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

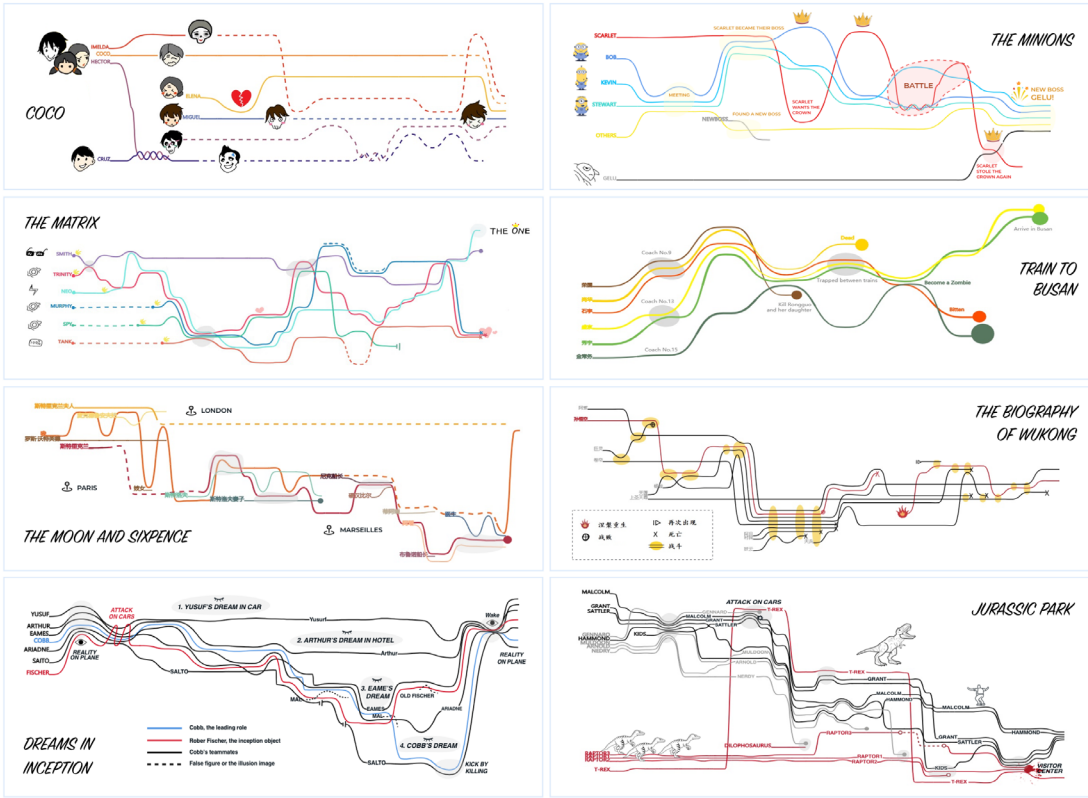

Fig. 1. Example storyline visualizations created using PlotThread. The layouts are generated through a collaborative design between the AI agent and designers, while the styles and visual labels are customized manually to embellish the storylines.

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 [28, 44] have gained wide popularity in presenting complex entity relationships. The ability to create visual narratives [19] makes it applicable in presenting fictions [28], analyzing dynamic networks [25], recalling meeting content [37], and understand-

- *T. Tang, R. Li, X. Wu, S. Liu, Y. Wu are with Zhejiang Lab and State Key Lab of CAD&CG, Zhejiang University. E-mail: tangtan, renzhongli, xinke_wu, shliu, ycwu@zju.edu.cn. Y. Wu is the corresponding author.*
- *J. Knittel, S. Koch, and T. Ertl are with VIS/VISUS, University of Stuttgart. E-mail: Johannes.Knittel, Steffen.Koch, Thomas.Ertl@vis.uni-stuttgart.de.*
- *L. Yu is with Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University. Email: Lingyun.Yu@xjtlu.edu.cn.*
- *P. Ren is with Alibaba Group. E-mail: renpeiran@gmail.com.*

Manuscript received 30 Apr. 2020; revised 31 July 2020; accepted 14 Aug. 2020.
Date of publication 13 Oct. 2020; date of current version 15 Jan. 2021.
Digital Object Identifier no. 10.1109/TVCG.2020.3030467

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.*

Inspired by graph layouts [42], it is necessary to minimize line crossings and deviations to avoid dense visual clutter. Thus, three aesthetic goals [44] 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 [4, 25, 43, 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 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 (**A**rtificial **I**ntelligence) 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 [28, 44] among entities that are inherently different from the 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 [15]. 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 **R**einforcement **L**earning (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 [4, 25, 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

overall layout using these fine-grained interactions. iStoryline automatically translates the high-level interactions into mathematical constraints which are further integrated into the optimization model [25] to generate storyline layouts. However, users may obtain unexpected layouts due to the unpredictability of the optimization process, which requires 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 [34,47] (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 [18, 46] because their ultimate goal is to obtain well-designed layouts. To achieve this goal, Wang et al. [46] employ a graph-LSTM-based model to map graph structures to graph layouts directly, and Kwon et al. [18] employ a deep generative model that uses an encoder-decoder framework to map training datasets into a latent space. However, the existing approaches 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 like human users to support collaborative design.

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:

DC1. Support a smooth collaborative design workflow. The AI agents could act as a stimulus for lateral thinking [8] to inspire co-creativity [48]. To foster effective human-AI collaboration, it is necessary to place the human at the center of visualization designs, while the AI agent should assist, rather than replace the designers [48]. Hence, users should be granted enough control in the decision-making stage. One common practice is that the user takes the task-initiate [30] in customizing an initial layout. Then, the agent proactively contributes to the design process by improving users' intermediate results and providing alternative designs based on users' input layouts. Moreover, users should be capable of further modifying and improving the AI-generated storyline instead of merely accepting or rejecting it. To follow this practice, it is essential to seamlessly integrate the AI agent into the authoring process and provide a smooth co-design workflow.

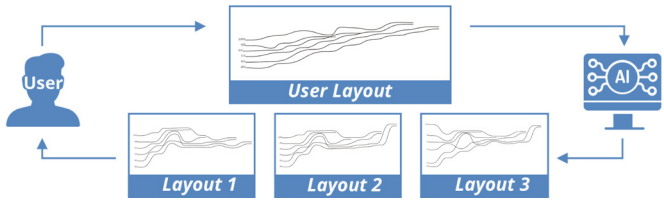

Fig. 2. System workflow that supports a smooth and iterative co-design process between users and the AI agent. Users start the design process by customizing an initial storyline while the AI agent provides a set of suggestive alternative designs according to the user-specified layout.

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 **DC 1**, 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.

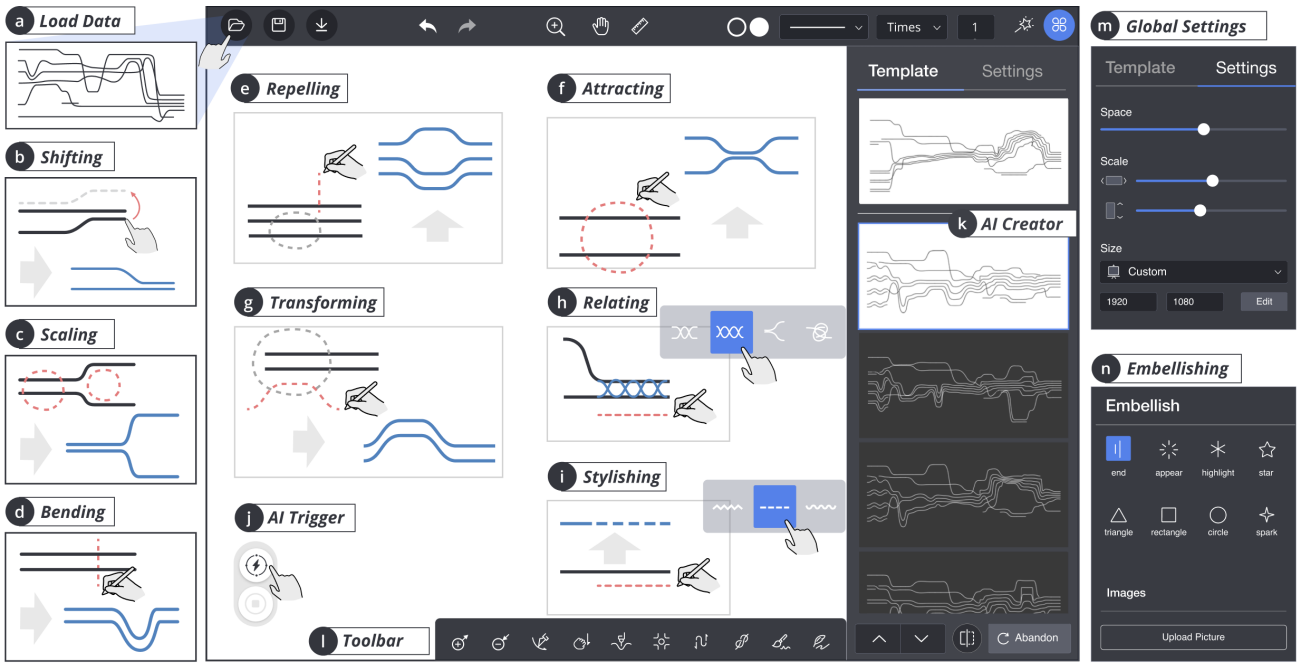

Fig. 3. PlotThread is composed of a menubar for (a) loading story scripts, setting canvas, and exporting storylines; (l) a toolbar that provides a set of easy-to-use interactions (b) to (i); The red lines indicate the interactions that change original layouts (black lines) into desired layouts (blue lines). (j) buttons for activating and stopping the AI agent and (k) a panel for presenting AI-generated layouts; (m) a setting panel for changing the parameters of storylines and (n) an embellishing panel for inserting icons or images into the canvas.

Shifting. The relationships among characters are visually revealed by the spatial proximity of the corresponding lines. To define the characters' relationship, *shifting* (Fig. 3b) enables users to drag an individual line to re-order the characters freely.

Bending. The plot evolution can be indicated by the overall layout of the storyline visualizations. For example, users can arrange the line groups in a certain direction to suggest that the story evolves into a positive or negative ending. *Bending* (Fig. 3d) enables users to easily bend a straight line into a curving line while the associated groups will be transformed automatically.

Scaling. The white space can be used to present different narrative elements, such as emphasizing separations between characters to present their relationships or making room for inserted images. *Scaling* (Fig. 3c) enables users to control the size of white space between lines or groups by dragging and moving the groups of lines.

3.3.2 Extended Interactions

We propose four types of extended interactions to support the design space [45] that describes the design of storylines at four levels, namely, character, relationship, plot, and structure levels.

Transforming (Fig. 3g) 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. 3f and 3e) 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. 3h) is designed to assist users in visually presenting the relationships among characters using various visual elements, such as merged or twined lines [45], 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.

Stylishing (Fig. 3i) 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.

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. 3j) at any time during the authoring process and then provides a list of suggestive layouts based on users' layouts. As shown in Fig. 3k, the user-specified 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, and *change* model parameters.

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 [25] to re-generate storyline layouts.

The problem is formulated as follows: given a user layout L_u crafted by a user and an origin layout L_o generated by the optimization model [25], the agent predicts the actions used to modify L_o according to L_u. 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 [28]. 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 $[y_i^0, y_i^1 ..., y_i^{M-1}]$. The overall layout can be denoted as a set of characters $L = \{C_i\}_{i=0}^{N-1}$ which can be further depicted as

$$M_{pos} = [y_i^j]_{i=0,...,N-1;j=0,...,M-1} \quad (1)$$

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 [25,43,45] has been considerably improved, we adopt iStoryline [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 using a constrained crossing minimization algorithm [11], which generates *ordering* constraints using a set of order pairs $[o_i^j, o_{i'}^j]_{i,i'<N;j<M}$ where o_i^j indicates the order of the i-th character at the j-th time slot. The constraint suggests that the i-th character should be "ahead" of the i'-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

$$M_{order} = [o_i^j]_{i=0,...,N-1;j=0,...,M-1} \quad (2)$$

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 are 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 $\{e_i^0 = 1\}_{i=0,...,N-1}$. After solving the dynamic programming [25], the alignment situations can be obtained using

$$M_{align} = [e_i^j]_{i=0,...,N-1;j=0,...,M-1} \quad (3)$$

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 considerate time and human resource. Inspired by the recent studies on graph drawings [18,46], 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 considerate and diverse datasets, researchers have employed a grid search that applies different combinations of random parameters on the graph models [18, 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 $< L_o, L_u >$ where L_o is the origin layout generated by the optimization model directly, and L_u is the "user" layout simulated by inserting randomly-selected narrative constraints into the optimization model [25]. However, the simulated "user" layout L_u 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 $\mathbb{S}$, an action space $\mathbb{A}$, a transition function $T(s_t, a_u)$ and a reward function $R(s_t, a_u)$ [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 $s_u = (L^{(k)}, L_u, k)$ 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.

[1] https://istoryline.github.io/gallery/

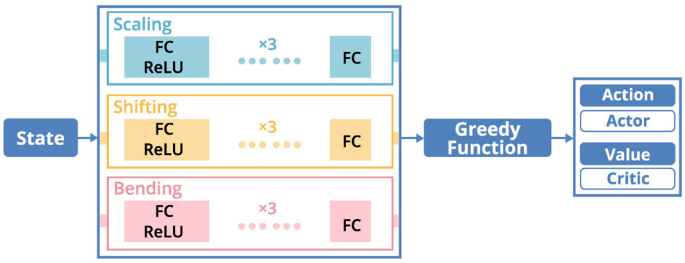

Fig. 4. Neural network architecture for the AI agent. **FC** refers to fully-connected layer, and **ReLU** represents an activation function. Three neural networks are employed to separately predict the three high-level interactions. A greedy function is used to obtain final actions and values.

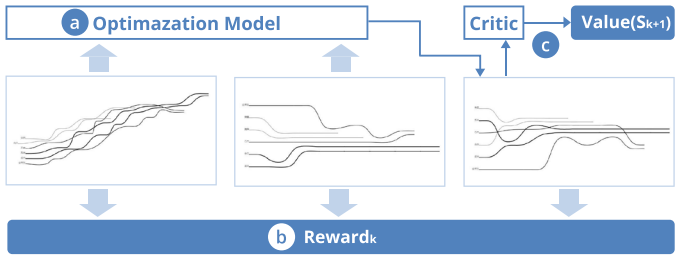

Fig. 5. Learning algorithm for the AI agent: (a) use the optimization model [25] to produce storylines; (b) measure the similarity between user-specified and "current" layouts to obtain the reward; (c) calculate the critic value to predict the "next" action.

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 [25, 45] 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_k 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 $P : \mathbb{S} \rightarrow \mathbb{A}$ 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 $o_i^j(L)$ indicates the ranking position of the i-th character at the j-th time slot for the layout L. Based on that, we formulate the ordering similarity between the user layout L_u and the layout $L^{(k)}$ at step k as

$$S_{order}^{(k)} = Comp(M_{order}^{L_u}, M_{order}^{L^{(k)}}) \tag{4}$$

Alignment Feature 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

$$S_{align}^{(k)} = Comp(M_{align}^{L_u}, M_{align}^{L^{(k)}}) \tag{5}$$

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

$$D_{pos}^{(k)} = Dist(M_{pos}^{L_u}, M_{pos}^{L^{(k)}}) \tag{6}$$

where $Comp(\cdot)$ is a counting function that self increment one if the corresponding values of two matrices are the same and $Dist(\cdot)$ is a distance function that calculates the difference between two matrices using Euclidean metric. We further employ sigmoid function $\overline{S}(\cdot)$ to normalize the three visual features. Based on that, we define the similarity between the user layout L_u and the k-th step layout $L^{(k)}$ using a linear scheme $S(k) = \omega_1 \overline{S}(S_{order}^{(k)}) + \omega_2 \overline{S}(S_{align}^{(k)}) + \omega_3 \overline{S}(D_{pos}^{(k)})$. The reward at k-th step can be obtained using $r(s_k, a_k) = S(k) - S(k+1)$. To make the final result resemble the user layout, we maximize the cumulative rewards in the whole episode using a discounted scheme that $R_k = \sum_{k'=k}^{K} \gamma^{k'} r(s_{k'}, a_{k'})$ with a discounting factor $\gamma \in [0, 1]$. The default parameters $[\omega_1, \omega_2, \omega_3, \gamma]$ are set to 1.

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. 4). In the end, we employ a greedy function to calculate the reward and determine the action.

$$R_k = \sum_{k'=k}^{K} \gamma^{k'} \max_{a_{k'}} r(s_{k'}, a_{k'}) \tag{7}$$

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 [26].

In a typical actor-critic model [32], researchers usually employ two neural networks to present *actor* and *critic*, respectively (see Fig. 5). 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 $\pi(a_t | s_t, \theta_\pi)$ 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 π can be defined as $V^\pi(s, \theta_V) = \mathbb{E}(R_k | s_k = s)$ that is the expected return for following policy π from state s. Hence, the problem of training a storyline agent is to obtain the parameters (θ_π, θ_V) of the neural networks for the policy function π and the value function V. The updates on the model parameters [6] can be written as

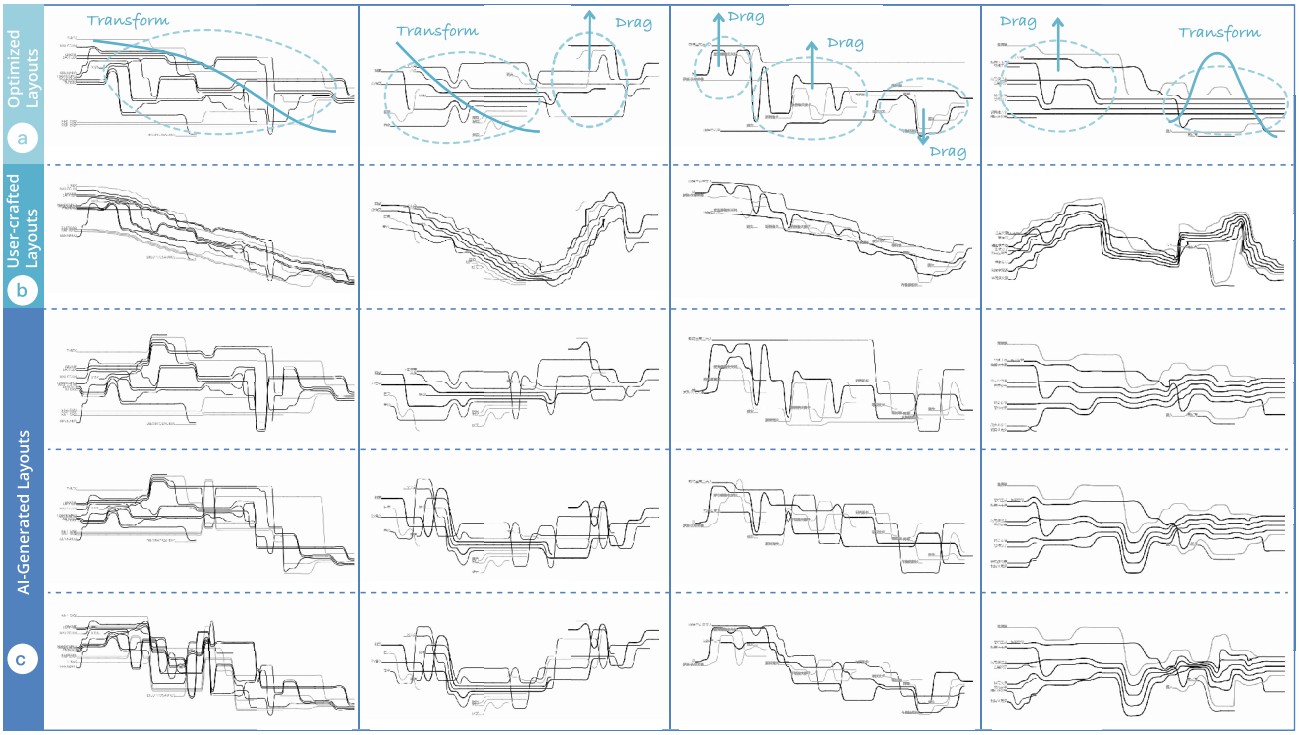

Fig. 6. Qualitative experiments: (a) the initially-optimized layouts generated by the optimization model [25]. (b) the user layouts modified by the interactions shown in (a). (c) the AI-generated layouts that resemble the user layouts but with improved aesthetic quality. The intermediate layouts at the k-th step ($k = 1, 5, 15$) are also presented to indicate how the AI agent reproduces the user layouts. The four cases are *Jurassic Park*, *WuKong*, *Moon and Sixpence*, *Justice League* (from left to right).

$$\Delta\theta_\pi \leftarrow \Delta\theta_\pi + \nabla_{\theta_\pi} log\pi(a_k|s_k;\theta_\pi)(R_k - V(s_k;\theta_V))$$
$$\Delta\theta_V \leftarrow \Delta\theta_V + \frac{\partial(R_k - V(s_k;\theta_V))^2}{\partial\theta_V}$$

where $\Delta\theta_\pi$ and $\Delta\theta_V$ are the updates applied to the model parameter θ_π and θ_V, respectively.

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.

6 Experiments

Implementation. We employ a client-server architecture to develop PlotThread. The web interface is implemented using TypeScript [3] and ECharts.js [9] while the server side is implemented using Python and the popular machine learning library PyTorch [2]. To support the flexible customization of storyline visualizations, we adopt the well-established graphic library, namely Paper.js [1]. We also develop a storyline layout optimizer which is implemented using C# to modularize PlotThread.

To validate the effectiveness of the reinforcement learning (**RL**) model, we conduct both quantitative and qualitative experiments on four datasets. The input stories are visualized using the optimization model [25] in Fig. 6a. We first show that the agent has leveraged both aesthetic and expressiveness in producing various types of storyline layouts. To simulate the real authoring process, we create four user layouts, including *incline-*, *stair-*, and *wave-layout* (see Fig. 6b), using the extended interactions that reshape the overall layouts without invoking optimization models. Apparently, the user layouts are twisted and do not satisfy the aesthetic goals, but they are regarded as more expressive in terms of the diverse visual forms. We then invoke the RL model to predict actions that can modify the initially-optimized layouts (Fig. 6a) to resemble the user layouts (Fig. 6b). The results (Fig. 6c) indicate that our RL model can successfully capture the visual features from the user layouts and produce more expressive layouts than the initially-optimized ones. Despite that the AI-generated layouts seem 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.

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. 8a, 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 difficulties in searching the design space in the beginning, it finally achieves better performances than the baseline method in the long term. The results indicate that our RL model has successfully learned how to "think" when designing storylines and can sacrifice short-term rewards to achieve long-term planning. Moreover, both the AI agent and the baseline method can converge to final layouts within 12 seconds (see Fig. 8b). In PlotThread, we set the default steps of the agent to 15 which ensures a satisfactory response time for users' interactions.

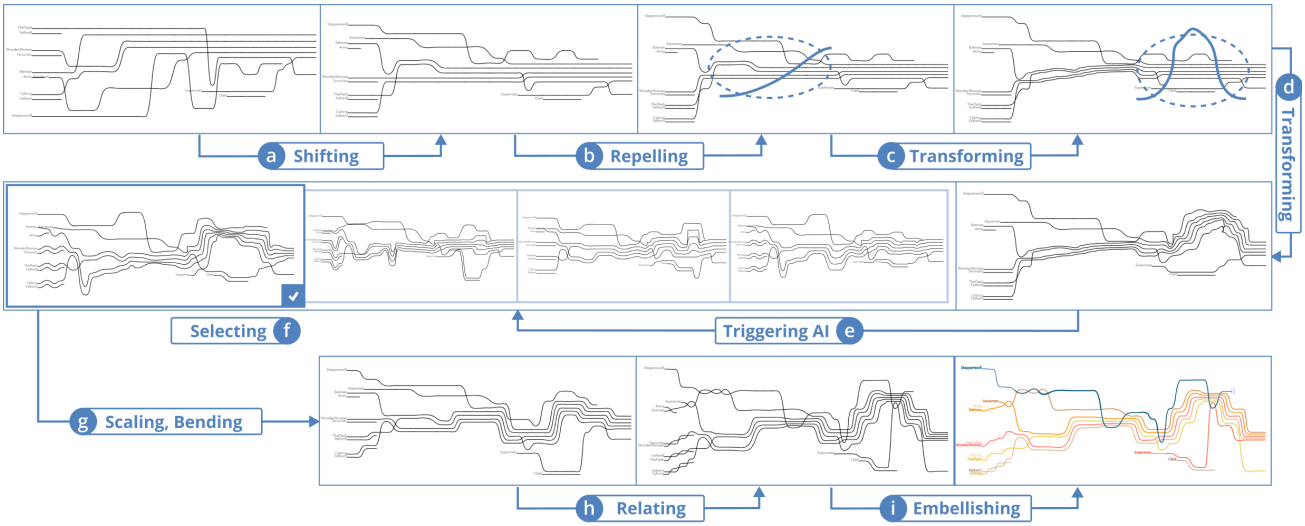

Fig. 7. This case illustrates the authoring process of the storyline visualization (*Justice League*) using PlotThread. The designer first customizes an initial layout through (a) *Shfiting*, (b) *Repelling*, (c) and (d) *Transforming* (Dashed ellipses indicate the transforming regions and solid paths represent the transforming shapes). The AI agent is then (e) triggered to produce suggestive storylines and (f) the desired one is selected. S/He further improves the AI-generated layout using high-level interactions, namely *Scaling* and *Bending* (g). The relationships among characters are revealed using (h) *Relating*, and the layout is (i) embellished to enrich the narration.

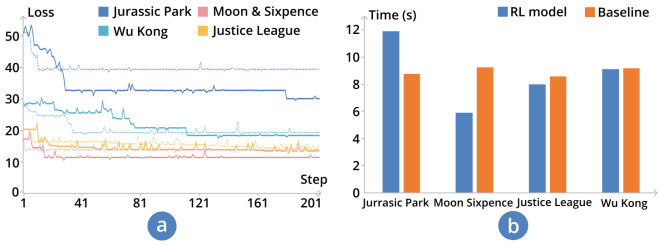

Fig. 8. Quantitative experiments: (a) x-axis represents the number of steps, and y-axis indicates the loss that is inverse of the cumulative rewards. The solid and dashed lines indicate the performance of the RL model and the baseline method, respectively. (b) y-axis indicates the running time of the RL model (blue) and the baseline method (Orange).

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 achieves 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 [38] 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 [45], 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. 7c). Next, he moves on to transform the shape of other stages, for example, in the climax stage, he draws a parabolic curve (Fig. 7d) 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. 7f) 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. 7b). He triggers the AI creator when he wants to improve the appearance and gain some inspirations about the storyline design (Fig. 7e). 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. 7g). 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. 7h). 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 (**EA**) was an artist who graduated from a national academy of art. She evaluated the output storylines crafted by PlotThread (see Fig. 1) and compared them with the storylines generated by the optimization model [25] and human artists [45]. The second 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 third expert (**EV**) was a senior researcher who studied visualization and visual analytics [29] for eight years. He evaluated the system development of PlotThread and discussed the potential applications for storyline visualizations. The interview includes a 30-min demonstration of PlotThread and a 30-min discussion.

EA 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. 1), 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.

ED mainly focused on the system design, including the human-AI collaborative workflow, the design of interactions, as well as the user interface. She was impressed by the interaction and interface design and commented that "the interactions are intuitive and the interface is easy to follow," but she also pointed out that users may need some training when they first use the system. To lower the learning cost, we will further improve the system with a user-friendly built-in user guide. When asked about the experience of human-AI collaboration, she commented that "users may doubt whether the AI agent can really understand their intentions, so they may be very reluctant to seek AI for help." This concern reveals a common trust issue widely existing in "black box" models. One possible solution is to provide an animation that demonstrates the evolution of layouts and how the AI agent modifies storylines. She also suggested that it would be helpful if users do not need to prepare story scripts because "it will challenge general users who do not have story scripts." Additionally, she provided a potential application of PlotThread that "it may be promising for preschool teachers to tell stories visually using PlotThread."

EV commented on the performance of the reinforcement learning algorithm and the applicability of PlotThread. He confirmed the effectiveness and expressiveness of the storylines (see Fig. 1) created by PlotThread. He mentioned that "the diverse visual forms of the storylines can arouse viewer's emotions, which I never expect from the optimization-based results." Due to the various visual elements proposed in the design space [7], we believe that we can further improve PlotThread and expand its applications. He suggested that "it can be helpful if the AI agent can guide users when they have no clues on how to start the design of storylines." This comment involves the trade-off between AI-driven and AI-guide systems where users or agents start the design process, respectively. To balance the two sides adequately, we plan to extend our RL framework to enable the agent to generate storylines from the input stories directly.

9 Discussion

We discuss the implications and limitations of PlotThread as follows:

Implications. Our work has several important implications. First, 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, they require users to have a deep understanding of the automatic generation process of storyline visualizations. Thus, non-expert users are usually limited in fully expressing their ideas and design talents when designing the layouts of storylines. Due to the assistance of the AI agent, PlotThread enables users to design storyline layouts flexibly without considering the aesthetic goals (**G1** to **G3**). The AI agent can resemble the user-specified layouts while preserving the aesthetic quality as much as possible. Thus, we believe PlotThread can serve numerous amateur users, which reflects the idea of "visualization for the mass."

Second, to the best of our knowledge, we are the first to apply reinforcement learning to the design of storyline visualizations. Despite recent studies indicate that machine learning techniques can be successfully applied to the design of data visualizations (e.g., graphs [18, 46] and charts [33]), it is still unknown whether storylines can be produced using machine learning approaches. To answer this issue, we employ reinforcement learning that formulates the design of storyline visualizations as a long-term delayed reward problem. The agent is trained to learn how designers typically "refine" initial storyline layouts to provide users possible suggestions of effective storyline layouts that follow the aesthetic goals. Our RL framework can inspire promising research frontiers in the field of visualization design. For example, researchers could first decompose a complicated design task into a set of design actions and then employ reinforcement learning to predict possible combinations of actions to construct data visualizations.

Third, we propose a mixed-initiative approach that incorporates predictive models and user feedbacks into interactive applications where users initiate and exploit the design task while computational agents explore the design space. Compared with a typical computer-assisted tool (e.g., iStoryline [45]), PlotThread intends to achieve a better trade-off between human creative work and automation by providing intelligence-level (not tool-level) assistance. Despite that the optimization-based approaches [25, 43, 44] have been improved significantly, we argue that it is necessary to integrate human intelligence into the design of storylines. Sitting in opposition to a perspective of pure automation, PlotThread provides a successful example where computational agents and people are seamlessly integrated to work on a shared problem, which can inspire the development of future visualization tools.

Limitations. Our work has several limitations. First, while there are various design tools to support the design of expressive storylines, PlotThread could be further improved to increase the artistry of the storyline visualizations. For instance, more diverse sketch styles could be employed to enrich the narration of storylines. Thus, we plan to develop more design tools to support the creative design of storylines. Second, even though the time efficiency of the AI agent is acceptable during the authoring process (see Fig. 8), it could be further improved to support more tightly collaborative designs. As a proof of concept, we have implemented PlotThread on a personal laptop, and we plan to improve the time performance of the AI agent via GPUs and parallel programming. Third, it is labor-intensive for users to prepare story scripts [45] that are necessary input for the storyline renderer [45]. To alleviate users' burden, we also plan to enable users to create storyline visualizations from scratch and investigate how to improve the collaborative design workflow progressively.

10 Conclusion

In this research, we develop PlotThread, a mixed-initiative authoring tool that seamlessly integrates computational agents and people to facilitate the easy design of storyline visualizations. The agent is designed to help users explore the design space [45] efficiently by providing a set of suggestive layouts, which can also inspire lateral thinking [8]. To develop such an intelligent agent, we formulate the design of storyline layouts as a reinforcement learning (**RL**) problem where the agent is trained to "refine" storyline layouts based on user-shared interactions. Moreover, we propose a novel framework and generate a collection of well-optimized storylines to address the two major challenges, namely *model architecture* and *model training*, raised by applying RL on designing storylines. We evaluate the effectiveness of our framework using qualitative and quantitative experiments and demonstrate the usage of PlotThread through a group of use cases. As future work, we plan to improve the time efficiency of the agent by employing parallel computing and extend the design tools of PlotThread to support more creative designs.

Acknowledgments

The work was supported by the joint Sino-German program of NSFC (61761136020), National Key R&D Program of China (2018YFB1004300), NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization (U1609217), Zhejiang Provincial Natural Science Foundation (LR18F020001) and the 100 Talents Program of Zhejiang University. This project was also partially funded by Microsoft Research Asia. Lingyun Yu is supported by XJTLU Research Development Funding RDF-19-02-11. Parts of this work were funded by German Science Foundation (DFG) as part of the project 'VAOST' (392087235).

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] Typescript, 2019. Retrieved Dec 1st, 2019 from https://www.typescriptlang.org/.

[4] D. Arendt and M. Pirrung. 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. Naddaf, 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):2471–2482, 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. Díaz, J. Petit, and M. Serna. A Survey of Graph Layout Problems. *ACM Computing Survey*, 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, 1999.

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

[17] M. Kempka, 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. A Deep Generative Model for Graph Layout. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):665–675, 2020.

[19] B. Lee, R. H. Kazi, and G. Smith. SketchStory: Telling More Engaging Stories with Data through Freeform Sketching. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2416–2425, 2013.

[20] A. Liapis, G. N. Yannakakis, C. Alexopoulos, and P. Lopes. Can computers foster human users' creativity? Theory and praxis of mixed-initiative co-creativity. 8(2):136–153, 2016.

[21] A. Liapis, G. N. Yannakakis, and J. Togelius. Sentient World: Human-based Procedural Cartography. In *Proceedings of the International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 180–191, 2013.

[22] 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. In *Proceedings of the International Conference on Learning Representations*, 2016.

[23] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE TVCG*, 23(1):91–100, 2017.

[24] S. Liu, G. Andrienko, Y. Wu, N. Cao, L. Jiang, C. Shi, Y.-S. Wang, and S. Hong. Steering data quality with visual analytics: The complexity challenge. *Visual Informatics*, 2(4):191–197, 2018.

[25] S. Liu, Y. Wu, E. Wei, M. Liu, and Y. Liu. StoryFlow: Tracking the Evolution of Stories. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2436–2445, 2013.

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 1928–1937, 2016.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[28] R. Munroe. Movie Narrative Charts. https://xkcd.com/657/, December 2009.

[29] T. Munzner. *Visualization Analysis and Design*. CRC press, 2014.

[30] D. G. Novick and S. Sutton. What is Mixed-Initiative Interaction? In *Proceedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction*, pages 114–116, 1997.

[31] M. Ogawa and K.-L. Ma. Software evolution storylines. In *Proceedings of the International Symposium on Software Visualization*, page 35–42, 2010.

[32] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[33] J. Poco and J. Heer. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Computer Graphics Forum*, 36(3):353–363, 2017.

[34] S. Racanière, T. Weber, D. Reichert, L. Buesing, A. Guez, D. Jimenez Rezende, A. Puigdomènech Badia, O. Vinyals, N. Heess, Y. Li, R. Pascanu, P. Battaglia, D. Hassabis, D. Silver, and D. Wierstra. Imagination-augmented agents for deep reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 5690–5701, 2017.

[35] 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.

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

[37] 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.

[38] Z. Snyder. *Justice League*. DC Films, 2017.

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

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

[41] 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.

[42] 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.

[43] Y. Tanahashi, C. H. Hsueh, 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.

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

[45] 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, 2019.

[46] 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.

[47] 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.

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

[49] 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.

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