DashBot: Insight-Driven Dashboard Generation Based on Deep Reinforcement Learning

Dazhen Deng, Aoyu Wu, Huamin Qu, and Yingcai Wu
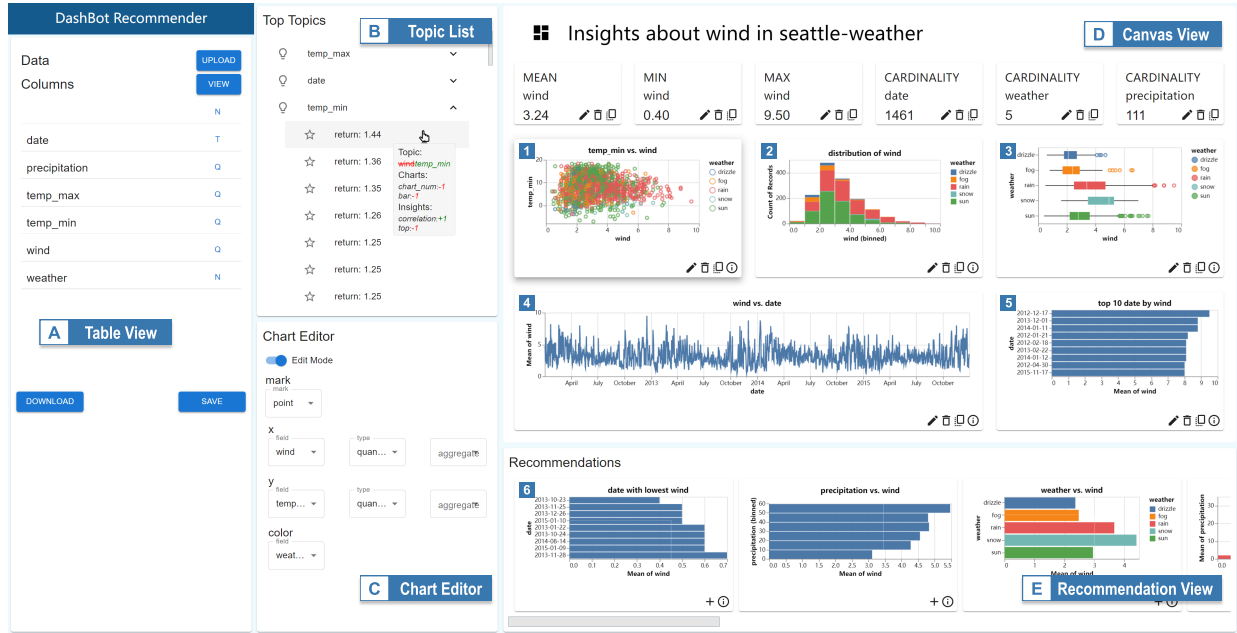
Fig. 1. Screenshot of the DashBot Interface showing the generation of a dashboard named "Insights about wind in seattle-weather". The interface consists of a table view (A), a topic list (B), a chart editor (C), a canvas view (D), and a recommendation view (E).

Abstract—Analytical dashboards are popular in business intelligence to facilitate insight discovery with multiple charts. However, creating an effective dashboard is highly demanding, which requires users to have adequate data analysis background and be familiar with professional tools, such as Power BI. To create a dashboard, users have to configure charts by selecting data columns and exploring different chart combinations to optimize the communication of insights, which is trial-and-error. Recent research has started to use deep learning methods for dashboard generation to lower the burden of visualization creation. However, such efforts are greatly hindered by the lack of large-scale and high-quality datasets of dashboards. In this work, we propose using deep reinforcement learning to generate analytical dashboards that can use well-established visualization knowledge and the estimation capacity of reinforcement learning. Specifically, we use visualization knowledge to construct a training environment and rewards for agents to explore and imitate human exploration behavior with a well-designed agent network. The usefulness of the deep reinforcement learning model is demonstrated through ablation studies and user studies. In conclusion, our work opens up new opportunities to develop effective ML-based visualization recommenders without beforehand training datasets.

Index Terms—Reinforcement Learning, Visualization Recommendation, Multiple-View Visualization

---

1 INTRODUCTION

Analytical dashboards have been broadly used in business intelligence to help data analysts explore and discover data insights with multiple-view visualizations (MVs) [46]. Even with the help of professional

- D. Deng and Y. Wu are with the State Key Lab of CAD&CG, Zhejiang University. Y. Wu is also with the Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies. E-mail: {dengdazhen, ycwu}@zju.edu.cn.
- A. Wu and H. Qu are with Department of Computer Science and Engineering, the Hong Kong University of Science and Technology. E-mail: {awuac, huamin}@cse.ust.hk.
- Yingcai Wu is the corresponding author.

authoring tools, such as Tableau and Power BI, creating an effective dashboard is still a highly demanding task, requiring expertise in data analysis and visualization. Specifically, the analyst needs to explore the dataset, select appropriate data columns and visual encodings to configure charts, and investigate whether the charts are insightful. In addition, the analyst has to consider the relationship between charts to exhibit different perspectives of the dataset [65]. Such a process of exploratory data analysis for dashboard generation is trial-and-error [8].

To reduce the burden, many studies have investigated rule-based and machine learning-based (ML-based) methods for visualization recommendation. Rule-based methods, such as APT [36], CompassQL [70], and Voyager [69, 71], translate well-established visualization design rules (e.g., expressiveness and effectiveness criteria [36]) to be programmable constraints for the recommendation. Differently, ML-based methods employ state-of-the-art models, such as decision trees [33] and deep neural networks [17, 21, 30], to learn common patterns of visual encodings from large-scale chart datasets [16, 22]. Though useful in generating effective charts, these methods focus on single visualizations instead of MVs, where the relations between charts are important.

Towards the generation of MVs, a series of studies investigate the use of hand-crafted rules. For example, to tell data stories, Datashot [64] and Calliope [53] adopt statistics metrics (e.g., Pearson correlation coefficient) to extract facts from datasets and then generate charts according to the facts. These successful cases prove the usefulness of hand-crafted rules. Recently, deep learning-based methods have also been used to improve the efficiency of generating MVs. For example, MultiVision [75] trains deep neural networks to score the goodness of single charts. Then the single chart scoring is combined with customized metrics to generate multiple charts. In this method, the generation of MVs is conducted indirectly due to the lack of high-quality MVs datasets, which could harm the training process and results.

In this paper, we propose to use deep reinforcement learning to generate analytical dashboards, taking advantage of established visualization knowledge and efficient machine intelligence. We argue that applying reinforcement learning to such a scenario has several advantages. 1) Intuitive modeling. Reinforcement learning agents learn from environments through exploration, which is similar to the mechanism of exploratory visual analysis. Therefore, modeling the exploratory visual analysis process with reinforcement learning is natural. 2) Self-play training. With carefully designed environments, action spaces, and reward functions, agents can be constantly trained with different datasets and obtain a shared experience of dashboard generation, with no need for beforehand training data with labels. 3) Online recommendation. Due to a similar mechanism, a reinforcement learning-based recommendation system supports a swift switching between human steering and automation. When users update the dashboards by preference, the agents can generate subsequent recommendations promptly.

However, designing a reinforcement learning model for analytical dashboards is challenging. First, it lacks a well-established environment for "visualization agents" to explore and train. Different from training agents to play games (e.g., AlphaGo [55]), in which the winning and losing can explicitly measure the goodness of agent actions, there is no deterministic rule to evaluate the generated dashboards. Second, it is difficult to design an agent to imitate complex human behavior in generating dashboards, including configuring charts by multiple parameters (e.g., mark types, encoding types, data fields, and data transformations) and exploring the large space of chart combinations.

To address the above challenges, we developed DashBot, a deep reinforcement learning-based recommendation system for analytical dashboards. For the first challenge, we investigated state-of-the-art design guidelines for MVs [44, 65] and conducted a preliminary study with a collection of dashboard designs of Tableau and Power BI. From the study, we investigated how to use design guidelines for assessing dashboards. Built upon the guidelines and knowledge gained, we designed a dashboard playground, which scores the generated dashboards and provides an interface for the agents to explore the dashboard design space. For the second challenge, we formulated the exploratory dashboard generation to be a sequence prediction problem. Specifically, at each state, agents can decide the actions to take (e.g., adding or removing a chart) and configure chart parameters (e.g., chart types and encoding types). We designed a novel deep neural network to achieve the prediction. To improve training efficiency, we proposed a constrained sampling strategy to ensure the validity of generated charts while preserving the exploration uncertainty of the agents. To validate the usefulness of the proposed method, we conducted an ablation study for the model design and comparative experiments with a state-of-the-art dashboard generation system. We also analyzed the user feedback and reflected on designing automatic agents to generate analytical dashboards. In summary, we have four major contributions.

- A preliminary study that reviews practical dashboard designs and summarizes the design considerations for the recommendation.
- A reinforcement learning formulation for dashboard generation that features the definition of reward functions for evaluating the expressiveness and insightfulness of the dashboards.
- A novel deep neural network for agents to explore the actions and parameters for generating dashboards.
- A series of quantitative and qualitative studies that validate the usefulness of the proposed approach and lessons learned in designing automatic agents for visualizations.

2 RELATED WORK

In this section, we introduce related studies from the perspectives of visualization recommendation, multiple-view visualization generation, and reinforcement learning for visualization.

2.1 Visualization Recommendation

Existing visualization recommendation approaches can be categorized into rule-based methods and ML-based methods [45, 74]. Rule-based methods utilize the principles in visualization theories to construct visual mapping. For example, APT [36] incorporates expressiveness and effectiveness criteria [11] into graphical languages to formulate visualizations. Show Me [37] and CompassQL [70] employ query techniques to enumerate visual encodings. Furthermore, Voyager [69, 71] adopts statistics and perceptual measures to rank the generated visualizations and supports interactive exploration.

ML-based methods incorporate machine learning models to predict the visual mapping. A number of methods formulate the visual mapping as a non-linear regression from hand-crafted data features to charts, such as VizML [21], NL4DV [41], and wide-and-deep recommendation network [43]. Other methods formulate the recommendation as different problems, such as sequence-to-sequence translation [17, 82], learning-to-rank [33, 40, 57, 76], and knowledge graph [30, 83]. However, these recommendation methods mainly focus on generating a single chart, which might be insufficient for solving the visual analysis problems with high-dimensional data.

2.2 Multiple-View Visualization Generation

Multiple-view visualizations (MVs) are useful in visual analysis for their capability in representing different perspectives of data simultaneously. Numerous MVs, which refer to visual analytics (VA) systems, have been created to discover patterns and insights [4, 72]. Existing studies of visualization recommendation for VA systems focus on layout problems regarding organizing multiple views [51]. For example, Al-maneea and Roberts [3] proposed a series of criteria to decompose the VA systems in the publications and quantify their layouts. Chen et al. [14] investigated view composition and configuration of the systems.

Existing studies of MV generation target to creating MVs from tabular data for insight discovery [25] or storytelling [19, 50]. For example, Voder [56] and QRec-NLI [63] adopts natural language processing models to extract data facts or recommending next-step queries for dashboard exploration. Zhao et al. [81] proposed ChartStory, a system that composes charts into comic-style visualizations. DataShot [64] generates data fact sheets with a template-based method for visual storytelling. Similarly, Calliope [53] obtains data insights with customized metrics and identifies the best ones using a Monte Carlo tree search. These rule-based methods can well integrate visualization domain knowledge into the design of metrics. Recent research also explores the generation of dashboards with deep learning. MultiVision [75] employs bidirectional long short-term memory models to score and rank single charts for MV generation. In this work, we integrate deep learning methods and visualization knowledge to generate analytical dashboards. Specifically, we utilize the capability of deep neural networks in simulating complex environments and take advantage of well-established visualization design rules to score the generated visualizations.

2.3 Reinforcement Learning and Visualization

Reinforcement learning aims to train agents to take actions in specific environments so that the agents can gain the highest accumulated rewards. Given a current observation, Q-learning [66] is designed to predict the rewards that can be gained and choose the actions with the highest rewards. However, Q-learning can only handle a small number of observations and actions. To cope with the problems with complex situations, more variants based on deep learning have been proposed. For example, to handle the large observation space of game screens during playing Atari video games, Mnih et al. [39] proposed deep Q-learning. Though useful, deep Q-learning fails when there is a high-dimensional action space. The problem of instability during training also arises. A series of policy gradient methods [31, 38, 48, 49] have been proposed to address these problems. In this work, we choose to
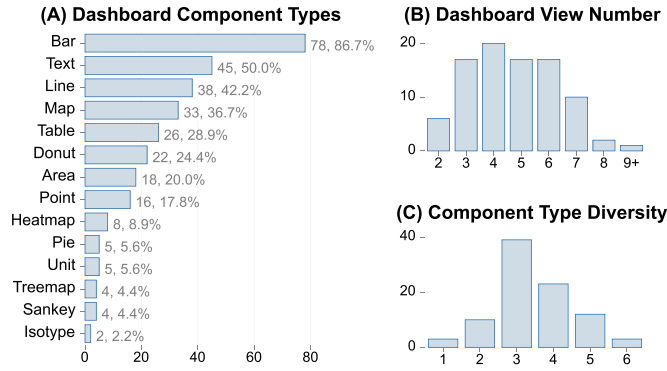
Fig. 2. Statistics in the preliminary study: distributions of dashboard component types, numbers of views in a dashboard, and numbers of different component types in a dashboard.

use asynchronous advantage actor-critic algorithm (A3C) [38], a state-of-the-art reinforcement learning framework, to handle the problem of dashboard generation, in which both observation space (i.e., dashboards with different chart combinations and variant chart numbers) and action space (i.e., generating chart configurations) are high-dimensional.

A few studies have used reinforcement learning models for visualization generation. For example, MobileVisFixer [73] adopts an explainable Markov decision model to optimize the layouts of visualizations on mobile devices. Bako et al. [6] also adopted a Markov decision process model to recommend potential D3 syntax for authoring visualizations. Tang et al. [59] adopted reinforcement learning to create storyline layouts. Shi et al. [52] and Wei et al. [67] used reinforcement learning to predict next-step operations of chart editing. In this work, we target the dashboard, a multiple-view visualization with larger design space, and formulate the problem of dashboard generation to be a reinforcement learning problem.

3 DESIGN OF DASHBOT

To design an effective recommendation system, we conducted a preliminary study to understand the current practices of analytical dashboards and derive the design considerations of recommendation systems.

3.1 Preliminary Study

Existing studies for visualization designs mainly focus on visualization genres such as infographics [12], storylines [60], and data stories [54, 64]. There are few studies that investigate the design patterns for analytical dashboards [5, 46]. Therefore, we start with a preliminary study to gain an overview of the design practices.

We first collected dashboards with the most number of "likes" from the official galleries of Tableau [2] and Microsoft Power BI [1], which contain hundreds of high-quality examples. Not all examples in the galleries are dashboards and some of them are posters (telling stories mainly with text and charts only for assistance) or infographics (with only one well-designed view). Therefore, we carefully filtered the dashboards from the galleries. Finally, we obtained 40 Tableau dashboards and 50 Power BI dashboards.

Second, we analyzed the dashboards from visual and data presentation perspectives. Based on the design guidelines of multiple-view visualizations [65], a good design should follow the rules of diversity, complementarity, decomposition, and parsimony. We opted to understand the collected designs concerning these rules.

Diversity. We analyzed how diverse chart types are used to represent the data columns. A dashboard usually contains multiple views, and each comprises one or multiple charts or components (e.g., text and table). Therefore, for each dashboard, we annotated the types of charts and components, as shown in Fig. 2(A). From the results, we discovered that the bar chart is the most commonly used chart type, followed by line charts, maps, and donut charts. Text components are the second most common, used to summarize the insights in the charts or show key indicators independently. The table components are usually used to show raw data directly.

Parsimony. The rule of parsimony refers to minimizing the number of views while preserving effectiveness and expressiveness. Therefore, we counted the view numbers of the collected dashboards (Fig. 2(C)). We discovered that most dashboards are composed of 3-6 views. Only a small number of dashboards contain more than 8 views.

Complementarity & Decomposition. Complementarity refers to how charts complement each other to exhibit different perspectives of the datasets. Based on the definition in the prior study [75], we regard two views to be complementary to each other when they visualize different data columns. For example, a view encodes columns A and B and another view encodes columns C and D. On the contrary, decomposition refers to analyzing complex data with multiple charts, such as chunking the data or applying different aggregation methods to a data column. These charts will share the same data column. When investigating the examples in-depth, we discovered that few dashboards include views complementary to each other, because the dashboards usually concern specific "key columns". In fact, in most dashboards (96.7%), all their views are related to one or two data columns.

From the analysis above, we understood that to design an effective dashboard, it is encouraged to identify a topic (i.e., a key column) and configure composed charts to discover insights surrounding the topic [64]. Besides, it is necessary to introduce adequate chart diversity to enhance expressiveness but avoid a large chart number.

3.2 Design Considerations

Based on previous empirical studies [44, 65], recommendation systems [53, 75], and our preliminary study on current practices, we derive design considerations of a recommendation system for dashboards.

DC1 Generate valid dashboards automatically. The dashboards should be automatically generated with specific characteristics. Based on the preliminary study, an analytical dashboard commonly features multiple charts with diverse types and components of text and tables. The charts should also follow effectiveness rules [36]. For example, bar charts are suitable to visualize the data of a nominal column and a quantitative column.

DC2 Facilitate self-steering data insight discovery. In addition to merely visualizing the data with appropriate visual encodings, an effective dashboard is supposed to convey data insights. When generating the dashboard, the system should recognize data insights, such as high correlations and temporal distributions, and prioritize selecting charts with insights into the dashboard.

DC3 Enable direct manipulation on the recommendations. It is unlikely to generate dashboards that fulfill the requirements of all users. Therefore, users should be allowed to modify the dashboards directly. The system should provide an interface for users to customize the dashboards according to their preferences and add new charts in an exploratory manner.

DC4 Support online recommendation during exploration. Editions on the dashboard inherently exhibit user preferences, which provide additional conditions for the recommendation. Therefore, the system should be able to start from current dashboard configurations and explore the best dashboards accordingly. Furthermore, the system should promptly generate new recommendations after the modifications to ensure interaction efficiency.

Guided by these design considerations, we develop a recommendation system empowered by a deep reinforcement learning-based computation module. The computation module is built with a deep neural network that can generate valid charts (**DC1**) and discover data insights (**DC2**) automatically. To ensure the validity of the generated charts, we propose a novel constrained sampling method to apply rules to the sampling state (**DC1**). Besides, we carefully design insight rewards to encourage the discovery of insights during the exploration (**DC2**). On top of the computation module, we develop an interactive interface that allows users to edit the recommended dashboards (**DC3**). The user editions are then fed back to the computation module, and the module generates new recommendations based on the editions (**DC4**).

4 PROBLEM FORMULATION

This section introduces how to formulate exploratory dashboard generation to be a reinforcement learning problem. We first introduce the Markov decision process (MDP), the foundation of reinforcement

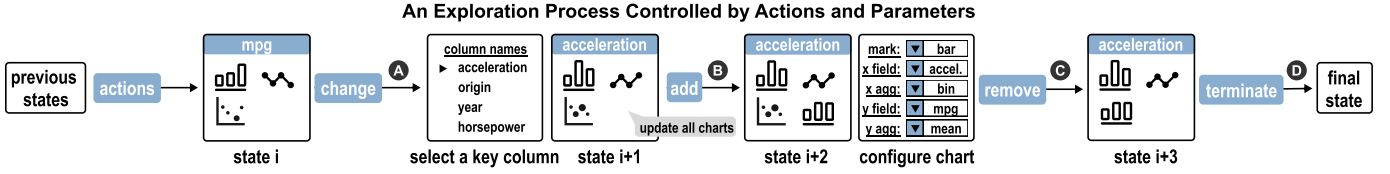**An Exploration Process Controlled by Actions and Parameters**

Fig. 3. An exploration process is controlled by actions and parameters. The agent first changes the key column at state i (A) and adds a new chart into the dashboard by configuring chart parameters (B). Then the agent removes a chart (C) and terminates the whole exploration process (D).

learning. Then we interpret human behaviors of designing analytical dashboards into MDP actions.

4.1 Background: Markov Decision Process

Markov decision process (MDP) [10] is a stochastic control process. An MDP can be formulated as a sequence of states and actions:

$$P = \{(s_i, a_i, p_i, r_i) | i \in [1, T]\}, \tag{1}$$

where T is the length of the process, $s_i \in \mathcal{S}$ is the state at step i, $a_i \in \mathcal{A}$ is a selected action from action space $\mathcal{A}$, $p_i \in \mathbb{R}^{|\mathcal{A}|}$ is the probabilities of different actions, and $r_i \in \mathbb{R}$ is the immediate reward after executing action a_i at state s_i.

The key of reinforcement learning (RL) is to train agents obtain the most cumulative rewards after the exploration, i.e., return $R = \sum_{i=1}^{T} \gamma^{i-1} r_i$, where r_i is the immediate reward obtained at each state and γ ($\gamma = 1$) is a discount rate. Therefore, the MDP is suitable to model exploratory processes, in which the goodness of the actions for the final result can be quantitatively evaluated at each state.

4.2 Formulating Dashboard Generation as MDP

We define the states and actions of designing an analytical dashboard. Based on the preliminary study, we model the generation of dashboards as a process to identify a key column (i.e., topic) from the dataset and configure a series of charts with additional explanation columns to reveal insights about the topic.

State Space. During dashboard generation, the agent decides the next actions according to the current configuration of the dashboard, which is a collection of charts. Therefore, the state space $\mathcal{S}$ enumerates all valid chart combinations with a given dataset.

$$\mathcal{S} = \{\{chart_j | j \in [0, n]\} | n \in [0, N]\}, \tag{2}$$

where $chart_j$ denotes a chart and N is the maximum chart number.

Action Space. From a specific state, the agent continues to explore the dashboard design space by taking action. To fully explore the design space, agents are allowed to *change* the key column, *add* a new chart, *remove* an existing chart, and decide whether to *terminate* the exploration session. The action space is defined as follows.

$$\mathcal{A} = \{change, add, remove, terminate\} \tag{3}$$

Different actions require different parameters for execution. When the agent decides to *change* the key column, it should further select a new one from the dataset. All charts will replace the key column with the new one. To *add* a new chart, the agent has to specify mark types and visual encodings for the chart. The action *remove* refers to removing an existing chart from the dashboard. The action *terminate* will end up the exploration session and finalize a dashboard. An example of executing actions with parameters is demonstrated in Fig. 3.

4.3 Reward Functions

After executing the action and obtaining a dashboard with a new state, the environment is supposed to return a reward to tell the agent how well is the generated dashboard from the previous action. Reward functions are critical for training effective agents to achieve the goals, which should be carefully designed based on existing knowledge about dashboard design. The immediate reward at step i is defined as follows,

$$r_i = f(s_i, a_i), \tag{4}$$

which is computed from the action a_i and state s_i. We have designed two categories of reward functions, namely, presentation rewards and insight rewards.

4.3.1 Presentation Rewards

The presentation rewards are derived from the rules of diversity and parsimony based on the preliminary study.

Diversity Reward. Dashboards commonly adopt different chart types to improve expressiveness. However, a higher diversity is not always better because users have to interpret the visual encodings of different charts, increasing users' cognition load [44]. From the preliminary study, most of the dashboards contain 2 or 3 different chart types. Therefore, we use a concave increasing function to award the increase of diversity with the idea of "diminishing returns":

$$dr(s_i, a_i) = 1 - exp(-\frac{\alpha \cdot c_{used}}{c_{total}}). \tag{5}$$

The variable c_{used} denotes the number of chart types used in the dashboard and c_{total} is the total number of chart types allowed, and α is a weight to adjust the diminishing degree. Similarly, we use the function to score the diversity of the visualized columns with regard to all columns of the dataset.

Parsimony Reward. Similar to chart diversity, the chart numbers should not be too large. From the preliminary study, most of the dashboards contain 3-5 different charts. However, different from chart types that have limited choices, the number of charts can be ambiguously large. Therefore, the parsimony reward is a piece-wise function that firstly increases and then decreases with regard to the chart numbers:

$$pr(s_i, a_i) = \begin{cases} \sin\frac{\pi}{2} \cdot \frac{n}{n_{best}}, & n \in [0, n_{best}] \\ \sin\frac{\pi}{2} \cdot (1 + \frac{n - n_{best}}{n_{max} - n_{best}}), & n \in (n_{best}, n_{max}]. \end{cases} \tag{6}$$

The variable n is the chart number of the dashboard, n_{best} is the best number of charts, and n_{max} is the maximum number of charts.

4.3.2 Insight Rewards

The use of a dashboard is to identify and visualize insights. Therefore, we design insight rewards to award the discovery of insightful charts. We enumerate the metrics proposed by existing methods [7,53,58,64] and categorize the insight metrics based on the number of columns involved, including single-column insights and double-column insights. Additionally, we propose to measure multiple-column insights on the basis of single and double-column insights.

Concerning a single column, it is common to exhibit the statistics (e.g., mean and cardinality) or value distribution. These insights can give an overview of the selected column, but users might be more interested in the relation between the two columns. Previous studies [53, 64] have adopted statistical metrics to measure double-column insights, such as the Pearson correlation coefficient. In this work, we model the double-column insights of a dashboard in a similar way.

Furthermore, we propose to model multiple-column insights for dashboard insight discovery. The multiple-column insights refer to the insights derived from multiple charts, where more than two explanation columns are involved. For instance, if column A has a high correlation with column B and column A has a high correlation with column C, it is possible that column B correlates to column C. Existing methods mainly focus on the insights in single charts [64] or relations between two neighboring charts [53]. Differently, we model the dashboards as a collection of charts and consider the relations between any two charts.

Table 1. Definitions of the Insights.

Insight	Definition
distribution	$A \in \mathcal{Q}$: visualize A with a histogram by applying bin count.
trend	$A \in \mathcal{Q}, B \in \mathcal{T}$: visualize A across B with a line chart.
correlation	$A \in \mathcal{Q}, B \in \mathcal{Q}$: visualize A across B with a line chart or scatterplot, and the correlation between A and B is higher than the threshold.
top/bottom k	$A \in \mathcal{N}, B \in \mathcal{Q}$: visualize top or bottom k entities of A with B.
co-correlation	$A \in \mathcal{Q}, B \in \mathcal{Q}, C \in \mathcal{Q}$: there are correlation insights about (A, B) and (A, C).
comparison	$A \in \mathcal{N}, B \in \mathcal{Q}$: there are top and bottom k insights about A and B.

*note: $\mathcal{Q}$, $\mathcal{T}$, and $\mathcal{N}$ stand for quantitative, temporal, and nominal columns.

For each insight, we demonstrate the conditions for chart types and column types and the statistical conditions to be fulfilled (Table 1). When recognizing a single/double/multiple-column insight, the reward value would be 1, 2, and 3, respectively.

4.3.3 Combined Rewards

We combine the rewards together by weighted sum:

$$cr_i = w_1 \cdot \sum_c^{\{col,vis\}} dr_i^c + w_2 \cdot pr_i + w_3 \cdot \sum_c^{\{insights\}} ir_i^c, \qquad (7)$$

where $\{w_k\}$ are constant values that balance the magnitude of the rewards. We empirically set $w_1 = w_2 = 0.33$ to normalize the presentation rewards and set $w_3 = 0.1$ to encourage the gaining of the insights. We compute the immediate reward gained at state i by

$$f(s_i, a_i) = cr_i - cr_{i-1}, \qquad (8)$$

where cr_i and cr_{i-1} are dashboard rewards at states i and $i - 1$.

5 DESIGN OF DASHBOT

Based on the formulation of dashboard generation, we further design proper machine learning models to facilitate the exploration.

5.1 Model Framework

In this work, we use the asynchronous advantage actor-critic algorithm (A3C) [38], a novel reinforcement learning framework, to train agents to explore the dashboard design space. A3C is a flexible framework designed for the problem with a large action and observation space, which is well-suited for the problem of visualization generation [59].

A3C framework consists of two modules, an action network predicting the action probabilities and a critic network evaluating the maximum expected return from the state. The main idea of A3C is to learn an accurate critic network to evaluate the expected return from the state. Then the critic network is used to train the action network to take the best actions from specific states. During training, the return R, probabilities of the actions p_i, value estimation for the expected return $v(s_i)$, and rewards r_i are fed into a loss function:

$$L(s_i, p_i) = (R - v(s_i))^2 - \log(p_i)A(s_i) - H(p_i), \qquad (9)$$

where $A(s_i)$ is advantage function and $H(p_i)$ is entropy of the action probabilities [38]. The framework of the A3C is shown below.
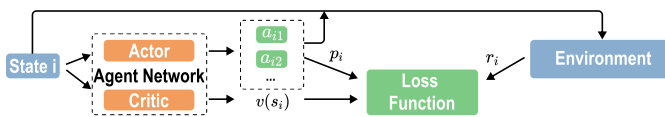
The asynchronous mechanism of A3C enables the training by exploration with multiple independent agents, which can accelerate the convergence of networks. The design of agent network comprises feature engineering of dashboards (Sect. 5.2), sequential network structure for action and parameter selection (Sect. 5.3), and constrained sampling with visualization design rules (Sect. 5.4). The detailed structure combining these modules together is demonstrated in Fig. 4.

5.2 Dashboard Feature Construction

The construction of a deep neural network requires numerical representations for the input data. However, it lacks a proper feature engineering method for dashboards. Recently, Wu et al. [75] proposed to model the dashboard features in an indirect way. They firstly obtained single chart features through a learning-to-rank neural network and then packed the features together as the representation of the dashboard. Though useful, such a method requires another network for feature engineering. At the same time, other studies, such as VizML [21] and KG4Vis [30], propose to represent data columns with hand-crafted features, which are proven to be effective through experiments. Therefore, we propose to combine the learning-based and hand-crafted features for dashboard representation. Specifically, we first incorporate hand-crafted column features to construct dashboard representations. Then the representations are fed into neural networks to learn high-level embedding for action and parameter prediction.

We first construct column features from column properties and statistics metrics based on VizML [21] (Fig. 4A). Column properties include data types (e.g., quantitative, nominal, or temporal), minimum value, cardinality, etc. Statistics metrics consist of values computed from statistics models, such as skewness and Gini impurity [68].

Based on the column features, we further construct chart features by representing chart attributes (Fig. 4B). Our representation is based on Vega-Lite [47], a declarative programming language popular for chart rendering. We primarily focus on mark types of bar, line, point, and boxplot, and visual channels of x, y, and color. The representation can be easily extended for more mark types and visual channels with a similar modeling method. For each chart, we represent mark type and the use of visual channels with one-hot encoding. If a visual channel is used, we append the feature of the encoded column (denoted as field feature). It is noted that the field feature is not the copy of the column feature, but the features of the encoded data after data transformation for rendering. Taking Fig. 7-B1 as an example, the features of "y" field would be the features of "*mean US Gross grouped by Major Genre*" instead of "*US Gross.*" Hereby, we intrinsically embed the fields of "aggregate" for each visual channel.

The dashboard features are constructed by packing the chart features together with context information about the key column and datasets (Fig. 4C). For each chart, we append the features of the current key column and all data columns to facilitate the column selection during exploration. We set a maximum number of 10 for data columns. For datasets with less than 10 columns, we pad the chart features with zeros. Finally, we obtain the dashboard features at step i, $e_i \in \mathbb{R}^{n \times l}$, where n is the current chart number and l is the length of chart features with contextual information added.

5.3 Agent Network Architecture

On the basis of dashboard features, we design an agent network for action prediction and parameter selection. The design of the network should fulfill three requirements. First, the network can learn mutual relationship between charts and generate a unified representation for the dashboard. Second, the network is supposed to achieve value estimation, action prediction, and parameter selection with an integrated structure. Besides, the network should predict parameters considering their interrelation. For example, when adding a new chart, the agent should focus on chart configurations, but when deciding to change the key column, the agent should consider the columns to be selected.

To achieve the first requirement, we introduce a long short-term memory (LSTM) layer to model the relations between charts. In our task, we aim to learn the chart combinations instead of sequence orders. However, it is computationally costly to enumerate the chart combinations with all possible orders. As a compromise, we use a bidirectional LSTM (Bi-LSTM) [23] layer that summarizes the information from both directions (Fig. 4E). In addition, we randomly shuffle the charts during training, which generates dashboards with different chart orders. The embedding extracted from the Bi-LSTM is then fused to be a unified shared embedding for the dashboard (Fig. 4F).

For the second requirement, we incorporate multiple classification blocks for value estimation, action prediction, and parameter prediction. First, the value estimation is achieved with a fully connected layer taking the shared embedding as input (Fig. 4G). Next, for action and

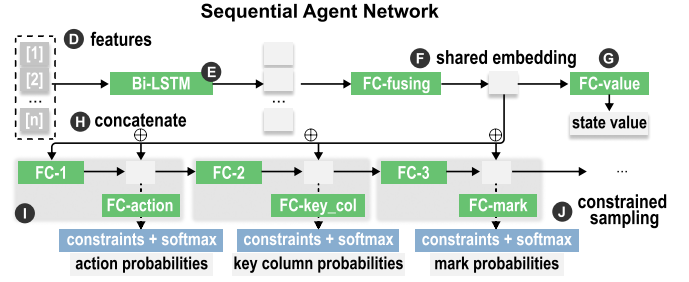**Dashboard Feature Engineering**
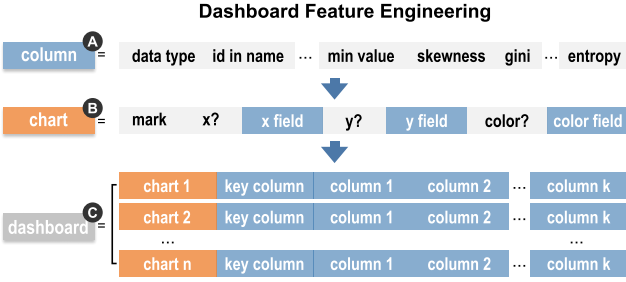
Sequential Agent Network

Fig. 4. Process of feature construction and structure of sequence generation network. The dashboard features (C) are constructed from column features (A) and chart features (B). The dashboard features (D) are fed into the sequence generation network with a Bi-LSTM layer (E) for the extraction of shared embedding (F). The shared embedding is forwarded to a fully-connected layer for value estimation (G). The shared embedding is then sequentially fed into several classification blocks (I) and concatenated with intermediate embedding (H) for the prediction of action and parameter probabilities with constrained sampling applied (J).

parameter prediction, we introduce a sequential model structure that predicts the parameters by incorporating the embedding from previous predictions of actions and parameters. Specifically, the shared embedding is sequentially passed into several classification blocks (Fig. 4I), each block responsible for predicting an action or a parameter. The inputs are forwarded into a fully connected layer in a classification block to obtain an intermediate embedding, which is then fused with the shared embedding (Fig. 4H) and fed into a fully-connected layer. The fused embedding is then forwarded into the next classification block for the prediction of the following parameter. With such a sequence model, the prediction of parameters can refer to the previous context information and thus accelerate the convergence. In each classification block, the outputs are fed into a categorical softmax layer, and probabilities for different actions or parameters are obtained (Fig. 4J).

5.4 Constrained Sampling

When using an agent network for action execution with parameters, it is critical to control the activation of different network components for three reasons. First, the column numbers of different datasets are different. It is supposed to avoid the selection of non-existing columns. Second, it is supposed to activate different branches of parameters based on the actions. For example, when the agent selects to "change" the key column, the branches for configuring charts should be disabled. Third, it is required to rule out improper parameter values based on the selection of previous parameters. The parameters in chart configurations are interrelated. However, the agent network possibly generates incorrect chart configurations which violate the grammar of visualization rendering. False configurations might result in failed charts without any analytical and aesthetic value. To alleviate the situation, we attempted to introduce a large penalty for invalid chart configurations. The agents succeeded in reducing the generation of invalid charts, but the balancing between penalties and rewards for insights and chart designs becomes another critical problem (Sect. 6.2). Nevertheless, even the best machine learning models might make mistakes. Therefore, to ensure the correctness of chart generation, constraints are commonly used to regulate the model weights. For example, to generate valid chart configurations from natural languages, NL2Vis [34] uses constraints to mask the attention of transformer networks to avoid false inference.

Similarly, we formulate visualization knowledge as constraints to ensure sampling reasonable parameters for dashboard exploration. Our design of constrained sampling is based on sequence modeling, in which the inference is performed sequentially. When an inference of previous action or parameter is obtained, constraints based on visualization knowledge will be applied in the next classification block to ensure the validity of the final configuration.

We demonstrate how constrained sampling works through an example of configuring a bar chart (Fig. 5). In the example, the action classification block of the agent decides to "add" a new chart for a current dashboard with the key column of "horsepower." The agent has to configure a new chart by specifying parameters. Therefore, constrained sampling is applied on the key column classification block disabling all weights and generating a dummy sampling. For mark types, all choices are available. The agent samples the "bar" based on

the softmax value, although the "bar" is not the one with the highest probability. After selecting the mark type, the agent network predicts an explanation column. The selection of "horsepower" is disabled because "horsepower" is the key column. The agent selects "origin," which is a nominal column, so the "aggregate" of mean and max are disabled. Next, the agent will generate an "aggregate" of the key column and color encodings with similar behaviors. With such an activation mechanism, we ensure the validity of the sampled configurations. It is noted that the constraints are applied to the predictions before softmax, which ensures the validity of entropy computation for back-propagation.

5.5 DashBot Interface & Interactions

To facilitate interactive creation, we design an interface consisting of a table view (Fig. 1A), a topic view (Fig. 1B), a chart editor (Fig. 1C), a canvas view (Fig. 1D), and a recommendation view (Fig. 1E).

The table view supports uploading a CSV data file for dashboard generation. After uploading, the columns and their types are shown in a list. Users can view the raw data in a table visualization by clicking the view button. Meanwhile, the agent network automatically explores and generates dashboards directly. Specifically, an agent is allowed to explore at most 50 steps for a dashboard in case of no early termination (e.g., the number of charts reaching the maximum limit). We assign the agents a quota of n steps ($n = 1000$), and the agents can generate a series of dashboards with different topics (i.e., key columns). The dashboards with different topics will be displayed in the topic view sorted by returns. Users can drill down to investigate each dashboard. When hovering on another result, a tooltip will pop up demonstrating the changes on charts and insights.

The canvas view displays the charts generated by the deep reinforcement model. Displaying multiple views of charts is a critical challenge for visual analysis and has been studied for a long [14]. Given that chart layout is not the research focus of this work, we use a rule-based method to provide an efficient solution. Specifically, we first aggregate the charts by their mark type. For the charts with the same mark types, we aggregate the ones with the same insight types. From the preliminary study, we understand that text visualizations are commonly used to show an overview of the data. Therefore, we summarize the statistics of key columns and other columns that are possibly interesting and visualize the statistics by a default setting of text visualizations. The text visualizations will be positioned in the top row of the dashboard. The chart sizes and positions are also changeable.

With chart editor, users can edit the dashboards with online recommendations. Users are allowed to edit chart configurations by editing the parameter values, adding a new chart, or deleting charts. Once edited, the agents will start exploring from the current dashboard state for k steps (k=200). The systems would identify the best charts that could be added to the dashboard and maximize rewards. The recommended charts will be shown in the recommendation view. Users can explore and add the ones of interest to the dashboard.

6 EVALUATION

To demonstrate the usefulness of DashBot, we first show three example dashboards created with DashBot (Fig. 1, Fig. 7). Second, we focus on
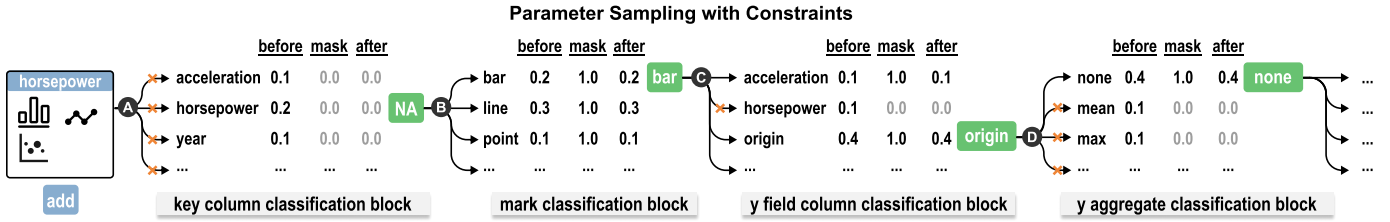
Fig. 5. An example of constrained sampling for configuring a new chart. Masks are applied in the classification blocks to avoid infeasible parameter selections for key columns (A), mark types (B), y field (C), and y aggregate (D).
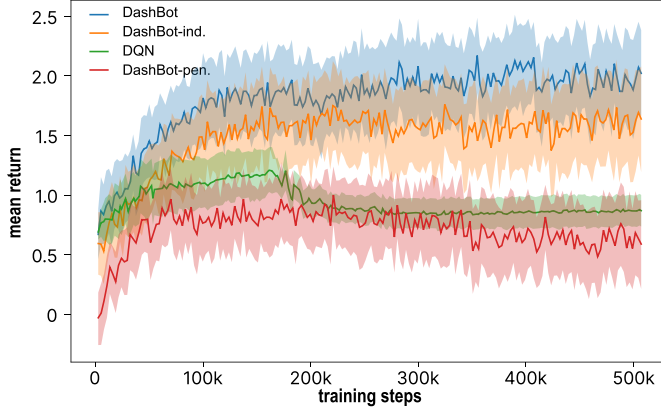

Fig. 6. Ablation study showing the mean returns of DashBot, DashBot-ind., DashBot-pen., and DQN across training steps.

evaluating the core component of DashBot, i.e., the deep reinforcement learning model that defines the dashboard design space, enumerates that space, and ranks the candidate designs [80], through an ablation study and a user study.

6.1 Example Dashboards

We first show how to exploratory generate a dashboard through the first example (Fig. 1). Then we show two automatically generated examples (Fig. 7) with style redesigned on the exported SVG files using Figma.

In the first example (Fig. 1), assume that Mary wants to investigate the insights into Seattle weather. After uploading the data file, the data columns and types are shown (Fig. 1A). Then a series of dashboards are generated by DashBot and listed in the topic list (Fig. 1B). Mary is more interested in the topic of "wind" in Seattle. Therefore, she selects the dashboard of "wind" with the highest return value (Fig. 1D). From Chart 1, she discovers that the minimum temperature seems to have no obvious correlation with the wind, but there is more red points on the right side. She infers that the days with the highest winds are mostly rainy. Chart 2 shows wind distribution by different weathers. The proportion of rainy days gets larger as the wind gets larger. This pattern partially conforms with the inference in Chart 1. Chart 3 provides more evidence with a boxplot visualization. The median wind on rainy days is higher than on drizzling, foggy and sunny days, but lower than on snowy days. Furthermore, the points on the rightmost indicate that days with the largest wind are all rainy. Chart 4 presents the wind change across dates. Mary further feels interested in the dates with the highest winds and thus configures a new chart, Chart 5, with chart editor (Fig. 1C). After the edition, DashBot automatically recommends Chart 6 about the days of lowest wind for comparison.

The second example (Fig. 7A) is about the horsepower in the "cars" dataset. Chart 1 is a boxplot showing horsepower distribution by cylinders. Charts 2-5 indicate that miles per gallon and accelerations are possibly negatively correlated with horsepower while weights and displacements tend to be positively correlated with horsepower. Chart 6 further gives an overview of horsepower with regard to years.

The third (Fig. 7B) is about US gross in the "movies" dataset. Chart 1 & 2 demonstrate the mean US gross by major genres and MPAA ratings. Chart 3 lists the directors with the highest US gross. Chart 4

& 5 show the US gross by IMDB ratings and Rotten Tomato ratings, which facilitate the comparison of different rating systems. Chart 6 presents the correlation between US gross and production budget.

6.2 Ablation Study

We opt to demonstrate the effectiveness of our model design, including the A3C framework, sequence modeling, and constrained sampling. We design experiments with three baseline models for the ablation study. The experiments were carried out on a workstation with 6 core Intel i7-8700K CPU, a GeForce RTX 1080Ti GPU, and 32GB memory. All models were trained for 500,000 steps with 27 datasets from Vega Datasets[1]. As suggested by Minh et al. [38], we train all models for 10 runs and average the return values. The results are shown in Fig. 6, where transparent bands represent standard variances of the return.

A3C versus Deep Q-Network. To demonstrate the effectiveness of the A3C training framework for the task of dashboard generation, we design a multi-step deep Q-Network (i.e., DQN) as a baseline. The DQN uses a similar agent network as DashBot. The DQN is trained with a temporal difference and replay memory [39]. From Fig. 6, we discover that DQN fails to achieve a stable increase in the return.

Sequence Model versus Independent Multi-Classification. We design a baseline model (i.e., DashBot-ind.) by removing the embedding concatenation in classification blocks. All classification blocks independently predict parameter values with the shared embedding. To ensure a fair comparison, constrained sampling is preserved to avoid invalid configurations. From the results (Fig. 6), we can see the return of DashBot-ind. constantly increases but finally converges at a mean return value lower than DashBot. The DashBot also converges faster with smaller training steps. The reason might be that the sequence model is able to learn mutual relations between consecutive tokens.

Constrained Sampling versus Invalidity Penalty. We design a baseline model without constrained sampling (i.e., DashBot-pen.). Alternatively, we add penalty rewards. Specifically, if an agent configures an invalid chart, it will receive a negative reward. The penalty rewards result in a relatively low return at the beginning because of the random sampling of invalid configurations. As the training proceeds, the agents obtain increasing returns, but finally, the return gets decreased. The penalty mechanism seemingly works, but the balancing between penalties and other rewards might be a critical problem.

In addition, we run the DashBot for 1,000 steps (about 20 episodes) on each dataset, which takes 15.4 (SD=1.52) seconds on average, and compute the statistics of generated dashboards. On average, the dashboards contain 5.42 (SD = 0.83) charts with 2.81 (SD = 0.74) chart types, which are consistent with the statistics in the preliminary study.

6.3 Comparative Experiment

We further conduct a user study to evaluate the usefulness of our deep reinforcement learning-based method in comparison with MultiVision [75], another deep learning-based method. Two systems have different interface designs and implementation details, but the machine learning models have the same inputs and outputs. The study focuses on evaluating the core machine learning components of the systems.

Participants. We conducted the study with 10 data workers (P1-P10, 4 females and 6 males) with diverse backgrounds, including statistics, chemistry, biology, environment engineering, transportation management, computer science, finance, and economics. All participants

---

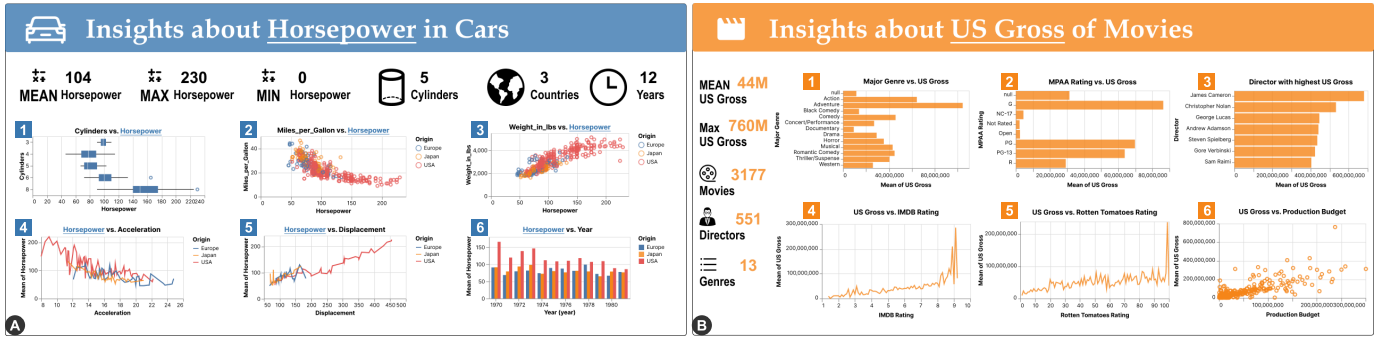[1] https://github.com/vega/vega-datasets

Fig. 7. Two example Dashboards created with DashBot: (A) a dashboard showing the properties of cars, such as cylinders, miles per gallon, and accelerations, about horsepower and (B) a dashboard about US gross of the movies with regard to IMDB ratings, Rotten Tomato ratings, genres, etc.

reported having more than two years of using programming languages (i.e., Python, R, MatLab, and Javascript) or interactive tools (i.e., Stata, SPSS, Origin, and Excel) to process and visualize data for analysis. All participants mentioned that they only had a basic knowledge of creating charts for analysis or presentation.

Data. To ensure fairness, we use the same datasets in MultiVision from Vega Datasets, which include data about cars, jobs, penguins, Seattle weather, and movies. For each dataset, two systems generate one dashboard. Given that DashBot can generate multiple dashboards with different topics, we select the one with the highest return values.

Experiment Setup. We design a comparative experiment to evaluate the initial results generated by two systems because the two systems have provided different interaction designs for the dashboard edition and human inputs are hard to evaluate. To ensure the fairness of comparison, the results generated by the two methods are all rendered with DashBot interface, all facilitated with text visualizations for dataset overview. This makes sure the same styles of charts and interfaces. Please note that the dashboards generated with the original MultiVision interface are facilitated with interactivity (i.e., cross-filtering), but the interactivity is not considered in this study because we focus on the model comparison. In the study, users could explore two dashboards freely without knowing how the dashboards are generated. We counterbalanced the presentation order in order to alleviate the carryover effect. We followed the think-aloud protocol to collect feedback about the generated dashboards. Specifically, during exploration, participants were requested to report insights she/he gains from the dashboards. After reading the dashboards of each dataset, the participants were asked to select the better one with regard to overall quality, insightfulness, understandability, and aesthetics. The experiment was followed with a post-study interview to understand participants' routine analysis workflow and suggestions about the dashboards.

Procedure. A study lasted for about 50 minutes with a 10-minute training session, a 30-minute experiment, and a 10-minute post-study interview. Before the study, participants were asked to sign a consent form agreeing to record their feedback and comments for analysis. All participants were paid with $20 after the study.

Results. In total, we collected 200 ratings from 10 participants. As shown in Fig. 8, 78% of ratings agree that dashboards generated by DashBot have higher overall quality when compared with the baseline method. We collected a large number of positive comments about our method and summarized the comments below.

Among the responses, 84% told that DashBot is more understandable. Participants liked our idea of key columns. P8 quickly identified that the charts in our generated dashboard share a common column, saying that "*the design accelerates the understanding of the dashboard because I only need to identify the columns combined with that (key) column*" (C1). Due to the constrained sampling to avoid ineffective encodings, the dashboards generated by DashBot might be easier to understand. P6 appreciated the use of color encodings on the nominal columns with an adequate cardinality and commented that "*the color is efficient for identifying different penguin species*" (C2).

Moreover, 76% of voters agreed that DashBot is more aesthetically pleasing. Participants liked the diversity of chart types in the dashboards generated by DashBot. P7, who had a background in digital

media design, commented that "*the first one (our method) looks good because of diverse chart types, while the second one contains too many scatterplots*" (C3). P6 also held a similar opinion. Reasonable visual encodings also improve the overall appearance. P4, who reported to have little experience in design, raised a comment that "*this dashboard looks more comfortable at my first glance because it does not use size to represent the column of Miles_per_Gallon (in the scatterplot)*" (C4). P1 liked the aggregation of columns, which benefits from the design of an agent network that covers different chart parameters, saying that "*averaging the values of this column makes the chart clear*" (C5).

More participants (88%) thought that DashBot can provide more insights compared with the baseline. P6 commented that "*the second one (our method) provides more column combinations other than the other one, which is more informative*" (C6). P5 liked the patterns exhibited in the dashboards we generated, saying that "*I can easily identify the correlations from the shape of scatterplots*" (C7). P2 inferred additional column relationships from the charts, commenting that "*these two scatterplots (with high correlation values) potentially reveal the relations between another two columns, although they are not visualized in the same charts*" (C8). P4 showed her preference for the comparative analysis on the top and bottom entries, with a comment that "*if I were a climate analyst, showing me the dates with highest and lowest temperatures will be meaningful*" (C9).

Analysis. We summarized participants' feedback and analyzed the technical differences that theoretically make DashBot outperforms MultiVision. First, numerically modeling insights helps to reveal important information about the data. A major difference between DashBot and MultiVision is that we evaluate the insightfulness of the charts with computational metrics. Thus, the recommended charts have data patterns that are more salient and easier to comprehend (C7, C8). Second, incorporating dashboard design patterns into the generation is another advantage of DashBot. For example, we require the charts in a dashboard to have a shared column as the topic of the dashboard, which makes the results more coherent and interconnected (C1). MultiVision incorporates criteria such as diversity and simplicity into the dashboard design. DashBot follows a similar approach, but further considers statistics obtained from empirical surveys to follow "common practices" (C3, C6). Third, constrained sampling help exclude charts with ineffective visual encodings. MultiVision recommends chart encodings by learning from the Excel datasets, which might result in some unreasonable recommendations. For example, MultiVision tends to visualize three data columns in a chart and encodes numerical data with point size, which is complained about by the participant (C4). Instead, DashBot takes visual effectiveness into the design of constrained sampling for better visual encodings and data transformation (C2, C5).

7 DISCUSSION AND LIMITATIONS

In this section, we discuss our limitations and potential improvements.

7.1 Extending Reward Function Sets

We design reward functions based on a preliminary study of dashboard designs, but there are additional considerations to extend the reward function set. For presentation, we only consider the diversity and parsimony between charts. Chart effectiveness is also an important aspect
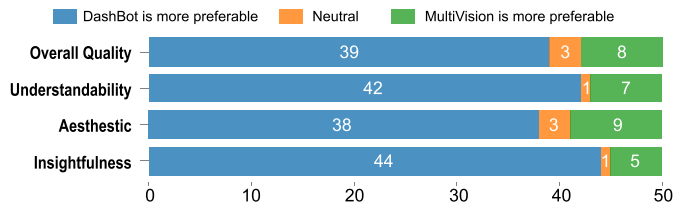
Fig. 8. Ratings of DashBot compared with MultiVision from the perspectives of overall quality, understandability, aesthetic, and insightfulness.

of chart comprehension [13, 32]. A potential solution is introducing Draco knowledge base [40], a comprehensive rule set of visualization design knowledge, to compute the effectiveness scores of charts and integrate them into reward functions. Moreover, the effectiveness of chart composition [15, 24] should be considered. For example, even with a large number of subplots, scatterplot matrices are still easy to understand. Thus, continued research on quantifying the quality of chart composition is needed.

For insights, we use several metrics (e.g. correlation) to measure the insightfulness of a chart based on existing studies [53, 58]. However, our current reward functions do not consider semantic information in the columns, which might lead to results of commonsense [25]. For example, the total sales of a merchant in a year are certainly correlated to the average sales. A potential fix for this problem is to process the header names for their semantic structure [18] for reward function design (e.g., normalizing the correlation with semantic distance). The introduction of semantic information is potentially helpful for addressing domain problems. For example, when exploring the dashboards of the penguin dataset, P10, who has a background in biology, commented that comparing statistics between different species would be extremely important, and using y positions might be more intuitive than colors. The environment should reward the agent if a chart visualizes a topic-related column with an effective visual channel.

Additional metrics could be incorporated into reward functions for better insight discovery. For example, as suggested by participants with a statistics background (P3 and P9), Shapiro-Wilk static, skewness, and kurtosis can be used to evaluate the normality of the data distribution. Visual patterns are another critical consideration for insight discovery by human experts. Computer vision [35] and other sophisticated algorithms [9, 29] could be used to measure the visual salience of the patterns, such as outliers and visual grouping [20, 62].

7.2 Evaluating DashBot in Real-World Scenarios

It lacks comparisons between DashBot and what users might do with real-world tasks. User performance in different recommendation systems is a critical aspect of evaluation [61, 80]. However, directly comparing two complex systems is difficult in controlling dependent variables (e.g., workflows, interface designs, and artifacts) [28]. In our case, DashBot has a different workflow and interface design. DashBot recommends a default setting and allows users to adjust the dashboards to their satisfaction, while real-world tools (e.g., Tableau and Power BI) provide efficient interactions to support generating dashboards from scratch. Moreover, different dashboard styles might introduce biases in interpreting the insightfulness and understandability of the results. DashBot is a proof-of-concept system that only covers the generation of dashboards with limited chart types and layouts and lacks support in style customization. In contrast, real-world authoring tools support the generation of sophisticated dashboards even with advanced pictograms. Controlled studies of multiple recommendation systems could be conducted with a consistent interface to evaluate user performance [80]. Future evaluation could consider integrating the DashBot model into existing tools and compare how the resulting dashboards are different under different workflows.

Real-world tasks also differ from lab studies in terms of datasets. We use Vega datasets throughout the paper, given that the datasets are commonly used in visualization generation [41, 71] and the area of data analysis, which is understandable for readers and study participants. It is also convenient for model comparison because the baseline model was trained on Vega datasets. However, real-world datasets could

contain hundreds or thousands of data columns, which challenges the scalability of models. Moreover, additional table operations (e.g., reshape and rearrange) should be considered for data cleansing and insight discovery [26]. Therefore, it would be a total different problem for feature engineering, model design, and the overall framework design (e.g., action space) when dealing with real-world datasets. Future study could focus on developing efficient agents to explore and identify insights from large-scale datasets.

7.3 Considering Additional Data Transformation

DashBot currently considers basic data aggregation calculators (e.g., mean and sum) and sorting for identifying the top and bottom values. P9 suggested encoding multiple data series in the same chart by introducing visualization compositions such as layering and faceting. Given that the visualization rendering of DashBot is achieved using Vega-Lite, these potential operations can be extended with additional modeling of Vega-Lite parameter space. Given that we primarily focus on specifying visual encodings, some chart appearances are not aesthetically pleasing. For example, the line chart of Fig. 1-D4 is too dense for investigation, requiring binning or sampling. Towards the generation of delicate charts, it would be a promising research direction to encode the knowledge of visualization debugging [13] and layout optimization [73] into the deep reinforcement learning framework.

7.4 Enhancing Dashboards with Interactivity

DashBot focuses on modeling data insights and chart configurations in dashboards but does not support interactivity, which limits the usability of the generated dashboards [46]. Bridging the charts with interactivity requires the modeling of data flow between charts. MultiVision [75] supports cross-chart filtering by detecting the data records related to user's filtering operations on a single chart. This solution is efficient when all charts share the same datasheet. However, there might be more complex situations. For example, the data of a chart might be transformed from the data of another chart. To handle this situation, the data flow model should be constructed when the exploration and data transformation are executed, as demonstrated by VisFlow [79]. Such kind of progressive exploration could be integrated into the framework of DashBot, where agents could choose to explore from the data of the previous chart or use the original table when generating a new chart. Reward functions that model the relationship between the consecutive charts should be considered [27].

8 CONCLUSION AND FUTURE WORK

We present DashBot, a deep reinforcement learning model, to generate analytical dashboards for insight discovery. To design an effective model, we conduct a preliminary study to understand the design practices in existing dashboard galleries. Then we formulate the problem of exploratory dashboard generation as a Markov decision process with appropriate action spaces and reward functions. Furthermore, we design a sequence generation network for the action selection and parameter configuration. Finally, we demonstrate the effectiveness of our model design through an ablation study and a user study. Our study opens up a research direction that facilitates visualization generation without the need for large-scale human-labeling training data, which is a pain point for learning-based visualization generation methods.

Further research can be conducted in two directions. First, we can extend the framework for the generation of more visualization genres, such as glyph [77, 78] and data stories [54], considering additional visualization types and data transformation operations. Moreover, the data flow could be considered to improve the interactivity. Second, we can incorporate the visual patterns into the modeling of insight discovery [35], given that visual patterns are unique values that visual analytics can provide for expert users. Third, inspired by the idea of inverse reinforcement learning [42], we can further incorporate user data to steer the generation of dashboards on the basis of visualization rules. User inputs can guide the agents to adapt to new analysis scenarios.

REFERENCES

[1] Microsoft Power BI Data Stories Gallery. `https://community.powerbi.com/t5/Data-Stories-Gallery/bd-p/DataStoriesGallery`. Accessed at Feb. 26, 2022.

[2] Tableau Viz Gallery. `https://public.tableau.com/en-gb/s/viz-gallery`. Accessed at Feb. 26, 2022.

[3] H. M. Al-maneea and J. C. Roberts. Towards quantifying multiple view layouts in visualisation as seen from research publications. In *Proceedings of IEEE Visualization Conference*, pp. 121–121, 2019.

[4] N. Andrienko, G. Andrienko, S. Miksch, H. Schumann, and S. Wrobel. A theoretical model for pattern discovery in visual analytics. *Visual Informatics*, 5(1):23–42, 2021.

[5] B. Bach, E. Freeman, A. Abdul-Rahman, C. Turkay, S. Khan, Y. Fan, and M. Chen. Dashboard design patterns. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[6] H. K. Bako, A. Varma, A. Faboro, M. Haider, F. Nerrise, J. P. Dickerson, and L. Battle. User-driven programming support for rapid visualization authoring in d3. *arXiv preprint arXiv:2112.03179*, 2021.

[7] O. Bar El, T. Milo, and A. Somech. Automatically generating data exploration sessions using deep reinforcement learning. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 1527–1537, 2020.

[8] L. Battle and J. Heer. Characterizing exploratory visual analysis: A literature review and evaluation of analytic provenance in tableau. *Computer Graphics Forum*, 38(3):145–159, 2019.

[9] M. Behrisch, R. Krueger, F. Lekschas, T. Schreck, N. Gehlenborg, and H. Pfister. Visual pattern-driven exploration of big data. In *Proceedings of International Symposium on Big Data Visual and Immersive Analytics*, pp. 1–11, 2018.

[10] R. Bellman. A Markovian decision process. *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[11] J. Bertin. Semiology of graphics: Diagrams, networks, maps. Technical report, 1983.

[12] M. A. Borkin, A. A. Vo, Z. Bylinskii, P. Isola, S. Sunkavalli, A. Oliva, and H. Pfister. What makes a visualization memorable? *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2306–2315, 2013.

[13] Q. Chen, F. Sun, X. Xu, Z. Chen, J. Wang, and N. Cao. VizLinter: A linter and fixer framework for data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):206–216, 2022.

[14] X. Chen, W. Zeng, Y. Lin, H. M. AI-maneea, J. Roberts, and R. Chang. Composition and configuration patterns in multiple-view visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1514–1524, 2021.

[15] D. Deng, W. Cui, X. Meng, M. Xu, Y. Liao, H. Zhang, and Y. Wu. Revisiting the design patterns of composite visualizations. *arXiv preprint arXiv:2203.10476*, 2022.

[16] D. Deng, Y. Wu, X. Shu, J. Wu, S. Fu, W. Cui, and Y. Wu. VisImages: A fine-grained expert-annotated visualization dataset. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[17] V. Dibia and Ç. Demiralp. Data2Vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE Computer Graphics and Applications*, 39(5):33–46, 2019.

[18] L. Du, F. Gao, X. Chen, R. Jia, J. Wang, J. Zhang, S. Han, and D. Zhang. TabularNet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 322–331, 2021.

[19] C. Edmond and T. Bednarz. Three trajectories for narrative visualisation. *Visual Informatics*, 5(2):26–40, 2021.

[20] L. Giovannangeli, R. Bourqui, R. Giot, and D. Auber. Color and shape efficiency for outlier detection from automated to user evaluation. *Visual Informatics*, 2022.

[21] K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo. VizML: A machine learning approach to visualization recommendation. In *Proceedings of CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2019.

[22] K. Hu, S. Gaikwad, M. Hulsebos, M. A. Bakker, E. Zgraggen, C. Hidalgo, T. Kraska, G. Li, A. Satyanarayan, and Ç. Demiralp. VizNet: Towards a large-scale visualization learning and benchmarking repository. In *Proceedings of CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2019.

[23] Z. Huang, W. Xu, and K. Yu. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[24] W. Javed and N. Elmqvist. Exploring the design space of composite visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pp. 1–8, 2012.

[25] B. Karer, H. Hagen, and D. J. Lehmann. Insight beyond numbers: The impact of qualitative factors on visual data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1011–1021, 2021.

[26] S. Kasica, C. Berret, and T. Munzner. Table Scraps: An actionable framework for multi-table data wrangling from an artifact study of computational journalism. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):957–966, 2021.

[27] Y. Kim, K. Wongsuphasawat, J. Hullman, and J. Heer. GraphScape: A model for automated reasoning about visualization similarity and sequencing. In *Proceedings of CHI Conference on Human Factors in Computing Systems*, pp. 2628–2638, 2017.

[28] J. Lazar, J. H. Feng, and H. Hochheiser. *Research methods in human-computer interaction*. Morgan Kaufmann, 2017.

[29] H. Li, Y. Wang, A. Wu, H. Wei, and H. Qu. Structure-aware visualization retrieval. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2022.

[30] H. Li, Y. Wang, S. Zhang, Y. Song, and H. Qu. KG4Vis: A knowledge graph-based approach for visualization recommendation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):195–205, 2021.

[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[32] L. Y.-H. Lo, A. Gupta, K. Shigyo, A. Wu, E. Bertini, and H. Qu. Misinformed by visualization: What do we learn from misinformative visualizations? *arXiv preprint arXiv:2204.09548*, 2022.

[33] Y. Luo, X. Qin, N. Tang, and G. Li. DeepEye: Towards automatic data visualization. In *Proceedings of IEEE International Conference on Data Engineering*, pp. 101–112. IEEE, 2018.

[34] Y. Luo, N. Tang, G. Li, J. Tang, C. Chai, and X. Qin. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):217–226, 2021.

[35] 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*, 26(3):1562–1576, 2020.

[36] J. Mackinlay. Automating the design of graphical presentations of relational information. *Acm Transactions On Graphics*, 5(2):110–141, 1986.

[37] J. Mackinlay, P. Hanrahan, and C. Stolte. Show Me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, 2007.

[38] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[40] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing visualization design knowledge as constraints: Actionable and extensible models in Draco. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):438–448, 2018.

[41] A. Narechania, A. Srinivasan, and J. Stasko. NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379, 2021.

[42] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of International Conference on Machine Learning*, pp. 663–670, 2000.

[43] X. Qian, R. A. Rossi, F. Du, S. Kim, E. Koh, S. Malik, T. Y. Lee, and J. Chan. ML-based visualization recommendation: Learning to recommend visualizations from data. *arXiv preprint arXiv:2009.12316*, 2020.

[44] Z. Qu and J. Hullman. Keeping multiple views consistent: Constraints, validations, and exceptions in visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):468–477, 2017.

[45] B. Saket, D. Moritz, H. Lin, V. Dibia, C. Demiralp, and J. Heer. Beyond heuristics: Learning visualization design. *arXiv preprint arXiv:1807.06641*, 2018.

[46] A. Sarikaya, M. Correll, L. Bartram, M. Tory, and D. Fisher. What do we talk about when we talk about dashboards? *IEEE Transactions on Visualization and Computer Graphics*, 25(1):682–692, 2019.

[47] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite:

A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2017.

[48] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of International conference on machine learning*, pp. 1889–1897, 2015.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[50] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, 2010.

[51] L. Shao, Z. Chu, X. Chen, Y. Lin, and W. Zeng. Modeling layout design for multiple-view visualization via bayesian inference. *Journal of Visualization*, 24(6):1237–1252, 2021.

[52] D. Shi, Y. Shi, X. Xu, N. Chen, S. Fu, H. Wu, and N. Cao. Task-oriented optimal sequencing of visualization charts. In *2019 IEEE Visualization in Data Science (VDS)*, pp. 58–66. IEEE, 2019.

[53] D. Shi, X. Xu, F. Sun, Y. Shi, and N. Cao. Calliope: Automatic visual data story generation from a spreadsheet. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):453–463, 2020.

[54] X. Shu, A. Wu, J. Tang, B. Bach, Y. Wu, and H. Qu. What makes a data-GIF understandable? *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1492–1502, 2020.

[55] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[56] A. Srinivasan, S. M. Drucker, A. Endert, and J. Stasko. Augmenting visualizations with interactive data facts to facilitate interpretation and communication. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):672–681, 2018.

[57] Y. Sun, J. Li, S. Chen, G. Andrienko, N. Andrienko, and K. Zhang. A learning-based approach for efficient visualization construction. *Visual Informatics*, 6(1):14–25, 2022.

[58] 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*, pp. 1509–1524, 2017.

[59] T. Tang, R. Li, X. Wu, S. Liu, J. Knittel, S. Koch, L. Yu, P. Ren, T. Ertl, and Y. Wu. PlotThread: Creating expressive storyline visualizations using reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):294–303, 2020.

[60] T. Tang, S. Rubab, J. Lai, W. Cui, L. Yu, and Y. Wu. iStoryline: Effective convergence to hand-drawn storylines. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):769–778, 2018.

[61] J. J. van Wijk. Evaluation: A challenge for visual analytics. *Computer*, 46(7):56–60, 2013.

[62] J. Wang, X. Cai, J. Su, Y. Liao, and Y. Wu. What makes a scatterplot hard to comprehend: data size and pattern salience matter. *Journal of Visualization*, 25(1):59–75, 2022.

[63] X. Wang, F. Cheng, Y. Wang, K. Xu, J. Long, H. Lu, and H. Qu. Interactive data analysis with next-step natural language query recommendation. *arXiv preprint arXiv:2201.04868*, 2022.

[64] Y. Wang, Z. Sun, H. Zhang, W. Cui, K. Xu, X. Ma, and D. Zhang. DataShot: Automatic generation of fact sheets from tabular data. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):895–905, 2019.

[65] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pp. 110–119, 2000.

[66] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[67] Y. Wei, H. Mei, W. Huang, X. Wu, M. Xu, and W. Chen. An evolutional model for operation-driven visualization design. *Journal of Visualization*, 25(1):95–110, 2022.

[68] Wikipedia contributors. Decision tree learning, 2022. [Online; accessed 16-June-2022].

[69] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):649–658, 2015.

[70] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Towards a general-purpose query language for visualization recommendation. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pp. 1–6, 2016.

[71] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of CHI Conference on Human Factors in Computing Systems*, pp. 2648–2659, 2017.

[72] A. Wu, D. Deng, F. Cheng, Y. Wu, S. Liu, and H. Qu. In defence of visual analytics systems: Replies to critics. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[73] A. Wu, W. Tong, T. Dwyer, B. Lee, P. Isenberg, and H. Qu. MobileVisFixer: Tailoring web visualizations for mobile phones leveraging an explainable reinforcement learning framework. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):464–474, 2021.

[74] A. Wu, Y. Wang, X. Shu, D. Moritz, W. Cui, H. Zhang, D. Zhang, and H. Qu. AI4VIS: Survey on artificial intelligence approaches for data visualization. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[75] A. Wu, Y. Wang, M. Zhou, X. He, H. Zhang, H. Qu, and D. Zhang. MultiVision: Designing analytical dashboards with deep learning based recommendation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):162–172, 2021.

[76] A. Wu, L. Xie, B. Lee, Y. Wang, W. Cui, and H. Qu. Learning to automate chart layout configurations using crowdsourced paired comparison. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2021.

[77] L. Ying, T. Tangl, Y. Luo, L. Shen, X. Xie, L. Yu, and Y. Wu. GlyphCreator: Towards example-based automatic generation of circular glyphs. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):400–410, 2021.

[78] L. Ying, Y. Yang, X. Shu, D. Deng, T. Tang, L. Yu, and Y. Wu. MetaGlyph: Automatic generation of metaphoric glyph-based visualization. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[79] B. Yu and C. T. Silva. VisFlow: Web-based visualization framework for tabular data with a subset flow model. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):251–260, 2016.

[80] Z. Zeng, P. Moh, F. Du, J. Hoffswell, T. Y. Lee, S. Malik, E. Koh, and L. Battle. An evaluation-focused framework for visualization recommendation algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):346–356, 2021.

[81] J. Zhao, S. Xu, S. Chandrasegaran, C. J. Bryan, F. Du, A. Mishra, X. Qian, Y. Li, and K.-L. Ma. ChartStory: Automated partitioning, layout, and captioning of charts into comic-style narratives. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[82] M. Zhou, Q. Li, Y. Li, X. He, Y. Liu, S. Han, and D. Zhang. Table2Charts: Learning shared representations for recommending charts on multi-dimensional data. *arXiv preprint arXiv:2008.11015*, 2020.

[83] H. Zhu, M. Zhu, Y. Feng, D. Cai, Y. Hu, S. Wu, X. Wu, and W. Chen. Visualizing large-scale high-dimensional data via hierarchical embedding of knn graphs. *Visual Informatics*, 5(2):51–59, 2021.