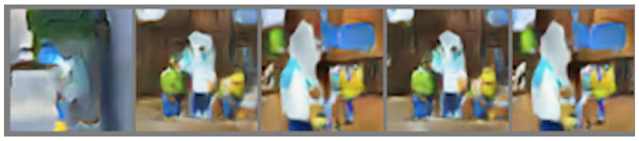
LongStoryGAN: A Chronological Illustration Generation Framework For Documents

Yiqing Xie, Yanbo Xu and Zhuoxuan Peng
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{yxieal, yxubu, zpengac}@connect.ust.hk

Abstract

Story visualization is a newly proposed task that combines computer vision with natural language processing. While neural image generation methods have enabled story visualization from image descriptions in previous works, they are not well suited to general text documents. To close this gap, we present a chronological illustration generation framework, LongStoryGAN, which expands the input of story visualization to general documents. Our framework contains two major modules: the text summarization module and the image generation module. The text summarization module applies the TextRank algorithm [8] to extract sentences in chronological order that contain the action of character. The image generation module is based on a modified state-of-the-art model on story visualization, StoryGAN [7], which generates a series of images describing these sentences. Extensive experiments show that LongStoryGAN is able to illustrate a relatively long story using images. In particular, we compare both modules with their individual ablations on corresponding datasets. Both qualitative and quantitative results show that our modules outperform their own ablation.


(a) The result generated by LongStoryGAN

(b) The ground truth

Figure 1: The input story is "Pororo and Crong agrees to what Loopy asked them of. The woods are covered with snow. The sky is blue and clear.","Loopy went away to get a drink for her friends Pororo and Crong. They are in Loopy's house. There is a plant flower on the table.","Pororo raised the plant flower that was on the table. Crong is watching the plant right next to him. They are in Loopy's house.","Loopy brings some drinks for her friends Pororo and Crong. They are in Loopy's house.","Pororo asks Loopy if Pororo can have the flower. Pororo Crong and Loopy are drinking at a table in Loopy's house."

1. Introduction

Storytelling has been one of the major ways by which humans pass knowledge, spread information, and is crucial in the entire history. During the process, we humans always visualize those texts using our pre-learned knowledge. With the rapid development in computer science areas including Computer Vision, Natural Language Processing, a natural question to ask is whether the machine could perform such a task automatically. In this work, we propose LongStoryGAN, a chronological illustration generation framework for documents to describe the task that given a relatively long story or text document, a series of images would be generated to describe it.

There are some major problems in this task. The first one is **Large and noisy input space**. In previous works [7], images are generated from a series of descriptive sentences. There are normally only 5-10 sentences in a full story. An example of input data is shown in the caption of Figure 1. However, the scale of general documents such as fairy tales or news reports is much larger, with around 50-100 sentences in each document. As a result, when taking generally long documents with complex sentences as inputs, it will take extremely long training time if we illustrate the entire story by giving every sentence a corresponding image. Even if we use a text summarization approach to extract salient sentences [8], the sentences in the summarization results are not guaranteed to present in chronological order, which makes the illustrations presented in a random order. Therefore, we need a text summarization module to extract salient sentences in chronological order from the documents.

The second challenge is **Image Inconsistency**. If there are no message passing between the generation process of subsequent sentences, the generated images may be inconsistent. For example, a character in different images of a story may look different. Therefore, it requires a sequential model structure, which carries information about previous states and passes it on.

To solve the two problems mentioned above, we divide our framework into two modules. The first one is the text summarization module, which takes the long story and summarizes it into a few sentences which are considered as important and representative among the whole story. The second part is the image generation module, which generates images from the summarized text using a GAN model. The module is based on StoryGAN, a state-of-the-art model on story visualization. The result of our model as well as the ground truth is shown in 6. It can be seen that our model can generate relatively consistent images with correct characters, their positions and background.

To sum up, our contributions are: (1) Extend the scope of text-to-image generation task from salient sentences to general documents. (2) Propose a novel model that contains a text summarization module and a image generation module. (3) Use comprehensive experiments on human evaluation to validate the quality of images generated by our model.

2. Related Work

2.1. Natural Language Processing

Natural Language Processing (NLP) is one of the major research areas in Computer Science. Approaches involve deep learning has been wildly used and performances of such models are quite promising. Tasks include translation, sentiment analysis, text understanding are well implemented. In this task, we need to summarize a relatively long story, which is also a major problem in NLP.

2.2. Generative Adversarial Net

The Generative Adversarial Net (GAN) is a widely used structure of neural networks in the field of image generation.[2] A regular GAN is comprised of a generator, which generates images with a realistic look, and a discriminator, which distinguish the generated images from real images. In every training loop, fake images are produced in the generator and passed to the discriminator for differentiation. The target of the generator is to maximize the error rate of the discriminator, while the discriminator are updated to prevent the mistakes. Finally the generator is expected to output images that the discriminator cannot distinguish. In our task another discriminator is added for better understanding of the generated images.

2.3. Text-to-image Generation

Text-to-image generation refers to the machine learning task in which a single image is generated from a segment of text in natural languages. [10] In recent years, GANs have been playing an essential role in the development of this task. Although a high quality is achieved in the state-of-the-art works, this task only focuses on single image generation. Nevertheless, in dialogue-to-image generation, the input text is dialogue transcripts instead of a single sentence [11], which is close to our idea.

2.4. Text-to-video Generation

Text-to-video generation is another task similar to ours since a video can be seen as a sequence of continuous images with a chronological order. Current methods to this task is only able to generate small video clips with static backgrounds. [3] In contrast, in our task the continuity is not considered but backgrounds are supposed to be diversified across different images. In addition, like text-to-image generation, one video usually corresponds to only one sentence.

2.5. Story Visualization

Story visualization is a newly proposed task that inspires this paper. Its target is to generate a sequence of images to illustrate a story with multiple sentences. [7] The problem is that it works only on a group of processed sentences instead of a natural paragraph. In contrast, LongStoryGAN expands the range of stories for image generation to any paragraphs with a chronological order. It keeps the consistency of the story while still generates corresponding images for each individual sentences.

3. Data

Three datasets are used in our project, one for story summarization and two for image generation.

3.1. Wikipedia 2014 + Gigaword 5

Wikipedia 2014 + Gigaword 5 is a large-scaled text dataset used in GloVe, an unsupervised algorithm for word embeddings.[9] Every words in the dataset are transformed into several vectors with different dimensions. In our work, the embedding vectors are then utilized to extract main information from a paragraph so that a concise summary can be produced. The approximate vocabulary size is 400,000, and each of them corresponds to a 50, 100, 200 and 300 dimensional vectors.

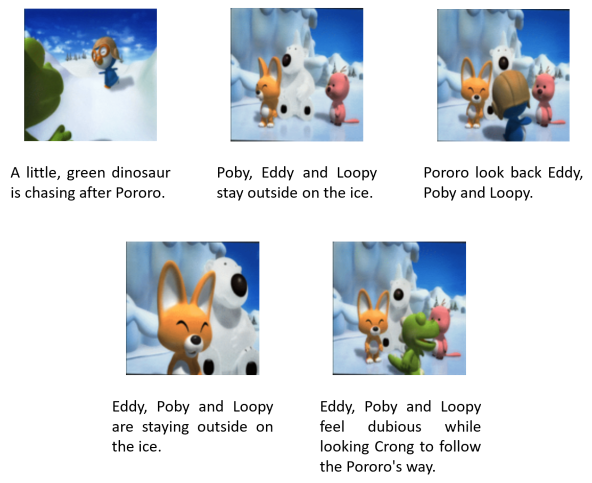

A little, green dinosaur is chasing after Pororo.
Poby, Eddy and Loopy stay outside on the ice.
Pororo look back Eddy, Poby and Loopy.
Eddy, Poby and Loopy are staying outside on the ice.
Eddy, Poby and Loopy feel dubious while looking Crong to follow the Pororo's way.

Figure 2: A sample group of images and sentences in the Pororo-SV dataset

3.2. CLEVR-SV

CLEVR-SV is a modified version of the CLEVR dataset[4], consisting of pictures containing only geometric objects and no backgrounds. These objects are classified into three shapes, two materials, eight colors and two sizes. Each picture contains at most four objects, and four pictures form a group corresponding to a "paragraph". Paragraphs are not written in natural languages, but sequences of attributes and positions of the objects, so it is only used to train the second half of the GAN. The dataset contains 10000 training groups and 3000 testing groups.

3.3. Pororo-SV

Like CLEVR-SV, Pororo-SV is also adapted from another dataset. It contains more complex pictures, the video frames of a cartoon called Pororo. In the original Pororo-QA dataset[6], one description is written for every second of the video, and Pororo-SV selects one picture from every second to associate with the description, so that the continuity between the pictures is interrupted but the context is maintained. Each paragraph consists of five sentences, but the absence of conjunction words makes it kind of unnatural . In total, there are 13000 training paragraphs and 2336 testing paragraphs. A sample from this dataset is shown in Figure 2

4. Methods

To illustrate a relatively long story, we use a text summarization model combined with the StoryGAN model. Given a story, the summarizer will output the most important sentences and pass the result to the image generation module,

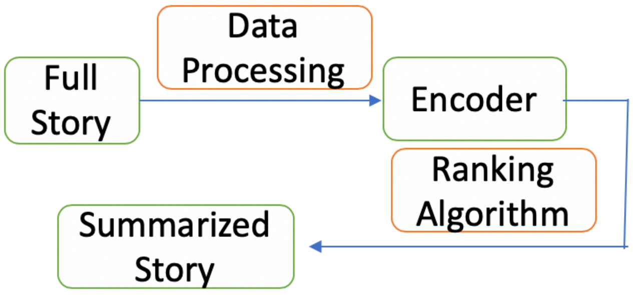


Figure 3: The Architecture of Text Summarizer

which will accept the entire story as input and generate images only for the sentences regarded as important and representative.

4.1. Text Summarization

Long paragraphs with complex expressions are more commonly seen in real life, which makes graphical illustrations of text even harder. This problem can be solved by summarizing the text, leaving or creating sentences that represent mean ideas of the entire story. Although it is possible to embed the summarizer into the GAN model later, making them separately will reduce the difficulty of training and easier for debugging and adjustments, as a GAN model takes a very long time to train.

Currently, there are 2 types of summarization models. The extractive summarization method will identify important sentences and then extract those out form the original paragraph. The extracted sentences will be identical to the ones in the given text. The second method is abstract summarization, whose output will be new sentences written by the model according to the original text. Both methods are sequence to sequence model and they could accomplish the task. However, the text data in Pororo consists of simple sentences with limited vocabulary. For example, a typical sentence is Pororo will be similar to ”Pororo wears orange glasses and a hat covering Pororo's ears.” Moreover, we find that abstract method may create new words that are out of range, which may not be understandable for StoryGAN. Therefore, we use an extract summarization model based on TextRank Algorithm in this task.

4.1.1 TextRank Algorithm

TextRank is a Graph-based algorithm deciding the importance in one or multiple documents.[8] Every sentence will be regarded as a note with the note value representing the sentence. By observing the relationship of each pair of sentences, a value will be computed, which represents the similarity, connectivity and other aspects of these sentences. Later those value will be ranked thus giving the rank of sentences in the story.

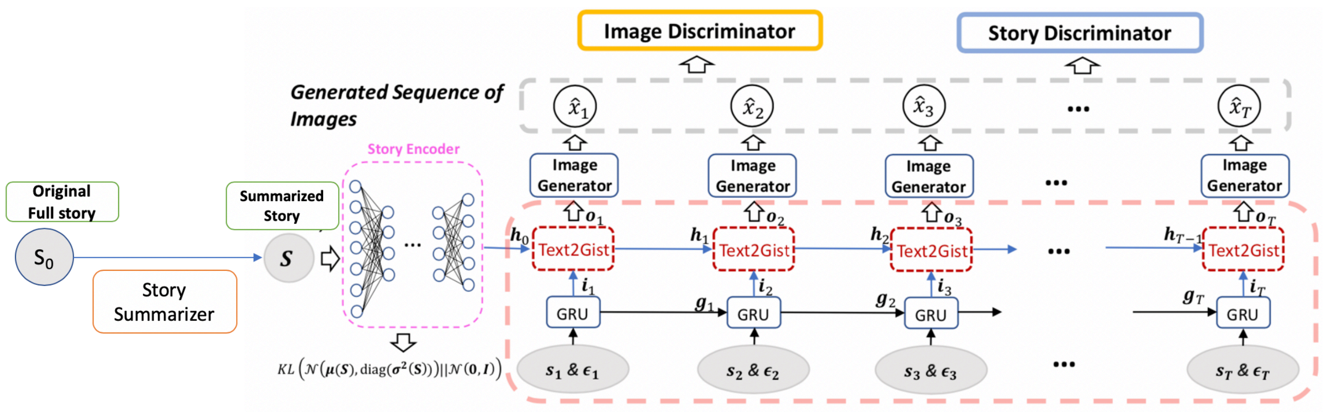


Figure 4: The Entire Structure of Our Model

4.1.2 TextRank with Pre-trained Encoder

The framework of the text summarization module of LongStoryGAN is shown in 3. The TextRank Algorithm will rank all sentences in the entire story and output them by rank. With a pre-trained encoder, every sentence in the document will be vectorized.[5] To rank those texts, the similarity between all vectors will be computed by a similarity matrix. The algorithm will work because after vectorization, the model will have an understanding of those sentences and if they are similar, they must be important in the paragraph. This method will largely reduce the size of text and works especially well in children's stories and simple stories in pororo with repeated expressions.

Traditionally each word will be represented with a unique vector and one sentence could be represented by the concatenation of these vectors. The method is straightforward, but it may do well in finding sentences with similar words instead of acquiring a understanding. By using a pre-trained encoder, the texts will be transformed into hidden stats that contain some features of the sentence. This process could be considered as understanding the texts, giving a much better result. To rank those texts, the similarity between all vectors will be computed by a similarity matrix. This method will largely reduce the size of text and works especially well in children's stories and simple stories in Pororo with repeated expressions. After the summarization, the input story will be replaced by a few sentences that are considered as important and representative. And the summarized text will be given to the GAN model. The model will then generate a representative illustration of the whole story, which is the goal of this task.

4.2. Image Generation

For image generation, we utilize a state-of-the-art model, StoryGAN but make several modifications. We name the modified model LongStoryGAN. Here are some major reasons for choosing StoryGAN instead of other GAN models. Firstly, the GAN model should support text to image generation. Secondly, regular GAN models will generated images separately according to the given sentences, not being able to provide global consistency across the story, which is crucial in graphical illustration. After comparing many networks, StoryGAN suits our demand best. The LongStoryGAN takes in a group of sentences $S = [s_1, s_2, \ldots, s_T]$ and for each sentence s_i one corresponding picture $\hat{x}_i$ is generated. Thus, the output of LongStoryGAN is a sequence of pictures $\hat{X} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_T]$ of the same length with S. It is worth noting that the sentences S are embedding vectors encoded by a universal sentence encoder rather than raw text.
As shown in Figure 3, LongStoryGAN consists of four parts: (i) a story encoder that compress S to a single hidden vector h_0 (ii) a context encoder that combines each sentence s_t with its context and outputs a gist vector o_t (iii) an image generator that generates an image $\hat{x}_t$ from the gist vector o_t (iv) an image discriminator to examine the quality of each image and a story discriminator to assess the consistency among all the images.

4.2.1 Story Encoder

The story encoder is basically a multi-layer fully connected network, which maps the input story S to a single hidden vector h_0. h_0 is sampled from a normal distribution for randomness and will be passed to the context encoder as the initial hidden state. The idea is that h_0 is supposed to serve as a summary vector containing main information of the story, so that the global consistency is kept in later process.

4.2.2 Context Encoder

Since multiple images with logical connection are the target in this model, a deep RNN with a two-layer structures is used to preserve and transmit the contextual information. The lower layer is a sequence of GRU and the other consists of Text2Gist cells, a variation of GRU specifically designed for this task. The process can be described as the following:

$$i_t, g_t = GRU\left(s_t, g_{t-1}\right) \tag{1}$$

$$o_t, h_t = Text2Gist\left(i_t, h_{t-1}\right) \tag{2}$$

where g_t and h_t are the hidden states of GRU and Text2Gist respectively. o_t is called the gist vector, since it combines the contextual information in h_{t-1} and the sentence information in i_t. Then it is used for image generation in the next step.

4.2.3 Image Generator

The image generator will not be specified since it can be any neural network that outputs an image, such as a multi-layer neural network or a network with transposed convolution layers. In consideration of performance, the original structure in StoryGAN is preserved.

4.2.4 Discriminators

In addition to the quality of a single image, the coherence of images in a group and the consistency between the images and the story are also important criteria for our task. Therefore, two discriminators are used in the model and the final loss will combine both.
The image discriminator computes the difference between each group of generated images and their corresponding ground truth, while the story discriminator has a more complicated structure. After each image is transformed into a feature vector, all the vectors in a group are then concatenated into a single vector $E_{img}(X)$. Using the same method, the group of sentences are also transformed into a vector $E_{text}(S)$. A linear transformation of the element-wised product of $E_{img}(X)$ and $E_{text}(S)$ will be the final scores, which is denoted by $D(X, S)$. The fomula is shown below:

$$D(X, S) = W(E_{img}(X) \odot E_{text}(S)) + b \tag{3}$$

where W and b are learn-able parameters and $\odot$ here means element-wise multiplication.
The scores computed with fake and real images are then compared to indicate the degree of global consistency.

4.2.5 Training Method

Since the model is large and composed of different networks, the training process is divided into two parts: the text encoder and the image generator are trained separately using different datasets. The text encoder is expected to produce a vector that fully covers the information and preserves the order, while the image generator should generate images both discriminators cannot distinguish.
Therefore, the following training strategy is used:

1. Train the text encoder with raw story-sentences pairs $(S_0, [s_1, s_2, \ldots, s_T])$

2. Train the LongStoryGAN with groups of sentences $([s_1, s_2, \ldots, s_T])$ and real images $([x_1, x_2, \ldots, x_T])$

After the training process, stories will be inputted into the entire model. The generated images will be evaluated manually to ensure the universality of the model.

5. Experiments

In this section, we evaluate our model in two parts. First, we evaluate the text summarization model to see if can extract the descriptive sentences from the document. Then, we evaluate the image generation model to see if it can generate clear image that follows the descriptions we extracted in the first phase. The experiments are done on one toy dataset, *CLEVR-SV* and one cartoon dataset, *Pororo-SV*. To the best of our knowledge, there is no previous works that focus on neither descriptive sentence extraction nor video generation, so our comparisons are mainly to ablated versions of our proposed models. For a fair comparison, all models use the same structure of the image generator, Context Encoder and discriminators when applicable.

5.1. Text Summarization Evaluation

In this section, we provide qualitative evaluation of the quality of the text summarization module. The compared method is:

LongStoryGAN-order: It is the ablation of our model that takes off the order-persistent module, which extract the sentences in chronological order. Instead, it ranks the sentences in the order of degree of importance.

5.1.1 Experiment Results

We compare the results generated by our model with the human extracted result, noted as **Ground truth** to show that our model effectively select the sentences that describing the motion of the characters and keep them in a chronological order. The results are shown in Table 1.

We can see that LongStoryGAN extracts more relevant sentences than LongStoryGAN-order. This is because the order-persistent module takes sentences that spread more evenly in each part of the whole document. In this way, the descriptive sentences from all the parts can be extracted. In contrast, LongStoryGAN-order tend to extract sentences that squeeze in a specific part of the document, which hurts

Ground truth:
A wolf happened to pass by the lane where the three little pigs lived;
The wolf opened his jaws very wide and bit down as hard as he could,
but the first little pig escaped and ran away to hide with the second little pig.
The wolf continued down the lane and he passed by the second house
His big jaws clamped down on nothing but air and the two little pigs scrambled away as fast as their little hooves would carry them.
they made it to the brick house and slammed the door closed before the wolf could catch them.
The wolf huffed, huffed, and he puffed, puffed; but he could not blow the house down.
So the little piggy put on the cover again, boiled the wolf up, and the three little pigs ate him for supper.

LongStoryGAN:
A wolf happened to pass by the lane where the three little pigs lived;
The wolf opened his jaws very wide and bit down as hard as he could,
but the first little pig escaped and ran away to hide with the second little pig.
The wolf continued down the lane and he passed by the second house made of sticks;
and he saw the house, and he smelled the pigs inside, and his mouth began to water as he thought about the fine dinner they would make. (X)
His big jaws clamped down on nothing but air and the two little pigs scrambled away as fast as their little hooves would carry them.
The wolf hadn't eaten all day and he had worked up a large appetite chasing the pigs around and now he could smell all three of them inside and he knew that the three little pigs would make a lovely feast. (X)
The wolf huffed, huffed, and he puffed, puffed; but he could not blow the house down.

LongStoryGAN-order:
The wolf opened his jaws very wide and bit down as hard as he could,
but the first little pig escaped and ran away to hide with the second little pig.
Not by the hairs on our chinny chin chin! (X)
His big jaws clamped down on nothing but air and the two little pigs scrambled away as fast as their little hooves would carry them.
The wolf showed his teeth and said: (X)
Then I'll huff, and I'll puff, and I'll blow your house down. (X)
The wolf hadn't eaten all day and he had worked up a large appetite chasing the pigs around and now he could smell all three of them inside and he knew that the three little pigs would make a lovely feast. (X)
So the little piggy put on the cover again, boiled the wolf up, and the three little pigs ate him for supper.

Table 1: The text summarization results of **LongStoryGAN**, **LongStoryGAN-order** and the ground truth. The original corpus is the fairytale "the three little pigs". For each of the methods, we extract 8 sentences from the corpus. The wrong sentences extracted are marked with (X)

the overall salience. Another difference between LongStoryGAN and LongStoryGAN-order is that LongStoryGAN sorts the sentences in chronological order. In contrast, the sentences extracted by LongStoryGAN-order are in random order, which makes subsequent sentences inconsistent.

5.2. Image Generation Evaluation

In this section, we provide qualitative evaluation of the quality of the image generation module. the compared method is:

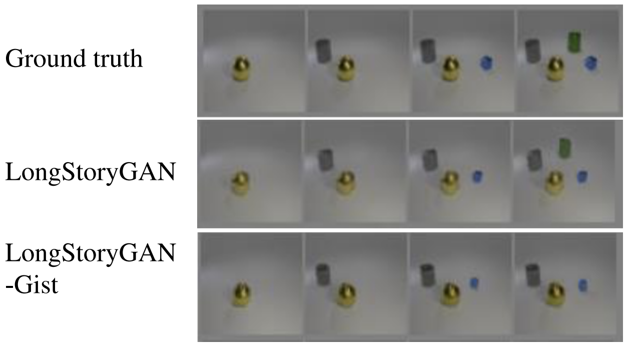


Figure 5: One generated sample chosen from the experiment results on the CLEVR-SV dataset. Each story has four images.

LongStoryGAN-Gist: In LongStoryGAN-Gist, the Text2Gist cell in LongStoryGAN is replaced by simple concatenation of the encoded story and description feature vectors. The structure of the discriminator and image generator are kept as the same.

Methods	LongStoryGAN	LongStoryGAN-Gist
SSIM	0.672	0.641

Table 2: SSIM score of different methods on CLEVR-SV dataset

5.2.1 Experiment Results on CLEVR-SV

The CLEVR-SV dataset contains a set of images and descriptions of the layout. 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 10, 000 image sequences for training and 3, 000 for testing. For our task, the story is the layout descriptions of objects.

The results are shown in Figure 5. The forth image in the sequence generated by LongStoryGAN-Gist is plausible. We hypothesize that using trivial concatenation in LongStoryGAN-Gist cannot effectively balance the information contained in the current sentence and the whole story. In contrast, the generation results of LongStoryGAN is almost the same as that in ground truth. This validates the effectiveness of Text2Gist component in our model, which encodes the information in the current sentence while keeping tracking on the progress of the whole story.

To quantitatively evaluate our model, we also compare the Structural Similarity Index (SSIM) score between the generated images and ground truth.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \quad (4)$$

where mu_x is the average of x and $sigma_x$ is the standard deviation. c_1 and c_2 are two variables to stabilize the division with weak denominator. SSIM is used for measuring the similarity between two images. Here, it is used to determine whether the generated images are consistent with the input description. The experiment results are shown in Table 2. We can see that LongStoryGAN outperforms LongStoryGAN-Gist in terms of SSIM score. It again validates the utility of the Text2Gist module.

5.2.2 Experiment Results on Pororo-SV

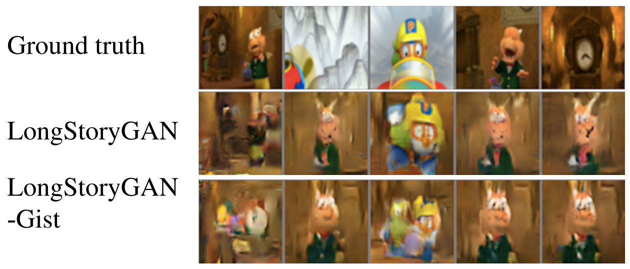
The Pororo-SV dataset contains a series of cartoon frames with manually written descriptions. In our experiments, we use five continuous images to visualize a single story. We encode the text by the Universal Encoder [1] with default parameters, which produces an 128-dim vector representation for each sentence.

Two visualized stories from the compared methods and the ground truth are given in Figure 6. By comparing the results, we can see that LongStoryGAN-Gist tend to generate plausible results in the first image of each story. For example, it generates the wrong characters in the first image on Story 1 (the clock is omitted), and generates the wrong background in the first image of Story 2 (generates the snow background instead of the land). Such flaws result in the inconsistency of the story. In contrast, the first image in Story generated by LongStoryGAN is more similar to the ground truth, in terms of both characters and positions. LongStoryGAN also generates the correct background in Story 2. This shows the advantage of using the output of the Story Encoder as first hidden state over random initialization.

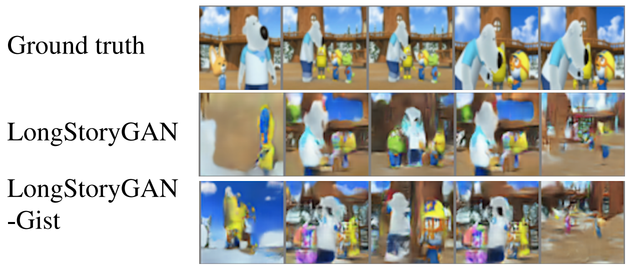
Similarly, we compare the SSIM score between the generated image and the ground truth. In this task, LongStoryGAN significantly outperforms LongStoryGAN-Gist, which is consistent with the qualitative result analysis.

Methods	LongStoryGAN	LongStoryGAN-Gist
SSIM	0.369	0.328

Table 3: SSIM score of different methods on Pororo-SV dataset



(a) Story 1



(b) Story 2

Figure 6: Two generated samples chosen from the experiment results on the Pororo-SV dataset. Each story has five images.

6. Conclusion

6.1. Project Summerization

We proposed a new machine learning task, chronological illustration generation from document, which broadened the application of image generation by combining GAN with multiple NLP techniques. More specifically, text summarization using the TextRank Algorithm is applied to the story preprocessing so that the task is transformed to story visualization. Then a revised StoryGAN generated understandable images based on the summarized text. Our model improved the accuracy of image generation when the input is an entire paragraph instead of separated sentences.

6.2. Future Improvements and Applications

The main limitation is data at this point. With the Proro dataset, our model could only generate cartoon images. It would be better if data which contain real life event and text could be used for training, enabling the model to generate vivid pictures. Such data could be generated by giving text discrimination to videos like what Proro did. Also, the current model have no attention in the event or object. Although the story has been summarized, the model could do better if main events in each sentences could gain attention from the model.

Such a model could be used in auto-image illustration. For example, a writer who just finished a story could use this model for automated illustration. It might also be helpful in situation like crime scene picturization. The most exciting aspect about this work is that it gives a machine the ability to imagine things from its experiences, more accurately, from the data it read and things it learned, just like us humans do.

References

[1] D. M. Cer, Y. Yang, S. yi Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil. Universal sentence encoder. *ArXiv*, abs/1803.11175, 2018.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[3] J. He, A. M. Lehrmann, J. Marino, G. Mori, and L. Sigal. Probabilistic video generation using holistic attribute control. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V*, volume 11209 of *Lecture Notes in Computer Science*, pages 466–483. Springer, 2018.

[4] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016.

[5] P. Joshi and A. Vidhya. Introduction to text summarization using the textrank algorithm, May 2019.

[6] K. Kim, M. Heo, S. Choi, and B. Zhang. Deepstory: Video story QA by deep embedded memory networks. In C. Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2016–2022. ijcai.org, 2017.

[7] Y. Li, Z. Gan, Y. Shen, J. Liu, Y. Cheng, Y. Wu, L. Carin, D. E. Carlson, and J. Gao. Storygan: A sequential conditional gan for story visualization. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6322–6331, 2018.

[8] R. Mihalcea and P. Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[9] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[10] S. E. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *CoRR*, abs/1605.05396, 2016.

[11] S. Sharma, D. Suhubdy, V. Michalski, S. E. Kahou, and Y. Bengio. Chatpainter: Improving text to image generation using dialogue. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.