StoryGAN: A Sequential Conditional GAN for Story Visualization

Yitong Li[1], Zhe Gan[2], Yelong Shen[4], Jingjing Liu[2], Yu Cheng[2], Yuexin Wu[5],
Lawrence Carin[1], David Carlson[1] and Jianfeng Gao[3]

[1]Duke University, [2]Microsoft Dynamics 365 AI Research, [3]Microsoft Research
[4]Tencent AI Research, [5]Carnegie Mellon University

Abstract

In this work, we propose a new task called Story Visualization. Given a multi-sentence paragraph, the story is visualized by generating a sequence of images, one for each sentence. In contrast to video generation, story visualization focuses less on the continuity in generated images (frames), but more on the global consistency across dynamic scenes and characters – a challenge that has not been addressed by any single-image or video generation methods. Therefore, we propose a new story-to-image-sequence generation model, StoryGAN, based on the sequential conditional GAN framework. Our model is unique in that it consists of a deep Context Encoder that dynamically tracks the story flow, and two discriminators at the story and image levels, to enhance the image quality and the consistency of the generated sequences. To evaluate the model, we modified existing datasets to create the CLEVR-SV and Pororo-SV datasets. Empirically, StoryGAN outperformed state-of-the-art models in image quality, contextual consistency metrics, and human evaluation.

1. Introduction

Learning to generate meaningful and coherent sequences of images from a natural language story is a challenging task that requires understanding and reasoning on both natural language and images. In this work, we propose a new *Story Visualization* task. Specifically, the goal is to generate a sequence of images to describe a story written in a multi-sentence paragraph, as shown in Figure 1.

There are two main challenges in this task. First, the sequence of images must consistently and coherently depict the whole story. This task is highly related to text-to-image generation [35, 28, 17, 36, 34], where an image is generated based on a short description. However, by sequentially applying text-to-image methods to a story will not generate a coherent image sequence, failing on the story visualization task. For instance, consider the story "A red metallic cylinder cube is at the center. Then add a green rubber cube at the right." The second sentence alone does not capture the entire scene.

The second challenge is how to display the logic of the storyline. Specifically, the appearance of objects and the layout in the background must evolve in a coherent way as the story progresses. This is similar to video generation. However, story visualization and video generation differ as: (*i*) Video clips are *continuous* with smooth motion transitions, so video generation models focus on extracting dynamic features to maintain realistic motions [32, 31].

Figure 1: The input story is "Pororo and Crong are fishing together. Crong is looking at the bucket. Pororo has a fish on his fishing rod." Each sentence is visualized with one image. In this work, the image generation for each sentence is enriched with contextual information from the Context Encoder. Two discriminators at different levels guide the generation process.

This work was done while the first author was an intern at Microsoft Dynamics 365 AI Research.

In contrast, the goal of story visualization is to generate a sequence of key static frames that present correct story plots where motion features are less important. (*ii*) Video clips are often based on simple sentence input and typically have a static background, while complex stories require the model to capture scene changes necessary for the plot line. In that sense, story visualization could also be viewed as a critical step towards real-world long-video generation by capturing sharp scene changes. To tackle these challenges, we propose a StoryGAN framework, inspired by Generative Adversarial Networks (GANs) [10], a two-player game between a generator and a discriminator. To take into account the contextual information in the sequence of sentence inputs, StoryGAN is designed as a sequential conditional GAN model.

Given a multi-sentence paragraph (story), StoryGAN uses a recurrent neural network (RNN) to incorporate the previously generated images into the current sentence's image generation. Contextual information is extracted with our Context Encoder module, including a stack of a GRU cell and our newly proposed Text2Gist cell. The Context Encoder transforms the current sentence and a story encoding vector into a high-dimensional feature vector (Gist) for further image generation. As the story proceeds, the Gist is dynamically updated to reflect the change of objects and scenes in the story flow. In the Text2Gist component, the sentence description is transformed into a filter and adapted to the story, so that we can optimize the mixing process by tweaking the filter. Similar ideas are also used in dynamic filtering [18], attention models [34] and meta-learning [27].

To ensure consistency across the sequence of generated images, we adopt a two-level GAN framework. We use an image-level discriminator to measure the relevance of a sentence and its generated image, and a story-level discriminator to measure the global coherence between the generated image sequence and the whole story.

We created two datasets from the existing CLEVR [19] and Pororo [21] datasets for our story visualization task, called CLEVR-SV and Pororo-SV, respectively. Empirically, StoryGAN more efficiently captures the full picture of the story and how it evolves, compared to existing baselines [36, 24]. Equipped with the deep Context Encoder module and the two-level discriminators, StoryGAN significantly outperforms previous state-of-the-art models, generating a sequence of higher quality images that are coherent with the story in both image quality and global consistency metrics, as well as human evaluation.

2. Related Work

Variational AutoEncoders (VAEs) [23], Generative Adversarial Nets (GANs) [10], and flow-based generative models [7, 8]) have been widely applied to a wide range of generation tasks including text-to-image generation, video generation, style transfer, and image editing. Story visualization falls into this broad categorization of generative tasks, but has several distinct aspects.

Very relevant for the story visualization task is conditional text-to-image transformation [28, 17, 38, 35], which can now generate high-resolution realistic images [36, 34]. A key task in text-to-image generation is understanding longer and more complex input texts. For example, this has been explored in dialogue-to-image generation, where the input is a complete dialogue session rather than a single sentence [29]. Another related task is textual image editing, which edits an input image according to a textual editing query [3, 30, 4, 9]. This task requires consistency between the original image and the output image. Finally, there is the task of placing pre-specified images and objects in a picture from a text description [20]. This task also relates text to a consistent image, but does not require a full image generation procedure.

A second closely related task to story visualization is video generation, especially that of text-to-video [24, 13] or image-to-video generation [1, 31, 32]. Existing approaches only generate short video clips [13, 5, 12] without scene changes. The biggest challenge in video generation is how to ensure a smooth motion transition across successive video frames. Trajectory, skeleton or simple landmarks are used in existing works to help model the motion feature [12, 37, 33]. To this end, researchers disentangle dynamic and static features for motion and background, respectively [32, 24, 31, 6]. In our modeling of story visualization, the whole story sets the static features and each input sentence encodes dynamic features. However, there are several differences: **(i)** conditional video generation has only one input, while our task has sequential, evolving inputs; and **(ii)** the motion in video clips is continuous, while images visualizing a story are discrete and often with different scene views.

There are also several other related tasks in the literature. For instance, story image retrieval from a pre-collected training set rather than image generation [26]. Cartoon generation has been explored with a "cut and paste" technique [11]. However, both of these techniques require large amounts of labeled training data. An inverse task to story visualization is visual storytelling, where the output is a paragraph describing a sequence of input images. Text generation models or reinforcement learning are often highlighted for visual storytelling [16, 25, 15].

3. StoryGAN

StoryGAN is designed to create a sequence of images to describe an input story $\mathbf{S}$. The story $\mathbf{S}$ consists of a sequence of sentences $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_T]$, where the length $\mathbf{T}$ may vary. There is one generated image per sentence, denoted as $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_T]$, that are both

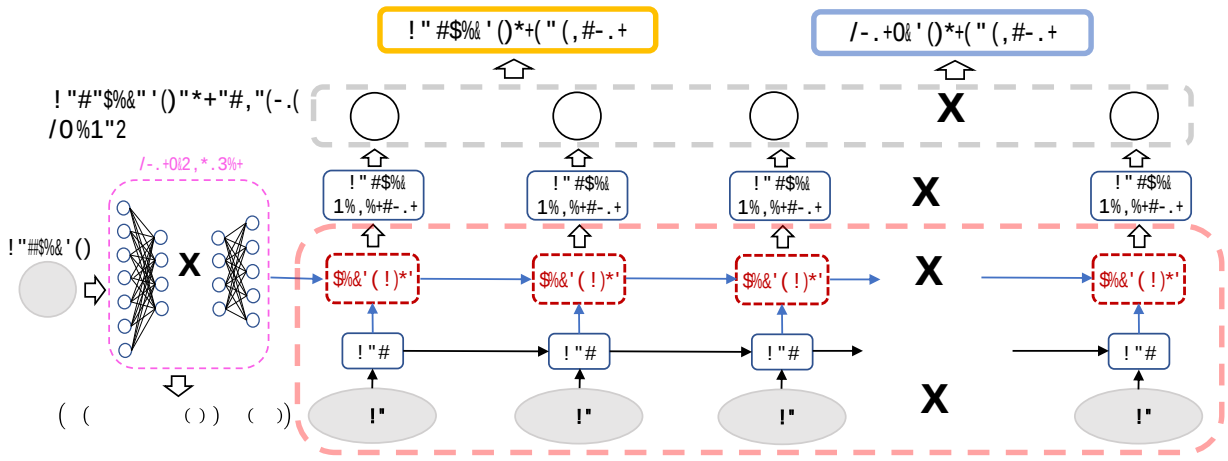

Figure 2: The framework of StoryGAN. The variables in gray solid circles are the input story S and individual sentences $s_1, \ldots, s_T$ with random noise $z_1, \ldots, z_T$. The generator network contains the Story Encoder, Context Encoder and image generator. The proposed component Text2Gist is introduced in detail in Section 3.2. There are two discriminators on top, which discriminate whether each image-sentence pair and each image-sequence-story pair are real or fake.

locally (sentence-image) and globally (story-images) consistent. For training, ground truth images are denoted as $\mathbf{X} = [x_1, x_2, \ldots, x_T]$. The image sequence is locally consistent if each image matches its corresponding sentence semantically. The image sequence is globally consistent if all the images globally hold together as coherent to the full story $\mathbf{S}$ it visualizes. In our approach, each sentence in story $\mathbf{S}$ has been encoded into an embedding vector using a pre-trained sentence encoder [2]. With slight abuse of notation, each sentence is an encoded via vector $\mathbf{s_t} \in R^{128}$. In the following, we assume $\mathbf{s_t}$ and $\mathbf{S}$ are both encoded feature vectors instead of raw text.

The overall architecture of StoryGAN is presented in Figure 2. It is implemented as a sequential GAN model, which consists of (*i*) a Story Encoder that encodes $\mathbf{S}$ into a low-dimensional vector $\mathbf{h_0}$; (*ii*) a two-layer recurrent neural network (RNN) based Context Encoder that encodes input sentence $\mathbf{s_t}$ and its contextual information into a vector $\mathbf{o_t}$ (Gist) for each time point $\mathbf{t}$; (*iii*) an image generator that generates image $\hat{\mathbf{x}}_\mathbf{t}$ based on $\mathbf{o_t}$ for each time step $\mathbf{t}$; and (*iv*) an image discriminator and a story discriminator that guide the image generation process so as to ensure the generated image sequence $\hat{\mathbf{X}}$ is locally and globally consistent, respectively.

3.1. Story Encoder

The Story Encoder is given in the dotted pink box of Figure 2. Following the conditioning mechanism in StackGAN [36], the Story Encoder $\mathbf{E}(\cdot)$ learns a stochastic mapping from story $\mathbf{S}$ to an low-dimensional embedding vector $\mathbf{h_0}$. $\mathbf{h_0}$ encodes the whole story and it serves as the initial state of the hidden cell of the Context Encoder. Specifically, the Story Encoder samples an embedding vector $\mathbf{h_0}$ from a normal distribution $\mathbf{h_0} \sim \mathbf{E(S)} = \mathbf{N}(\boldsymbol{\mu}(\mathbf{S}), \Sigma(\mathbf{S}))$, with $\boldsymbol{\mu}(\cdot)$ and $\Sigma(\cdot)$ implemented as two neural networks. In this work, we restrict $\Sigma(\mathbf{S}) = \mathrm{diag}(\sigma^2(\mathbf{S}))$ to a diagonal matrix for computational tractability. With the reparameterization trick, the encoded story $\mathbf{h_0}$ can be written as $\mathbf{h_0} = \boldsymbol{\mu}(\mathbf{S}) + \sigma^2(\mathbf{S})^{\frac{1}{2}} \odot \mathbf{z_S}$, where $\mathbf{z_S} \sim \mathbf{N}(0, \mathbf{I})$. $\odot$ represents elementwise multiplication, and the square root is also taken elementwise. $\boldsymbol{\mu}(\mathbf{S})$ and $\sigma^2(\mathbf{S})$ are parameterized as Multi-Layer Perceptrons (MLPs) with a single hidden layer. Convolutional networks could also be used depending on the structure of $\mathbf{S}$. The sampled $\mathbf{h_0}$ is provided to the RNN-based Context Encoder as the initial state vector.

By using stochastic sampling, the Story Encoder deals with the discontinuity problem in the original story space, thus not only leading to a compact, semantic representation of $\mathbf{S}$ for story visualization, but also adding randomness to the generation process. The encoder's parameters are optimized jointly with the other modules of StoryGAN via back propagation. Therefore, to enforce the smoothness over the conditional manifold in latent semantic space and avoid collapsing to a single generative point rather than a distribution, we add the regularization term [36],

$$L_{KL} = KL\left(N\left(\boldsymbol{\mu}(\mathbf{S}), \mathrm{diag}(\sigma^2(\mathbf{S}))\right) \| N(0, \mathbf{I})\right), \quad (1)$$

which is the Kullback-Leibler (KL) divergence between the learned distribution and the standard Gaussian distribution.

3.2. Context Encoder

Video generation is closely related to story visualization, and it typically assumes a static background with smooth motion transitions, requiring a disjoint embedding of static and dynamic features [32, 16, 31]. In story visualization, the

challenge differs in that the characters, motion, and background often change from image to image, as illustrated in Figure 1. This requires us to address two problems: (*i*) how to update the contextual information to effectively capture background changes; and (*ii*) how to combine new inputs and random noise when generating each image to visualize the change of characters, which may shift dramatically.

We address these issues by proposing a deep RNN based Context Encoder to capture contextual information during sequential image generation, shown in the red box in Figure 2. The context can be defined as any related information in the story that is useful for the current generation. The deep RNN consists of two hidden layers. The lower layer is implemented using standard GRU cells and the upper layer using the proposed Text2Gist cells, which are a variant to GRU cells and are detailed below. At time step $\mathbf{t}$, the GRU layer takes as input the concatenation of the sentence $\mathbf{s_t}$ and isometric Gaussian noise ϵ_t, and outputs the vector $\mathbf{i_t}$. The Text2Gist cell combines the GRU's output $\mathbf{i_t}$ with the story context $\mathbf{h_t}$ (initialized by Story Encoder) to generate $\mathbf{o_t}$ that encodes all necessary information for generating an image at time $\mathbf{t}$. $\mathbf{h_t}$ is updated by the Text2Gist cell to reflect the change of potential context information.

Let $\mathbf{g_t}$ and $\mathbf{h_t}$ denote the hidden vectors of the GRU and Text2Gist cells, respectively. The Context Encoder works in two steps to generate its output:

$$\mathbf{i_t}, \mathbf{g_t} = \text{GRU}(\mathbf{s_t}, \epsilon_t, \mathbf{g_{t-1}}), \tag{2}$$

$$\mathbf{o_t}, \mathbf{h_t} = \text{Text2Gist}(\mathbf{i_t}, \mathbf{h_{t-1}}). \tag{3}$$

We call $\mathbf{o_t}$ the *"Gist" vector* since it combines all the global and local context information, from $\mathbf{h_{t-1}}$ and $\mathbf{i_t}$ respectively, at time step $\mathbf{t}$ (i.e. it captures the "gist" of the information). The Story Encoder initializes $\mathbf{h_0}$, while $\mathbf{g_0}$ is randomly sampled from an isometric Gaussian distribution.

Next, we give the underlying updates of Text2Gist. Given $\mathbf{h_{t-1}}$ and $\mathbf{i_t}$ at time step $\mathbf{t}$, Text2Gist generates a hidden vector $\mathbf{h_t}$ and an output vector $\mathbf{o_t}$ as follows:

$$\mathbf{z_t} = \sigma_z(\mathbf{W_z i_t} + \mathbf{U_t h_{t-1}} + \mathbf{b_z}), \tag{4}$$

$$\mathbf{r_t} = \sigma_r(\mathbf{W_r i_t} + \mathbf{U_r h_{t-1}} + \mathbf{b_r}), \tag{5}$$

$$\mathbf{h_t} = (1 - \mathbf{z_t}) \odot \mathbf{h_{t-1}} + \mathbf{z_t} \odot \sigma_h(\mathbf{W_h i_t} + \mathbf{U_h}(\mathbf{r_t} \odot \mathbf{h_{t-1}}) + \mathbf{b_h}), \tag{6}$$

$$\mathbf{o_t} = \text{Filter}(\mathbf{i_t}) \odot \mathbf{h_t}, \tag{7}$$

where $\mathbf{z_t}$ and $\mathbf{r_t}$ are the outputs from the update and reset gates, respectively. The update gate decides how much information from the previous step should be kept, and the reset gate determines what to forget from $\mathbf{h_{t-1}}$. σ_z, σ_r and σ_h are sigmoid non-linearity functions. In contrast to standard GRU cells, output $\mathbf{o_t}$ is the convolution between Filter$(\mathbf{i_t})$ and $\mathbf{h_t}$. The filter $\mathbf{i_t}$ is learned to adapt to $\mathbf{h_t}$. Specifically, Filter$(\mathbf{i_t})$ transforms vector $\mathbf{i_t}$ to a multi-channel filter of size $\mathbf{C_{out}} \times \mathbf{1} \times \mathbf{1} \times \text{len}(\mathbf{h_t})$ using a neural network, where $\mathbf{C_{out}}$ is the number of output channels. Since $\mathbf{h_t}$ is a vector, this filter is used as a 1D filter as in a standard convolutional layer.

The convolution operator in Eq. (7) infuses the global contextual information from $\mathbf{h_t}$ and local information from $\mathbf{i_t}$. $\mathbf{o_t}$ is the output of the Text2Gist cell at time step $\mathbf{t}$. Since $\mathbf{i_t}$ encodes information from $\mathbf{s_t}$ and $\mathbf{h_t}$ from $\mathbf{S}$, which reflects the whole picture of the story, the convolutional operation in Eq. (7) can be seen as helping $\mathbf{s_t}$ to pick out the important part from the story in the process of generation. Empirically, we find that Text2Gist is more effective than traditional RNNs for story visualization.

3.3. Discriminators

StoryGAN uses two discriminators, an image and a story discriminator, to ensure the local and global consistency of the story visualization, respectively. The image discriminator measures whether the generated image $\hat{\mathbf{x}}_\mathbf{t}$ matches the sentence $\mathbf{s_t}$ given its initial context information encoded in $\mathbf{h_0}$. It does this by comparing the generated triplet $\{\mathbf{s_t}, \mathbf{h_0}, \hat{\mathbf{x}}_\mathbf{t}\}$ to the real triplet $\{\mathbf{s_t}, \mathbf{h_0}, \mathbf{x_t}\}$. In contrast to prior work on text-to-image generation [36, 28], the same sentence can have a significantly different generated image depending on the context, so it is important to give the encoded context information to the discriminator as well. For example, consider the example given in Section 1, "A red metallic cylinder cube is at the center. Then add a green rubber cube at the right of it." The second image will vary wildly without the context (i.e. the first sentence).

Figure 3: Structure of the story discriminator. The feature vectors of the images/sentences in the story are concatenated. $\odot$ means elementwise product. The product of image and text features are input to a fully connected layer with sigmoid non-linearity to predict whether it is a fake or real story pair.

The story discriminator helps enforce the global consistency of the generated image sequence given story $\mathbf{S}$. It differs from the discriminators used for video generation, which often use 3D convolution [32, 31, 24] to smooth

the changes between frames. The overall architecture of the story discriminator is illustrated in Figure 3. The left part is an image encoder, which encodes an image sequence into a sequence of feature vectors $\mathbf{E_{img}(X)} = [\mathbf{E_{img}}(\mathbf{x_1}), \ldots, \mathbf{E_{img}}(\mathbf{x_T})]$, where $\mathbf{X}$ are either real or generated images (which are denoted by $\hat{\mathbf{X}}$). These vectors are concatenated into a single vector, shown as the blue rectangular in Fig. 3. Similarly, the right part is a text encoder, which encodes the multi-sentence story $\mathbf{S}$ into a sequence of feature vectors $\mathbf{E_{txt}(S)} = [\mathbf{E_{txt}}(\mathbf{s_1}), \ldots, \mathbf{E_{txt}}(\mathbf{s_T})]$. Likewise, these are concatenated into one big vector, shown as the red rectangle in Fig. 3. The image encoder is implemented as a deep convolutional network and the text encoder as a multi-layer perceptron. Both output a same dimensional vector.

The global consistency score is computed as

$$\mathbf{D_S} = \sigma(\mathbf{w}^{\top}(\mathbf{E_{img}(X)} \odot \mathbf{E_{txt}(S)}) + \mathbf{b}), \quad (8)$$

where $\odot$ is element-wise product. The weights $\mathbf{w}$ and bias $\mathbf{b}$ are learned in the output layer. σ is a sigmoid function that normalizes the score to a value in $[0, 1]$. By pairing each sentence and image, the story discriminator can consider both local matching and global consistency jointly.

Both image and story discriminators are trained on positive and negative pairs. The latter are generated by replacing the image (sequence) in the positive pairs with generated ones.

3.4. Algorithm Outlines

Let θ, ψ_I, and ψ_S denote the parameters of the whole generator $\mathbf{G}(\cdot;\theta)$, the image discriminator, and the story discriminator, respectively. The objective function for StoryGAN is

$$\min_{\theta} \max_{\psi_I, \psi_S} \alpha \mathbf{L_{Image}} + \beta \mathbf{L_{Story}} + \mathbf{L_{KL}}, \quad (9)$$

where α and β balance the three loss terms. $\mathbf{L_{KL}}$ is the regularization term of the Story Encoder defined in (1). $\mathbf{L_{Image}}$ and $\mathbf{L_{Story}}$ are defined as

$$\mathbf{L_{Image}} = \sum_{t=1}^{T} (\mathbb{E}_{(\mathbf{x_t}, \mathbf{s_t})}[\log \mathbf{D_I}(\mathbf{x_t}, \mathbf{s_t}, \mathbf{h_0}; \psi_I)] + \mathbb{E}_{(\hat{\mathbf{x}}_t, \mathbf{s_t})}[\log(1 - \mathbf{D_I}(\mathbf{G}(\hat{\mathbf{x}}_t, \mathbf{s_t}; \theta), \mathbf{s_t}, \mathbf{h_0}; \psi_I))]) \quad (10)$$

$$\mathbf{L_{Story}} = \mathbb{E}_{(\mathbf{X}, \mathbf{S})}[\log \mathbf{D_S}(\mathbf{X}, \mathbf{S}; \psi_S)] + \mathbb{E}_{(\hat{\mathbf{X}}, \mathbf{S})}[\log(1 - \mathbf{D_S}([\mathbf{G}(\hat{\mathbf{x}}_t, \mathbf{s_t}; \theta)]_{t=1}^{T}), \mathbf{S}; \psi_S))]. \quad (11)$$

$\mathbf{D_I}(\cdot;\psi_I)$ and $\mathbf{D_S}(\cdot;\psi_S)$ are the image and story discriminator, parameterized by ψ_I and ψ_S, respectively.

The pseudo-code for training StoryGAN is given in Algorithm 1. The parameters of the story and image discriminators, ψ_I and ψ_S, are updated in two separate **for** loops, respectively, while the parameters of the image generator θ are updated in both loops. The initial hidden state of the Text2Gist layer is the encoded story feature vector $\mathbf{h_0}$ produced by the Story Encoder. The detailed configuration of the network is provided in Appendix A.

Algorithm 1 Training Procedure of StoryGAN

Input: Encoded sentence vectors $\mathbf{S_n} = [\mathbf{s_{n1}}, \mathbf{s_{n2}}, \ldots, \mathbf{s_{nT}}]$ and corresponding images $\mathbf{X_n} = [\mathbf{x_{n1}}, \ldots, \mathbf{x_{nT}}]$ for $\mathbf{n} = 1, \ldots, \mathbf{N}$.
Output: Generator parameters θ and discriminator parameters ψ_I and ψ_S.

```
for iter = 1 to max_iter do
  for iter_I = 1 to k_I do
    Sample a mini-batch of story-sentence pairs {(s_t, S, x_t)} from the training set.
    Compute h_0 as the initialization of the Text2Gist layer and the KL regularization term as Eq. (1).
    Generate a single output image x̂.
    Update ψ_I and θ.
  end for
  for iter_S = 1 to k_S do
    Sample a mini-batch of story-image pair {(S, X)} from training set.
    Compute h_0 and update h_t at each time step t
    Generate image sequence X̂.
    Update ψ_S and θ.
  end for
end for
```

In our experiments, we use Adam [22] for parameter updates[1]. We also find that using different mini-batch sizes for image and story discriminators may accelerate training convergence, and that it is beneficial to update generator and discriminator in different time steps in one epoch.

4. Experiment

In this section, we evaluate the StoryGAN model on one toy and one cartoon dataset. To the best of our knowledge, there is no existing work on our proposed story visualization task. The closest alternative for story visualization is conditional video generation [24], where the story is treated as single input and a video is generated in lieu of the sequence of images. However, we empirically found that the video generation result is too blurry and not comparable to StoryGAN. Thus, our comparisons are mainly to ablated versions of our proposed model. For a fair comparison, all models use the same structure of the image generator, Context Encoder and discriminators when applicable. The compared

[1]Code is available at https://github.com/yitong91/StoryGAN

baseline models are:

ImageGAN: ImageGAN follows the work in [28, 36] and does not use the story discriminator, story encoder and Context Encoder. Each image is generated independently. However, for a reasonable comparison, we concatenate $\mathbf{s_t}$, encoded story $\mathbf{S}$ and a noise term as input. Otherwise, the model fails on the task. This is the simplest version of StoryGAN.

SVC: In "Story Visualization by Concatenation" (SVC), the Text2Gist cell in StoryGAN is replaced by simple concatenation of the encoded story and description feature vectors [31]. Compared to ImageGAN, SVC includes the additional story discriminator, and is visualized in Figure 4.

Figure 4: The framework of the baseline model SVC, where the story and individual sentence are concatenated to form the input.

SVFN: In "Story Visualization by Filter Network" (SVFN), the concatenation in SVC is replaced by a filter network. Sentence $\mathbf{s_t}$ is transformed into a filter and convolved with the encoded story. Specifically, the image generator input is $\mathbf{o_t} = \text{Filter}(\mathbf{i_t}) \circledast \mathbf{h_0}$ instead of Eq. 7.

4.1. CLEVR-SV Dataset

The CLEVR [19] dataset was originally used for visual question answering. We modified this data for story visualization by generating images from randomly assigned layouts of the object (examples in the top row of Figure 5). We named this dataset CLEVR-SV to distinguish it from the existing CLEVR dataset. Specifically, four rules were used to construct the CLEVR-SV: **(i)** The maximum number of objects in one story is limited to four. **(ii)** Objects are made of metallic/rubber with eight different colors and two different sizes. **(iii)** The object shape can be cylinder, cube or sphere. **(iv)** The object is added one at a time, resulting in a four-image sequence per story. We generated $\mathbf{10,000}$ image sequences for training and $\mathbf{3,000}$ for testing. For our task, the story is the layout descriptions of objects.

The input $\mathbf{s_t}$ is the current object's attribute and the relative position given by two real numbers indicating its coordinates. For instance, the first image of the left column of Fig. 5 is generated from "yellow, large, metal, sphere, (-2.1, 2.4)." The following objects are described in the same way. Given the description, the generated objects' appearance should have little variation from the ground truth and their relative positions should be similar.

Figure 5 gives the results comparison. ImageGAN [28] fails to keep the consistency of the 'story' and it mixes up the attributes when the number of objects increases. SVC solves this consistency problem by including the story discriminator and GRU cell at the bottom, as the third row of Figure 5 has consistent objects in the image sequence. However, SVC generates an implausible forth image in the sequence. We hypothesize that using simple vector concatenation cannot effectively balance the importance of the current description with the whole story. SVFN can alleviate this problem to some extent, but not completely. In contrast, StoryGAN generates more feasible images than the competitors. We attribute the performance improvement to three components: **(i)** Text2Gist cell tracks the progress of story; **(ii)** story and image discriminators keep the consistency of objects in the generation process; **(iii)** using the Story Encoder to initialize the Text2Gist cell gives better result on first generated image. Greater empirical evidence for this final point appears in the cartoon dataset in Section 4.2.

In order to further validate the StoryGAN model, we designed a task to evaluate whether the model can generate consistent images by changing the first sentence description. Specifically, we randomly replaced the first object's description while keeping the other three the same during generation, which we visualize in Supplemental Figure 8 in Appendix B. This comparison shows that only StoryGAN can keep the story consistency by correctly utilizing the attributes of the first object in later frames, as discussed above. In Supplemental Figure 9, we give additional examples on changing the initial attributes only using StoryGAN. Regardless of the initial attribute, StoryGAN is consistent between frames.

	ImageGAN [28]	SVC	SVFN	StoryGAN
SSIM	0.596	0.641	0.654	0.672

Table 1: SSIM comparison on CLEVR-SV dataset.

We also compare the Structural Similarity Index (SSIM) score between the generated images and ground truth [14]. SSIM was originally used to measure the recovery result from distorted images. Here, it is used to determine whether the generated images are aligned with the input description. Table 1 gives the SSIM metric for each method on the test set. Note that though this is a generative task, using SSIM to measure the structure similarity is reasonable because there is little variation given the description. In this task, StoryGAN significantly outperforms the other baselines.

4.2. Cartoon Dataset

The Pororo dataset [21] was originally used for video question answering, where each one second video clip is as-

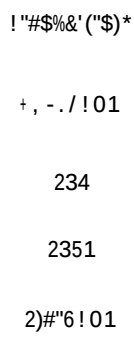

Figure 5: Comparison among different methods on CLEVR-SV dataset.

Figure 6: Two generated samples on the Pororo-SV dataset.

sociated with more than one manually written description. About **40** video clips forms a complete story. Each story has several QA pairs. In total, the Pororo dataset contains **16**K clips of one second videos about **13** distinct characters. The manually written description has an average length of **13.6** words that describes what is happening and which characters are in each video clip. These **16**K video clips are sorted into **408** movie stories [21].

We modified the Pororo dataset to fit story visualization task by considering the description for each video clip as the story's text input. For each video clip, we randomly pick out one frame (sampling rate is 30Hz) during training as the real image sample. Five continuous images form a single story. Finally, we end up with **15, 336** description-story pairs, where **13, 000** pairs are used as training, the remaining **2, 336** pairs for testing. We call this dataset Pororo-SV to differ it from the original Pororo QA dataset [21].

The text encoder uses universal encoding [2] with fixed pre-trained parameters. Training a new text encoder empirically gave little performance gain. Two visualized stories from the competing methods are given in Figure 6. The text input is given on the top. ImageGAN does not generate consistent image sequences; for instance, the generated images switch from indoors to outdoors randomly. Additionally, the characters' appearance is inconsistent in the sequence of images (e.g. Pororo's hat). SVC and SVFN can improve the consistency to some extent, but their limitations can be seen in the unsatisfactory first images. In contrast, StoryGAN's first image has a much higher quality than other baselines because of the use of the Story Encoder to initialize the recurrent cell. This shows the advantage of using the output of the Story Encoder as first hidden state over random initialization.

To explore how different models represent the story, we ran experiments where only the character names in the story were changed, shown in Figure 7. Visually, StoryGAN outperforms the other baselines on the image quality and consistency.

Further, we perform two distinct quantitative tasks. The first is to determine whether the generation is able to cap-

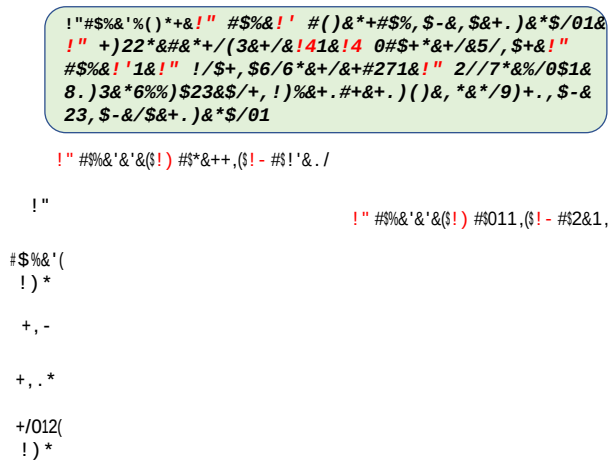

Figure 7: Generation result by changing character names in the same story. The story template is given at the top, with the character names c1, c2 and c3 in the two instances of the story, each one shown in a column.

ture the relevant characters in the story. The nine most common characters are selected from the dataset. Their names and pictures are provided in Supplemental Figure 9 in Appendix D. Next, a character image classifier is trained on real images from training set and applied on both real and generated images from the test set. We compare the classification accuracy (only exact matches across all characters counts as correct) of each image/story pair as an indicator of whether the generation is coherent to the story description. The classifier's performance on the test set is 86%, which is considered an upper bound for the task. From these results, it is clear that StoryGAN has increased character consistency compared to the baseline models. Note that there is peculiarity in the labels, as the human labeled description can sometimes include characters not shown in the frame. Further, the training classifier is on real images. Domain gap between real and generated images also harm the performance. However, these should hurt all algorithms equally and it is a fair comparison.

	Upper Bound	ImageGAN [28]	SVC	SVFN	StoryGAN
Acc.	0.86	0.23	0.21	0.24	0.27

Table 2: Character classification accuracy (exact match ratio) comparison on Pororo-SV dataset. The upper bound is the classifier accuracy on the real images associated with the stories.

Human Evaluation Automatic metrics cannot fully evaluate the performance of StoryGAN. Therefore, we performed both pairwise and ranking-based human evaluation studies on Amazon Mechanical Turk on Pororo-SV. For both tasks, we use 170 generated image sequences sampled from the test set, each assigned to 5 workers to reduce human variance. The order of the options within each assignment is shuffled to make a fair comparison.

Table 3: Results of pairwise human evaluation. The ± denotes standard error on the metrics.

	StoryGAN vs ImageGAN		
Choice (%)	StoryGAN	ImageGAN	Tie
Visual Quality	74.17 ± 1.38	18.60 ± 1.38	7.23
Consistence	79.15 ± 1.27	15.28 ± 1.27	5.57
Relevance	78.08 ± 1.34	17.65 ± 1.34	4.27

Table 4: Results of ranking-based human evaluation. The ± denotes standard error on the metrics.

Method	ImageGAN	SVC	SVFN	StoryGAN
Rank	2.91 ± 0.05	2.42 ± 0.04	2.77 ± 0.04	1.94 ± 0.05

We first performed a pairwise comparison between StoryGAN and ImageGAN. For each input story, the worker is presented with two generated image sequences and asked to make decisions from the three aspects: visual quality[2], consistency[3], and relevance[4]. Results are summarized in Table 3. The standard error on these estimates is small, demonstrating that StoryGAN drastically outperformed ImageGAN on this task.

We next performed ranking-based human evaluation. For each input story, the worker is asked to rank images generated from the four compared models on their overall quality. Results are summarized in Table 4. StoryGAN achieves the highest average rank, while ImageGAN performs the worst. There is little uncertainty in these estimates, so we are confident that humans prefer StoryGAN on average.

5. Conclusion

We studied the story visualization task as a sequential conditional generation problem. The proposed StoryGAN model deals with the task by jointly considering the current input sentence with the contextual information. This is achieved by the proposed Text2Gist component in the Context Encoder. From the ablation test, the two-level discriminator and the recurrent structure on the inputs helps ensure the consistency across the generated images and the story to be visualized, while the Context Encoder efficiently provides the image generator with both local and global conditional information. Both quantitative and human evaluation studies show that StoryGAN improves the generation compared to the baseline models. As image generators improve, the story visualization's quality will improve also.

[2]The generated images look visually appealing, rather than blurry and difficult to understand.

[3]The generated images are consistent with each other, have a common topic hidden behind, and naturally forms a story, rather than looking like 5 independent images.

[4]The generated image sequence accurately reflects the input story and covers the main characters mentioned in the story.

References

[1] Haoye Cai, Chunyan Bai, Yu-Wing Tai, and Chi-Keung Tang. Deep video generation, prediction and completion of human action sequences. *arXiv preprint arXiv:1711.08682*, 2018. 2

[2] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018. 3, 7

[3] Jianbo Chen, Yelong Shen, Jianfeng Gao, Jingjing Liu, and Xiaodong Liu. Language-based image editing with recurrent attentive models. *CVPR*, 2018. 2

[4] Yu Cheng, Zhe Gan, Yitong Li, Jingjing Liu, and Jianfeng Gao. Sequential attention gan for interactive image editing via dialogue. *arXiv preprint arXiv:1812.08352*, 2018. 2

[5] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *ICML*, 2018. 2

[6] Emily L Denton et al. Unsupervised learning of disentangled representations from video. In *NIPS*, 2017. 2

[7] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 2

[8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 2

[9] Alaaeldin El-Nouby, Shikhar Sharma, Hannes Schulz, Devon Hjelm, Layla El Asri, Samira Ebrahimi Kahou, Yoshua Bengio, and Graham W Taylor. Keep drawing it: Iterative language-based image generation and editing. *arXiv preprint arXiv:1811.09845*, 2018. 2

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[11] Tanmay Gupta, Dustin Schwenk, Ali Farhadi, Derek Hoiem, and Aniruddha Kembhavi. Imagine this! scripts to compositions to videos. *ECCV*, 2018. 2

[12] Zekun Hao, Xun Huang, and Serge Belongie. Controllable video generation with sparse trajectories. In *CVPR*, 2018. 2

[13] Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic video generation using holistic attribute control. *arXiv preprint arXiv:1803.08085*, 2018. 2

[14] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *ICPR*, 2010. 6

[15] Qiuyuan Huang, Zhe Gan, Asli Celikyilmaz, Dapeng Wu, Jianfeng Wang, and Xiaodong He. Hierarchically structured reinforcement learning for topically coherent visual story generation. *arXiv preprint arXiv:1805.08191*, 2018. 2

[16] Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. Visual storytelling. In *NAACL*, 2016. 2, 3

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1, 2

[18] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NIPS*, 2016. 2

[19] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017. 2, 6

[20] Jin-Hwa Kim, Devi Parikh, Dhruv Batra, Byoung-Tak Zhang, and Yuandong Tian. Codraw: Visual dialog for collaborative drawing. *arXiv preprint arXiv:1712.05558*, 2017. 2

[21] Kyung-Min Kim, Min-Oh Heo, Seong-Ho Choi, and Byoung-Tak Zhang. Deepstory: Video story qa by deep embedded memory networks. In *IJCAI*, 2017. 2, 6, 7

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[24] Yitong Li, Martin Renqiang Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. *AAAI*, 2018. 2, 4, 5

[25] Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, and Eric P Xing. Recurrent topic-transition gan for visual paragraph generation. *arXiv preprint arXiv:1703.07022*, 2017. 2

[26] Hareesh Ravi, Lezi Wang, Carlos Muniz, Leonid Sigal, Dimitris Metaxas, and Mubbasir Kapadia. Show me a story: Towards coherent neural story illustration. In *CVPR*, 2018. 2

[27] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *CVPR*, 2018. 2

[28] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *ICML*, 2016. 1, 2, 4, 6, 8

[29] Shikhar Sharma, Dendi Suhubdy, Vincent Michalski, Samira Ebrahimi Kahou, and Yoshua Bengio. Chatpainter: Improving text to image generation using dialogue. *arXiv preprint arXiv:1802.08216*, 2018. 2

[30] Rakshith Shetty, Mario Fritz, and Bernt Schiele. Adversarial scene editing: Automatic object removal from weak supervision. *arXiv preprint arXiv:1806.01911*, 2018. 2

[31] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. *CVPR*, 2018. 1, 2, 3, 4, 6

[32] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NIPS*, 2016. 1, 2, 3, 4

[33] Wei Wang, Xavier Alameda-Pineda, Dan Xu, Pascal Fua, Elisa Ricci, and Nicu Sebe. Every smile is unique: Landmark-guided diverse smile generation. In *CVPR*, 2018. 2

[34] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. *CVPR*, 2018. 1, 2

[35] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016. 1, 2

[36] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017. 1, 2, 3, 4, 6

[37] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris Metaxas. Learning to forecast and refine residual motion for image-to-video generation. In *ECCV*, 2018. 2

[38] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017. 2