PlotThread: Creating Expressive Storyline Visualizations using Reinforcement Learning

Tan Tang, Renzhong Li, Xinke Wu, Shuhan Liu, Johannes Knittel, Steffen Koch, Thomas Ertl, Lingyun Yu, Peiran Ren, and Yingcai Wu

Abstract— Storyline visualizations are an effective means to present the evolution of plots and reveal the scenic interactions among characters. However, the design of storyline visualizations is a difficult task as users need to balance between aesthetic goals and narrative constraints. Despite that the optimization-based methods have been improved significantly in terms of producing aesthetic and legible layouts, the existing (semi-) automatic methods are still limited regarding 1) efficient exploration of the storyline design space and 2) flexible customization of storyline layouts. In this work, we propose a reinforcement learning framework to train an AI agent that assists users in exploring the design space efficiently and generating well-optimized storylines. Based on the framework, we introduce PlotThread, an authoring tool that integrates a set of flexible interactions to support easy customization of storyline visualizations. To seamlessly integrate the AI agent into the authoring process, we employ a mixed-initiative approach where both the agent and designers work on the same canvas to boost the collaborative design of storylines. We evaluate the reinforcement learning model through qualitative and quantitative experiments and demonstrate the usage of PlotThread using a collection of use cases.

Index Terms—Storyline visualization, reinforcement learning, mixed-initiative design

1 INTRODUCTION

Storyline visualizations have gained wide popularity in presenting complex entity relationships. The ability to create visual narratives makes it applicable in presenting fictions, analyzing dynamic networks, recalling meeting content, and understanding.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.

Digital Object Identifier: xx.xxx/TVCG.201x.xxxxxx
ing software evolutions [31]. However, designing storyline visualizations has long been considered as a difficult and tedious task which involves balancing the trade-off between narrative constraints [45] and aesthetic goals [25]. To illustrate the evolutions of entity relationships, two primary narrative constraints [45] should be followed:

- C1: The lines that represent characters who appear in the same scene should be grouped.
- C2: Otherwise, the grouped lines should be divided.

Inferred by graph layouts [32], it is necessary to minimize line crossings and deviations to avoid dense visual clutter. Thus, three aesthetic goals [43] are proposed to create legible layouts:

- G1: reducing line crossings
- G2: reducing line wiggles
- G3: reducing white space

To ease the difficulty of designing storyline visualizations, previous studies [42-44] have developed optimization-based methods that produce storylines according to the design factors mentioned above. However, these methods mainly focus on producing aesthetic and legible layouts without considering the whole design space of storylines. With limited design choices, the storylines generated by the optimization models cannot cover the full diverse narrative elements compared to the manually-created ones [45]. For example, the hand-drawn storylines [28-45] adopt various layouts to indicate different plots.

To support the design of expressive storylines from the narrative aspect, researchers [45] developed iStoryline that incorporates human design knowledge and creativity into the optimization models. Specifically, iStoryline formulates user interactions as mathematical constraints to control the optimization model [25] so that users can focus on constructing storyline layouts that conform to their understandings of the stories. However, the interactions proposed in iStoryline mainly concentrate on modifying the local regions, which makes it time-consuming and labor-intensive to customize the overall layouts. For example, users may need to take a considerable number of actions to refine storyline layouts, which hinders the efficient exploration of the design space. Besides, the unpredictability of the optimization process may give rise to unexpected results, which requires trial-and-error practices to obtain the desired storylines.

To facilitate the easy design of storyline layouts, we envision whether a human-AI (Artificial Intelligence) collaborative approach can be helpful. Specifically, we intend to employ machine learning to develop an intelligent agent. Similar to a recommendation engine, the agent can reduce human efforts by providing users possible suggestions of compelling storylines that follow the aesthetic goals (G1 to G3). However, we are not aware of any prior work on designing storylines using machine learning, which raises two major challenges:

**Model Architecture** Storylines depict the temporal relationships among entities that are inherently different from Euclidean data (e.g., images, videos, and texts) that can be processed by existing machine learning applications [18, 46]. Thus, it remains unclear whether storylines can be generated using machine learning or how to extend the existing models to deal with storylines. Recent studies [18, 46] have adapted neural networks for graph drawings, but they mainly focus on the topological structure of graph layouts. While storylines and graphs pursue some common aesthetic goals (e.g., minimizing crossings [10]), storylines require a higher aesthetic standard for legible layouts. Moreover, it is also necessary to develop a novel learning framework that takes narrative constraints into considerations for the storyline generation problem.

**Model Training** Training a machine learning model requires an appropriate loss function and a high-quality benchmark dataset [12]. In image classification, for instance, the loss function can be easily defined as counting incorrect labels while the training data can be obtained by labeling real-world images [13]. However, the training of the storyline model becomes more complicated than typical machine learning tasks. First, it is challenging to define “correct” layouts in terms of the different narratives since designers usually have different understandings about the stories. Thus, it is difficult to identify a unique loss function for the storyline generation problem. Second, there are not enough storyline visualizations available to train a machine learning model, even though previous work [45] extended the collection of hand-drawn storylines.

In this work, we propose a novel reinforcement learning framework that trains an AI agent to design storylines “like” human designers. To support the collaborative design, the agent should follow two principles:

- **D1** Storylines generated by agents should resemble the ones on which users are currently working to preserve their mental map.
- **D2** The agent should share the same action space as human users so that they can work on the same canvas collaboratively.

Thus, the goal of the AI agent is to imitate and improve users’ intermediate results instead of generating storylines from scratch. To achieve this goal, the agent should be capable of decomposing a given storyline into a sequence of actions, understanding the state of intermediate layouts, and have a foresightful plan for future actions. Therefore, we employ Reinforcement Learning (RL) to solve the challenges. Specifically, we define the states as the intermediate storyline layouts and define the actions of the agent as the same interactions implemented by iStoryline due to its success in producing diverse storylines that conform to different narratives. We further define loss function by maximizing the accumulative rewards that are vital for training RL models. To obtain sufficient training data, we follow the common practices [18, 46] that generate well-optimized storylines with diverse visual layouts using the existing optimization approach [25].

As a proof of concept, we implement PlotThread that integrates the agent into the authoring process of storyline visualizations. We extend the interaction set of iStoryline to support a more flexible design of storylines and foster close collaboration between the agent and designers. We present the usage of PlotThread through a set of use cases (see Fig. 1) and validate its usability via expert interviews.

The main contributions are summarized as follows:

- We propose a novel reinforcement learning framework and generate a collection of high-quality storylines to train an agent that supports the collaborative design of storylines with designers.
- We develop PlotThread, a mixed-initiative system that facilitates the easy creation of expressive storyline visualizations, and demonstrates its usage through a set of use cases.

## 2 Related Work

We summarize critical techniques used in producing storyline visualizations and the state-of-the-art reinforcement learning techniques.

### 2.1 Storyline Visualization

Storyline visualizations have become prevalent in revealing the evolution of stories [25] and presenting various narrative elements [45]. To ease the difficulties in designing storyline layouts, researchers have proposed many (semi-) automatic approaches [12, 43, 44] that achieve the trade-off between aesthetic goals and narrative constraints using optimization models. Ogawa and Ma [31] firstly proposed an automatic approach for generating storyline visualizations but their algorithm failed to produce aesthetic layouts due to the ignorance of the heuristic criteria. Tanahashi and Ma [44] suggested a more comprehensive set of design considerations for storyline visualizations and proposed a layout generation approach based on genetic algorithms. Despite the success of producing relatively aesthetically-appealing and legible storyline layouts, their technique is inefficient to support interactive editing of storyline visualizations. For a better performance in both efficiency and the overall aesthetic quality, StoryFlow [25] was developed to generate storyline visualizations using a hybrid approach that combines discrete and continuous optimization models. Moreover, it supports real-time interactions (e.g., bundling, removing, straightening) for users to edit storyline layouts. However, the automatically-generated storylines are not comparable to the hand-drawn illustrations [45] in terms of the expressiveness because the automatic methods cannot cover abundant narrative elements, including plots, tones, etc.

To create more meaningful storyline visualizations that conform to designers’ requirements, Tang et al. [45] extended the design space of storylines that associates narrative elements with visual elements. They further developed iStoryline that integrates a set of high-level post-editing interactions to support the flexible customization of storyline layouts. They developed a set of easy-to-use high-level interactions, but it is still inefficient to explore the design space and construct the
We develop a mixed-initiative tool [39], PlotThread, to facilitate the AI-generated storyline instead of merely accepting or rejecting it. To achieve this, the AI agent should assist, rather than replace the designers [48]. Hence, the mixed-initiative application [21] refers to a system where automated services (e.g., agents) and users work iteratively (i.e., taking turns) to perform tasks in the same context [20, 30]. Design principles for mixed-initiative systems have been explored [13, 14] to achieve effective collaboration between users and computers. To guide the design of PlotThread, we summarize two primary design considerations:

**DC1. Support a smooth collaborative design workflow.** The AI agents could act as a stimulus for lateral thinking [8] because their ultimate goal is to obtain well-designed layouts. To achieve this goal, Wang et al. [46] employ a graph-based model to map graph structures to page layouts directly, and Kwon et al. [13] employ a deep generative model that uses an encoder-decoder framework to map training datasets into a latent space. However, the existing optimizations are not applicable to our work because storylines pursue higher aesthetic criteria [35] and need to balance narrative constraints [45]. Thus, we intend to develop a novel reinforcement learning framework that trains an AI agent to design storylines similar to the user-specified one. Next, users can simply go ahead for trial-and-error practices to refine the results. To improve user experiences, we employ reinforcement learning to reduce users’ effort in iteratively refining storyline visualizations.

**2.2 Reinforcement Learning**

Reinforcement learning refers to a system where an agent performs a task using a set of actions in an environment that can be represented by a set of states [16][49]. The learning process can be depicted by an agent predicting the “next” action based on the observed “current” state and obtain a reward [40], and the goal of the agent is to maximize cumulative rewards. Due to the emergent development of deep learning techniques [12][36], deep reinforcement learning [44][47] (DRL) has burgeoned in fields like games [17][27] and painting [15]. Mnih et al. [27] proposed a deep Q-network to perform a group of challenging tasks in classic 2D games for the Atari 2600 console [5] and achieved great success in surpassing the previous algorithms when performing the same tasks. To simulate a semi-realistic 3D world, Kempaka et al. [17] introduced a new AI research platform called ViZDoom and further employed deep Q-learning and experience replay to train competent agents. To demonstrate how to teach machines to paint like human painters, Huang et al. [15] employed a neural renderer in model-based DRL to train an agent that creates fancy drawings using a small group of strokes. Despite that reinforcement learning has become prevalent in various fields, we are not aware of any prior works on designing storylines. The issue of producing storylines is similar to the graph drawing problem [13][46] because their ultimate goal is to obtain well-designed layouts. To achieve this goal, Wang et al. [46] employ a graph-based model to map graph structures to page layouts directly, and Kwon et al. [13] employ a deep generative model that uses an encoder-decoder framework to map training datasets into a latent space. However, the existing optimizations are not applicable to our work because storylines pursue higher aesthetic criteria [35] and need to balance narrative constraints [45]. Thus, we intend to develop a novel reinforcement learning framework that trains an AI agent to design storylines similar to the user-specified ones.

**3 PLOTTHREAD**

We develop a mixed-initiative tool [39], PlotThread, to facilitate the easy creation of storyline visualizations. We believe it is essential to combine both human and AI intelligence so that designers can produce creative storylines based on their design preferences and understandings about stories while the agent can reduce labor-intensive efforts.

**3.1 Design Considerations**

The mixed-initiative application [21] refers to a system where automated services (e.g., agents) and users work iteratively (i.e., taking turns) to perform tasks in the same context [20][30]. Design principles for mixed-initiative systems have been explored [13,14] to achieve effective collaboration between users and computers. To guide the design of PlotThread, we summarize two primary design considerations:

**DC2. Balance fine-grained and high-level interactions.** It is burdensome for users to create storyline layouts while pursuing the aesthetic goals (G1 to G3). For reducing human efforts, the previous study [45] proposed high-level interactions that can invoke the optimization model to re-layout storylines. The high-level interactions enable users to change the overall layouts while the aesthetic quality is ensured by the optimization model [25]. While they are easy to use, the high-level interactions cannot fully support users’ design requirements due to their limited flexibility. Conversely, fine-grained interactions focus on modifying the individual lines, so they are flexible enough to support various design requirements. However, they are also tedious and even require professional skills. Since “users may often wish to complete or refine an analysis provided by an agent” [13], we need to achieve a better balance between the high-level and fine-grained interactions.

**3.2 System Workflow**

Our system has two actors, namely users and AI agents (see Fig. 2), to support the collaborative design of storyline visualizations. The existing storyline tools [25][45] employ a solo-authoring workflow where users are the only actor to invoke the design process while computers mainly provide flexible design tools to ease users’ efforts. By incorporating the AI agent, we transformed the typical solo-authoring workflow into a divergent, collaborative design workflow where the agent can help users to explore the design space by providing alternative layouts. Following DC1, users should first input a story script (see Appendix A1) into the system and an initial layout would be automatically generated by the storyline optimization model [45] which conforms to the three aesthetic goals. Users can next modify the initial layout and then trigger the AI agent to generate various storylines proactively. The AI-generated storylines are displayed in a list so that diverse designs can inspire users. By default, we recommend the storyline layout which looks most similar to the user-specified one. Next, users can simply go ahead for further refinements or smoothly switch between different AI layouts. They can also reset to the original storyline when they are unsatisfied with the AI layouts. Compared with the solo-authoring workflow, the co-design workflow may invoke more novel and creative ideas because both users and AI agents can contribute to the design of storylines [48].

**3.3 Interactions**

The core part of a mixed-initiative system entails user interactions which are vital to integrate human-AI co-creativity into the authoring process [20]. To ease the difficulty of constructing storyline layouts, we first implement three high-level interactions inherited from a previous study [45], namely, shifting, bending, and scaling. Second, to support the design of expressive storylines, we also propose a set of novel interactions. According to DC2, the new interactions should enable users to modify the overall layouts without considerable efforts in designing the individual lines. Moreover, they should be more flexible than the high-level interactions since they do not invoke any storyline optimization model.

**3.3.1 High-level Interactions**

We only inherit the three interactions from the previous study [45] because they can formulate user interactions as mathematical constraints which are further integrated into the optimization model to control the generation of storyline layouts.
We propose four types of extended interactions to support the design of storylines. **Shifting** enables users to control the size of white space between lines to indicate whether the selected lines should be attracted or repelled. **Attracting/Repelling** allows users to change the overall trend of the storyline visualization. **Transforming** is developed for users to bend a straight line into a curving line while the associated groups will be transformed automatically. **Relating** enables users to visually present their relationships or making room for inserted images.

### 3.4 Interface

Users first need to load data (Fig. 3a) that are scripts recording characters and their scenic interactions. The AI-based creator can be triggered (Fig. 3b) at any time during the authoring process and then provides a list of suggestive layouts based on users’ layouts. As shown in Fig. 3, the user-specific layout is shown at the top of the list to be compared with alternative layout designs. To inspire lateral thinking [8], we not only present the “final” layout that looks most similar to the user layout but also exhibit the intermediate layouts that demonstrate how the AI agent modifies storylines. Users can freely browse and adopt alternative layouts. We develop two panels (Fig. 3m and 3n) to support the creative design of storylines where users can insert images, add icons, or change model parameters.

### 3.3.2 Extended Interactions

We propose four types of extended interactions to support the design space that describes the design of storylines at four levels, namely, character, relationship, plot, and structure levels. **Transforming** (Fig. 3d) is designed to change the overall trend of storyline layouts, which is at the plot level. Users should first select the scope with a circular brush, and then sketch a trajectory as the trend of the target layout. The specified path will be segmented automatically to guide the translation of line groups of the original storyline. **Attracting/Repelling** (Fig. 3c) are designed to indicate the closeness between the line groups, which is at the structure level. After selecting line groups with a circular brush, users can draw a straight line to indicate whether the selected lines should be attracted or repelled. **Relating** (Fig. 3) is designed to assist users in visually presenting the relationships among characters using various visual elements, such as merged or twined lines, which is at the relationship level. After selecting the desired visual elements, users can choose the group of lines they want to relate with each other. **Styling** (Fig. 3) is developed for users to decorate the lines with diverse stroke styles (e.g., dash and zigzag), which is at the character level. Users first need to select a line style and then brush the target line which they want to embellish. Users can also highlight certain events or characters by directly inserting graphics or icons.

### 4 PROBLEM OVERVIEW

Following the two principles (D1 and D2) of collaborative design, our goal is to train an agent that learns how to resemble users’ intermediate layouts using a set of user-shared interactions. Furthermore, we want to leverage the aesthetic goals (G1 to G3) to produce well-optimized layouts. Thus, the agent is trained to predict the high-level interactions that can modify an automatically-optimized layout to resemble a user-created layout. The high-level interactions preserve the aesthetic quality of layouts as much as possible since we employ the optimization model to re-generate storyline layouts. The problem is formulated as follows: given a user layout L_u generated by the optimization model, the agent predicts the actions used to modify L_u according to L_0. Like human designers, the agent is designed to predict the “next” action by observing the “current” layout and imitating the user layout. To avoid local minima, the agent should balance current actions and future actions by maximizing the cumulative rewards after finishing the given number of actions, rather than the current gain. Inspired by the similar task of reproducing paintings [15], we employ reinforcement learning to achieve this long-term delayed-reward design.
5 REINFORCEMENT LEARNING

In this section, we describe the entire process for designing the reinforcement learning framework from constructing storyline layouts, generating training datasets, building neural networks, and learning the storyline agent.

5.1 Storyline Layout Construction

Storyline visualizations depict how characters interact with each other. Generally, each line represents a character, and a group of lines indicates that the associated characters are together at a given time slot [25].

Given a story with $N$ characters and $M$ time slots, the path of the $i$-th character $C_i$ can be described as a sequence of points $[y^0_i, y^1_i, ..., y^{M-1}_i]$. The overall layout can be denoted as a set of characters $L = \{ C_i \}_{i=0}^1$ which can be further depicted as

$$ M_{pos} = [y^j_i] = 0, ..., N-1; j = 0, ..., M-1 \tag{1} $$

The main difficulty in obtaining a storyline layout is the calculation of its position matrix $M_{pos}$, which pursues the maximization of the aesthetic metrics (G1 to G3) while satisfying the primary narrative constraints (C1 and C2). Given that the performance of the existing storyline algorithms [25,45,46] has been considerably improved, we adopt Storyline [45] as the renderer to calculate the layout. First, iStoryline is implemented on the basis of StoryFlow to achieve a real-time generation for a large collection of storylines [25], which is vital for training an agent that needs to reproduce storylines for over thousands of hundreds of times. Second, iStoryline extends the optimization model of StoryFlow to integrate a more diverse set of narrative constraints, which is crucial for the agent to fully explore the overall design space and customize storyline layouts without losing too much aesthetic quality.

In PlotThread, we inherit three high-level interactions, namely, shifting, bending, scaling, from iStoryline, which insert three novel types of narrative constraints to the three optimization stages, namely ordering, alignment, and compaction [45]. Next, we introduce how these high-level interactions are incorporated into the optimization model to enable an efficient customization of storyline layouts.

Shifting determines the vertical order of characters along the timeline via the dynamic programming algorithm [25], which generates ordering constraints using a set of order pairs $[o^i_j, o^i_{j'}]_{i, j < N; j' < M}$ where $o^i_j$ indicates the order of the $i$-th character at the $j$-th time slot. The constraint suggests that the $j$-th character should be “ahead” of the $j'$-th character at the $j$-th time slot. After solving the ordering algorithm [11], the order of characters during the whole timeline can be obtained using

$$ M_{order} = [o^i_j] = 0, ..., N-1; j = 0, ..., M-1 \tag{2} $$

Bending determines the straightness of characters along the timeline via the dynamic programming algorithm [25], which generates alignment constraints using a set of indicators $[e^i_{j'}]_{i < N; j' < M}$. The variable $e^i_{j'}$ is set to 1 when the $i$-th character is aligned at both the $j$-th and its previous time slots. By default, the indicators at the first time slot are set to 1 so that $[e^0_j = 1]_{j = 0, ..., N-1}$. After solving the dynamic programming [25], the alignment situations can be obtained using

$$ M_{align} = [e^i_{j'}] = 0, ..., N-1; j = 0, ..., M-1 \tag{3} $$

Scaling determines the white space among characters through the least-square method [25], which generates compaction constraints using a set of inequalities $[d_1 < y^i_j - y^{i-1}_j < d_2]_{i < N; j < M}$ where $d_1$ and $d_2$ are numerical values to indicate the lower and upper bounds of the white space among the $i$-th and its last characters. After obtaining the results (Eq. 2 and 3) of the two previous optimization stages, the position matrix (Eq. 1) can be obtained by solving a constrained convex optimization problem which is detailed in Appendix A2.

5.2 Training Data Collection

Training neural networks require a large number of high-quality datasets [18,24]. Although Tang et al. [45] have extended the collection of hand-drawn storyline illustrations, the size of the dataset is too small for a machine learning task. Moreover, the manual production of training data is a labor-intensive task which requires considerable time and human resource. Inspired by the recent studies on graph drawings [19,46], we generate a set of well-optimized storyline layouts using the optimization model [25]. Although automatically-generated storylines are not comparable to the hand-drawn illustrations in terms of both aesthetic quality and expressiveness [45], our goal is to train an agent that can imitate users layouts instead of generating storylines that are comparable or superior to hand-drawn ones.

To obtain considerable and diverse datasets, researchers have employed a grid search that applies different combinations of random parameters on the graph models [19,46]. Following this common practice, we use iStoryline [45] to generate the training datasets due to its ability to produce aesthetic storyline layouts in a short time. Notice that iStoryline only receives two parameters, namely inner gap and outer gap, to determine the white space between individual lines and the groups of lines, respectively. Thus, merely modifying the model parameters cannot produce sufficient storylines with diverse layouts.

We apply random searching in generating different narrative constraints described in Sec. 5.1, which are further integrated into the optimization model [25] to control the diversity of storyline layouts. Mathematically, the training data is a set of storyline pairs $<L_o, L_r>$, where $L_o$ is the origin layout generated by the optimization model directly, and $L_r$ is the “user” layout simulated by inserting randomly-selected narrative constraints into the optimization model [25]. However, the simulated “user” layout $L_r$ may not be visually “better” than the origin layout $L_o$ because more narrative constraints are used to restrict the optimization of storylines. Since the goal of the AI agent is to provide a list of possible layouts according to users’ layouts, the key of our RL model is to teach the agent to refine origin layouts and imitate users’ layouts instead of producing extremely-optimized storylines.

Following the design considerations mentioned above, we first extract story scripts that describe characters and their scenic interactions from the hand-drawn illustrations [1]. Each story script records a set of time slots that indicate who are together at a given time. To ensure the diversity of training data, we evenly produce three groups of narrative constraints, namely, ordering, alignment, and compaction constraints with random parameters. We then randomly select $K$ constraints from the three groups to obtain different layouts for the same story script. The selected constraints are the ground truth that the agent needs to learn and predict when modifying origin layouts $L_o$ to imitate user layouts $L_u$. The variable $K$ indicates how many steps the agent can have to reproduce user layouts. In our case, we set $K = 15$ because the agent should complete the authoring task within reasonable time to avoid losing users’ attention. We obtain 20 story scripts from the published gallery and generate 1000 storyline layouts for each story. In total, we generate 20000 layouts to train the AI agent.

5.3 Model Architecture

Given a user layout $L_u$ and an origin layout $L_o$, the agent aims to predict a sequence of actions $(a_k)_{k=0}^{K-1}$ where rendering $a_k$ on $L(k)$ leads to $L(k+1)$. The initial layout $L(0)$ can be obtained from the origin layout $L_o$. The final layout $L(K-1)$ can be obtained by rendering the consecutive actions, which should be visually similar to $L_u$ as much as possible. This design issue can be formulated as a Markov Decision Process [15] with a state space $S$, an action space $A$, a transition function $T(s_k, a_k)$ and a reward function $R(s_k, a_k)$ [34].

5.3.1 State and Transition Function

The state space describes all possible layouts that an agent can obtain after rendering actions. Mathematically, we define a state $s_k = (L(k), L_u, k)$ where $L(k)$ and $L_u$ refer to the layouts that can be represented by the position matrix $M_{pos}$ and the variable $k$ indicates the $k$-th step for the agent. We further define the transition function $s_{k+1} = T(s_k, a_k)$ that describes the transition process between states $s_k$ and $s_{k+1}$, which is implemented by applying action $a_k$ on state $s_k$.

https://istoryline.github.io/gallery/
We calculate the position difference of the two layouts using Euclidean metric. We further employ sigmoid function to quantify the alignment similarity as:

\[ S_{\text{align}}(k) = \text{Comp}(M_{\text{align}}^{(k)}, M_{\text{align}}^{(k)}) \]

Position Feature The last step for generating storyline layouts is to calculate the exact positions of characters by minimizing the white space of the overall layout. The position variable \( e_i^j(L) \) indicates that the position of the \( i \)-th character at the \( j \)-th time slot in the layout \( L \). We calculate the position difference of the two layouts using:

\[ D_{\text{pos}}^{(k)} = \text{Dist}(M_{\text{pos}}^{(k)}, M_{\text{pos}}^{(k)}) \]

where \( \text{Comp}(\cdot) \) is a counting function that self-increment one if the corresponding values of two matrices are the same and \( \text{Dist}(\cdot) \) is a distance function that calculates the difference between two matrices using Euclidean metric. We further employ sigmoid function for

\[ \text{sigmoid}(\cdot) \]

5.3.2 Action
To support the collaborative design, we define the action space as the high-level interactions (discussed in Sec. 5.1) for three reasons. First, it is necessary for the agent to share the same action space with users so that they can work concurrently to design storyline visualizations. Second, the high-level interactions are implemented on the basis of the constrained optimization model so that the agent can produce well-optimized layouts in terms of the aesthetic goals. Third, it is sufficient to modify storyline layouts with these interactions so that we do not include the other interactions proposed in PlotThread. Formally, an action \( a \) of the storyline agent is a set of parameters that define a narrative constraint (e.g., ordering, alignment, compaction constraint). The behaviors of the agent can be further described using a policy function \( \pi : \mathbb{S} \rightarrow \mathbb{A} \) that maps states to deterministic actions [40]. After predicting action \( a_k \) at step \( k \), the state can evolve using the transition function \( s_{k+1} = T(a_k, s_k) \), which runs for \( K \) steps [47].

5.3.3 Reward
Reward is a stimulus for the agent to improve its prediction ability [27]. Our goal is to guide the agent to resemble the layout on which users are working and produce alternative layouts to inspire co-creativity. We formulate the reward as the similarity between the user layout \( L_u \) and the layout \( L^{(k)} \) produced by the agent at step \( k \). To quantify the layout similarity, we follow the well-established framework [25] that measures storyline layouts in three aspects:

Ordering Feature The first step to obtain an aesthetic storyline layout is to determine the vertical order of characters. The ordering variable \( e_i^j(L) \) indicates the ranking position of the \( i \)-th character at the \( j \)-th time slot for the layout \( L \). After obtaining the orders of characters, the second step is to determine the alignment situation of characters along the whole timeline. Given a layout \( L \), the alignment variable \( e_i^j(L) \) indicates whether the \( i \)-th character is aligned at the \( j \)-th time slot and the previous slot. Following the same mathematical notations, we quantify the alignment similarity as:

\[ S_{\text{align}}(k) = \text{Comp}(M_{\text{align}}^{(k)}, M_{\text{align}}^{(k)}) \]

Position Feature The last step for generating storyline layouts is to calculate the exact positions of characters by minimizing the white space of the overall layout. The position variable \( e_i^j(L) \) suggests that the position of the \( i \)-th character at the \( j \)-th time slot in the layout \( L \). We calculate the position difference of the two layouts using:

\[ D_{\text{pos}}^{(k)} = \text{Dist}(M_{\text{pos}}^{(k)}, M_{\text{pos}}^{(k)}) \]

5.3.4 Network Architecture
Due to the high variability and complexity of narratives, we first normalize the input layout into a \( H \times H \) matrix which can be regarded as a one-channel image (By default, we set \( H = 100 \)). To extract the visual features from storyline layouts, we employ the network structure that is similar to ResNet-18 [12]. Given that storyline layouts are less complicated than real-world images, we simplify the network structure by removing all convolution layers to preserve visual information. In our experiments, we discover that the fully-connected layers are capable of predicting actions for generating storyline layouts. To ease the difficulty of exploring the mix-type action space and stabilize learning, we separate the network architecture into three parallel components [17] that aim at exploring the different parts of the action space. Specifically, every component is designed only to explore the action space of one of the high-level interactions (see Fig. 6). In the end, we employ a greedy function to calculate the reward and determine the action.

\[ R_k = \sum_{k=0}^{K} \gamma^k \max_{a_k} r(s_k, a_k) \]

5.4 Learning
We first introduce the standard setting for reinforcement learning [41] where an agent interacts with an environment over a certain number of time steps, and then describe how to train the agent using the state-of-the-art framework, namely, asynchronous advantage actor-critic [25].

In a typical actor-critic model [32], researchers usually employ two neural networks to present actor and critic, respectively (see Fig. 6). An actor observes the environment by receiving an state \( s_k \) and then predict an action \( a_k \) at time step \( k \), while a critic obtains the state \( s_k \) to predict cumulative reward in the future. In general, the policy function \( \pi(s_k | s, \theta_\pi) \) characterizes the actor’s behaviors which can be formulated as a mathematical probability function. Since an agent aims to maximize the expected cumulative reward [16], the value of state \( s_k \) under policy \( \pi \) can be defined as \( V^\pi(s, \theta_\pi) = E(R_k | s_k = s) \) that is the expected return for following policy \( \pi \) from state \( s \). Hence, the problem of training a storyline agent is to obtain the parameters \( (\theta_\pi, \theta_\gamma) \) of the neural networks for the policy function \( \pi \) and the value function \( V \). The updates on the model parameters [6] can be written as
To overcome these issues, Minih et al. [26] propose a novel asynchronous framework that enhances the existing RL algorithms, such as Q-learning [22], and actor-critic methods [32]. The key idea is to use asynchronous actor-learners that run in parallel to explore different parts of the environment [23][26]. Instead of using different machines, the actor-learners are running on the different processes to remove the communication costs and improve training efficiency. Moreover, the researchers observe that it is more likely for the multiple actor-learners to be uncorrelated than a single agent when applying the overall changes to the model parameters. The updates applied to the parallel agent [26] will be updated on the main agent to combine the asynchronous changes of the model parameters on different processes.

However, training an agent in a complicated high-dimensional mix-type action space is difficult due to the unstable learning problem and the requirements of large computational resources [26][50]. To overcome these issues, Minih et al. [26] propose a novel asynchronous framework that enhances the existing RL algorithms, such as Q-learning [22], and actor-critic methods [32]. The key idea is to use asynchronous actor-learners that run in parallel to explore different parts of the environment [23][26]. Instead of using different machines, the actor-learners are running on the different processes to remove the communication costs and improve training efficiency. Moreover, the researchers observe that it is more likely for the multiple actor-learners to be uncorrelated than a single agent when applying the overall changes to the model parameters. The updates applied to the parallel agent [26] will be updated on the main agent to combine the asynchronous changes of the model parameters on different processes.

\[
\begin{align*}
\Delta \theta_x & \leftarrow \Delta \theta_x + \nabla_{\theta_x} \log \pi(a_k|s_k; \theta_x) \langle R_k - V(s_k; \theta_y) \rangle \\
\Delta \theta_y & \leftarrow \Delta \theta_y + \frac{\partial (R_k - V(s_k; \theta_y))^2}{\partial \theta_y} 
\end{align*}
\]

where \(\Delta \theta_x\) and \(\Delta \theta_y\) are the updates applied to the model parameter \(\theta_x\) and \(\theta_y\), respectively.

We also conducted quantitative experiments on a desktop with a CPU (3.7GHz) to evaluate the search power and time performance of the AI agent by comparing it with a baseline method. We repeated the experiments 4 times and calculated average values to avoid the influences of CPU scheduling. Since there are no prior RL models on designing storyline visualizations, we implemented a greedy algorithm that randomly selects a group of actions and then adopts the one that can improve the reward. We compare the AI agent with the greedy algorithm by measuring their convergence rates and time when performing the same tasks. As shown in Fig. 8b, the baseline method can improve the rewards dramatically in the short term but they are trapped in the local optimums finally. While the AI agent seems to have more edge crossings than the initially-optimized layouts, they still preserve a satisfactory aesthetic quality compared to the user layouts. Thus, we believe our agent achieves a better trade-off between expressiveness and aesthetics even though it increases expressiveness at the cost of some aesthetic quality.
7 Use Cases

In this section, we illustrate the usage of PlotThread. A storyline visualization called Justice League is created to describe the authoring process of customizing the layout with the assistance of the AI agent. This use case indicates how people and the agent can work together to achieve users’ design requirements. Following the same procedure, we create more use cases (see Fig. 1) to demonstrate that PlotThread can be used to design various stories and produce diverse layouts.

As a proof of concept, we simplify the story of Justice League and only depict events that are vital for the evolution of the narrative. The story depicts how superheroes stand together and establish Justice League to fight against Steppenwolf, which can be roughly divided into three stages. First, Batman and Wonder Woman decided to recruit team members. They recruited The Flash, Cyborg, and Aquaman to save the world from the catastrophic threat caused by Steppenwolf. The second stage begins with the fight against Steppenwolf’s invasion and the rescue act for Superman. But Superman attacked the other superheroes who try to rescue him since Steppenwolf had twisted his mind. After he recovered, he decided to join Justice League. The third stage was the climax where the superheroes struggled with the fight against Steppenwolf, and finally won with the return of Superman.

As shown in Fig. 7 we illustrate how to create the storyline visualization step by step using PlotThread. We use “George” who refers to the user to describe the authoring process. George first loads the story data which is a formatted document for the storyline renderer, and he obtains an initial storyline. Then, he uses Shifting to change the order of characters, and separate Steppenwolf and the members of Justice League (Fig. 7a). Based on his understanding of the story, he wants to transform the layout of the first stage into an “up-step” shape, where Batman and Wonder Woman tried to increase their team by recruiting new superheroes. He uses Transforming, selects the superheroes involved, and draws an ascending step-like line to obtain the initial step layout (Fig. 7b). Next, he moves on to transform the shape of other stages, for example, in the climax stage, he draws a parabolic curve (Fig. 7c) to illustrate how Justice League beats Steppenwolf. George thinks the appearance is not legible or aesthetically appealing enough, so he seeks for AI assistance by triggering the AI creator. After AI gives the results, he quickly switches between different layouts and finds one (Fig. 7e), which has few crossings and deviations but preserving the narrative trends specified by him.

George continues to clarify the relationship between the groups of characters in a more coarse-grained way. Using Repelling, he emphasizes the part when Superman leaves the other superheroes, and when he reunites (Fig. 7h). He triggers the AI creator when he wants to improve the appearance and gain some inspirations about the storyline design (Fig. 7j). After clarifying the trend of the story as well as the general narrative structure, George starts to work on detailed refinements using Bending and Scaling (Fig. 7f). For example, he emphasizes the closeness of Justice League at the end of the whole story using Scaling. After completing the layout, George uses Relating to embellish the plots and make them more expressive (Fig. 7g). For example, he uses twined lines to illustrate the intense fights. Finally, he embellishes the picture by adding icons, changing line colors and stroke weight, as well as adding text annotations (Fig. 7i).

8 User Feedback

To evaluate the effectiveness and usability of PlotThread, we conducted semi-structured interviews with three experts. The first expert (ED) was a professional UI/UX designer who worked for an international software company. She helped to test the usability of PlotThread because she had rich experiences in using various commercial design tools (e.g., Adobe AI/PS). The second expert (EV) was a senior researcher who studied visualization and visual analytics for eight years. He evaluated the output storylines crafted by PlotThread (see Fig. 1) and compared them with the storylines generated by the optimization model and human artists. The second expert (EA) was a professional UI/UX designer who worked for an international software company. She helped to test the usability of PlotThread because she had rich experiences in using various commercial design tools (e.g., Adobe AI/PS). She evaluated the output storylines crafted by PlotThread and discussed the potential applications for storyline visualizations. The interview includes a 30-min demonstration of PlotThread and a 30-min discussion.
ED mainly evaluated the storylines generated by different agents from the aesthetic and narrative aspects. We provided three kinds of storylines: the AI-assisted storylines created by PlotThread (see Fig. 7), the extremely-optimized storylines generated by the optimization model without human involvement [25], and the hand-drawn storylines created by artists [45]. She thoroughly compared the visual designs of the different storylines and surprisingly found that the extremely-optimized storylines are hardest to read although they have fewest crossings and deviations. She inferred that “viewers intend to pay attention to line groups which are hard to be distinguished in extremely-optimized storylines because they are too compact.” This observation validates the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

We discuss the implications and limitations of PlotThread as follows: we develop PlotThread that facilitates the easy creation of storyline visualizations. Despite that existing tools [25, 45] have incorporated human creativity into the optimization models and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. The one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

In this research, we develop PlotThread, a mixed-initiative authoring system where the agent is trained to “refine” storyline layouts based on user feedback. We have implemented PlotThread on a personal laptop, and we plan to improve the time efficiency of the AI agent by applying RL on designing storylines. We evaluate the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

We discuss the implications and limitations of PlotThread as follows: we develop PlotThread that facilitates the easy creation of storyline visualizations. Despite that existing tools [25, 45] have incorporated human creativity into the optimization models and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

ED mainly evaluated the storylines generated by different agents from the aesthetic and narrative aspects. We provided three kinds of storylines: the AI-assisted storylines created by PlotThread (see Fig. 7), the extremely-optimized storylines generated by the optimization model without human involvement [25], and the hand-drawn storylines created by artists [45]. She thoroughly compared the visual designs of the different storylines and surprisingly found that the extremely-optimized storylines are hardest to read although they have fewest crossings and deviations. She inferred that “viewers intend to pay attention to line groups which are hard to be distinguished in extremely-optimized storylines because they are too compact.” This observation validates the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

In this research, we develop PlotThread, a mixed-initiative authoring system where the agent is trained to “refine” storyline layouts based on user feedback. We have implemented PlotThread on a personal laptop, and we plan to improve the time efficiency of the AI agent by applying RL on designing storylines. We evaluate the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

We discuss the implications and limitations of PlotThread as follows: we develop PlotThread that facilitates the easy creation of storyline visualizations. Despite that existing tools [25, 45] have incorporated human creativity into the optimization models and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

In this research, we develop PlotThread, a mixed-initiative authoring system where the agent is trained to “refine” storyline layouts based on user feedback. We have implemented PlotThread on a personal laptop, and we plan to improve the time efficiency of the AI agent by applying RL on designing storylines. We evaluate the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

We discuss the implications and limitations of PlotThread as follows: we develop PlotThread that facilitates the easy creation of storyline visualizations. Despite that existing tools [25, 45] have incorporated human creativity into the optimization models and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

In this research, we develop PlotThread, a mixed-initiative authoring system where the agent is trained to “refine” storyline layouts based on user feedback. We have implemented PlotThread on a personal laptop, and we plan to improve the time efficiency of the AI agent by applying RL on designing storylines. We evaluate the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

We discuss the implications and limitations of PlotThread as follows: we develop PlotThread that facilitates the easy creation of storyline visualizations. Despite that existing tools [25, 45] have incorporated human creativity into the optimization models and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

In this research, we develop PlotThread, a mixed-initiative authoring system where the agent is trained to “refine” storyline layouts based on user feedback. We have implemented PlotThread on a personal laptop, and we plan to improve the time efficiency of the AI agent by applying RL on designing storylines. We evaluate the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

We discuss the implications and limitations of PlotThread as follows: we develop PlotThread that facilitates the easy creation of storyline visualizations. Despite that existing tools [25, 45] have incorporated human creativity into the optimization models and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

In this research, we develop PlotThread, a mixed-initiative authoring system where the agent is trained to “refine” storyline layouts based on user feedback. We have implemented PlotThread on a personal laptop, and we plan to improve the time efficiency of the AI agent by applying RL on designing storylines. We evaluate the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

We discuss the implications and limitations of PlotThread as follows: we develop PlotThread that facilitates the easy creation of storyline visualizations. Despite that existing tools [25, 45] have incorporated human creativity into the optimization models and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

In this research, we develop PlotThread, a mixed-initiative authoring system where the agent is trained to “refine” storyline layouts based on user feedback. We have implemented PlotThread on a personal laptop, and we plan to improve the time efficiency of the AI agent by applying RL on designing storylines. We evaluate the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

We discuss the implications and limitations of PlotThread as follows: we develop PlotThread that facilitates the easy creation of storyline visualizations. Despite that existing tools [25, 45] have incorporated human creativity into the optimization models and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

In this research, we develop PlotThread, a mixed-initiative authoring system where the agent is trained to “refine” storyline layouts based on user feedback. We have implemented PlotThread on a personal laptop, and we plan to improve the time efficiency of the AI agent by applying RL on designing storylines. We evaluate the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

We discuss the implications and limitations of PlotThread as follows: we develop PlotThread that facilitates the easy creation of storyline visualizations. Despite that existing tools [25, 45] have incorporated human creativity into the optimization models and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.

In this research, we develop PlotThread, a mixed-initiative authoring system where the agent is trained to “refine” storyline layouts based on user feedback. We have implemented PlotThread on a personal laptop, and we plan to improve the time efficiency of the AI agent by applying RL on designing storylines. We evaluate the effectiveness of PlotThread which intends to better balance the aesthetic goals and the narrative constraints by sacrificing some aesthetic quality to enrich the narration. On the one hand, the AI agent is inherently driven by the optimization model so that it can produce well-optimized results. On the other hand, the agent resembles the input layouts which can be flexibly customized to indicate more narrative details.
REFERENCES

[1] Paper.js. 2019. Retrieved Dec 1st, 2019 from http://paperjs.org/.

[2] Pytorch. 2019. Retrieved Dec 1st, 2019 from https://pytorch.org/.

[3] Typescriptlang.org/}

[4] D. Arendt and M. Pirsing. The “y” of it Matters, Even for Storyline Visualization. In Proceedings of the IEEE Conference on Visual Analytics Science and Technology, pages 81–91, 2017.

[5] M. G. Bellemare, Y. Nadjar, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. Journal of Artificial Intelligence Research, 47(1):253–279, 2013.

[6] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor-critic algorithms. Automatica, 45(11):2482–2492, 2009.

[7] Z. Chen, Y. Wang, T. Sun, X. Gao, W. Chen, Z. Pan, H. Qu, and Y. Wu. Exploring the design space of immersive urban analytics. Visual Informatics, 1(2):132–142, 2017.

[8] E. De Bono. Lateral thinking: a textbook of creativity. Penguin UK, 2010.

[9] L. Deqing, M. Honghui, S. Yi, S. Shuang, Z. Wenli, W. Junting, Z. Ming, and C. Wei. Echarts: A declarative framework for rapid construction of web-based visualization. Visual Informatics, 2:136–146, 2018.

[10] J. Diaz, J. Petit, and M. Serna. A Survey of Graph Layout Problems. ACM Computing Surveys, 34(3):313–356, 2002.

[11] M. Forster. A Fast and Simple Heuristic for Constrained Two-Level Crossing Reduction. In Proceedings of the International Conference on Graph Drawing, pages 206–216, 2004.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[13] E. Horvitz. Principles of Mixed-Initiative User Interfaces. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems, pages 159–166, 1990.

[14] E. Horvitz. Uncertainty, Action, and Interaction: In Pursuit of Mixed-initiative Computing. IEEE Intelligent Systems, 14(5):17–20, 1999.

[15] Z. Huang, W. Heng, and S. Zhou. Learning to Paint With Model-Based Deep Reinforcement Learning. In Proceedings of the IEEE International Conference on Computer Vision, pages 8709–8718, 2019.

[16] L. P. Kaehbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. Journal of artificial intelligence research, 4:237–285, 1996.

[17] M. Kerzuka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. Vizdoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. In Proceedings of the IEEE Conference on Computational Intelligence and Games, pages 1–8, 2016.

[18] O.-H. Kwon and K.-L. Ma. An Efficient Framework for Optimizing Storyline Visualizations. In Proceedings of the Advances in Neural Information Processing Systems, pages 6493–6501, 2017.

[19] Rong-Long Wang and Okazaki. Artificial Neural Network for Minimum Crossing Number Problem. In Proceedings of the International Conference on Machine Learning and Cybernetics, pages 4201–4204, 2005.

[20] Schmidhuber and Jürgen. Deep Learning in Neural Networks: An Overview. Neural Networks, 61:85–117, 2015.

[21] Y. Shi, C. Bryan, S. Bhamidipati, Y. Zhao, Y. Zhang, and K.-L. Ma. MeetingVis: Visual Narratives to Assist in Recalling Meeting Context and Content. IEEE Transactions on Visualization and Computer Graphics, 24(6):1918–1929, 2018.

[22] Z. Snyder. Justice League. DC Films, 2017.

[23] I. Steffoniss and D. Thue. Authoring Tools Should be Mixed-initiative Systems. In Proceedings of the ICIDS Workshop on Authoring for Interactive Storytelling, 2017.

[24] R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. MIT press, 2018.

[25] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Proceedings of International Conference on Neural Information Processing Systems, pages 1057–1063, 1999.

[26] R. Tamassia, G. D. Battista, and C. Batini. Automatic Graph Drawing and Readability of Diagrams. IEEE Transactions on Systems, Man and Cybernetics, 18(1):61–79, 1988.

[27] Y. Tanahashi, C. H. Hsu, and K.-L. Ma. An Efficient Framework for Generating Storyline Visualizations from Streaming Data. IEEE Transactions on Visualization and Computer Graphics, 21(6):730–742, 2015.

[28] Y. Tanahashi and K. Ma. Design Considerations for Optimizing Storyline Visualizations. IEEE Transactions on Visualization and Computer Graphics, 18(12):2679–2688, 2012.

[29] T. Tang, S. Rubab, J. Lai, W. Cui, L. Yu, and Y. Wu. iStoryline: Efective Convergence to Hand-drawn Storylines. IEEE Transactions on Visualization and Computer Graphics, 25(1):769–778, 2019.

[30] Y. Wang, Z. Jin, Q. Wang, W. Cui, T. Ma, and H. Qu. DeepDrawing: A Deep Learning Approach to Graph Drawing. IEEE Transactions on Visualization and Computer Graphics, 26(1):676–686, 2020.

[31] Z. Yang, K. Merrick, H. Abbass, and L. Jin. Multi-task deep reinforcement learning for continuous action control. In Proceedings of the International Joint Conference on Artificial Intelligence, pages 3301–3307, 2017.

[32] G. N. Yannakakis, A. Liapis, and C. Alexopoulos. Mixed-Initiative Creativity. In Proceedings of International Conference on the Foundations of Digital Games, pages 1–8, 2014.

[33] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu. A survey of visual analytics techniques for machine learning. Computational Visual Media, 7(1), 2021.

[34] Z. Zhou. Abductive learning: towards bridging machine learning and logical reasoning. Science China Information Sciences, 62(7), 2019.