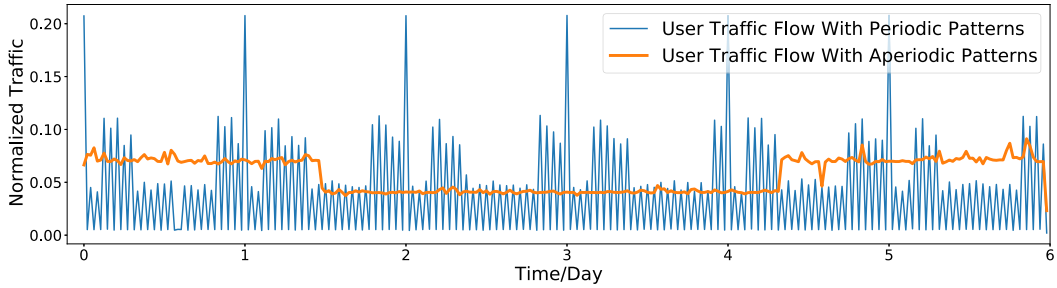
1 Introduction

Mobile networks have gained widespread popularity in recent years. In 2022, there were over five billion unique mobile internet users [1], indicating that over 60% of the world's population uses a mobile device to access the internet. This data underscores the indispensable role that mobile networks play in modern society. Under this circumstance, collecting and utilizing the network traffic of mobile users is of increasing importance and popularity [2, 3]. Specifically, mobile user traffic facilitates diverse applications, including user behavior modeling [4, 5], network digital twinning [6–8], malware detection [9, 10], advertising fraud detection [11], sociological inference [12], website fingerprinting [13], and so forth. Mobile user traffic also contributes to the progress of mobile networks by inspiring the improvement of network planning [14, 15] and optimization [16–18].

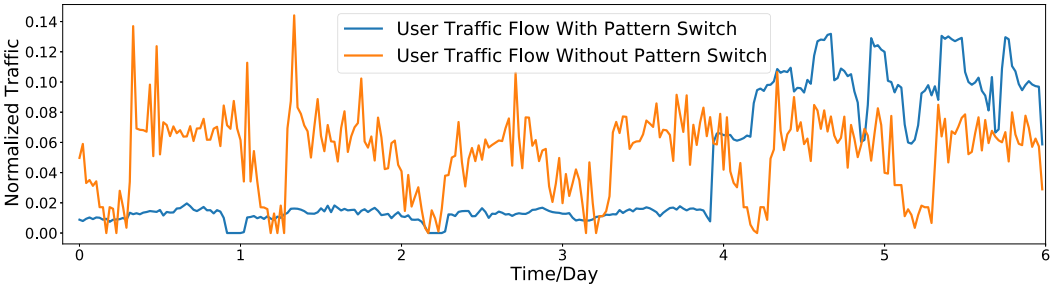
Mobile user traffic is hardly available in practice. Due to privacy concerns for mobile users, mobile network providers with access to mobile user traffic are often unwilling to make this data public to the research community [19, 20]. Alternatively, they may add noise to the data before its utilization, reducing its quality, and usability [21–23]. Moreover, collecting mobile user traffic data via crowdsourcing requires careful handling of the tradeoff between data quality and expenses [24, 25]. To mitigate the above gaps, we propose to generate synthetic mobile user traffic data, avoiding privacy issues in data sharing while reducing costs in data collection. Recently, several models have been proposed to generate various forms of traffic data, including network packet sequences [26–28] and traffic volume series [29–32]. These previous models predominantly concentrate on generating network traffic at an aggregated level, such as network packets of a local network or the traffic volume of a specific region. However, their limitations lie in their inability to generate network traffic traces for individual users, which is the primary focus of our work.

Network traffic usage by mobile users displays complex characteristics from various perspectives, including individual micro-scale traffic usage patterns and macro-scale group patterns observed among mobile users. Specifically, generating traffic usage for mobile users requires careful consideration of both characteristics, presenting challenges.

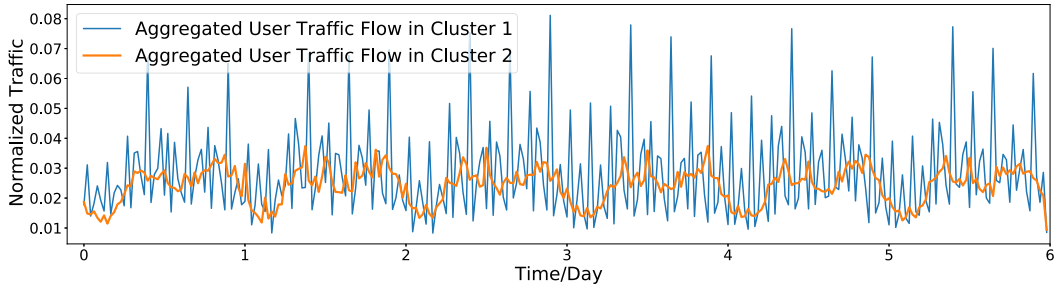
— *Individual micro-scale traffic usage patterns*. The traffic usage of mobile users showcases individual characteristics that emerge from a blend of multiple behavior patterns, termed micro-scale behavior patterns. These patterns often result from basic mobile app usage activities like online messaging, video streaming, and online shopping. Notably, as only one app is typically active in the foreground of a mobile phone and primarily consumes the mobile traffic, these micro-scale behavior patterns are more pronounced in the mobile domain than in wired networks [33]. For example, Figure 1(a) presents two selected user traffic flows, where the blue curve shows multi-periodicity while the yellow curve shows slight fluctuations.
— *Macro-scale group patterns*. Beyond individual patterns, mobile user traffic also demonstrates group-level patterns, indicating strong clustering characteristics in their behavior. This phenomenon is described as macro-scale group patterns. Distinct user groups sharing similar features exhibit micro-scale behaviors in specific constellations. Figure 1(c) illustrates the



(a) User Traffic Flows With/Without Periodic Patterns.



(b) User Traffic Flows With/Without Pattern Switch.



(c) Aggregated User Traffic Flows in Different Clusters.

Fig. 1. Examples of the real user traffic flows: (a) with/without pattern switch, (b) with/without periodic patterns, and (c) aggregated traffic.

aggregated user traffic flows of two such clusters. While both curves display daily periodic patterns, the blue curve additionally reveals high-frequency periodicity.

This article proposes a **Multi-Scale Hierarchical Generative Adversarial Network (MSH-GAN)** for mobile user traffic generation. First, we simulate the micro-scale behavior patterns in user traffic via multiple pattern generators. We utilize the **Bidirectional Long Short-Term Memory (BiLSTM)** networks and self-attention mechanism to capture the long-term and short-term temporal correlations. Then, we bridge the gap between micro-scale behavior patterns and macro-scale user clusters by introducing the switch mode generators, which describe the traffic usage behavior in switching between different patterns. For example, Figure 1(b) presents two selected user traffic flows, where the yellow curve shows stable patterns corresponding to daily life,

while the blue curve switches between distinct patterns on the fourth day. Thus, users' traffic usage behavior is formed by the cooperation of pattern generators and switch mode generators. All users share the pattern generators, while a specific cluster only shares the switch mode generators. In this way, we model the multi-scale hierarchical structure of the traffic usage behavior of massive numbers of users. Our multi-class discriminator provides the true or false results to discriminate micro-scale behavior patterns and classifies the user traffic into macro-scale user clusters, where we adopt **Temporal Convolutional Networks (TCNs)** with daily and weekly kernels to capture the multi-scale temporal dynamics. Finally, we update MSH-GAN using a delicately designed loss function, including adversarial loss, clustering loss, aggregated loss, and regularity terms.

Our contributions are summarized as follows:

—We propose MSH-GAN to simulate the multi-scale hierarchical structure of the traffic usage behavior of massive users, generating the traffic patterns with multiple pattern generators and pattern switch modes with multiple switch mode generators while discriminating and clustering the mobile user traffic with a multi-class discriminator.
—We design a combined loss function to urge the MSH-GAN to learn individual- and aggregate-level multi-scale temporal dynamics of user traffic, including adversarial loss, clustering loss, aggregated loss, and regularity terms.
—We conduct experiments on a real-world user traffic dataset, where we train the MSH-GAN and other baseline models and utilize the trained models to generate synthetic user traffic for individual-level and aggregate-level evaluation. Extensive results show that MSH-GAN outperforms other state-of-art baselines by more than 118.7% in terms of key metrics on fidelity and usability while successfully simulating individual-level and aggregate-level traffic behaviors.

2 Problem Definition and System Overview

2.1 Definitions

Definition 1. (User Traffic Flow). Given a mobile user U, the traffic flow is defined as the series of network traffic volumes consumed in equal time intervals $V = \{v_t\}_{t=1}^{T}$, where v_t represents the network traffic volume consumed in the t_{th} interval and T is the amount of these intervals.

Focusing on simulating the traffic variations of each user, we apply normalization to the user traffic flows.

Definition 2. (Normalized User Traffic Flow). Given a mobile user U, the normalized traffic flow is defined as $X = \{x_t\}_{t=1}^{T} = \{v_t / \|V\|_2\}_{t=1}^{T}$, where x_t represents the normalized network traffic volume in the t_{th} time interval, $\|V\|_2$ is the two norm of user traffic flow V.

Definition 3. (Aggregated Normalized User Traffic Flow). Given a cluster of mobile users $\{U^i\}_{i=1}^{N}$ and their normalized traffic flow $\{X^i\}_{i=1}^{N}$, the aggregated normalized traffic flow is defined as $A = \{a_t\}_{t=1}^{T} = \{\sum_{i=1}^{N} x_t^i\}_{t=1}^{T}$, where a_t represents the aggregated normalized network traffic volume in the t_{th} time interval.

As mobile users can change their behavior patterns significantly in the normalized user traffic flow, we define the consistent behavior patterns and the switch modes among these patterns to decompose the user traffic generation problem at micro and macro scales.

Definition 4. (Traffic Pattern). A traffic pattern is a normalized user traffic flow with consistent temporal patterns, which is defined as $P = \{p_t\}_{t=1}^{T}$, where p_t represents the normalized network traffic volume of the traffic pattern in the t_{th} time interval.

Table 1. Notations and Descriptions for User Traffic Generation

Notation	Description
U	A mobile user
$V = \{v_t\}_{t=1}^{T}$	The traffic flow of mobile user U
$X = \{x_t\}_{t=1}^{T}$	The normalized traffic flow of mobile user U
$A = \{a_t\}_{t=1}^{T}$	The aggregated normalized traffic flow of mobile users $\{U^i\}_{i=1}^{N}$
$P = \{p_t\}_{t=1}^{T}$	The traffic pattern
$S = \{s_t\}_{t=1}^{T}$	The traffic switch mode of mobile user U
C	The cluster of mobile users $\{U^i\}_{i=1}^{N}$ according to their $\{X^i\}_{i=1}^{N}$
Z	The random noise vector for a generation

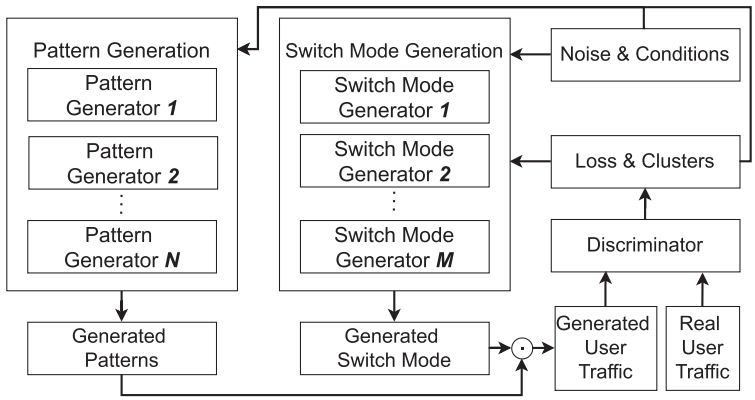


Fig. 2. Overview of user traffic generation via MSH-GAN.

Definition 5. (Traffic Switch Mode). Each user's traffic switch mode is defined as $S = \{s_t\}_{t=1}^{T}$, where s_t is a one-hot encoding vector representing the traffic pattern obeyed by the user in the t_{th} time interval.

2.2 Problem Formulation

Based on the above-defined notations and concepts, the user traffic generation problem can be expressed as follows:

Definition 6. (User Traffic Generation Problem). Given the real normalized user traffic flow set $\{X\}$, the goal is to generate a synthetic normalized user traffic flow set $\{\hat{X}\}$ that exhibits similar characteristics to the real training data. In addition, auxiliary information of the users can be optionally utilized in this process.

2.3 System Overview

Figure 2 shows the workflow of MSH-GAN, where we generate synthetic normalized user traffic flows and discriminate between them and the real flows. Specifically for each user, we first generate

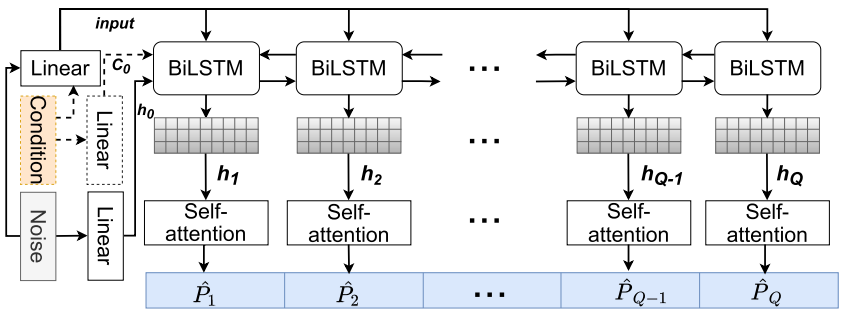


Fig. 3. Illustration of the pattern generator.

several traffic patterns $\{\hat{P}^i\}_{i=1}^N$ and the traffic switch mode $\hat{S}$, where the dimension of each one-hot encoding vector $\hat{s}_t$ is equal to the number of patterns N. Then, we take the element-wise multiplication of the traffic patterns $\{\hat{P}^i\}_{i=1}^N$ and the traffic switch mode $\hat{S}$ as the generated normalized user traffic flow $\hat{X}$, and discriminate between $\hat{X}$ and the real traffic X. Particularly, we utilize multiple switch mode generators to simulate the variable switch modes of mobile users. As we found that the aggregated normalized user traffic flow A of mobile users $\{U^i\}_{i=1}^N$ can show different patterns corresponding to the clusters, we adjust the discriminator to output the true or false results along with the cluster for each flow X or $\hat{X}$. Finally, we update generators and the discriminator via loss functions computed according to the discrimination results of X and $\hat{X}$.

3 Method

As discussed in the introduction section, mobile user traffic shows multi-scale temporal behaviors on the individual level and multi-periodic patterns on the aggregate level. Therefore, we design multiple pattern generators, switch mode generators, and a multi-class discriminator. We utilize BiLSTM networks and self-attention mechanisms in each pattern generator to capture the long-term and short-term temporal dynamics in traffic patterns. In each switch mode generator, we utilize a linear **Switch Learner (SL)** to memorize and update the pattern switch modes and use Gumbel-Softmax to output the pattern switch results. Then, we generate synthetic mobile user traffic as the element-wise product of the generated traffic patterns and switch modes. Next, we input the real and generated mobile user traffic into the discriminator, providing the discrimination result as the probabilities of the input traffic being fake and belonging to each cluster. According to the discrimination results, we calculate the combined loss function to update the generators and discriminator, simultaneously considering individual and aggregate levels.

3.1 Pattern Generator

To simulate the long-term and short-term temporal dynamics in the traffic pattern, we utilize BiLSTM [34] networks and self-attention mechanism [35] to generate the traffic patterns $\{\hat{P}\}$, as illustrated in Figure 3. **Long Short-Term Memory (LSTM)** networks [36] are **Recurrent Neural Network (RNN)** architectures known for their ability to remember historical values over arbitrary intervals and generate series data step by step. In a typical LSTM network, the output at each step is influenced not only by the current input but also by the input and states from previous steps. BiLSTM network [34] takes this a step further by allowing the current step to be influenced by the following steps. In other words, the BiLSTM can capture past and future context by processing the input sequence in both forward and backward directions. This capability enables the BiLSTM better to understand the temporal dependencies within user network traffic. Therefore, we adopt BiLSTM

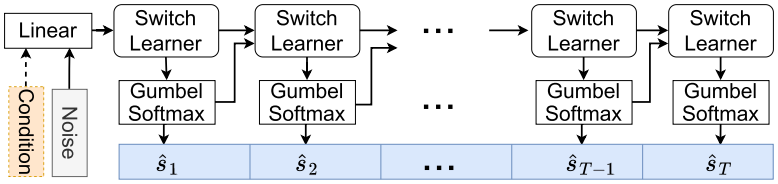


Fig. 4. Illustration of the switch mode generator.

networks to learn the long-term and short-term temporal dynamics of traffic patterns. Specifically, we map the condition vector and noise vector to the initial cell state c_0 and the initial hidden state h_0 of the BiLSTM via two linear layers and map the condition and noise vector to the input of each step for the BiLSTM via another linear layer. As presented by the dashed yellow box and lines in Figure 3, the condition vector is optional according to the existence of auxiliary information of the users, and the initial cell state c_0 is set to zeros without the condition vector.

To simulate the short-term temporal dynamics in the traffic pattern more delicately while improving the efficiency, we adopt self-attention mechanism to process the output hidden states and generate multiple elements in each step

$$\begin{aligned} R_K = \text{ReLU}(h_k W_K), R_Q = \text{ReLU}(h_k W_Q), R_V = \text{ReLU}(h_k W_V), \\ \hat{P_k} = \text{softmax}\left(\frac{R_Q \cdot R_K^T}{\sqrt{d}}\right) \cdot R_V \cdot W_s. \end{aligned} \tag{1}$$

As presented in Equation (1), the self-attention layer maps the hidden state h_k in each step to the key R_K, query R_Q, and value R_V representations via linear projections, and the dot products of the query with all keys and the weights on the values are computed via a softmax activate function after normalizing the products by the dimension of keys. Finally, it takes Q steps to generate the traffic pattern $\hat{P}$, and $\hat{P_k} = \{p_t\}_{t=(k-1)\cdot Q+1}^{k\cdot Q}$ in each step. Notably, the traffic pattern generator G^P has a single BiLSTM unit and self-attention layer. The expanded structure presented in Figure 3 stands for the Q steps in the generation. Finally, we generate the traffic patterns $\{\hat{P^i}\}_{i=1}^N$ with N pattern generators.

3.2 Switch Mode Generator

To simulate the switch modes among traffic patterns for each user, we utilize a linear SL to memorize and update the switch modes and adopt Gumbel-Softmax [37] to generate the one-hot encoding vector while avoiding discontinuity step by step as follows,

$$\begin{aligned} m_0 &= \text{Linear}(Z; condition^*), \\ \hat{s_t} &= \text{GumbelSoftmax}(m_t), \\ m_{t+1} &= \text{SL}(m_t, \hat{s_t}), \end{aligned} \tag{2}$$

where Z is the random noise vector, $condition$ is the auxiliary information of the users, m_0 the initial switch state, m_t the switch state in the t_{th} step, $\hat{s_t}$ is the generated one-hot encoding vector representing the traffic pattern in t_{th} step, and SL represents the switch learner.

As presented in Figure 4, we first map the noise vector and condition vector to the initial switch state m_0 for the SL. In the SL, we memorize the switch modes via a matrix, where the elements represent the previous switches among traffic patterns. With a one-hot encoding vector representing the traffic pattern $\hat{s_t}$ as input, the SL outputs the next switch state m_{t+1} based on the previous switches and $\hat{s_t}$. Gumbel-Softmax [38] is a continuous distribution that can approximate categorical

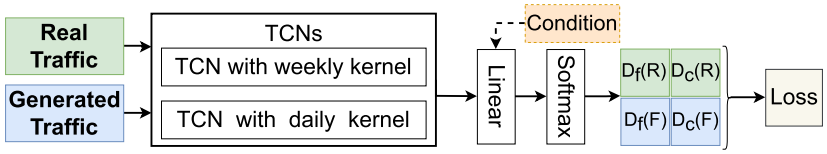


Fig. 5. Illustration of the discriminator.

samples, which applies the softmax function to the Gumbel-Max trick [39]. Gumbel-Max trick helps to draw samples $\hat{s_t}$ from a multinomial distribution with probabilities m_t as follows,

$$\hat{s_t} = \text{one\_hot}\left(\arg\max_i \left[g^i + \log m_t^i\right]\right), \; g^i = -\log(-\log(u^i)), \tag{3}$$

where m_t^i is the i_{th} element in m_t, g^i is the independent identically distributed samples drawn from the $Gumbel(0,1)$ distribution, u^i is the independent identically distributed samples drawn from the $Uniform(0,1)$ distribution. Then, replacing the arg max function with the softmax function, the Gumbel-Softmax trick can be expressed as follows,

$$\hat{s_t}^i = \frac{\exp\left(\left(\log\left(m_t^i\right) + g^i\right)/\tau\right)}{\sum_{j=1}^{N} \exp\left(\left(\log\left(m_t^j\right) + g^j\right)/\tau\right)} \quad \text{for } i = 1, \ldots, N, \tag{4}$$

where $\hat{s_t}^i$ is the i_{th} element in $\hat{s_t}$, N is the dimension of $\hat{s_t}$, which equals the total number of traffic patterns. Then, $\hat{s_t}$ becomes a one-hot encoding vector with the softmax temperature τ approximating zero, while the parameter gradients can be easily computed via the reparameterization trick. Particularly, as presented by the dashed yellow box and lines in Figure 4 and the * in Equation (2), the condition vector is optional according to the existence of auxiliary information of the users.

With the generated traffic patterns $\{\hat{P^i}\}_{i=1}^N$ and switch mode $\hat{S}$, the generated multi-scale normalized user traffic flow is computed via element-wise multiplication $\hat{X} = [\hat{P^1}, \hat{P^2}, ..., \hat{P^N}] \odot \hat{S}$.

3.3 Discriminator

The aggregated normalized user traffic flow A of mobile users $\{U^i\}_{i=1}^N$ can show different patterns corresponding to distinct clusters. To bridge the gap between individual-level and aggregate-level behaviors, the discriminator in our model should act as a multi-class classifier to provide the cluster label for each flow X or $\hat{X}$ if it is discriminated as real. Hence, the output of the discriminator can be expressed as follows,

$$\begin{aligned} D(X) &= (D_f(X), D_c(X)), \\ D_c(X) &= [D_c^1(X), D_c^2(X), ..., D_c^M(X)], \end{aligned} \tag{5}$$

where $D_f(X)$ is the probability that the traffic flow X is discriminated as fake by the discriminator, and $D_c^i(X)$ is the probability that the traffic flow X is discriminated as real in the i_{th} cluster by the discriminator. Then, the sum of these probabilities equals one, i.e., $D_f(X) + \sum_{i=1}^M D_c^i(X) = 1$, representing that each traffic flow is either fake or real in one of the clusters. Notably, there is no pre-determined ground truth or labels for clusters. The discriminator automatically determines the clustering results to maximize inter-cluster distances while minimizing intra-cluster distances, as detailed in the loss function (7).

To capture the temporal correlations in the real and generated traffic, we adopt TCN [40] with daily and weekly kernels. TCN is a one-dimension convolutional network with casual and dilated convolutions between the input layer, hidden layers, and output layer, capturing features from each layer via convolutions and passing the feature to the next layer. In each layer, the feature

is calculated based on values within the convolutional kernel size. Therefore, TCNs with daily and weekly kernels help to capture the temporal correlations in previous days and weeks across multiple layers. As presented by Figure 5, the real traffic flow X and generated traffic flow $\hat{X}$ are input to the TCNs, and probabilities are given through a linear layer and a softmax layer. Similar to the pattern generator and switch mode generator, the condition vector is optional according to the auxiliary information of users, as presented by the dashed yellow box and lines in Figure 5.

3.4 Loss Functions

To simultaneously simulate the individual-level and aggregate-level multi-scale traffic, we consider clustering loss and aggregation loss in addition to the traditional adversarial loss. Moreover, we regularize generators and the discriminator for more realistic results.

3.4.1 Discrimination Loss. The loss function for discrimination includes: adversarial loss, clustering loss, and regularity term. To discriminate the real and generated traffic flows, the adversarial loss can be expressed as follows,

$$loss_D^{adversarial} = \mathbb{E}_{X\sim p_r}\left[\log\left(D_f\left(X\right)\right)\right] + \mathbb{E}_{\hat{X}\sim G}\left[\log\left(1 - D_f\left(\hat{X}\right)\right)\right], \tag{6}$$

where X is the real traffic flow, p_r represents the distribution of real data, $\hat{X}$ is the generated traffic flow, G represents the generator, D_f is the probability that the sample is false given by the discriminator.

The clustering loss urges the inner-cluster distances to be less than the intra-cluster distances, which can be expressed as follows,

$$\begin{aligned} loss_D^{clustering} &= \sum_{i=1}^{M}\left[\left(\sum_{D_c(X)=C_i} X - O_i\right) - \left(\sum_{D_c(X)\neq C_i} X - O_i\right)\right], \\ O_i &= \sum_{D_c(X)=C_i} X/|C_i|, \end{aligned} \tag{7}$$

where $D_c(X)$ is the cluster of the real traffic flow X given by the discriminator, C_i is the i_{th} cluster, O_i is the center of cluster C_i, and M is the total number of clusters.

A good discriminator should provide as much information as possible for each sample, where the output $D(X)$ or $D(\hat{X})$ should be more similar to a one-hot encoding, which requires the weight of the linear layer in the discriminator be more similar to an orthogonal matrix. Therefore, the regularity term can be expressed as follows,

$$loss_D^{regularity} = \mathrm{AE}(D'_w D_w, I), \tag{8}$$

where D_w is the weight of the linear layer in the discriminator, D'_w is the transpose of D_w, I is the identity matrix, and AE means the Absolute Error.

The loss function for discrimination can be expressed as follows,

$$loss_D = loss_D^{adversarial} + \beta_c * loss_D^{clustering} + \beta_r * loss_D^{regularity}, \tag{9}$$

where β_c and β_r are the relative magnitudes of the clustering and regularization coefficients.

3.4.2 Generation Loss. The loss function for generation includes three parts: adversarial loss, aggregation loss, and continuity loss for the switch mode generators. To fool the discriminator with generated traffic flows, the adversarial loss can be expressed as follows,

$$loss_G^{adversarial} = \mathbb{E}_{\hat{X}\sim G}\left[\sum_{i=1}^{N}\log\left(D_c^i\left(\hat{X}\right)\right)\right], \tag{10}$$

where $D_c^i\left(\hat{X}\right)$ is the probability that the traffic flow $\hat{X}$ is discriminated as real in the i_{th} cluster by the discriminator.

The aggregated traffic of the generated traffic flows in each cluster should be similar to the aggregated traffic of the real traffic flows in the same cluster, and the aggregation loss can be expressed as follows,

$$loss_G^{aggregation} = \sum_{i=1}^{N} \text{AE}(\hat{A}_i, A_i) = \sum_{i=1}^{N} \text{AE}\left(\sum_{D_c(\hat{X})=C_i} \hat{X}, \sum_{D_c(X)=C_i} X\right), \tag{11}$$

where A_i is the aggregated traffic of the real traffic flows $\{X\}$ in cluster C_i, and $\hat{A}_i$ is the aggregated traffic of the generated traffic flows $\{\hat{X}\}$ in cluster C_i.

As most users tend not to switch frequently among different traffic patterns, the generated switch modes should retain continuity to some extent. Therefore, the continuity loss for the switch mode generators can be expressed as follows,

$$loss_G^{switch} = \left\|\text{diff}(\hat{S})\right\|_2 = \left\|\sum_{t=1}^{T-1} \hat{s_{t+1}} - \hat{s_t}\right\|_2, \tag{12}$$

where $\hat{S}$ is the generated switch mode, $\hat{s_t}$ is the generated one-hot encoding vector representing the traffic pattern in the t_{th} time interval, diff means the first-order difference, and $\|\cdot\|_2$ means the two norm.

The loss function for generation can be expressed as follows,

$$loss_G = loss_G^{adversarial} + \beta_a * loss_G^{aggregation} + \beta_s * loss_G^{switch}, \tag{13}$$

where β_a and β_s are the relative magnitudes of the aggregation and switch continuity coefficients.

3.5 Model Training

We train the generators and discriminator alternately until convergence. In each iteration, we first input the real traffic flows $\{X\}$ to the discriminator. Then, according to the discriminated clusters of the real traffic flows $\{X\}$, we generate fake switch modes $\{\hat{S}\}$ with different switch mode generators and fake traffic patterns $\{\hat{P}\}$ with multiple traffic pattern generators. With the generated switch modes $\{\hat{S}\}$ and traffic patterns $\{\hat{P}\}$, we take element-wise multiplication and input the generated fake traffic flows $\{\hat{X}\}$ to the discriminator. After discriminating all the real traffic flows and generating the corresponding fake ones, we compute the loss functions to update the model. We use a mini-batch to improve the training efficiency, and the details are presented in Algorithm 1.

4 Experiments

We conduct experiments on a real-world user traffic dataset, evaluating MSH-GAN on individual, aggregate, and application levels.

4.1 Experiment Setting

4.1.1 Dataset. We obtained a user traffic dataset collected by one of the major Internet Service Providers from Shanghai, a big city in China. The dataset contains nearly two billion network traffic records of over one million mobile users in Shanghai, collected between April 20th and April 26th, 2016. However, the records of most users are sparse in the time domain, and several users have no records for days. Thus, we select 6,055 users with dense records in the time domain.

Algorithm 1 MSH-GAN. Default values : $N = 6$, $M = 6$, $\tau = 0.1$, $\alpha = 0.0001$, $\alpha_s = 0.01$, $\beta_1 = 0.5$, $\beta_2 = 0.9$, $\beta_c = 1$, $\beta_r = 100$, $\beta_a = 0.001$, $\beta_s = 1$, $n_D = 1$, $B = 1024$.

Require: The pattern number N, the cluster number M, the softmax temperature τ, Adam hyperparameters α, α_s, β_1, β_2, the clustering coefficient β_c, the regularity coefficient β_r, the aggregation coefficient β_a, the switch continuity coefficient β_s, the number of discriminator iterations per generator iteration n_D,
Initialize: Initial pattern generator parameters ϕ_0, initial switch model generator parameters φ_0, initial discriminator parameters θ_0.

while ϕ, φ, θ has not converged **do**
  for i=1, ... , n_D **do**
    for j=1, ... , B **do**
      D_θ Training:
      Sample real traffic flow $X \sim P_r$,
        noise $Z \sim P_z$,
      for k = 1, ... , N **do**
        $\widehat{P}^k = G_\phi^k(Z)$
      end for
      $l = \arg\max D_\theta(X), \quad l = 1, \dots, M$
      $\widehat{S} = G_\varphi^l(Z)$
      $\widehat{X} = [\widehat{p}^1, \widehat{p}^2, \dots, \widehat{p}^N] \bigodot \widehat{S}$
      $L_D^j \leftarrow L_D^{adversarial}\ (D_\theta(X), D_\theta(\widehat{X}))$
    end for
    $L_D \leftarrow \frac{1}{B} \sum_{j=1}^{B} L_D^j + \beta_c\, L_D^{clustering}(\{X\}, \{D_\theta(X)\}) +$
      $\beta_r\, L_D^{regularity}(D_\theta)$
    $\theta \leftarrow \text{Adam}\ (\nabla_\theta L_D, \alpha, \beta_1, \beta_2)$
  end for
  G_ϕ, G_φ Training:
  for j=1, ... , B **do**
    Sample real traffic flow $X \sim P_r$,
      noise $Z \sim P_z$
    for k = 1, ... , N **do**
      $\widehat{P}^k = G_\phi^k(Z)$
    end for
    $l = \arg\max D_\theta(X), \quad l = 1, \dots, M$
    $\widehat{S} = G_\varphi^l(Z)$
    $\widehat{X} = [\widehat{p}^1, \widehat{p}^2, ..., \widehat{p}^N] \bigodot \widehat{S}$
    $L_G^j \leftarrow L_G^{adversarial}\ (D_\theta(\widehat{X}))$
  end for
  for l = 1, ... , M **do**
    $A^l \leftarrow \sum_{D_\theta(X)=l} X$
    $\widehat{A}^l \leftarrow \sum_{D_\theta(\widehat{X})=l} \widehat{X}$
  end for
  $L_G \leftarrow \frac{1}{B} \sum_{j=1}^{B} L_G^j + \beta_a \sum_{l=1}^{M} L_G^{aggregation}(\widehat{A}^l, A^l) +$
    $\beta_s \sum_{l=1}^{M} L_G^{switch}(G_\varphi)$
  for k = 1, ... , N **do**
    $\phi(K) \leftarrow \text{Adam}(\nabla_{\phi(K)}\ L_G, \alpha, \beta_1, \beta_2)$
  end for
  for l = 1, ... , M **do**
    $\varphi(l) \leftarrow \text{Adam}(\nabla_{\phi(l)}\ L_G, \alpha_s, \beta_1, \beta_2)$
  end for
end while

4.1.2 Baselines. We compare MSH-GAN with the following three baselines.

Recurrent GAN (RGAN) [41]. RGAN makes use of RNNs in the generator and the discriminator, which can generate sequences of real-valued data. We use $\{X\}$ as input to train this model and generate the synthetic traffic flow set $\{\hat{X}\}$.

Continuous RNN-GAN (C-RNN-GAN) [42]. Similar to RGAN, this model proposes an RNN architecture trained with adversarial training to model the joint probability of sequences and generate continuous sequence data. Also, $\{X\}$ are used to train the model and get the synthetic data $\{\hat{X}\}$.

Time-series GAN (TimeGAN) [43]. This model introduces an additional step-wise supervised loss using the original data as supervision to learn the temporal dynamics of the original data and use an embedding network to get a reversible mapping between features and latent representations. We use $\{X\}$ as the original data and get the synthetic data $\{\hat{X}\}$.

DoppelGANger [30]. This model leverages a conditional RNN-based GAN for generating traffic forms data, combining the data attributes and feature series with variable length. We use $\{X\}$ as the original data to train this model and get the synthetic data $\{\hat{X}\}$.

4.1.3 Metrics. We compare MSH-GAN with the baseline models via the following five metrics:

Traffic Volume. We compare the distribution of traffic volume of the generated user traffic with the real distribution via **Jensen–Shannon Divergence (JSD)** [44]. JSD is a commonly used metric to evaluate the similarity between two distributions, which can be expressed as follows,

$$\mathrm{JSD}(\mathbf{P}_{\{\hat{X}\}}, \mathbf{P}_{\{X\}}) = \sqrt{\frac{\mathbf{KL}(\mathbf{P}_{\{\hat{X}\}} \| \mathbf{P}_{\{\tilde{X}\}}) + \mathbf{KL}(\mathbf{P}_{\{\tilde{X}\}} \| \mathbf{P}_{\{X\}})}{2}}, \tag{14}$$

where $\mathbf{P}_{\{X\}}$ represents the distribution of the real samples $\{X\}$, $\mathbf{P}_{\{\hat{X}\}}$ is the distribution of the generated samples $\{\hat{X}\}$, $\mathbf{P}_{\{\tilde{X}\}}$ represents the point-wise mean of $\mathbf{P}_{\{X\}}$ and $\mathbf{P}_{\{\hat{X}\}}$, and $\mathbf{KL}$ is the Kullback–Leibler divergence. A lower JSD means a closer distribution to the real data, which indicates a better generation model.

First-Order Difference. To evaluate the dynamics of the user traffic, we compute its first-order difference as $X' = \{x_{t+1} - x_t\}_{t=1}^{T-1}$. Then we use JSD to evaluate the similarity between the distribution of the first-order difference of the real and generated traffic, which is denoted as $\mathrm{JSD}(\mathbf{P}_{\{\hat{X}'\}}, \mathbf{P}_{\{X'\}})$.

Daily Frequency Component. We compute the frequency spectrum of the aggregated user traffic flow A as $F = FFT(A)$. Next, the proportion of daily frequency component can be calculated as $F^D = \|F[d]\|_2 / \|F\|_2$, where $\|F\|_2$ is the two norm of the frequency spectrum, and $\|F[d]\|_2$ is the two norm of daily frequency component, and d is the total number of days. Then, we use AE to evaluate the similarity between the daily frequency proportion of the real and generated aggregated traffic, which can be expressed as $\mathrm{AE}(F^D, \hat{F^D})$.

Daily Periodic Error (DPE). To evaluate the daily periodicity in the aggregated traffic, we compute the DPE as the pairwise distance of the aggregated traffic on each day as follows,

$$\begin{aligned} \mathrm{DPE}(A) &= \sum_{i=1}^{d} \sum_{j=1}^{d} |A(i) - A(j)|, \\ A(i) &= \{a_t\}_{t=(i-1)\cdot T/d+1}^{i\cdot T/d}, \end{aligned} \tag{15}$$

where $A(i)$ is the aggregated traffic in the i_{th} days, and d is the total number of days. Then, we use AE to evaluate the similarity between the DPE of the real and generated aggregated traffic, which can be expressed as $\mathrm{AE}(\mathrm{DPE}(A), \mathrm{DPE}(\hat{A}))$.

Table 2. Evaluation Results of User Traffic Generated by Different Models, Where Lower Results are Better

Metrics	Traffic Volume		First-order Difference		Daily Frequency Component		Daily Periodic Error		Traffic Prediction		$\overline{\Delta}$
	JSD	Δ	JSD	Δ	AE	Δ	AE	Δ	MSE	Δ	
RGAN	*0.1625*	13.64%	0.1150	100.18%	0.0817	125.11%	0.1175	793.27%	0.2792	156.62%	237.76%
C-RNN-GAN	0.6422	349.09%	0.1001	74.09%	0.0830	128.74%	*0.0141*	7.21%	0.1462	34.38%	118.70%
TimeGAN	0.4870	240.56%	0.0918	59.65%	0.0724	99.37%	0.1181	798.32%	0.1338	22.98%	244.18%
DoppelGANger	0.4087	185.80%	*0.0835*	45.22%	*0.0716*	97.25%	0.0916	593.94%	*0.1278*	17.46%	187.93%
MSH-GAN	**0.1430**	0	**0.0575**	0	**0.0363**	0	**0.0132**	0	**0.1088**	0	0

Bold denotes the best(lowest) results and italics denotes the second-best results.

Traffic Prediction. To evaluate the usability of MSH-GAN, we conduct a traffic prediction experiment on the real and generated user traffic data, where we use **Mean Square Error (MSE)** between the predicted traffic and real traffic for evaluation.

4.1.4 Parameter Setting. We compute the user traffic with intervals of half an hour, hence $d = 6$ and $T = 288$. Then, we set the number of patterns to 6 and have 6 pattern generators. Meanwhile, we set the number of clusters to 6 according to the unsupervised clustering results of the real data, and we have 6 switch mode generators. More details can be found in Algorithm 1.

4.2 Individual-Level Evaluation

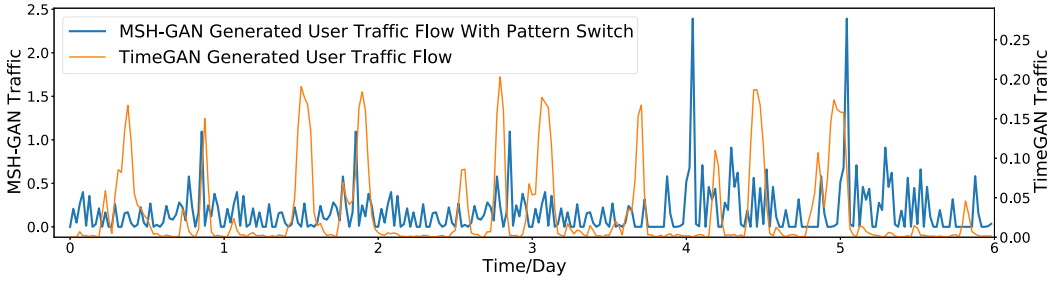
On the individual level, we mainly focus on the distribution of mobile users' traffic flows and switch modes. Notably, as several individual users' traffic flows do not show daily periodicity, we consider the daily periodicity on the aggregate level instead of the individual level.

Table 2 presents the JSD between the real and generated distributions for the traffic volume. MSH-GAN outperforms the baseline models by over 13.64%, demonstrating that MSH-GAN is superior in simulating the volume distribution of user traffic flows. Regarding short-term temporal dynamics, we compare the distributions of the first-order difference between the real and generated traffic flows. MSH-GAN outperforms the baseline models by at least 45.22%, indicating that MSH-GAN also excels in simulating the short-term temporal dynamics in user traffic flows. While DopplGANger is a competitive method among the baselines, our proposed model surpasses DoppelGANger because DoppelGANger adopts a simple RNN-based structure that cannot effectively capture the various usage patterns exhibited by mobile users.

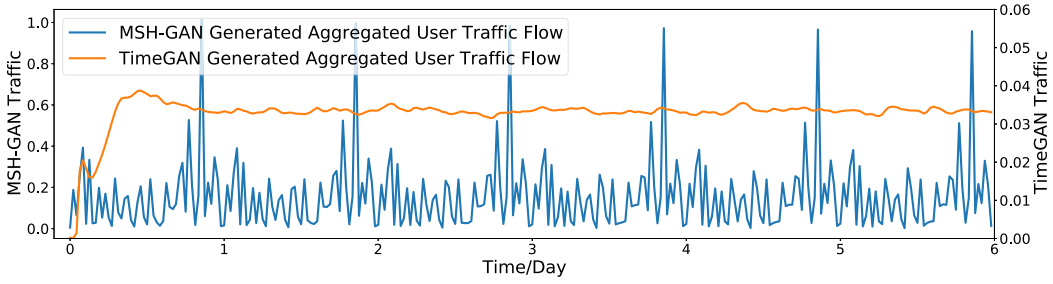
To present the performance visually, Figure 6(a) provides examples of the individual user traffic flows generated by MSH-GAN and TimeGAN in blue and yellow curves, respectively. The blue curve in Figure 6(a) shows pattern switch behavior, where the traffic flows in the previous four days and the latter two days have different patterns, indicating that MSH-GAN can simulate the switch modes in user traffic flows.

4.3 Aggregate-Level Evaluation

On the aggregate level, we focus on the daily periodicity of the aggregated traffic and consider the aggregate pattern switch probabilities. Table 2 presents the AE of the proportion of the daily frequency component and the DPE, where MSH-GAN outperforms the baseline models by over 97.25% and 7.21%, respectively. Moreover, Figure 6(b) provides the aggregated generated traffic flows generated by MSH-GAN and TimeGAN in blue and yellow curves, respectively, where daily periodicity can be observed in the MSH-GAN generated traffic. The results demonstrate that MSH-GAN performs better in simulating the daily periodicity of the aggregated user traffic flows. We

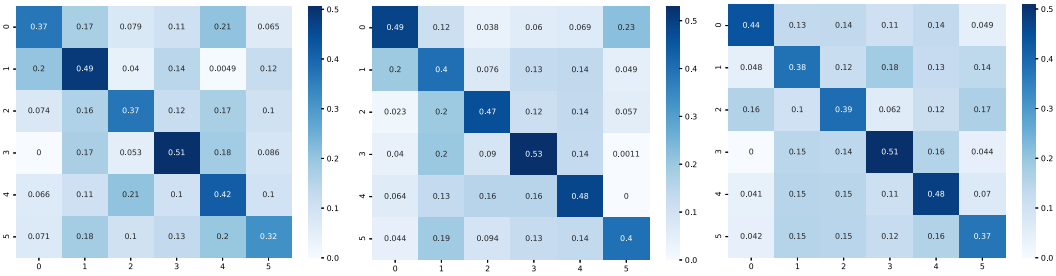


(a) Generated Individual User Traffic Flow.

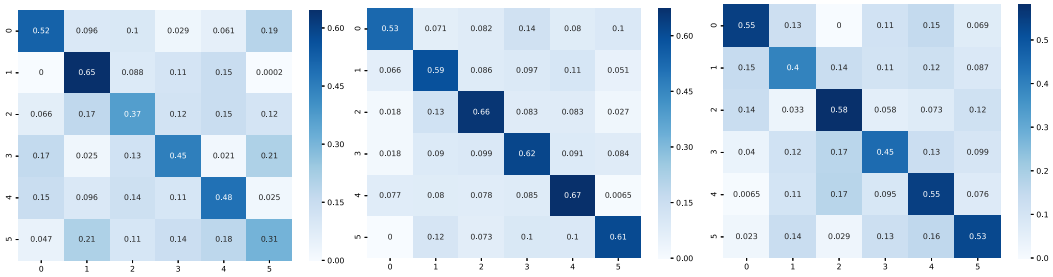


(b) Generated Aggregated User Traffic Flow.

Fig. 6. Examples of the user traffic flows generated by MSH-GAN and TimeGAN.



(a) Weights of Switch Learner 1. (b) Weights of Switch Learner 2. (c) Weights of Switch Learner 3.



(d) Weights of Switch Learner 4. (e) Weights of Switch Learner 5. (f) Weights of Switch Learner 6.

Fig. 7. Weights of the SL in each switch mode generator.

Table 3. Evaluation Results of User Traffic Generated by Variants of MSH-GAN, where Lower Results are Better

Metrics	Traffic Volume		First-order Difference		Daily Frequency Component		Daily Periodic Error		Traffic Prediction		$\overline{\Delta}$
	JSD	Δ	JSD	Δ	AE	Δ	AE	Δ	MSE	Δ	
with LSTM	0.2414	68.81%	0.0728	26.61%	0.0476	31.13%	0.0343	159.85%	0.1148	5.51%	58.38%
w/o switch model	0.3713	159.65%	0.0606	5.39%	0.0416	14.60%	0.0362	174.24%	0.2052	88.60%	88.50%
w/o $loss_D^{clustering}$ and $loss_G^{aggregation}$	0.4216	194.83%	0.1017	76.87%	0.0544	49.86%	0.0313	137.12%	0.1466	34.74%	98.68%
MSH-GAN	**0.1430**	0	**0.0575**	0	**0.0363**	0	**0.0132**	0	**0.1088**	0	0

present the weights of the SL in each switch mode generator in Figure 7, where each value $W(i, j)$ has a positive correlation with the probability of switching from pattern P^j to pattern P^i. The weights show that most users tend to keep the current traffic pattern, while some users are more likely to switch their traffic mode, which is qualitatively in line with the observation.

4.4 Application-Level Evaluation

In recent years, generated data has been used to enhance the training dataset in downstream applications. We conduct a traffic prediction experiment on the real and generated user traffic data, where we predict each user's traffic flow on the last day via learning behaviors from the traffic flow in the previous five days. Specifically, we train TCN models on each traffic dataset generated by MSH-GAN and the baseline models. Table 2 presents the MSE between the traffic predicted by TCNs trained on different datasets and the real traffic, where MSH-GAN outperforms the baselines by at least 17.46%. The results demonstrate that MSH-GAN can provide training data enhancement for downstream applications. Overall, MSH-GAN outperforms the baseline models by more than 118.7%, demonstrating the individual-level and aggregate-level fidelity and usability of MSH-GAN.

4.5 Ablation Study

To demonstrate the effectiveness of the designs we adopted, we compare the performance of MSH-GAN with the following three variants:

with LSTM. This variant uses unidirectional LSTM to capture temporal dependencies in the user network traffic instead of using bidirectional LSTM.

w/o Switch Model. In this variant, the switch mode is removed. This is equivalent to using the pattern generator output as the final output.

w/o $loss_D^{clustering}$ *and* $loss_G^{aggregation}$. This variant removes the discriminator's clustering loss and the generator's aggregation loss. It only retains the adversarial loss to capture individual-level behaviors.

The experimental results are presented in Table 3. It can be observed that the LSTM network performs significantly worse in all metrics, particularly in capturing the daily periodicity of network traffic. This suggests that bidirectional LSTM is more effective in understanding temporal dependencies. The switch mode explicit mixture modeling and utilizing a switch model to choose behavior patterns adaptively are necessary. Furthermore, without clustering loss and aggregation loss, the generation model fails to effectively capture aggregate-level characteristics, resulting in poor performance in modeling the daily frequency component and the daily periodicity.

We also assess the impact of cluster number settings. Table 4 demonstrates that six is the optimal number of clusters for the Shanghai dataset we collected. However, we acknowledge that optimal hyperparameter settings, such as the number of clusters, may vary in different usage scenarios when applying MSH-GAN. Therefore, the ideal number of clusters may not consistently be six across various contexts.

Table 4. Evaluation Results of User Traffic Generated by MSH-GAN Under Different Settings of the Number of Clusters, Where Lower Results are Better

Number of cluster	Traffic Volume	First-order Difference	Daily Frequency Component	Daily Periodic Error
2	0.403	0.0790	0.0527	0.0153
4	0.287	0.0683	0.0417	0.0142
6	**0.143**	**0.0575**	**0.0363**	**0.0132**
8	0.148	0.0606	0.0387	0.0136
10	0.173	0.0648	0.0416	0.0144
12	0.178	0.0656	0.0424	0.0145

5 Related Work

5.1 Network Data Generation

Network data generation is applied to test network equipment, network services, and security products [45]. Traditional methods rely on statistic models to describe network traffic (e.g., Poisson model [46]), traffic characteristics (e.g., packet size and packet arrival distributions [47]), and network protocols (e.g., TCP [48]) in the early stage. However, the representative ability of these models is too limited to capture the complicated temporal dynamics in real-world traffic.

In recent years, machine learning models have been increasingly applied to network data generation, focusing on two main aspects. The first aspect involves generating network packet traces, which include information like IP addresses, port numbers, protocol types, and packet size. For example, Ring et al. [29] developed three different pre-processing approaches based on **Generative Adversarial Network (GANs)** to generate the five-tuple network flows. Dowoo et al. [27] proposed a GAN-based model named PcapGAN to generate network packets using both cyber-attack data and normal data. However, it is important to note that these approaches primarily focus on modeling network data at the IP layer and have different objectives compared to our study. Alternatively, the second aspect involves generating network traffic with goals similar to our research's. Most existing methods focus on generating traffic for a specific region rather than individual users. These methods use **Convolutional Neural Networks (CNNs)**, treating the problem of regional traffic generation as an image generation issue. For instance, CartaGenie [49] uses information about a region's population density and points of interest as conditional features and leverages CNN to generate network traffic for that region. Moreover, SpectraGAN [50] considers the periodicity of traffic patterns and uses a CNN-based conditional GAN to generate spectra of mobile traffic at all locations of the target region. However, these CNN-based models are specifically designed for regional traffic generation and cannot serve as models for user traffic generation. Recently, Lin et al. [30] proposed an RNN-based GAN model named DoppelGANger to generate traffic forms data combining the data attributes and feature series with variable length. They conducted experiments on several traffic forms datasets, including web traffic time series, geographically distributed broadband measurements, and compute cluster usage measurements. DoppelGANger demonstrates state-of-the-art performance in network traffic generation. However, it merely focuses on individual features and neglects aggregate-level temporal dynamics.

5.2 GANs for Time Series Generation

Recently, GANs have been increasingly used to generate time series data. For instance, Mogren et al. [42] propose C-RNN-GAN to generate continuous-valued musical time series. They apply RNN architectures in both generator and discriminator to capture temporal dynamics and take context information into account for decisions. For medical time series, Esteban et al. [41] propose RGAN and **Recurrent Conditional GAN (RCGAN)** based on RNN architectures to generate real-valued

time series subject to some conditional inputs. For privacy-sensitive personal health time series data, Hartmann et al. [51] propose **Electroencephalographic (EEG)**-GAN to generate EEG brain signals, Golany et al. [52] propose simulator-based GANs for Electrocardiogram generation to improve classification tasks. Che et al. [53] present a modified GAN to generate Electronic Health Records data with plausible labels. Generally speaking, a good time series generation model should preserve temporal dynamics and generate sequences from the original patterns between time series variables. For this purpose, Yoon et al. [43] propose TimeGAN, which combines the flexibility of the unsupervised GAN with the control afforded by step-wise supervised loss.

As we generate mobile user traffic in terms of time series, these models provide us with extensive experience. Moreover, C-RNN-GAN [42], RCGAN [41], and TimeGAN [43] apply to our mobile user traffic generation problem, therefore, we compare MSH-GAN with them.

6 Conclusions

In this work, we propose MSH-GAN for multi-scale hierarchical user traffic generation, which generates the traffic patterns with multiple pattern generators and pattern switch modes while discriminating and clustering the mobile user traffic with a multi-class discriminator. Then, we design a combined loss function urging MSH-GAN to learn the multi-scale temporal dynamics on individual and aggregate levels. Extensive results show that MSH-GAN outperforms other state-of-art baselines by more than 118.7% in terms of key metrics on fidelity and usability while successfully simulating individual-level and aggregate-level traffic behaviors. For future work, one promising direction is to address the challenge of modeling mixture distributions in user behaviors. Richardson et al. [54] have highlighted that traditional GANs struggle to capture the entire distribution. Therefore, we propose utilizing multiple pattern generators with a switch mode to model the mixture distribution explicitly. Dilokthanakul et al. [55] has successfully combined Gaussian mixture distributions and variational autoencoders to generate data. Hence, an interesting avenue would be exploring the combination of Gaussian mixture distributions and GAN models to handle the mixture distribution in user behaviors.

References

[1] Statista. 2023. Number of Smartphone Mobile Network Subscriptions Worldwide From 2016 To 2022. Retrieved from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] Diala Naboulsi, Marco Fiore, Stephane Ribot, and Razvan Stanica. Large-scale mobile traffic analysis: A survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2015), 124–161.

[3] Mauro Conti, Qian Qian Li, Alberto Maragno, and Riccardo Spolaor. 2018. The dark side (-channel) of mobile devices: A survey on network traffic analysis. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 2658–2713.

[4] Tapio Soikkeli and Antti Riikonen. 2015. Session level network usage patterns of mobile handsets. In *Proceedings of the 13th International Conference on Telecommunications (ConTEL '15)*. IEEE, 1–8.

[5] Xuetao Wei, Nicholas C. Valler, Harsha V. Madhyastha, Iulian Neamtiu, and Michalis Faloutsos. 2017. Characterizing the behavior of handheld devices and its implications. *Computer Networks* 114 (2017), 1–12.

[6] Guangyi Liu, Juan Deng, and Qingbi Zheng. 2023. 6G autonomous mobile network enabled by digital twin network. *ZTE Technology Journal* 29, 3 (2023), 2–7.

[7] Xiangyang Duan, Honghui Kang, Xingzai Lyu, and Rui Hua. 2023. Digital twin technology for wireless access network oriented to 6G. *ZTE Technology Journal* 29, 3 (2023), 32–37.

[8] Jiahui Gong, Qiaohong Yu, Tong Li, Haoqiang Liu, Jun Zhang, Hangyu Fan, Depeng Jin, and Yong Li. 2023. Scalable digital twin system for mobile networks with generative ai. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*. 610–611.

[9] Shanshan Wang, Zhenxiang Chen, Lei Zhang, Qiben Yan, Bo Yang, Lizhi Peng, and Zhongtian Jia. 2016. TrafficAV: An effective and explainable detection of mobile malware behavior using network traffic. In *Proceedings of the IEEE/ACM 24th International Symposium on Quality of Service (IWQoS '16)*. IEEE, 1–6.

[10] Anshul Arora and Sateesh K. Peddoju. 2017. Minimizing network traffic features for android mobile malware detection. In *Proceedings of the 18th International Conference on Distributed Computing and Networking*. 1–10.

[11] Jonathan Crussell, Ryan Stevens, and Hao Chen. 2014. MAdFraud: Investigating ad fraud in android applications. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*. 123–134.
[12] Marco V. Barbera, Alessandro Epasto, Alessandro Mei, Vasile C. Perta, and Julinda Stefa. 2013. Signals from the crowd: Uncovering social relationships through smartphone probes. In *Proceedings of the 2013 Conference on Internet Measurement Conference*. 265–276.
[13] Raphael Spreitzer, Simone Griesmayr, Thomas Korak, and Stefan Mangard. 2016. Exploiting data-usage statistics for website fingerprinting attacks on android. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. 49–60.
[14] Guiyang Luo, Quan Yuan, Jinglin Li, Shangguang Wang, and Fangchun Yang. 2022. Artificial intelligence powered mobile networks: From cognition to decision. *IEEE Network* 36, 3 (2022), 136–144.
[15] Shuangfeng Han, I. Chih-Lin, Gang Li, Sen Wang, and Qi Sun. 2017. Big data enabled mobile network design for 5G and beyond. *IEEE Communications Magazine* 55, 9 (2017), 150–157.
[16] Wenzhen Huang, Tong Li, Yuting Cao, Zhe Lyu, Yanping Liang, Li Yu, Depeng Jin, Junge Zhang, and Yong Li. 2023. Safe-NORA: Safe reinforcement learning-based mobile network resource allocation for diverse user demands. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 885–894.
[17] Tong Li and Yong Li. 2023. Artificial intelligence for reducing the carbon emissions of 5G networks in China. *Nature Sustainability* 6, 12 (2023), 1522–1523.
[18] Zhu Xiao, Jianzhi Yu, Tong Li, Zhiyang Xiang, Dong Wang, and Wenjie Chen. 2016. Resource allocation via hierarchical clustering in dense small cell networks: A correlated equilibrium approach. In *Proceedings of the IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC '16)*. IEEE, 1–5.
[19] Tong Li, Yali Fan, Yong Li, Sasu Tarkoma, and Pan Hui. 2023. Understanding the long-term evolution of mobile app usage. *IEEE Transactions on Mobile Computing* 22, 2 (2023), 1213–1230.
[20] Tong Li, Yong Li, Mingyang Zhang, Sasu Tarkoma, and Pan Hui. 2023. You are how you use apps: User profiling based on spatiotemporal app usage behavior. *ACM Transactions on Intelligent Systems and Technology* 14, 4 (2023), 1–21.
[21] Gergely Acs, Gergely Biczók, and Claude Castelluccia. 2018. Privacy-preserving release of spatio-temporal density. In *Handbook of Mobile Data Privacy*. A. Gkoulalas-Divanis and C. Bettini (Eds.), Springer, 307–335.
[22] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 1054–1067.
[23] Tong Li, Tong Xia, Huandong Wang, Zhen Tu, Sasu Tarkoma, Zhu Han, and Pan Hui. 2022. Smartphone app usage analysis: Datasets, methods, and applications. *IEEE Communications Surveys & Tutorials* 24, 2 (2022), 937–966.
[24] Ke Huang, Chunhui Zhang, Xiaoxiao Ma, and Guanling Chen. 2012. Predicting mobile application usage using contextual information. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. 1059–1065.
[25] Jian-Hui Duan, Wenzhong Li, Xiao Zhang, and Sanglu Lu. 2022. Forecasting fine-grained city-scale cellular traffic with sparse crowdsourced measurements. *Computer Networks* 214 (2022), 109156.
[26] Adriel Cheng. 2019. PAC-GAN: Packet generation of network traffic using generative adversarial networks. In *Proceedings of the IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON '19)*. IEEE, 0728–0734.
[27] Baik Dowoo, Yujin Jung, and Changhee Choi. 2019. PcapGAN: Packet capture file generator by style-based generative adversarial networks. In *Proceedings of the 18th IEEE International Conference On Machine Learning And Applications (ICMLA '19)*. IEEE, 1149–1154.
[28] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. 2022. Practical GAN-based synthetic IP header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 458–472.
[29] Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. 2019. Flow-based network traffic generation using generative adversarial networks. *Computers & Security* 82 (2019), 156–172.
[30] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using GANs for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*. 464–483.
[31] Shuodi Hui, Huandong Wang, Tong Li, Xinghao Yang, Xing Wang, Junlan Feng, Lin Zhu, Chao Deng, Pan Hui, Depeng Jin, and Yong Li. 2023. Large-scale urban cellular traffic generation via knowledge-enhanced GANs with multi-periodic patterns. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4195–4206.
[32] Shiyuan Zhang, Tong Li, Shuodi Hui, Guangyu Li, Yanping Liang, Li Yu, Depeng Jin, and Yong Li. 2023. Deep transfer learning for city-scale cellular traffic generation through urban knowledge graph. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4842–4851.
[33] Tong Li, Yong Li, Tong Xia, and Pan Hui. 2021. Finding spatiotemporal patterns of mobile application usage. *IEEE Transactions on Network Science and Engineering*.

[34] Alex Graves, Navdeep Jaitly, and Abdel-Rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 273–278.
[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30, 1–11.
[36] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[37] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. arXiv:1611.01144. Retrieved from https://arxiv.org/abs/1611.01144
[38] Chris J. Maddison, Andriy Mnih, and Yee W. Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. arXiv:1611.00712. Retrieved from https://arxiv.org/abs/1611.00712
[39] Emil J. Gumbel. 1954. *Statistical Theory of Extreme Values and Some Practical Applications: A Series of Lectures*, Vol. 33. US Government Printing Office.
[40] Shaojie Bai, Jeremy Z. Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271. Retrieved from https://arxiv.org/abs/1803.01271
[41] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. 2017. Real-valued (medical) time series generation with recurrent conditional GANs. arXiv:1706.02633. Retrieved from https://doi.org/10.48550/arXiv.1706.02633
[42] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. arXiv:1611.09904. Retrieved from https://arxiv.org/abs/1706.02633
[43] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. 2019. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems* 32, 1–11.
[44] Maria Luisa Menéndez, Julio Angel Pardo, Leandro Pardo, and María del C. Pardo. 1997. The Jensen-Shannon divergence. *Journal of the Franklin Institute* 334, 2 (1997), 307–318.
[45] Junhui Zhang, Jiqiang Tang, Xu Zhang, Wen Ouyang, and Dongbin Wang. 2015. A survey of network traffic generation. In *Proceedings of the 3rd International Conference on Cyberspace Technology (CCT '15)*. IET, 1–6.
[46] Thomas Bonald. 2006. The erlang model with non-poisson call arrivals. *ACM SIGMETRICS Performance Evaluation Review* 34, 1 (2006), 276–286.
[47] Kashi V. Vishwanath and Amin Vahdat. 2009. Swing: Realistic and responsive network traffic generation. *IEEE/ACM Transactions on Networking* 17, 3 (2009), 712–725.
[48] Michele C. Weigle, Prashanth Adurthi, Félix Hernández-Campos, Kevin Jeffay, and F. Donelson Smith. 2006. Tmix: A tool for generating realistic TCP application workloads in ns-2. *ACM SIGCOMM Computer Communication Review* 36, 3 (2006), 65–76.
[49] Kai Xu, Rajkarn Singh, Hakan Bilen, Marco Fiore, Mahesh K. Marina, and Yue Wang. 2022. Cartagenie: Context-driven synthesis of city-scale mobile network traffic snapshots. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom '22)*. IEEE, 119–129.
[50] Kai Xu, Rajkarn Singh, Marco Fiore, Mahesh K. Marina, Hakan Bilen, Muhammad Usama, Howard Benn, and Cezary Ziemlicki. 2021. SpectraGAN: Spectrum based generation of city scale spatiotemporal mobile network traffic data. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*. 243–258.
[51] Kay G. Hartmann, Robin T. Schirrmeister, and Tonio Ball. 2018. EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals. arXiv:1806.01875. https://arxiv.org/abs/1806.01875
[52] Tomer Golany, Kira Radinsky, and Daniel Freedman. 2020. SimGANs: Simulator-based generative adversarial networks for ECG synthesis to improve deep ECG classification. In *Proceedings of the International Conference on Machine Learning*. PMLR, 3597–3606.
[53] Zhengping Che, Yu Cheng, Shuangfei Zhai, Zhaonan Sun, and Yan Liu. 2017. Boosting deep learning risk prediction with generative adversarial networks for electronic health records. In *Proceedings of the IEEE International Conference on Data Mining (ICDM '17)*. IEEE, 787–792.
[54] Eitan Richardson and Yair Weiss. 2018. On GANs and GMMs. *Advances in Neural Information Processing Systems* 31, 1–11.
[55] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. 2016. Deep unsupervised clustering with gaussian mixture variational autoencoders. arXiv:1611.02648. https://arxiv.org/abs/1611.02648

Received 5 January 2023; revised 15 February 2024; accepted 8 May 2024