ChartNavigator: An Interactive Pattern Identification and Annotation Framework for Charts

Tianye Zhang*, Haozhe Feng*, Wei Chen, Zexian Chen, Wenting Zheng, Xiaonan Luo, Wenqi Huang, Anthony Tung

Abstract—Patterns in charts refer to interesting visual features or forms. Identifying patterns not only helps analysts understand the 'shape' of the data but also supports better and faster decision-making. Existing solutions for identifying patterns in charts require a large number of labeled data instances, making it intractable without user supervision. In this paper, we propose ChartNavigator, an interactive pattern identification and annotation framework for unlabeled visualization charts. ChartNavigator leverages a novel chart-sensitive deep factor model to map patterns into a low-dimensional factor representation space, and facilitates rich analysis with the derived representations. We design and implement a visual interface to support efficient identification and annotation of potential patterns in charts. Evaluations with multiple datasets show that our approach outperforms the baseline models in identifying and annotating patterns.

Index Terms—pattern identification, chart, variational autoencoder, user interaction, visual analysis.

1 Introduction

Charts denote computer-generated visual forms used to represent data by a combination of visual encodings like shape, position, color, size, and layout [16]. A well-designed chart facilitates efficient information communication by leveraging human perception and understanding, especially when expected or unexpected patterns are to be extracted from the raw data [16]. It has proven that reading charts is much more efficient than using numbers when the data is fuzzy, irregular or complicated [20], [21]. In many cases, charts are preferable forms of the underlying data [17]. Established visualization authoring software or toolkits, such as Tableau,[1] Microsoft PowerBI,[2] and D3.js [19], have greatly eased the creation of visualization charts. This results in pervasive chart resources in news articles, business reports, research papers and etc.. The need to identify and annotate patterns from charts has been dramatically increasing in many fields like business intelligence, e-learning, scientific reporting, and media communication.

Although people can readily identify patterns in charts, it could still be a laborious process. To address this issue, pioneered works have demonstrated the feasibility of applying supervised methods to analyze chart images [4], [5], [15], [17]. For example, Ma et al. propose a deep learning based model for learning the visual cognition of scatterplots from labelled data [15]. Poco et al. adopt a convolution neural network to automatically recover visual encodings from charts [17]. Despite the success of these works, which relies on a number of labeled data instances to learn chart features, it is still difficult to perform pattern identification for charts with little or no supervision.

In this paper, we focus on identifying patterns from visual charts instead of the tabular data. Note that, a chart is represented as an image, whose patterns can be easily characterized by means of representation learning approaches [41]. In particular, we propose a novel deep factor model, which works unsupervisedly and learns the features hidden in the data by disentangling it into explanatory factors from which the data is built. The strong interpretability of deep factor models provides an insightful understanding into the potential patterns. In particular, we address two challenges that have not been revealed by previous works: (1) Given a specific dataset, people often have no idea about the target, making it difficult to determine a specific definition of a pattern. Therefore, pattern identification from charts demands the incorporation of human knowledge and prefers interactive methods instead of automatic ones. (2) Pattern identification raises haystack-in-a-haystack queries [22]. A specific pattern may appear

- *Tianye Zhang, Haozhe Feng, Wei Chen, Zexian Chen, Wenting Zheng are with the State Key Lab of CAD & CG, Zhejiang University, China, 310058.*
  Wei Chen is also with the Zhejiang University - China Southern Power Grid Joint Research Centre on AI.
  E-mail: {zhangtianye1026, fenghz, zexianchen}@zju.edu.cn, {chenwei, wtzheng}@cad.zju.edu.cn
  Wei Chen is the corresponding author.
- *Xiaonan Luo is with The Guilin University of Electronic Technology, China, 541214.*
  E-mail:luoxn@guet.edu.cn
- *Wenqi Huang is with the Digital Grid Research Institute, China Southern Power Grid, China, 510670.*
  E-mail:huangwqcsg@163.com
- *Anthony Tung is with the Department of Computer Science, National University of Singapore, Singapore, 117417.*
  Email: atung@comp.nus.edu.sg

* *Equal Contribution.*

Manuscript received April 19, 2005; revised August 26, 2015.

1. https://www.tableau.com/
2. https://powerbi.microsoft.com/en-us/

in a group of data instances (a haystack) instead of in an individual case (a needle). Thus, it is needed to discover interesting haystacks from the entire chart dataset (a bigger haystack).

We propose a novel interactive pattern identification and annotation framework for unlabeled visualization charts, denoted as ChartNavigator. As shown in Figure 1, ChartNavigator consists of three major components: (1) A chart-sensitive deep factor model that maps the high-dimensional chart data into high quality low-dimensional representations from which the chart can be reconstructed (Figure 1 (a)). (2) A suite of visualization tools which supports interactive exploration and analysis of patterns in the factor space (Figure 1 (b)). (3) An unsupervised or a semi-supervised algorithm that helps annotate the entire dataset with the identified patterns (Figure 1 (c)). ChartNavigator supports statistical visualization charts, such as scatterplots, line charts and bar charts etc., and more complicated charts, such as heatmaps and graphs etc. Without loss of generality, we demonstrate its effectiveness on three kinds of charts (scatterplots, pixel maps and heatmaps). It can easily also adapt to other charts. To the best of our knowledge, this is the first attempt on interactive pattern identification and annotation for charts and also on learning refined representations of patterns.

This paper presents the following contributions:

- We propose a novel chart-sensitive deep factor model, Optimal Transport Variational AutoEncoder (OT-VAE), to capture chart-sensitive factors and generate high quality representations.
- We design a visual interface that provides two key components: (1) a series of OT based factor operations on the factor space that supports chart-sensitive extraction and reorganization of features in charts. (2) a Gradient-based Factor Activation Map (Grad-FAM) that provides visual explanations for arbitrary factor combinations.
- We propose unsupervised and semi-supervised multi-label solutions for annotating the entire dataset with the identified patterns.

We apply ChartNavigator to real-world scenarios such as traffic analysis and power system control. The experimental results demonstrate the effectiveness and efficiency of our framework.

2 PRELIMINARIES

In this section, we introduce the basic concept on patterns and factors.

2.1 Deep Factor Model

Factor model [41] assumes that data can be generated from different explanatory factors, e.g., object position and hue in an image dataset. Given a dataset $\mathbb{D}$, a factor model learns the mapping from the high-dimensional data space to a low-dimensional representation space so that the data can be reconstructed from the low-dimensional representation.

A deep learning based factor model, variational autoencoder (VAE) [23], has been widely applied because of its excellent interpretability. As shown in Fig. 1 (a), it constructs a probabilistic model that consists of an inference and a generation process. The inference process learns the low-dimensional factor representation $\mathbf{z}$ of the high-dimensional data instance $\mathbf{X}$ by estimating the posterior distribution of $\mathbf{z}$. And the generation process reconstructs instances $\mathbf{X}$ from the factors $\mathbf{z}$ by estimating the posterior distribution of $\mathbf{X}$ given by $\mathbf{z}$. The evidence lower bound (ELBO) is used as the objective function, which consists of two parts: (1) the expectation of generating $\mathbf{X}$ from the factors $\mathbf{z}$ and (2) the difference between the distribution of derived factors and true factors. The ELBO is computed as

$$L_{ELBO} = D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{X})\|p(\mathbf{z})) - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{X})}\log p_{\boldsymbol{\theta}}(\mathbf{X}|\mathbf{z}), \quad (1)$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{X})$ and $p_{\boldsymbol{\theta}}(\mathbf{X}|\mathbf{z})$ denotes the inerence and generation process respectively. Table 2.1 summarizes related notations.

TABLE 1
Symbols and Definitions.

Symbol	Interpretation
$\mathbb{D}$	A dataset
$\mathbf{X} \in \mathbb{D}$	A data instance $\in \mathbb{D}$
$\mathbf{z}$	The explanatory factors
$\mathbb{E}_{\mathbf{X}\sim p}[f(\mathbf{X})]$	The expectation of $f(\mathbf{X})$
$D_{\mathrm{KL}}(q\|p)$	Kullback-Leibler divergence between q and p
$\mathbf{I}$	Identity matrix
$\mathcal{N}(\mathbf{X};\boldsymbol{\mu},\boldsymbol{\sigma}^2)$	Diagonal Gaussian distribution
p_{emp}	The empirical distribution $\mathbb{D}$
p_{mixup}	The mixup distribution [35] $\mathbb{D}$
$\beta(a,b)$	The beta distribution
$1_{condition}$	The conditional function
$\boldsymbol{\phi}$	The inference process parameters
$\boldsymbol{\theta}$	The generation process parameters
ELBO	The evidence lower bound

2.2 Factor-to-pattern

A pattern can be explained as a particular combination of the explanatory factors $\mathbf{z}$, where each $z_i \in \mathbf{z}$ are carefully sampled from the posterior distribution of $\mathbf{z}$. We derive the formal definition of a pattern as:

Definition 2.1. A pattern is represented by a triplet $(\mathbf{X}', \boldsymbol{\mu}', \boldsymbol{\sigma}')$, which satisfies

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{X}') = \mathcal{N}(\mathbf{z};\boldsymbol{\mu}', \boldsymbol{\sigma}'^2);\ \mathbf{X}' = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\boldsymbol{\mu}',\boldsymbol{\sigma}'^2)} f(\mathbf{z},\boldsymbol{\theta}), \quad (2)$$

where $\boldsymbol{\mu}'$ and $\boldsymbol{\sigma}'$ are the parameters of the Gaussian distribution $\mathbf{z}$ subject to, and $\mathbf{X}'$ is an instance generated from $\mathbf{z}$.

3 CHART-SENSITIVE DEEP FACTOR MODEL

We propose a chart-sensitive Optimal Transport VAE (OT-VAE) model to learn the factor distribution of the input chart images. Reasons for that are twofolds. First, it leverages an optimal transport (OT) scheme so that meaningful factors in charts can be captured. Second, we propose a set of factor operations based on the OT-VAE model to support pattern extraction and reorganization of charts (Section 4.3).

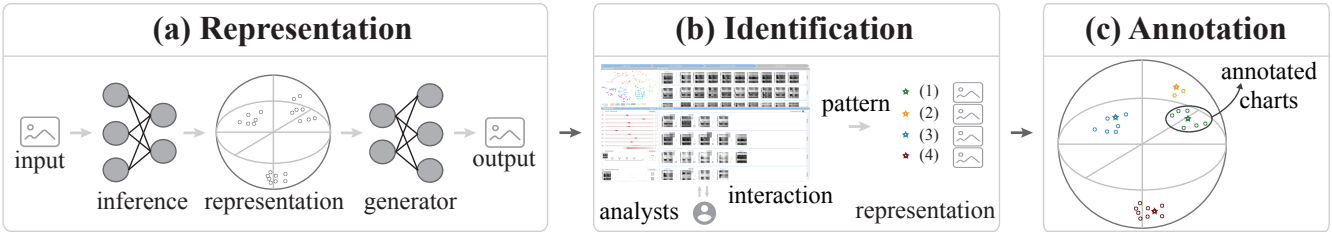

Fig. 1. The pipeline of ChartNavigator. (a) An enhanced deep factor model used to learn the factor representation of charts. (b) A visual interface used to visualize the learned representations, identify patterns and generate representations for patterns. (c) An annotation module.

3.1 Optimal Transport VAE

For conventional VAEs, there is a 'hidden' threshold of the ELBO value and continuing to optimize the model after this threshold barely improves the inference results. The reason is that ELBO only provides an upper bound of the inference margin between the distribution of inferred factors and the true factors (Please refer to Appendix A for proof). When the ELBO reaches the threshold [28], [42], the inference margin can still be large.

As such, we apply an optimal transport scheme to provide a better inference margin and more interpretable factors by regularizing the distributions of the factors. The estimation is achieved in two steps. First, we extend the empirical distribution p_{emp} to the mixup distribution p_{mixup} as [35], from which we sample a number of virtual instances $\mathbf{X}_\lambda$:

$$\mathbf{X}_\lambda = (1-\lambda) * \mathbf{X}_0 + \lambda * \mathbf{X}_1, \lambda \sim \beta(\alpha, \alpha) \tag{3}$$

where $\mathbf{X}_0, \mathbf{X}_1$ are data instances randomly drawn from the dataset $\mathbb{D}$. Second, considering that this interpolation process in the data space can be mapped to the factor space, we apply the norm-2 based optimal transport scheme [36] to create the posterior distribution $\tilde{p}(\mathbf{z}|\mathbf{X}_\lambda)$ as an approximation of the true posterior $p(\mathbf{z}|\mathbf{X})$. According to the optimal transport scheme, for factors $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, the shortest-path between $\mathbf{z}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2)$ and $\mathbf{z}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\sigma}_1^2)$ is:

$$\boldsymbol{\mu}_\lambda = (1-\lambda)\boldsymbol{\mu}_0 + \lambda\boldsymbol{\mu}_1; \boldsymbol{\sigma}_\lambda = (1-\lambda)\boldsymbol{\sigma}_0 + \lambda\boldsymbol{\sigma}_1 \tag{4}$$

where $\lambda \in [0, 1]$. For proof details please refer to Appendix B.

As shown in Algorithm 1, the estimated inference margin is denoted as $L_{M_\mathbf{z}}$. By adding $L_{M_\mathbf{z}}$ to the ELBO loss, we revise the conventional loss function as

$$L = L_{\text{ELBO}} + w_{M_\mathbf{z}} * L_{M_\mathbf{z}},$$

where $w_{M_\mathbf{z}}$ is the optimal transport estimation weight. Implementation details of OT-VAE is explained in Appendix D. Experiments in Section 5.2 demonstrates the effectiveness of OT-VAE in learning higher quality representations for charts than conventional VAEs.

Algorithm 1 The optimal transport estimation process of OT-VAE.

Input:
A batch of data $\mathbf{X}$ sampled from $p_{emp}(\mathbf{X})$;
Inferred factors $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{X}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}(\mathbf{X}; \boldsymbol{\phi}), \boldsymbol{\sigma}^2(\mathbf{X}; \boldsymbol{\phi}))$;
Hyperparameter α for mixup distribution $p_{mixup}(\mathbf{X})$
Output:
$\mathbf{X}_\lambda$ sampled from $p_{mixup}(\mathbf{X})$;
Estimation $L_{M_\mathbf{z}}$ of the margin $D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{X}_\lambda) \| p(\mathbf{z}|\mathbf{X}_\lambda))$

$\hat{\mathbf{X}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}^2 = \text{RandomPermutation}(\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
$\mathbf{X}_\lambda = (1-\lambda) * \mathbf{X} + \lambda * \hat{\mathbf{X}}, \lambda \sim \beta(\alpha, \alpha)$
$(\boldsymbol{\mu}_\lambda, \boldsymbol{\sigma}_\lambda^2) = \text{OptimalTransport}((\boldsymbol{\mu}, \boldsymbol{\sigma}^2), (\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}^2), \lambda)$
$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{X}_\lambda) = \text{VAE}(\mathbf{X}_\lambda)$
$\tilde{p}(\mathbf{z}|\mathbf{X}_\lambda) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\lambda, \boldsymbol{\sigma}_\lambda^2)$
$L_{M_\mathbf{z}} = D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{X}_\lambda) \| \tilde{p}(\mathbf{z}|\mathbf{X}_\lambda))$
return $\mathbf{X}_\lambda, L_{M_\mathbf{z}}$

4 ChartNavigator

ChartNavigator functions in three steps (Figure 1). First, the OT-VAE model derives low-dimensional factor representations from the input chart images. A factor representation is denoted as a triplet $(\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, where $\mathbf{X}$ denotes the input chart, and $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ are the Gaussian distribution parameters of factors $\mathbf{z}$. Second, the input charts and their factor representations (triplets) are visualized to support interactive identifications of patterns. Users can modify the factors of interested charts based on the input triplets and generate sketches of the desired patterns (represented by $(\mathbf{X}', \boldsymbol{\mu}', \boldsymbol{\sigma}')$, where $\mathbf{X}'$ is the sketch and $(\boldsymbol{\mu}', \boldsymbol{\sigma}')$ is its factor representation). Third, we annotate the dataset with the identified patterns by comparing pairwise triplets $(\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ and $(\mathbf{X}', \boldsymbol{\mu}', \boldsymbol{\sigma}')$.

Figure 2 illustrates the interface. It achieves the aforementioned step 2 and 3 through the following operations: loading data, finding candidate, representing pattern and annotating.

4.1 Data Selection

Given a chart dataset (which contains a set of chart images) and the associated factor representations $((\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\sigma}))$, charts are embed into a two-dimensional space by using the t-distributed Stochastic Neighbor Embedding (*t*-SNE) method and clustered by using the *k*-Means method (Figure 2 (A)).

Embedding: Charts are embed based on their factor distributions. For each chart, we sample the mean value of its factors and construct a multi-dimensional vector, based on which it is projected. To avoid visual occlusion, we employ the blue noise sampling [6] to reduce the number of displayed points and preserve the overall data distribution.

Distance Measure: In the *k*-Means clustering, the optimal transport distance (also known as the Wasserstein

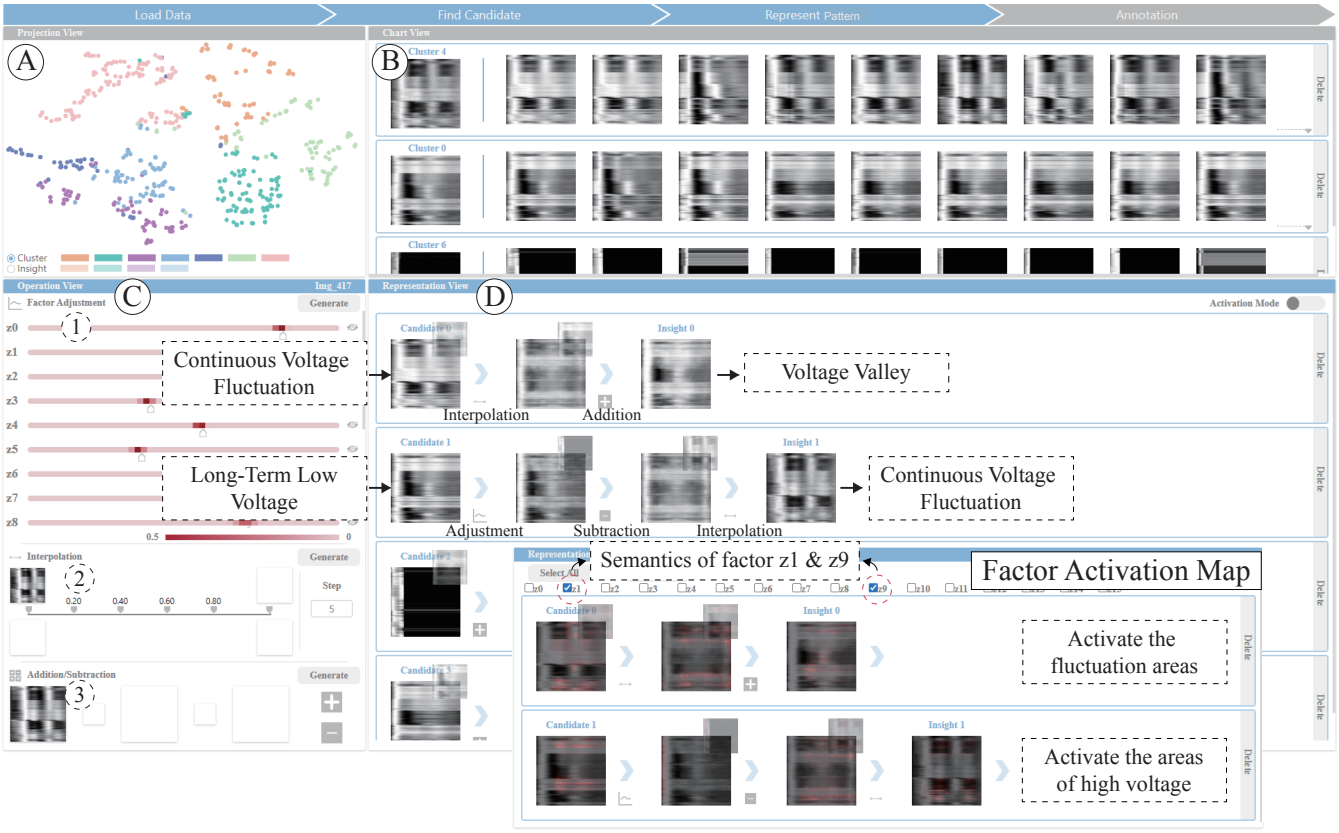

Fig. 2. The ChartNavigator interface. (A) The projection view that projects visualization charts to a two-dimensional space. (B) The chart view shows the original charts in the dataset. (C) The operation view supports arithmetic operations of factor representations. (D) The representation view shows semantic changes brought by the executed arithmetic operations. The shown example is the PG-S dataset.

distance) is used under the restriction of normal distribution, which is consistent with the OT-VAE model and yields reasonable results. Given a pair of charts $\mathbf{X}_0, \mathbf{X}_1 \in \mathbb{D}$, the distance between them is defined as:

$$\begin{aligned} \text{dist}^2_{OT}(\mathbf{X}_0, \mathbf{X}_1) = \|\boldsymbol{\mu}(\mathbf{X}_0; \boldsymbol{\phi}) - \boldsymbol{\mu}(\mathbf{X}_1; \boldsymbol{\phi})\|_2^2 \\ + \|\boldsymbol{\sigma}(\mathbf{X}_0; \boldsymbol{\phi}) - \boldsymbol{\sigma}(\mathbf{X}_1; \boldsymbol{\phi})\|_2^2 \end{aligned} \tag{5}$$

where the normal distribution $\mathcal{N}(\boldsymbol{\mu}(\mathbf{X}_1; \boldsymbol{\phi}), \boldsymbol{\sigma}^2(\mathbf{X}_1; \boldsymbol{\phi}))$ and $\mathcal{N}(\boldsymbol{\mu}(\mathbf{X}_2; \boldsymbol{\phi}), \boldsymbol{\sigma}^2(\mathbf{X}_2; \boldsymbol{\phi}))$ denote the factor distribution of $\mathbf{X}_0$ and $\mathbf{X}_1$ respectively.

4.2 Candidate Identification

Once data is loaded, users are expected to find the chart candidates that contain patterns. To address this problem, we allow users to select clusters from the projection view (Figure 2 (A)) by brushing an interested area or double-clicking a horizontal bar (each of which represent a cluster) at the bottom of the projection view. These selected clusters are then displayed in the chart view (Figure 2 (B)). Charts of the same cluster is bundled together. Due to the limit of the space, we only display 10 charts for each cluster and users can click the inverted-triangle button on the bottom right corner to see more charts in each cluster. We display the class centroid of each cluster on the most left of its bundle. The charts in this cluster are displayed on the right side according to their distances to the centroid. The class centroid and the other charts are separated by a blue vertical bar. The class centroid $\mathbf{X}_{ctd}$ for cluster $\mathbb{T} = \{\mathbf{X}_i\}_{i=1}^m$ is defined as the optimal transport based barycenter [36]:

$$\begin{aligned} &\mathbf{X}_{ctd} = \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_{ctd}, \boldsymbol{\sigma}^2_{ctd})} f(\mathbf{z}; \boldsymbol{\theta}) \\ &\text{s.t. } \boldsymbol{\mu}_{ctd} = \frac{1}{m}\sum_{i=1}^m \boldsymbol{\mu}(\mathbf{X}_i; \boldsymbol{\phi}), \boldsymbol{\sigma}_{ctd} = \frac{1}{m}\sum_{i=1}^m \boldsymbol{\sigma}(\mathbf{X}_i; \boldsymbol{\phi}), \end{aligned} \tag{6}$$

where $\{\mathcal{N}(\mathbf{z}_i; \boldsymbol{\mu}(\mathbf{X}_i; \boldsymbol{\phi}), \boldsymbol{\sigma}^2(\mathbf{X}_i; \boldsymbol{\phi}))\}_{i=1}^m$ are the factor distributions of the instances in $\mathbb{T}$ and $f(\mathbf{z}; \boldsymbol{\theta})$ is the generation function that generates $\mathbf{X}_{ctd}$ from its corresponding factors.

In this way, it is easier for users to find the following two types of candidates: 1) the predominant candidates that are able to represent a considerable part of the data, for example, the centroid of a cluster, 2) the rare candidates that could be the outlier of the data, for example, a chart that is completely different from the class centroid and other nearby charts.

4.3 Pattern Representation

The selected candidates are then displayed in the representation view (Figure 2 (D)). We introduce a factor activation map and three types of factor operations (Figure 2 (C)). The goal is to generate a chart for each individual pattern by disentangling multiple patterns in the same chart or creating a desired pattern based on the existing patterns. Particularly, the specific factor combination that constitutes a pattern (represented by a chart) is considered as the factor representation of this pattern.

4.3.1 Factor Activation Map

We introduce a factor activation map that provides visual explanations for factors. It highlights the areas activated by different factors in each chart.

We generalize the supervised Grad-CAM [34] to our unsupervised OT-VAE, and propose the Gradient-based Factor Activation Map (Grad-FAM). Given an instance $\mathbf{X}$ and its factors $\mathbf{z} = (z_1, \ldots, z_k)$, the Grad-FAM $\mathbf{M}$ is derived within two steps. First, following Grad-CAM, we produce a coarse-grained activation map $\mathbf{M}_c$. Second, we compute the gradient of the instance in the final convolutional layer and combine it with $\mathbf{M}_c$ to produce a fine-grained activation map $\mathbf{M}_f$. Figure 3 shows an example of the coarse-grained and fine-grained activation map. Details of the Grad-FAM is explained in Appendix C.

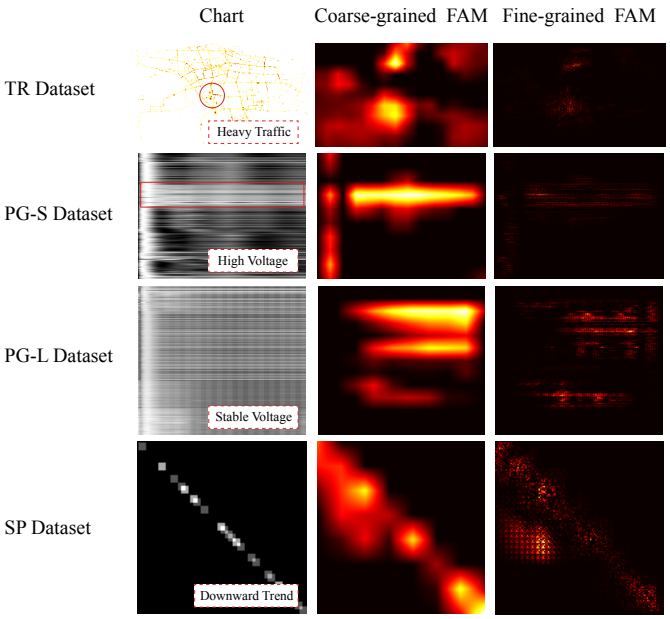

Fig. 3. The Grad-FAM examples on different datasets.

4.3.2 Factor Operations

We also introduce the following factor operations:

Factor Adjustment allows users to adjust the details of a pattern, including its position, shape and shade etc. (Figure 2 (C1)). Given a candidate $\mathbf{X}$ and its factor distribution $p(\mathbf{z}|X) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, the distribution of each $z_i \in \mathbf{z}$ is encoded by a horizontal heatmap axis whose color indicates the probability of z_i taking this value. A redder color indicates a higher probability. Users are allowed to flexibly adjust the sampling value of each factor, during which a chart is generated and changed in real-time based on the given values of factors. Particularly, users can hide the factors of less interest by clicking the hidden button on the right of each axis. The hidden factors will then be noted by a triangular button and can be re-displayed at anytime.

Interpolation supports the transition between pairwise patterns and can efficiently generate an intermediate pattern (Figure 2 (C2)). It performs linear interpolation between pairwise charts (X_i, X_j) in the optimal transport based factor space (Appendix B). Algorithm 2 shows the details of the linear interpolation process. The default interpolation step size is set to 5. Users can also flexibly set the interpolation scheme by adjusting the step size or setting the the interpolation positions in the interface.

Algorithm 2 Linear interpolation with the OT-VAE.

Input:
Instances $\mathbf{X}_0, \mathbf{X}_1 \in \mathbb{D}$;
Inferred factors $\mathcal{N}(\mathbf{z}_i; \boldsymbol{\mu}(\mathbf{X}_i; \boldsymbol{\phi}), \boldsymbol{\sigma}^2(\mathbf{X}_i; \boldsymbol{\phi})), i = 0, 1$;
Interpolation parameter $t \in [0, 1]$;
Generation function $f(\mathbf{z}; \boldsymbol{\theta})$
Output:
Interpolation result $\mathbf{X}_t$

$\boldsymbol{\mu}_t = (1 - t) * \boldsymbol{\mu}(\mathbf{X}_0; \boldsymbol{\phi}) + t * \boldsymbol{\mu}(\mathbf{X}_1; \boldsymbol{\phi})$
$\boldsymbol{\sigma}_t = (1 - t) * \boldsymbol{\sigma}(\mathbf{X}_0; \boldsymbol{\phi}) + t * \boldsymbol{\sigma}(\mathbf{X}_1; \boldsymbol{\phi})$
$\mathbf{X}_t = \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)} f(\mathbf{z}; \boldsymbol{\theta})$
return $\mathbf{X}_t$

Addition/Subtraction allows users to combine pairwise patterns or disentangle multiple patterns (Figure 2 (C3)). It performs linear transformations on the factor space of the selected charts $\{\mathbf{X}_i\}$. Algorithm 3 shows the details of this process.

Algorithm 3 Linear transformation with OT-VAE.

Input:
Instances $\mathbf{X}_i \in \mathbb{D}, i = 1, \ldots, m$;
Parameters $w_i \in \mathbb{R}, i = 1, \ldots, m$
Factors $\mathcal{N}(\mathbf{z}_i; \boldsymbol{\mu}(\mathbf{X}_i; \boldsymbol{\phi}), \boldsymbol{\sigma}^2(\mathbf{X}_i; \boldsymbol{\phi})), i = 1, \ldots, m$;
Generation function $f(\mathbf{z}; \boldsymbol{\theta})$ of OT-VAE
Output:
Linear transformation result $\mathbf{X}_T$

$\boldsymbol{\mu}_T = \sum_{i=1}^{m} w_i * \boldsymbol{\mu}(\mathbf{X}_i; \boldsymbol{\phi})$
$\boldsymbol{\sigma}_T = \sqrt{\sum_{i=1}^{m} w_i^2 * \boldsymbol{\sigma}^2(\mathbf{X}_i; \boldsymbol{\phi})}$
$\mathbf{X}_T = \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{\sigma}_T^2)} f(\mathbf{z}; \boldsymbol{\theta})$
return $\mathbf{X}_T$

4.3.3 Pattern Representation Generation

As shown in Figure 2 (D), each row in the representation view contains a candidate selected from the chart view. The candidates are listed at the leftmost side of each row. Users can display the factor activation maps by switching to the FAM mode on the top-right corner of the representation view. In the FAM mode, users are allowed to select a specific factor or a combination of factors, and the FAMs of the selected factors are computed and displayed by superposing an FAM on its corresponding chart. An FAM provides visual explanations of the semantics of the factors, which guides the process of using the factor operations to identify patterns.

Then, users can switch back to the pattern representation mode. Users can change the visualization of a candidate by selecting it to perform factor operations. For each candidate, the execution of any factor operations will generate a new chart by the OT-VAE model. This chart is reconstructed from the user-defined factor distribution. An arrow is shown between the candidate and the newly generated chart. Under the arrow, there is an icon indicating which kind of operation is taken. Operating on the newly generated chart will generate another new chart. Users can iteratively repeat

this process until the expected pattern is exactly represented by a generated chart. Particularly, users need to select a pair of charts to perform the addition/subtraction/interpolation operation and the another selected chart is superposed on the top right side of the candidate chart.

To be mentioned, the factor operations are used to enlarge the dataset by generating new charts (each of which represents a pattern), and the original charts of the dataset (candidates used to generate new charts) remain unchanged during the entire identification and annotation process.

Pattern Identification Strategy. Because there is a set of factors along with their different combinations, we suggest users to first apply the factor activation map (FAM). By analyzing the activation map of each factor and their combinations, users can gain a general understanding of the semantics of each factor. Then, users can select interested factors and adjust the factors to learn the in-depth correspondence between the factors and the basic elements in charts. Users are suggested to start pattern identification after they are familiar to the factors.

4.4 Annotation

Once the factor representation of an pattern $(\mathbf{X}', \boldsymbol{\mu}', \boldsymbol{\sigma}')$ is confirmed, users can rightclick the corresponding chart and record it. Descriptions of the pattern can be added. The recorded patterns are displayed on the bottom of the projection view. Whenever an pattern is recorded, the ChartNavigator interface will annotate the entire dataset with the existing patterns. We propose the following two methods to annotate the dataset.

The Semi-supervised Method For the semi-supervised way, we identify patterns in the ChartNavigator interface but do not necessarily generate new factor representations for them. Instead, users can manually annotate part of the charts in the dataset directly with the identified pattern categories during the exploration process. The rest of the dataset is then automatically annotated in a semi-supervised way in which the OT-VAE model learns from the labeled data and classifies on the unlabeled data.

The Unsupervised Method For the unsupervised way, we annotate the dataset by computing the similarity between each chart $\mathbf{X}_i$ and each pattern $\mathbf{X}_i^{'}$. In our experiment, we use the 2-Wasserstein distance to compute the distance between the factor distributions of $\mathbf{X}_i$ and $\mathbf{X}_i^{'}$.

Once the dataset is annotated, the color of each point in the projection view will indicate the pattern categories instead of the cluster categories.

5 Experiments

We conduct the following experiments to demonstrate the effectiveness of ChartNavigator. Each experiment verifies one of the major components of ChartNavigator.

5.1 Data Description

We conduct experiments on datasets that contains three different kinds of charts. Table 2 shows the basic information of these datasets.

Trajectory dataset. The TR dataset is a real-world dataset, which contains a set of heatmaps describing the distribution of taxi trajectories in a Chinese city for one month. The dataset is partitioned by hour and each heatmap represents the taxi trajectories in a specific hour of this month. The color of the heatmap represents the number of passing taxis. A redder color indicates a larger number of taxis. We invite two PhD students majoring in urban trajectory analysis to manually label the TR dataset. Five patterns are identified in the dataset, including 1) a heavy traffic in downtown, 2) a sparse night traffic in the entire city, 3) a heavy traffic near the mall, 4) a heavy traffic in the commercial area and 5) a heavy traffic in the tourism area. Each chart is labeled with one pattern.

Power grid datasets. We collect two simulation datasets from two real-world power grids: a small-scale one (PG-S) and a large-scale one (PG-L). Both datasets contains a set of pixel maps, which describe the changes in bus voltage when the power grid suffers different kind of electrical failures. For each pixel map, the horizontal axis represents the time and the vertical axis represents the buses, sorted by their ID. The grayscale of pixel (i, j) encodes the voltage of bus i at time j. A higher grayscale indicates a higher voltage value. We invite four power-grid engineers from the industry to manually label the PG-S dataset. Eight patterns are identified, including 1) a shutdown of the entire power gird, 2) a voltage valley for all buses immediately after the failure takes place, 3) a voltage peak after the failure, 4) an obvious voltage increase after the failure, 5) a long-term low voltage for buses near the failure, 6) a long-term low voltage for buses far from the failure, 7) a continuous voltage fluctuation of the entire power grid and 8) a continuous voltage fluctuation of half buses. Each chart in the dataset contains 2.5 patterns on average. Because it is extremely time-consuming to label the PG-L dataset, we conduct quantitative experiments on the PG-S dataset.

Scatterplot dataset. We collect the scatterplot dataset (SP) from [15], which is carefully selected from a number of real-world datasets. Each scatterplot is featured with one pattern proposed by Wilkinson et al. [14]. Seven patterns are considered in our paper, including 1) a monotonic upward trend, 2) a monotonic downward trend, 3) stringy periodic fluctuations, 4) a striated distribution 5) a convex distribution, 6) a skewed distribution and 7) a clumpy distribution. Each chart is labeled with one of these patterns.

TABLE 2
Details of the four real-world datasets used to evaluate the ChartNavigator framework.

Dataset	#Charts	Chart Size (px)	#patterns	#patterns (per Chart)
TR	744	512×352	5	1
PG-S	3,504	368×464	8	2.5
PG-L	25,925	650×800	-	-
SP	831	128×128	7	1

5.2 Factor Representation

Firstly, we evaluate the performance of OT-VAE model through the ELBO value and the quality of the generated images on the above four benchmark datasets.

TABLE 3
Comparison results of ELBO ($\leq logp(X)$) of our method and the baselines. The presented ELBO values are all negative and we do not display the negative sign of each result for consideration of clear reading. The best results are marked in bold.

Backbone	Model	PG-S	PG-L	TR	SP
MLP	VAE	314.87(±6.42)	345.16(±2.72)	1333.43(±42.42)	64.88(±, 8.15)
	LadderVAE	430.62(±9.75)	83.58(±3.06)	2546.83(±130.11)	119.48(±8.08)
	OT-VAE	**288.15**(±1.32)	**77.79**(±1.86)	**1292.00**(±58.87)	**62.24**(±7.79)
WideResNet-28-2	VAE	106.16(±5.25)	43.49(±1.40)	1169.59(±3.66)	58.04(±, 0.82)
	OT-VAE	**88.85**(±3.01)	**36.79**(±1.74)	**1152.2**(±4.44)	**55.11**(±, 1.42)
PreActResNet18	VAE	101.57(±3.75)	41.90(±1.50)	1244.06(±12.72)	58.15(±, 1.99)
	OT-VAE	**82.97**(±6.21)	**38.08**(±1.44)	**1151.67**(±8.40)	**56.42**(±, 2.82)
Densenet121	VAE	104.92(±9.48)	40.72(±2.58)	**1198.08**(±34.02)	58.46(±, 1.73)
	OT-VAE	**101.46**(±5.87)	**36.23**(±0.81)	1203.13(±55.59)	**57.29**(±, 2.18)

Baselines. We use two advanced VAE models as the baselines: the autoencoding variational bayes model (VAE) [43] and the hierarchical VAE model (LadderVAE) [42]. We consider four widely-used backbones in deep learning models, the multi-layer perceptron (MLP) [44], the wide residual networks (WideResNet) [45], the pre-acitivated residual networks (PreActResNet) [46] and the densely connected networks (DenseNet) [47]. To be fairness, the parameters of all models are on the same order of magnitude (about 10M).

Experimental settings. In training process, we use Adam [48] as the optimizer and set learning rate as $1e-5$. For the OT-VAE model, we increase the weight w_{M_z} from 0 to 1 gradually with an exponential scheduler (Appendix D). The experiments were carried out on GeForce RTX 2080Ti GPUs and all models were fully trained to achieve the best ELBO value.

Results. The ELBO on the test set are presented in Table 3. We find that the OT-VAE outperforms all competitors by at least 10% ELBO with all backbones. Moreover, with the optimal transport estimation, VAE models can achieve a tighter ELBO lower bound on almost the all cases except one in the TR dataset, which demonstrate our assumption in Section 3.1. We do not apply the LadderVAE model for the other three backbones because LadderVAE is a hierarchical model and these backbones do not possess a hierarchical architecture. Despite this, OT-VAE achieves a better generation performance with significantly small residual values compared with other baseline models (Please refer to Appendix F for the generation examples).

5.3 Pattern Identification

We now demonstrate the effectiveness of the factor operations and visual interface through several case studies and an expert interview. (For more examples please refer to Appendix E.)

5.3.1 Case Study

Case 1: Understand semantic changes

We first explain how *Factor Adjustment* can be used to help understand the semantic changes brought by different factors. As shown in Figure 4, the original chart selected from the TR dataset illustrates the distribution of taxi trajectories in a Chinese city. By decrease the fourth factor z_4 from 0.8674 to -2.7354 and decrease the eleventh factor z_{11} from 1.9174 to -3.0726, the trajectories in the highlighted area will disappear. As shown in the FAM mode, the highlighted area is no longer activated when z_4 and z_{11} decreases. According to the map, the highlighted area is a tourism area. Therefore, we draw the conclusion that these two factors correspond to the trajectory changes of the tourism area in the TR dataset.

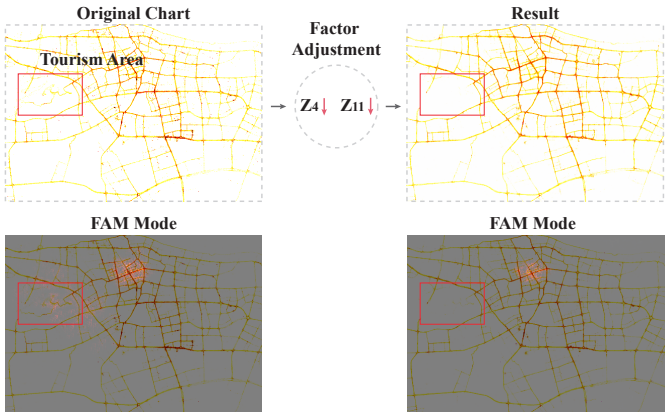

Fig. 4. Our factor operations can be used to understand the semantic changes brought by different factors.

Case 2: Walk in the factor space

Interpolation provides a more efficient way to help users understand the sophisticated factor space by walking in it. Apart from generating smooth transitions between similar charts, our interpolation operation can also find the breaking points for a pair of totally distinctive charts, which acts as a powerful tool for discovering new patterns. As shown in Figure 5, by applying interpolation between the source and target chart (step number 5, step size 0.2), we discover a breaking point in the middle of the interpolation results. By adjusting five of the factors of the breaking point chart, we derive a new chart X. As shown, chart X possesses opposite features to the source chart and share similar features with the target chart. In particular, the features presented by chart X is brand new in the PG-S dataset.

Case 3: Crop uninterested features

Addition/Subtraction operations can filter out the uninterested patterns in a given chart. As shown in Figure 6, the selected chart (A) in the PG-S dataset contains two type of patterns: 1) a sudden voltage drop at the very beginning as highlighted in red, and 2) repeated voltage increase for buses in two areas as highlighted in blue. To obtain the factor representation of the former type of pattern, we add another selected chart (B) to (A). Chart (B) have pattern (1) in common with chart (A) but behaves totally opposite to (A) after the sudden voltage drop. By adding chart (B) to

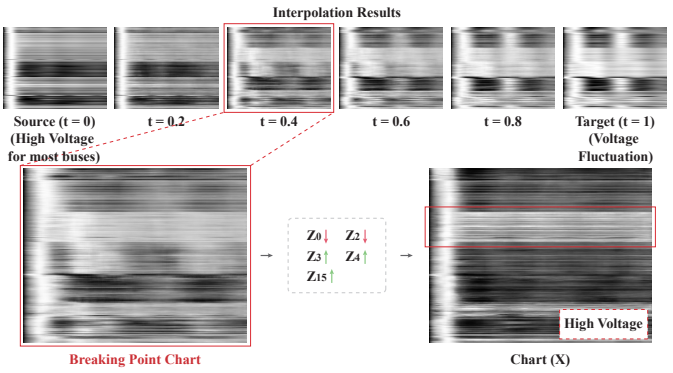

Fig. 5. An example of utilizing the interpolation operation to find the breaking point.

chart (A), we derive a synthetic chart (C) that only contains pattern (1).

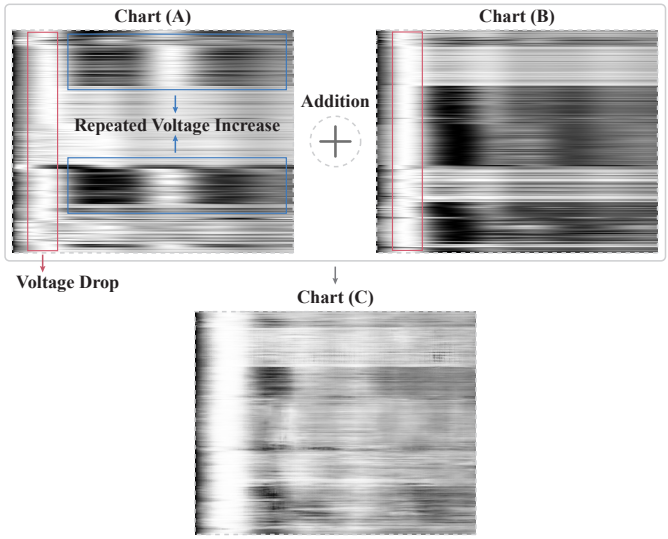

Fig. 6. An example of using addition operation to crop the uninterested features.

Case 4: Pattern Identification

Now we show a comprehensive example of using the ChartNavigator interface to identify a pattern. As shown in Figure 7, we select the chart (A) as the source to design new patterns. It contains two vertical stripes, which indicates an extreme data distribution in an attribute. The target is to define a pattern that keeps a similar distribution of the right points in Chart (A), which indicates that the data only takes a few large values for a certain attribute (the x-axis attribute). To achieve this, a straightforward idea is to subtract the left points from Chart (A). Therefore, we select a chart (B) with most points distributed on the left and perform subtraction between Chart (B) and Chart (A). The result is a set of disordered and aggregated points (result 1 in Figure 7) instead of the expected pattern. Then we adjust some factors to re-arrange these points and obtain result 2, which indicates a monotonic upward trend in the upper left corner of the chart. To approach the target pattern, we apply the FAM to result 2 and discover that the activation area of (z_0, z_4, z_5) matches the positions of the points. By adjusting these factors, points will gradually tilt to the right part (result 3 in Figure 7), which is the desired pattern.

The identified target pattern is not the same with points in the right part of Chart (A) because Chart (B) used to perform the subtraction operation contains a group of additional points in the bottom region. To facilitate pattern customization, we propose to support sketch-based interactions, which will be discussed in Section 7.

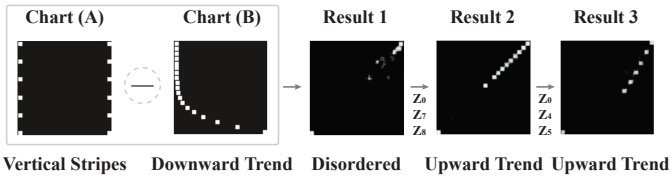

Fig. 7. A comprehensive example of pattern identification.

5.3.2 Expert Interview

ChartNavigator was evaluated by three domain experts, whose expertise was in power grid simulation (E1), traffic management (E2) and visualization (E3) respectively. They were not involved in the design and implementation of ChartNavigator. Each interview was conducted in three steps. 1) We introduced the background and the workflow of our framework. 2) We explained the interface with the dataset that the expert was familiar with (PG-S and PG-L datasets for E1, the TR dataset for E2, and the SP dataset for E3). During the interview, the expert could explore the dataset freely in the interface and was recommended to follow the pattern identification strategy (Section 4.3.3). 3) We collected their comments on the workflow and the visualization of ChartNavigator. Each interview lasted about 1 hour. Please refer to Appendix G for the interview questions.

WorkFlow: The workflow of ChartNavigator was appreciated by all experts. They agreed that ChartNavigator was effective in identifying and annotating patterns in visualization charts. E3 was satisfied with the three-step pipeline (representation, identification and annotation) and commented that *"ChartNavigator can improve the efficiency of pattern identification. The deep factor model provides the factor representation of the charts, which can be understood and operated in the interface. This helps me to quickly define a desired pattern in the interface even if it is not included in the chart dataset."* E2 was impressed by the OT-VAE model and commented that *"This model gives me a new understanding of the patterns by manually arranging, combining and changing charts."*. E1 mentioned that ChartNavigator could contribute significantly to their daily work. *"We need to analyze a large number of charts generated from numerous simulation experiments every day, ChartNavigator can effectively reduce our burden."*

Visualization: The visual interface received overall positive feedback from experts. One problem was raised during the interviews: experts found that the semantics of factors were unclear and took time to understand. As such, they were recommended to follow our strategy to explore in the interface. By applying our strategy, they noted that it become easier to understand the factors. As E1 mentioned *"The factor activation map explains the area (in the chart) affected by each factor or combination of factors, and therefore helps to*

interpret the factors." E2 appreciated the factor operations and commented *"The projection view and the cluster view provide an overview of the traffic conditions, which guides me to define a desired traffic map by using the factor operations. I especially prefer the addition, subtraction and interpolation operations, which more intuitively correspond to the increase or decrease of the traffic, while the factor adjustment operation takes some time to achieve the desired effect."* E3 appreciated the functionalities of the interface. He commented that *"The traditional way of searching patterns in the raw data is not transparent, and your interface offers an intuitive way for users to explore on their own. I consider ChartNavigator more trustworthy and reliable."*

Limitations: Experts provided suggestions to improve ChartNavigator. E1 pointed out that in each scenario, the charts were of the same category. It could help if various kinds of charts were included, e.g., analyzing the topological graphs and the pixel maps of the power grid at the same time. E3 suggested to represent each factor with a corresponding heatmap to illustrate its activated area in the chart.

5.4 Pattern Annotation

We evaluate the annotation accuracy of the ChartNavigator framework on two labeled datasets: PG-S and TR.

Baselines. We compare OT-VAE with other unsupervised data mining and deep factor models: Non-Parametric Instance Discrimination [7] (NPID) which is the state-of-the-art on ImageNet classification, and the aforementioned VAE, LadderVAE. For fair comparison, We use the same DenseNet-121 as the CNN backbone for NPID. Other experimental settings are the same with Section 5.2.

Evaluation Metric. We directly compare the annotation accuracy for the TR dataset because it is a single-label dataset. As for the PG-S dataset, we use the area under the receiver operating characteristic curve (AUC) metric to evaluate the annotation performance, which are widely used in multi-label learning algorithms [8] .

Semi-supervised Results. For fair comparison, we build KNN classifiers using the learned factor representations for all models to annotate the dataset. We present the 5-fold cross validation result of the OT-VAE model and present the best results we obtain from the other three models. The proportion of the labeled dataset varies from 1% to 10% of the entire dataset. Figure 8 shows that OT-VAE outperforms the other models by a large margin on both datasets. We also notice that when only 10% of the dataset is labeled, OT-VAE can achieve a considerable mAUC which is very close to the one in a 100%-labeled situation, while the other models are sensitive to the annotation proportion. This advantage can significantly reduce the annotation cost of the ChartNavigator framework.

Unsupervised Results. To evaluate the unsupervised annotation performance, we first identify the factor representations of the labeled patterns by using the ChartNavigator framework. Then we let all models learn the charts reconstructed from the identified factor representations. Table 4 and Table 5 show the annotations results of the PG-S and TR dataset, respectively. As shown in Table 4, our method achieves the best AUC value on most pattern categories and achieves the best mAUC. As for the TR dataset in Table 5, our method outperforms the other models by a large margin of 10%. We also notice that, OT-VAE achieves a comparable mAUC to the 1%-labeled semi-supervised annotation case. It indicates that with the support of ChartNavigator framework, the identified factor representations of patterns are effective to approach a ground-truth level.

6 Related Work

6.1 Automatic Pattern Identification

Patterns, in the sense of knowledge discovery and data mining, are interesting facts underly the data [9]. Among all pattern identification works, time series mining and frequent pattern mining are the most important and popular ones. Time-series mining aims to identify peak/valleys [49], motifs [50], trends [51] and concept drifts [52] etc. Frequent pattern mining seeks to mining frequent itemsets [53], association rules [54], sequential patterns [55] and structure patterns [56] etc. Han et al. [57] provides a detailed survey of existing frequent pattern mining methods. Particularly, to enhance the efficiency of the pattern identification process, existing works have proposed to store data in a data cube and apply approximate query techniques to speed up querying [9], [10].

However, these methods identify patterns directly in the raw data. As such, identified patterns are not always easy to read. To enhance the readability and interpretability of the identified patterns, existing works have proposed a two-step mining framework [58], [59]. Patterns are first identified from the raw data. Then, based on the data that contains patterns, a collection of visualizations are composed to help analysts compare, sort and reason the results. For example, Zenvisage [59] introduces a novel visual exploration language, ZQL, to support pattern querying and generate predefined visualizations. SeedDB [58] explores the entire data space to search for interesting patterns, and automatically generate visualizations based on the search results.

Different from the aforementioned works, there are works that skip the process of identifying patterns in the raw data. Such works focus on automatically generating visualizations according to specific visualization design knowledge and guidelines [60], [61], [62]. The main idea is that generated visualizations should contain interesting patterns, or it is easy for analysts to identify patterns in generated visualizations. These works, however, mainly focus on how to generate visualization charts, rather than how to identify patterns from generated charts.

While there has been a variety of methods for automatic pattern identification, none has addressed the problem of pattern identification in chart images instead of in the raw data. In this paper, we propose to solve this problem by using deep factor models, which not only supports pattern identification but also facilitates pattern interpretation.

6.2 VAE based deep factor model

Factor models learn vector representations of the input data. The independent variables in the learned representation captures independent factors that generate the input data, making factor models an important tool for pattern identification.

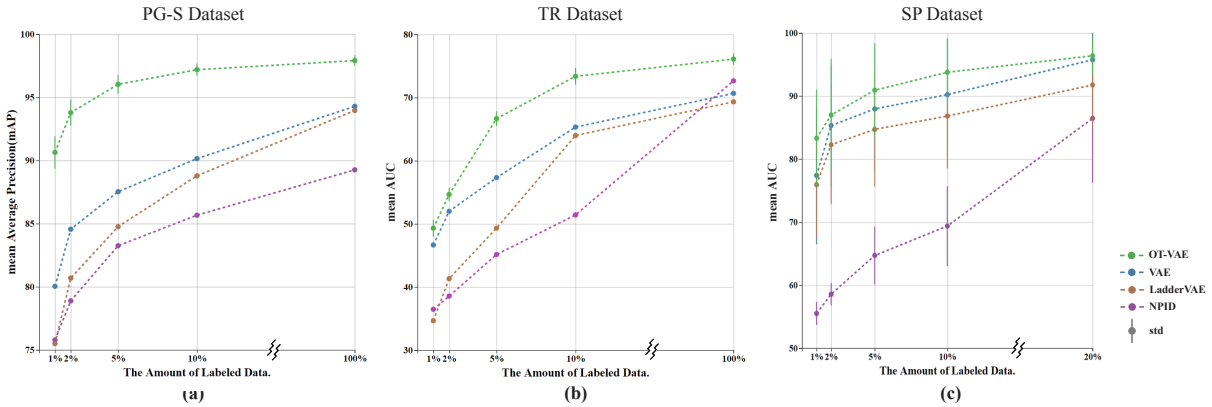

Fig. 8. Comparison Results of mean AUC of the semi-supervised annotation method with an increasing fraction of labeled data (x axis). (a) Results on the PG-S dataset. (b) Results on the TR dataset. (c) Results on the SP dataset.

TABLE 4
Comparison results of AUC and mAUC in % on the PG-S dataset. Column P_i indicates the category of pattern i. The best results are marked in bold.

Model	P1	P2	P3	P4	P5	P6	P7	P8	mAUC
VAE	64.29	**60.13**	71.71	51.96	52.80	51.78	63.37	49.88	58.24
NPID	49.93	48.85	54.14	49.02	**58.61**	49.30	42.99	49.85	50.34
LadderVAE	60.80	59.25	70.13	49.92	49.84	50.43	62.08	49.63	56.52
OT-VAE	**65.62**	59.15	**76.51**	**52.03**	49.92	**53.58**	**64.86**	**50.51**	**59.02**

TABLE 5
Comparison results of accuracy in % on the TR dataset.

Model	Accuracy
VAE	41.33
NPID	33.18
LadderVAE	40.00
OT-VAE	**53.33**

In order to learn explainable and disentangled factors, a series of VAE based deep factor model have been proposed. For example, β-VAE [25], [26] learns a conditionally independent factors of variation in the data by gradually increasing the weight of the second item in the loss function. Info-VAE [27], [28], [29] decomposes the loss function into two parts: a mutual information part and a prior divergence part. Therefore, it can strengthen the disentanglement among factors by increasing the weight of the second part. However, all these works focus on the ELBO loss function. Instead, we propose an OT-VAE model which is drastically different: It learns the independent factor representation by minimizing the margin between $\log p(\mathbf{X})$ and the ELBO loss.

6.3 Visual Explanation for Deep Inference

While the usage of deep neural networks enables superior performance, their inference processes are still black boxes [30]. In order to make the deep inference process interpretable, visualization-based techniques have been proposed to provide "visual explanations" for the inference results from a large class of Convolutional Neural Network (CNN)-based deep models. Recent works can be grouped into two categories: transpose convolution methods and gradient-based methods. Transpose convolution methods [31], [32] visualize the output of a specific layer by mapping it to the input space with the transpose convolution operation and the visualization results are often used to explain the output semantics of the intermediate layers in the deep model. The gradient-based [33], [34] methods produce multi-grained visualization results that corresponds to the different layers of the deep model by calculating the derivations of the inference bias to the related features. In our work, we extend the gradient-based methods to VAE and propose Grad-FAM to produce visual explanations for the inferred factors.

6.4 Chart Image Recognition

Existing chart image recognition mainly focuses on two tasks: chart type classification and visual contents decoding [11]. Pattern identification shares a common framework with these two tasks that starts from feature extraction and ends at pattern extraction. For the feature extraction part, prior models use hand-crafted features [12] but does not scale well with large amount of unbalanced data. Amara et. al. [13] leverage convolutional neural network (CNN) on image sets and achieve better classification results. Different from these methods, our method supports unsupervised feature extraction and arithmetic feature operations to generate higher-quality features. For the pattern extraction part, existing works mainly rely on supervised classification or unsupervised clustering which output label numbers instead of semantic patterns. In contrast, we adopt visualization-based approaches to interactively and directly define and extract patterns based on the learned representations.

7 Discussion and Conclusions

In this paper, we propose ChartNavigator, an interactive pattern identification and annotation framework for visualization charts. We develop an optimal transport variational autoencoder combined with interactive visualization-based approaches for representation learning. Experiments using real-world datasets indicate the improvement of our method compared to baselines.

Target User and Factor Setting: ChartNavigator is designed for data analysts who need to efficiently find patterns from a

large amount of chart images, for example, visualization analysts, journalists, business competitors and domain experts from different application areas. To provide meaningful factors, we suggest a trade-off between the number of factors and the performance of the OT-VAE model. On one hand, the number of factors should be as small as possible, so that duplicate semantics can be avoided. On the other hand, the ELBO of the OT-VAE model should be as good as possible, so that the trained model can better approximate the data distribution and generate meaningful factors.

Future Work: In this paper, we evaluate the effectiveness of ChartNavigator in three types of visualization charts, including scatterplot, pixel map and heatmap. We plan to incorporate more types of visualization charts, such as graph, treemap and streamgraph etc. Additionally, future iterations of ChartNavigator will adopt automated methods to draw semantics from the factor activation maps to further enhance the interpretability.

ACKNOWLEDGMENTS

This work is supported by National Key Research and Development Program (2018YFB0904503), National Natural Science Foundation of China (U1866602, 61772456, 61761136020).

REFERENCES

[1] Q. Lin, W. Ke, J. Lou, H. Zhang, K. Sui, Y. Xu, Z. Zhou, B. Qiao, and D. Zhang, "Bigin4: Instant, interactive insight identification for multi-dimensional big data," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Y. Guo and F. Farooq, Eds., 2018, pp. 547–555.

[2] B. Tang, S. Han, M. L. Yiu, R. Ding, and D. Zhang, "Extracting top-k insights from multi-dimensional data," in *Proceedings of the 2017 ACM International Conference on Management of Data*, S. Salihoglu, W. Zhou, R. Chirkova, J. Yang, and D. Suciu, Eds., 2017, pp. 1509–1524.

[3] A. Wasay, M. Athanassoulis, and S. Idreos, "Queriosity: Automated data exploration," in *IEEE International Congress on Big Data*, B. Carminati and L. Khan, Eds., 2015, pp. 716–719.

[4] M. Savva, N. Kong, A. Chhajta, F. Li, M. Agrawala, and J. Heer, "Revision: automated classification, analysis and redesign of chart images," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2011, pp. 393–402.

[5] K. Kafle, B. L. Price, S. Cohen, and C. Kanan, "DVQA: understanding data visualizations via question answering," in *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, 2018, pp. 5648–5656.

[6] L. Wei, "Multi-class blue noise sampling," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, pp. 79:1–79:8, 2010.

[7] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. L. Ball, K. S. Shpanskaya, J. Seekins, D. A. Mong, S. S. Halabi, J. K. Sandberg, R. Jones, D. B. Larson, C. P. Langlotz, B. N. Patel, M. P. Lungren, and A. Y. Ng, "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison," in *The AAAI Conference on Artificial Intelligence*, 2019, pp. 590–597.

[8] Y. Li, J. Hu, Y. Jiang, and Z. Zhou, "Towards discovering what patterns trigger what labels," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012.

[9] Q. Lin, W. Ke, J. Lou, H. Zhang, K. Sui, Y. Xu, Z. Zhou, B. Qiao, and D. Zhang, "Bigin4: Instant, interactive insight identification for multi-dimensional big data," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, 2018, pp. 547–555.

[10] B. Tang, S. Han, M. L. Yiu, R. Ding, and D. Zhang, "Extracting top-k insights from multi-dimensional data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2017, pp. 1509–1524.

[11] A. Mishchenko and N. Vassilieva, "Chart image understanding and numerical data extraction," in *IEEE International Conference on Digital Information Management ICDIM*, 2011, pp. 115–120.

[12] M. Savva, N. Kong, A. Chhajta, F. Li, M. Agrawala, and J. Heer, "Revision: automated classification, analysis and redesign of chart images," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2011, pp. 393–402.

[13] J. Amara, P. Kaur, M. Owonibi, and B. Bouaziz, "Convolutional neural network based chart image classification," in *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2017.

[14] L. Wilkinson, A. Anand, and R. L. Grossman, "Graph-theoretic scagnostics," in *IEEE Symposium on Information Visualization (InfoVis)*, J. T. Stasko and M. O. Ward, Eds., 2005, pp. 157–164.

[15] Y. Ma, A. K. H. Tung, W. Wang, X. Gao, Z. Pan, and W. Chen, "Scatternet: A deep subjective similarity model for visual analysis of scatterplots," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 3, pp. 1562–1576, 2020.

[16] T. Munzner, *Visualization Analysis and Design*, ser. A.K. Peters visualization series. A K Peters, 2014.

[17] J. Poco and J. Heer, "Reverse-engineering visualizations: Recovering visual encodings from chart images," *Computer Graphics Forum.*, vol. 36, no. 3, pp. 353–363, 2017.

[18] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer, "Reactive vega: A streaming dataflow architecture for declarative interactive visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 659–668, 2016.

[19] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[20] Larkin J H, Simon H A. Why a diagram is (sometimes) worth ten thousand words[J]. Cognitive science, 1987, 11(1): 65-100.

[21] Cook K A, Thomas J J. Illuminating the path: The research and development agenda for visual analytics[R]. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2005.

[22] Wasay A, Athanassoulis M, Idreos S. Queriosity: Automated data exploration[C]//2015 IEEE International Congress on Big Data. IEEE, 2015: 716-719.

[23] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the International Conference on Machine Learning, ICML*, vol. 32, 2014, pp. 1278–1286.

[24] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, "Vicinal risk minimization," in *Advances in Neural Information Processing Systems*, 2000, pp. 416–422.

[25] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations, ICLR*, 2017.

[26] E. Dupont, "Learning disentangled joint continuous and discrete representations," in *Advances in Neural Information Processing Systems*, 2018, pp. 708–718.

[27] H. Kim and A. Mnih, "Disentangling by factorising," in *Proceedings of the International Conference on Machine Learning, ICML*, vol. 80, 2018, pp. 2654–2663.

[28] T. Q. Chen, X. Li, R. B. Grosse, and D. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *Advances in Neural Information Processing Systems*, 2018, pp. 2615–2625.

[29] S. Zhao, J. Song, and S. Ermon, "Infovae: Balancing learning and inference in variational autoencoders," in *The AAAI Conference on Artificial Intelligence*, 2019, pp. 5885–5892.

[30] Z. C. Lipton, "The mythos of model interpretability," *Communications of the ACM*, vol. 61, no. 10, pp. 36–43, 2018.

[31] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, vol. 8689, 2014, pp. 818–833.

[32] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," in *International Conference on Learning Representations, ICLR*, 2015.

[33] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016, pp. 2921–2929.

[34] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *IEEE International Conference on Computer Vision*, 2017, pp. 618–626.

[35] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations, ICLR*, 2018.
[36] L. Ambrosio and N. Gigli, "A user's guide to optimal transport," in *Modelling and optimisation of flows on networks*. Springer, 2013, pp. 1–155.
[37] Kuang, Max, and Esteban G. Tabak. "Preconditioning of optimal transport." SIAM Journal on Scientific Computing 39.4 (2017): A1793-A1810.
[38] Kolouri, Soheil, et al. Optimal Mass Transport: Signal processing and machine-learning applications. IEEE Signal Processing Magazine 34.4 (2017): 43-59.
[39] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations, ICLR*, 2017.
[40] LeCun, Yann. "The MNIST database of handwritten digits." http://yann. lecun. com/exdb/mnist/ (1998).
[41] Basilevsky, Alexander T. Statistical factor analysis and related methods: theory and applications. Vol. 418. John Wiley & Sons, 2009.
[42] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Ladder variational autoencoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 3738–3746.
[43] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations, ICLR*, 2014.
[44] Schiffmann, W., M. Joost, and R. Werner. "Optimization of the backpropagation algorithm for training multilayer perceptrons." University of Koblenz: Institute of Physics (1994).
[45] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference, BMVC*, 2016.
[46] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, vol. 9908, 2016, pp. 630–645.
[47] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017, pp. 2261–2269.
[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations, ICLR*, 2015.
[49] Schneider R. "Survey of peaks/valleys identification in time series[J].". Department of Informatics, University of Zurich, Switzerland, 2011.
[50] Chiu B, Keogh E, Lonardi S. "Probabilistic discovery of time series motifs[C]." Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. 2003: 493-498.
[51] Povinelli R J, Feng X. "A new temporal pattern identification method for characterization and prediction of complex time series events[J]." IEEE Transactions on Knowledge and Data Engineering, 2003, 15(2): 339-352.
[52] Žliobaitė I, Pechenizkiy M, Gama J. "An overview of concept drift applications[J]." Big data analysis: new algorithms for a new society, 2016: 91-114.
[53] Goethals B. "Survey on frequent pattern mining[J]." University of Helsinki, 2003, 19: 840-852.
[54] Zhang H, Padmanabhan B, Tuzhilin A. "On the discovery of significant statistical quantitative rules[C]." Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. 2004: 374-383.
[55] Agrawal R, Srikant R. "Mining sequential patterns[C]." Proceedings of the eleventh international conference on data engineering. IEEE, 1995: 3-14.
[56] Washio T, Motoda H. "State of the art of graph-based data mining[J]." Acm Sigkdd Explorations Newsletter, 2003, 5(1): 59-68.
[57] Han J, Cheng H, Xin D, et al. "Frequent pattern mining: current status and future directions[J]." Data mining and knowledge discovery, 2007, 15(1): 55-86.
[58] M. Vartak, S. Madden, A. G. Parameswaran, and N. Polyzotis, "SEEDB: automatically generating query visualizations," *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, vol. 7, no. 13, pp. 1581–1584, 2014.
[59] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. G. Parameswaran, "Effortless data exploration with zenvisage: An expressive and interactive visual analytics system," *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, vol. 10, no. 4, pp. 457–468, 2016.
[60] X. Qin, Y. Luo, N. Tang, and G. Li, "Making data visualization more efficient and effective: a survey," *The Very Large Data Bases (VLDB) Journal*, vol. 29, no. 1, pp. 93–117, 2020.
[61] Luo Y, Qin X, Tang N, et al. "Deepeye: Towards automatic data visualization[C]." IEEE 34th international conference on data engineering (ICDE). IEEE, 2018: 101-112.
[62] Moritz D, Wang C, Nelson G L, et al. "Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco[J]." IEEE transactions on visualization and computer graphics, 2018, 25(1): 438-448.

Tianye Zhang received her B.S. in mathematics from the Zhejiang University, China in 2016. She is currently a PhD student in the College of Computer Science and Technology at the Zhejiang University. Her research interests include data mining and visual analytics.

Haozhe Feng received his B.S. in mathematics statistics from the Zhejiang University, China in 2018. He is currently a PhD student in the College of Computer Science and Technology at the Zhejiang University. His research interests include representation learning and distributed machine learning.

Wei Chen is a professor at the State Key Lab of CAD & CG, Zhejiang University. His research interests is on visualization and visual analysis, and has published more than 30 IEEE/ACM Transactions and IEEE VIS papers. He actively served as guest or associate editors of IEEE Transactions on Visualization and Computer Graphics, IEEE Transactions on Intelligent Transportation Systems, and Journal of Visualization. For more information, please refer to http://www.cad.zju.edu.cn/home/chenwei/

Zexian Chen received his B.S. in computer science and technology from the Zhejiang University, China in 2018. He is currently a master student in the College of Computer Science and Technology at the Zhejiang University. His research interests include information visualization and visual analytics.

Wenting Zheng is an assistant professor at the State Key Lab of CAD & CG, Zhejiang University. His research interests is on computer graphics and virtual reality.

Xiaonan Luo is a professor with the School of Computer Science and Information Security, Guilin University of Electronic Technology, China. He is the director of National Engineering Research Center of Digital Life and the director of Digital Home Standards Committee on Interactive Applications of China Electronics Standardization Association. He won the National Science Fund for Distinguished Young Scholars granted by the National Nature Science Foundation of China. His research interests include computer graphics, CAD, image processing and mobile computing.

Wenqi Huang is the leader of Artificial Intelligence and Intelligent Software Team in R&D Center, Digital Grid Research Institute, China Southern Power Grid. She holds B.S. (2010) and Ph.D. (2015) degrees from the Department of Information Science and Electronic Engineering, Zhejiang University. Her research interests span Artificial Intelligence, Data Mining and Blockchain application technology in the field of power industry.

Anthony K. H. Tung is a Professor in the Department of Computer Science, National University of Singapore (NUS) and a junior faculty member in the NUS Graduate School for Integrative Sciences and Engineering and a SINGA supervisor. He received both his B.Sc.(2nd Class Honour) and M.Sc. in computer sciences from the National University of Singapore in 1997 and 1998 respectively. In 2001, he receive the Ph.D. in computer sciences from Simon Fraser University (SFU). His research interests include various aspects of databases and data mining (KDD) including buffer management, frequent pattern discovery, spatial clustering, outlier detection, and classification analysis.