SEMANTIC VIDEO SYNTHESIS FROM VIDEO SCENE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Video synthesis has recently attracted a lot of attention, as the natural extension to the image synthesis task. Most image synthesis works use class labels or text as guidance. However, neither labels nor text can provide explicit temporal guidance, such as when action starts or ends. To overcome this limitation, we introduce video scene graphs as input for video synthesis, as they represent the spatial and temporal relationships between objects in the scene. Since video scene graphs are usually temporally discrete annotations, we propose a video scene graph (VSG) encoder that not only encodes the existing video scene graphs but also predicts the graph representations for unlabeled frames. The VSG encoder is pre-trained with different contrastive multi-modal losses. A video scene graph-to-video synthesis framework (SGVS) based on the pre-trained VSG encoder, VQ-VAE, and auto-regressive Transformer is proposed to synthesize a semantic video given an initial scene image and a non-fixed number of video scene graphs. We evaluate SGVS and other state-of-the-art video synthesis models on Action Genome dataset and demonstrate the positive significance of video scene graphs in video synthesis.

1 INTRODUCTION

With the tremendous breakthroughs in image synthesis (Karras et al., 2020; Saharia et al., 2022; Yu et al., 2022) in recent years, more and more researchers are focusing on the natural extension yet more challenging task of video synthesis. In contrast to image synthesis, video synthesis requires that each generated frame has spatial fidelity and also that these frames conform to temporal continuity. While some work has made incredible progress in several simple scenarios, such as robot interaction (Ebert et al., 2017), there is much space for improvement for semantic video synthesis.

To generate a semantically meaningful video, the guidance is essential. Text-guided image generation models have achieved convincing performance, and there has been recent works (Li et al., 2018; Ho et al., 2022; Hong et al., 2022) of video synthesis using text as condition. However, text has the drawback of not providing explicit guidance for time dependencies. For example, when we want to generate a 16-frame video from the text condition “a girl standing up after sitting on a chair”, the synthesis model needs not only to generate frames with visual quality and temporal consistency, but also to infer which of the 16 frames correspond to the “sitting” state and which to the “standing up” action. This highly increases the complexity of video synthesis task.

Is there a better guiding condition, which contains not only semantic information, but also clearly reflects temporal dependencies? Based on this observation, we make use of video scene graphs for the task of semantic video synthesis. An image scene graph is a structural representation that generalizes the objects of interest in an image as nodes and their relationships as edges. It has been utilized as conditional information for image generation (Johnson et al., 2018). Video scene graphs can be viewed as several static scene graphs on an temporal axis, which describe the semantic content of the frames in the video. The standard video scene graphs are continuous. In other words, each frame in the video is annotated with a corresponding scene graph. This data structure containing spatial and temporal information is very promising for video synthesis. However, giving continuous video scene graphs during inference is harsh compared to giving other temporal conditions such as trajectories (Le Moing et al., 2021), action labels (Menapace et al., 2021), or overall scene conditions such as text, because scene graphs are relatively difficult to create.

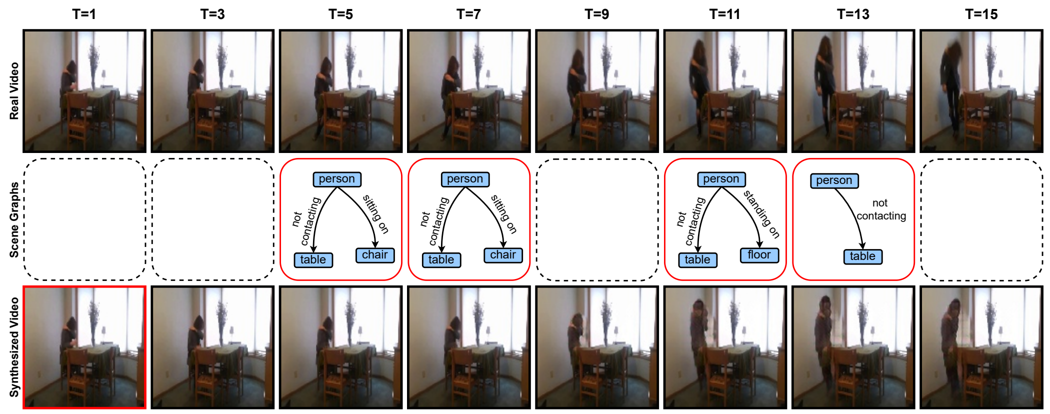


Figure 1: Given a starting frame and a non-fixed number of video scene graphs (with red bounding boxes), SGVS can synthesize a fix length complex semantic video with 128×128 resolution, not just simple repetitive motions. The number and temporal position of the input video scene graphs are freely controllable. For clarity, only 8 of 16 frames are shown.

In this paper, we propose a novel video scene graph-to-video synthesis framework (SGVS) that can synthesize a fixed-length semantic video with an initial scene image and discrete video scene graphs, as shown in Figure 1. To address the discontinuity of the input video scene graphs and to learn graph representations to better guide video synthesis, we propose a video scene graph (VSG) encoder pre-trained in a contrastive multi-modal framework. The VSG encoder can not only provide the graph representations for the existing video scene graphs but also predicts the representations of the frames that are missing a video scene graph. Therefore, our approach does not impose strict restrictions on the number and temporal position of the given video scene graphs, which makes SGVS more feasible in practice. For the generative model, we utilize a popular combination of VQ-VAE and auto-regressive Transformer (Yan et al., 2021; Le Moing et al., 2021; Ge et al., 2022) since this likelihood-based model fits our purpose and is easy to optimize. The graph representations can be inserted into the sequence of the discrete latents provided by the encoder of VQ-VAE. These latents and graph representations are then modeled by a GPT-like Transformer using an auto-regressive prior. The latents generated from the auto-regressive prior are converted to video frames by the decoder of VQ-VAE. Our main contributions are listed as follows:

- We are the first to synthesize semantic videos from video scene graphs.
- A contrastive multi-modal learning framework is proposed to pre-train a Transformer-based video scene graph (VSG) encoder which can provide high-quality video scene graph representations to condition semantic video synthesis.
- We propose a video scene graph-to-video synthesis method (SGVS) consisting of a VQ-VAE, an auto-regressive Transformer, and the VSG encoder. SGVS can synthesize a fixed-length semantic video with an initial scene image and discrete video scene graphs.
- We split a sub-dataset from the dataset Action Genome (Ji et al., 2020) and conduct experiments to demonstrate the potential and benefits of using video scene graphs as condition for semantic video synthesis. Compared to other state-of-the-art works, SGVS using video scene graphs can generate better quality semantic videos.

2 Related Work

Image and video scene graph. Scene graphs have first been proposed by Johnson et al. (2015) for image retrieval and have received a lot of attention in the field of scene understanding. The graphical representation whose nodes indicates objects and edges indicates the semantic relationships between objects can clearly describe the scene. There are many impressive two-stage works (Zellers et al., 2018; Zhang et al., 2019; Tang et al., 2019; Lin et al., 2020; Chiou et al., 2021; Li et al., 2021; Dhingra et al., 2021; Lyu et al., 2022; Liu et al., 2022) that have achieved incredible results of image scene graph generation on the datasets (Lu et al., 2016; Krishna et al., 2017; Kuznetsova et al., 2020). Recently, some one-stage methods (Liu et al., 2021; Li et al., 2022a; Teng & Wang, 2022) are emerging that infer the scene graph directly from images without the need for a pre-trained object detector. These methods reduce the complexity of image scene graph generation and have a faster inference speed compared to those two-stage methods.

Ji et al. (2020) proposes the dataset Action Genome, which extends image scene graphs to video scene graphs by adding the temporal dimension. Cong et al. (2021) captures spatial and temporal dependencies and generates video scene graphs with Transformers. Li et al. (2022b) proposes an anticipatory pre-training paradigm for video scene graph generation. Xu et al. (2022) solves the problem of spatial-temporal conditional bias in video scene graph generation.

Since scene graphs not only contain the objects present in the scene but also demonstrates the interactions between the objects, they are exploited in image retrieval (Johnson et al., 2015), image captioning (Milewski et al., 2020), visual question answering (Damodaran et al., 2021) and image synthesis (Johnson et al., 2018; Yang et al., 2022a; Herzig et al., 2020). In contrast, although video scene graphs have been used for video question answering (Cherian et al., 2022) and video captioning (Cao et al., 2020), the applications have not been effectively explored.

Video synthesis. A video can be regarded as a high-dimensional image with an additional temporal dimension. Therefore, the methods for image synthesis can be extended to video synthesis as well, although longer training time and higher memory consumption are required. Some GAN-based methods adapt the adversarial framework to generate videos with 3D convolutions (Acharya et al., 2018; Wang et al., 2020) or recurrent neural networks (Tulyakov et al., 2018; Clark et al., 2019; Saito et al., 2020). However, as the length of the generated videos becomes longer, the quality of the videos synthesized by these models decreases significantly. Auto-regressive models (Kalchbrenner et al., 2017; Weissenborn et al., 2019) have become popular in this field, even though their inference speed is slow. Many of these methods (Wu et al., 2021; Yan et al., 2021; Le Moing et al., 2021; Ge et al., 2022; Hong et al., 2022) introduce VQVAE (Van Den Oord et al., 2017) and Transformers (Vaswani et al., 2017) to improve performance. Furthermore, Ho et al. (2022); Yang et al. (2022b) propose video diffusion models which is a natural extension of the image architecture.

There are many sub-tasks in the field of video synthesis. In unconditional synthesis, a video is produced without any prior information. For conditional synthesis, a popular task is to leak a few video frames to the model, which predicts future videos. Some aforementioned works support the generation of videos conditional on class labels. areTo further control the video content, some text-to-video models (Balaji et al., 2019; Hong et al., 2022; Ho et al., 2022) were developed. Le Moing et al. (2021) uses the real trajectory of a robotic arm to generate videos. Ren & Wang (2022) propose a novel method to synthesize a video given a single scene image and camera motions. Menapace et al. (2021) introduces the task of playable video generation in an unsupervised setting.

Given the characteristics of video scene graphs, we introduce video scene graphs as the guidance of video synthesis. We use a scene image and video scene graphs as the input of the auto-regressive model SGVS to synthesize a video. Different from text-to-video generation, video scene graphs allow precise control of the moment when the episodes occur. Although video scene graphs are usually temporally discrete, the proposed video scene graph encoder pre-trained with different contrastive losses can predict continuous video scene graph embeddings for guiding video synthesis.

3 Video scene graph representation learning

Many text-to-image generation works (Ramesh et al., 2022; Saharia et al., 2022) using CLIP text encoder (Radford et al., 2021) have achieved outstanding results. It demonstrates that pre-trained condition encoders provide high-quality representations that facilitate the generation task. With the same motivation, we propose a video scene graph-video contrastive pre-training framework including a video scene graph (VSG) encoder and a frame encoder (see Figure 2). The pre-trained VSG encoder converts the given video scene graphs into single-vector representations and reasons about the representations of the ungiven scene graph. The auxiliary frame encoder, that is discarded during synthesis, provides the frame representations and feature maps. We establish a mapping of the semantic graph representation space to the visual latent space by using the graph-frame representation similarity while the feature maps provide fine-grained information at the node and edge level.

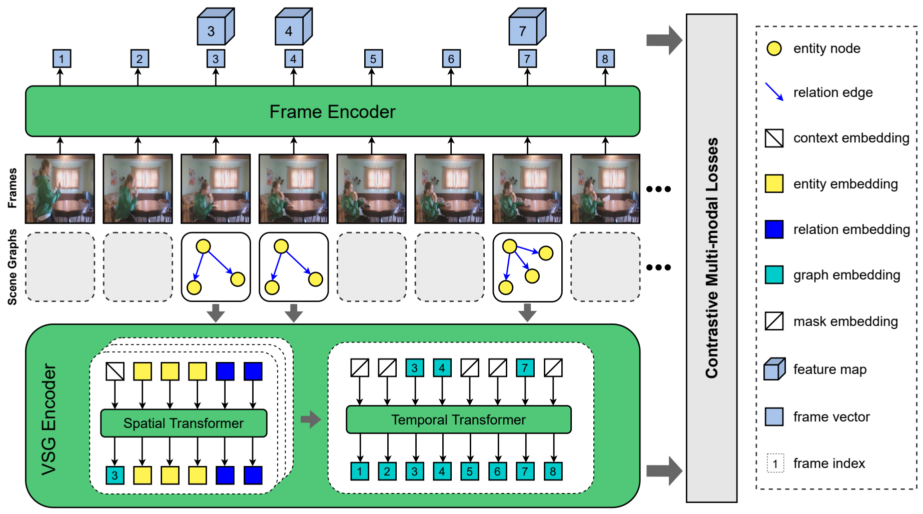


Figure 2: Framework of video scene graph representation learning.

3.1 Video scene graph encoder

Our video scene graph encoder consists of a spatial Transformer and a temporal Transformer. The spatial transformer captures the context of the input scene graphs and the fine-grained representations of their nodes and edges, while the temporal Transformer infers the graph representations at all times based on the graph context provided by the spatial Transformer.

Spatial Transformer. Given a scene graph $G = (\boldsymbol{N}, \boldsymbol{E})$, the nodes $\boldsymbol{N}$ and edges $\boldsymbol{E}$ in the graph are viewed as tokens with semantic information. We decompose the structure of the graph into a sequence $\boldsymbol{S}$ consisting of node tokens and edge tokens. In order to obtain the graph context through self attention mechanism, we introduce a special `[context]` token and place it always first in the sequence $\boldsymbol{S}$. Each token corresponds to a learned embedding $\boldsymbol{e} \in \mathbb{R}^d$ that is randomly initialized before being trained from scratch. In order for the structural properties of the graph to be preserved in Transformer which is permutation invariant, we construct the context encoding $\boldsymbol{E}(c) \in \mathbb{R}^d$ and node encodings $\boldsymbol{E}(\boldsymbol{N})$ with learned embeddings. For the edge e_{ij} from the node n_i to the node n_j, the edge encoding is calculated as $\boldsymbol{E}(e_{ij}) = \boldsymbol{E}(n_i) - \boldsymbol{E}(n_j)$. These encodings are element-wise added to the corresponding embeddings. For a graph with N_n nodes and N_e edges, the length of the input sequence is $(1+N_n+N_e)$. We adopt the common multi-layer Transformer as the architecture of the Transformers in this paper. Since the use of the Transformer (Vaswani et al., 2017) is already widespread and our implementation has the same structure as the popular, we omit the exhaustive background introduction and show its architecture in Appendix A.1. The graph context from the last layer of the spatial Transformer is forwarded to the temporal Transformer. The node and edge representations are used to learn fine-grained visual information from the feature maps provided by the frame encoder with the fine-grained graphical contrastive loss introduced in Section 3.3.

Temporal Transformer. Although the temporal Transformer has the identical architecture as the spatial Transformer, it plays a completely different role, inferring graph representations for unavailable scene graphs. A special `[mask]` token is introduced whose learned embedding has the same dimension as the graph context. We construct a sequence of the mask tokens with the same length T as the video and and replace the mask embeddings in the position of the given graphs with their graph context provided by the spatial Transformer. Temporal encodings $E_t \in \mathbb{R}^{T\times d}$ are customized to inject temporal location information into the sequence. These encodings are also learned embeddings. This sequence is used as the input to the temporal Transformer that reasons about the representations of all graphs based on temporal dependencies. Our motivation is similar to the masked language model(Devlin et al., 2018) that masks some words in the sentence and reconstructs them using the context. The difference is that these graph representations $\boldsymbol{g} \in \mathbb{R}^d$ are learned with the contrastive learning using the frame vectors provided by the frame encoder.

3.2 Frame encoder.

To relate the graph representations to the visual appearance, we utilize a frame encoder to encode the video frames and extract the feature maps. It is built upon a pre-trained convolutional neural network (Szegedy et al., 2016). A 1×1 convolution layer is exploited to reduce the dimension of the feature maps extracted by the pre-trained CNN. The frame vectors $\boldsymbol{v} \in \mathbb{R}^d$ are derived from the feature maps $\in \mathbb{R}^{h \times w \times d}$ through a 2D global average pooling layer and a linear transformation. The frame vectors of all frames are involved in the loss calculation, while only the feature maps of the frames with graph annotations are activated for fine-grained learning.

3.3 Contrastive multi-modal losses

We introduce 3 contrastive loss functions for graph representation learning, namely graphical intra-video contrastive loss, graphical inter-video contrastive loss, and graphical fine-grained contrastive loss. The total loss function is the equal sum of these three loss functions.

Graphical intra-video contrastive loss. Temporal dependencies and continuity are present in both visual appearance and semantic representations. Therefore, we propose to learn scene graph representations $\boldsymbol{g}$ from visual representations $\boldsymbol{v}$ through temporal dependencies. We compute the graphical intra-video contrastive loss of a video with T frames as:

$$L_{intra} = -\sum_{i=1}^{T}(\log \frac{\exp(\text{sim}(\boldsymbol{g}_i, \boldsymbol{f}_i))}{\sum_{j=1}^{T} \exp(\text{sim}(\boldsymbol{g}_j, \boldsymbol{f}_i))} + \log \frac{\exp(\text{sim}(\boldsymbol{g}_i, \boldsymbol{f}_i))}{\sum_{j=1}^{T} \exp(\text{sim}(\boldsymbol{g}_i, \boldsymbol{f}_j))}), \tag{1}$$

where $\text{sim}(\boldsymbol{g}_i, \boldsymbol{f}_j)$ indicates the cosine similarity between the graph representation of the i-th frame and the frame vector of the j-th frame. The first term denotes that the frame vector should have a higher similarity to the graph representation of the current frame compared to the graph representation of other frames, while the second term is symmetrical for the graph representation.

Graphical inter-video contrastive loss. Because a scene graph is the semantic description of a scene, videos with different visual appearances correspond to different scene graphs. The objective here is to learn graph representations by comparing scenes from different videos. Given a batch of B videos, the graphical inter-video contrastive loss of the t-th frame can be formulated as:

$$L_{inter}^{t} = -\sum_{i=1}^{B}(\log \frac{\exp(\text{sim}(\boldsymbol{g}_i, \boldsymbol{f}_i))}{\sum_{j=1}^{B} \exp(\text{sim}(\boldsymbol{g}_j, \boldsymbol{f}_i))} + \log \frac{\exp(\text{sim}(\boldsymbol{g}_i, \boldsymbol{f}_i))}{\sum_{j=1}^{B} \exp(\text{sim}(\boldsymbol{g}_i, \boldsymbol{f}_j))}), \tag{2}$$

where the frame index t on the right side of the equation is omitted for brevity. The graph representation $\boldsymbol{g}_i$ and frame vector $\boldsymbol{f}_j$ are inferred from the the t-th frames of the i-th video and the j-th video in the batch. The complete L_{inter} are the sum of L_{inter}^{t} of T frames.

Graphical fine-grained contrastive loss. Inspired by Xu et al. (2018), we propose a fine-grained loss function that supervises the consistency between the frame and the semantic nodes as well as edges of the scene graph. The graph representations are improved at a fine-grained level by associating the node and edge representations with the sub-regions in the video frame. We view both the node and edge representations of the scene graph as semantic embeddings $\boldsymbol{e}$. For a scene graph with N_n nodes and N_e edges, there are totally $N_g = N_n + N_e$ semantic embeddings. We compute the visual context $\boldsymbol{c}_i$ of the i-th node or edge in the graph as:

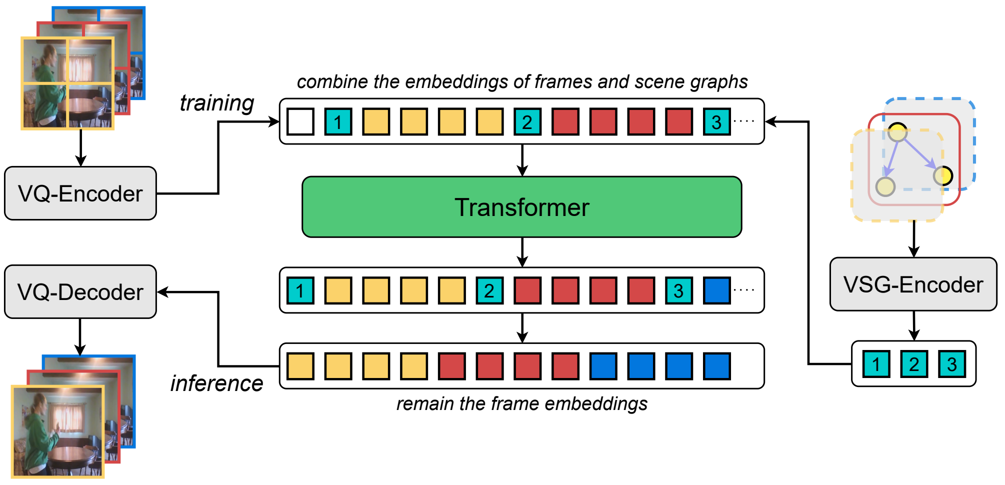
$$\boldsymbol{c}_i = \sum_{j=1}^{hw} \frac{\exp(\text{sim}_{dot}(\boldsymbol{e}_i, \boldsymbol{r}_j))}{\sum_{k=1}^{hw} \exp(\text{sim}_{dot}(\boldsymbol{e}_i, \boldsymbol{r}_k))} \boldsymbol{r}_j, \text{ where } \text{sim}_{dot}(\boldsymbol{e}_i, \boldsymbol{r}_j) = \frac{\exp(\boldsymbol{e}_i^{\mathrm{T}} \boldsymbol{r}_j)}{\sum_{g=1}^{N_g} \exp(\boldsymbol{e}_g^{\mathrm{T}} \boldsymbol{r}_j)}. \tag{3}$$

$\boldsymbol{r}_j \in \mathbb{R}^d$ is the vector of the j-th sub-region in the feature map with $h \times w$ shape provided by the frame encoder, whereas $\text{sim}_{dot}(\boldsymbol{e}_i, \boldsymbol{r}_j)$ denotes the normalized dot-product similarity between the i-th semantic embedding of the scene graph and the visual vector of the j-th sub-region. The matching score $\mathrm{S}(G, F)$ between the scene scene G and the frame F can be formulated as $\mathrm{S}(G, F) = \log(\sum_{i=1}^{N_g} \exp(\text{sim}(\boldsymbol{e}_i, \boldsymbol{c}_i))$. When there are P pairs of video frames and scene graphs in the batch of B videos, the graphical fine-grained contrastive loss is defined as:

$$L_{finegrain} = -\sum_{i=1}^{P}(\log \frac{\exp(\mathrm{S}(G_i, F_i))}{\sum_{j=1}^{P}\exp(\mathrm{S}(G_j, F_i))} + \log \frac{\exp(\mathrm{S}(G_i, F_i))}{\sum_{j=1}^{P}\exp(\mathrm{S}(G_i, F_j))}). \tag{4}$$

4 Video scene graph-to-video synthesis framework

In this section, we introduce a video scene graph-to-video synthesis framework (SGVS) consisting of a video scene graph (VSG) encoder, a VQ-VAE, and an auto-regressive Transformer (see Figure 3).



training
combine the embeddings of frames and scene graphs
VQ-Encoder
Transformer
VQ-Decoder
VSG-Encoder
inference
remain the frame embeddings
1
2
3

Figure 3: Overview of our video scene graph-to-video synthesis framework SGVS. During training, Transformer learns the prior distribution of the sequence consisting of the latent embeddings provided by VQ-VAE and the scene graph embeddings provided by the VSG encoder. In inference, the latent embeddings of the starting frame and graph embeddings are given. The latent embeddings of future frames are sampled from the prior and decoded into the video by VQ-VAE.

VQ-VAE. We adopt the VQ-VAE proposed by Le Moing et al. (2021) and pre-train it using videos in the training set. Given a frame, the CNN-based encoder produces the output with the down-sampled spatial resolution. The discrete latent variables $\boldsymbol{z}$ for the output are quantified by the nearest neighbor look-up using the shared embedding space. The latent embeddings are forwarded to the decoder to reconstruct the input frame. We freeze the pre-trained VQ-VAE in other stages.

Auto-regressive Transformer. We utilize an auto-regressive Transformer with full attention mechanism to learn a prior over the VQ-VAE latent embeddings $\boldsymbol{z}$ and video scene graph representations $\boldsymbol{g}$. The basic architecture of the auto-regressive Transformer is almost identical to the Transformers in Section 3.1, but with more layers. During the training stage, the latent embeddings of the video frames are flattened frame by frame into a 1D sequence in row-major order. The learned temporal encodings are added to the latent embeddings. Then we insert the graph representations provided by the VSG encoder into the sequence. Note that each frame has a corresponding scene graph representation. For frames where no scene graph is given, the VSG encoder is responsible for inferring the representations. To prevent information leakage, an empty embedding $\boldsymbol{z}_0$ is given at the beginning of the sequence by convention. The Transformer learns to model the prior distribution of $\boldsymbol{z}$ by minimizing the negative log-likelihood for latent codes with $L_{latent} = \mathbb{E}_{\boldsymbol{z}\sim \mathrm{p}(\boldsymbol{z}_{data})}(-\log \mathrm{p}(\boldsymbol{z}))$. Although scene graph embeddings are known in the inference stage, we also optimize the mean square error loss between the input and output graph embeddings.

During the inference, the starting frame and video scene graphs are given. The discrete latent embeddings of the starting frame are computed by the VQ-encoder, while the VSG encoder infers the scene graph representations. The initial sequence is constructed with the empty embedding, these latent embeddings, and graph representations. The latent embeddings of the future frames are randomly sampled from the categorical distribution learned by the auto-regressive Transformer. The

VQ-decoder synthesizes the video with the known and predicted latent embeddings. There are three approaches to inserting graph representations: 1) Insert the graph representation in front of the latent embeddings of the corresponding frame as shown in Figure 3; 2) insert the graph representation after the latent embeddings of the corresponding frame; 3) insert all graph representations in front of the latent embeddings of all frames. We use the first approach because it has the best performance.

5 Experiments

Dataset and evaluation metrics. We split a sub-dataset from the video scene graph dataset Action Genome (Ji et al., 2020), which is built upon the In-Home dataset Charades (Sigurdsson et al., 2016). There are $70k$ training videos and $7k$ test videos with 36 object categories and 17 relationship categories. Each video with a resolution of 128×128 contains 16 frames and a variable number of video scene graphs. The maximum number of nodes in scene graphs is set to 5. We sampled every 5 frames from the original videos to capture large motion. More details are given in Appendix A.2.

To evaluate the model performance, we adopt the Fréchet Video Distance (FVD) proposed by Unterthiner et al. (2018) that estimates the distribution distance between real and synthesized videos in the feature space. Following (Le Moing et al., 2021), we calculate the mean and standard deviation of FVD in 5 evaluations. Furthermore, we evaluate the structural similarity index measure (SSIM) (Wang et al., 2004) which evaluates the similarity between the original and synthetic frames.

Implementation details. We train all our models using ADAM optimizer (Kingma & Ba, 2014). The video scene graph encoder is trained on 2 RTX 2080 TI GPUs with a batch size of 12 images for about 20 hours. Both the VQ-VAE and the auto-regression Transformer are trained on 8 RTX 3090 TI GPUs for about 48 hours, with batch sizes of 32 and 64 videos respectively. It takes about 14 hours to synthesize $7k$ videos for evaluation using a single RTX 2080 TI GPU. See Appendix A.4 for more details about the hyperparamters and model architecture.

5.1 Comparison with state-of-the-art video synthsis methods

To the best of our knowledge, there is no previous work that synthesizes videos from video scene graphs. To verify whether video scene graphs have a positive effect on video synthesis, we evaluated the performance of some advanced models on our dataset that can synthesize future frames given a starting frame. Please note that we have selected only those works whose official code is published, and the amount of computation required to reproduce them is within our capabilities. Table 1 demonstrates that our model SGVS outperforms other state-of-the-art methods that only use the first frame to synthesize a video. With the help of video scene graphs, FVD between the real videos and videos synthesized by SGVS is 382.2, which is 44.5 lower than CCVS and 190.5 lower than LVT. For SSIM, SGVS also has the best performance. In addition, when we train the models on the sub-dataset split from Action Genome, VideoGPT does not perform as well as expected. We speculate that this is because our dataset is more complex than the widely-used datasets for video synthesis (Soomro et al., 2012; Ebert et al., 2017). The scenes are diverse, the camera pose is not fixed, and the motions in the videos are large.

Table 1: Comparison with state-of-the-art video synthesis methods on the split Action Genome. Given the starting frame, a 16-frame video with 128×128 resolution is synthesized. For our method SGVS, video scene graphs are provided as additional conditions. With the help of the VSG encoder, SGVS achieves the best performance in all metrics.✓ denotes video scene graphs are provided.

Method	Graph	FID (↓)	SSIM (↑)
MoCoGAN (Tulyakov et al., 2018)	-	911.2 ± 18.6	0.459
LVT (Rakhimov et al., 2020)	-	572.7 ± 25.4	0.493
VideoGPT (Yan et al., 2021)	-	888.6 ± 19.7	0.472
CCVS (Le Moing et al., 2021)	-	426.7 ± 21.4	0.516
SGVS (Ours)	✓	$\mathbf{382.2\pm15.2}$	**0.565**

Table 2: We evaluate the synthesize performance by integrating the VSG encoders pre-trained with different contrastive loss functions in SGVS. The first row indicates that the VSG encoder is not pre-trained, but optimized in the training of the auto-regressive Transformer. The second to fourth rows denote different losses are activated (✓) in the pre-training.

L_{intra}	L_{inter}	$L_{finegrain}$	**FVD (↓)**	**SSIM (↑)**
-	-	-	457.1 ± 10.6	0.509
✓	-	-	403.1 ± 12.7	0.541
✓	✓	-	395.1 ± 16.1	0.551
✓	✓	✓	**382.2** ± **15.2**	**0.565**

Table 3: Ablation study of graph representations. Ø indicates video scene graphs are not provided. The second to fourth rows show the different representation insertion order used by SGVS, the numbers correspond to the order numbers in Section 4.

Order	**FVD (↓)**	**SSIM (↑)**
Ø	426.7 ± 21.4	0.516
1	**382.2 ± 15.2**	**0.565**
2	391.8 ± 17.5	0.553
3	387.2 ± 13.1	0.559

5.2 Graph representation learning analysis

In contrast to the previous generative methods, we introduce video scene graphs as a condition to guide video synthesis. In order to clarify how graph representation learning contributes to the synthesis performance, we first ablate different contrastive multi-modal losses and present the results in Table 2. The first row indicates that the video scene graph encoder is integrated into SGVS without pre-training and optimized with the auto-regressive transformer. The FVD score increases significantly from 382.2 to 457.1. In this case, the quality of the generated videos is even worse than if only the first frame was given. We conjecture that video scene graph representations cannot be learned without a reasonably designed loss function. In particular, it is simultaneously optimized when training a complex generative model. Although we pre-train the VSG encoder with only the graphical intra-video contrastive loss, FVD rapidly drops to 403.1. Because the VSG encoder learns temporal dependencies, which are crucial for graph representation inference. The graphical inter-video contrastive loss also helps the VSG encoder pre-training, while graph representation quality can be further improved by using the graphical fine-grained contrastive loss. The FVD score decreases to 382.2, while SSIM increases to 0.565.

To demonstrate how the node and edge representations are associated with regions of interest using the graphical fine-grained contrastive loss, an instance of the attention weights in Equation 3 is visualized in Figure 4. The first row shows the scene graph with different highlighted nodes and edges, while the second row shows the corresponding attention maps. With our contrastive multi-modal learning, not only node representations can be localized to entities, but also semantic relationships can be projected to critical regions. For example, `holding` and `touching` are usually closely associated with the hand or arm of `person`.

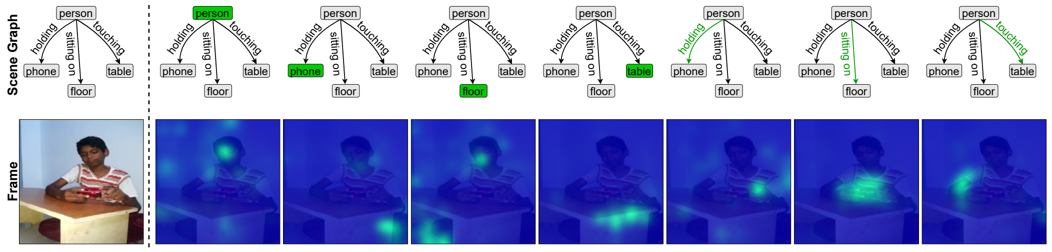


Figure 4: Attention maps of the node and edge representations to sub-regions in the frame. The node or edge in the scene graph corresponding to the attention map in the second row is highlighted in green. `phone` cannot match the correct region due to the small bounding box and low resolution.

5.3 Ablation study

To verify the effect of video scene graphs on video synthesis, we ablate the video scene graph encoder during the inference. In addition, we evaluate the performance of SGVS using different

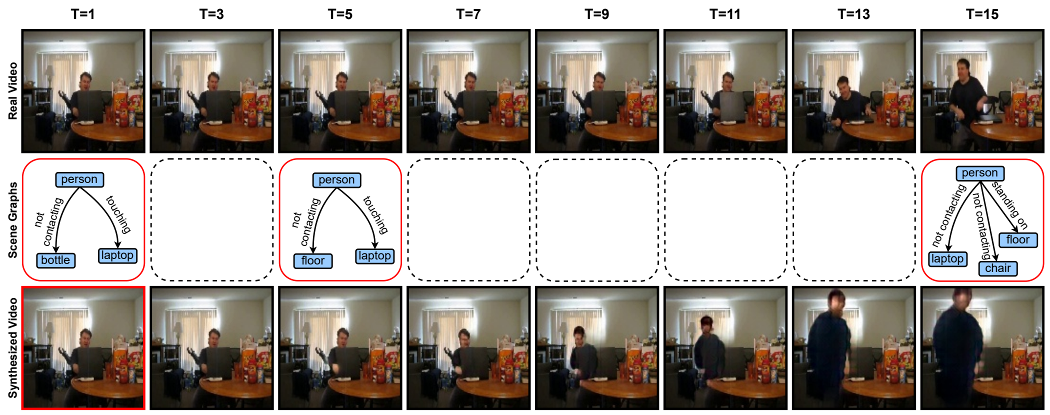


Figure 5: Qualitative result of complex semantic video synthesis. Given the first frame and a several video scene graphs (with red bounding boxes), SGVS can synthesize not only simple videos with small or repetitive movements, but also long-term videos with complex semantic content.

insertion graph representation orders introduced in Section 4: 1) Insert the graph representation in front of the latent embeddings of the corresponding frame; 2) insert the graph representation after the latent embeddings of the corresponding frame; 3) insert all graph representations in front of the latent embeddings of all frames. The results are shown in Table 3. The first row indicates no scene graphs are given, while the auto-regressive Transformer predicts latent embeddings only based on the first frame. Without video scene graphs, it is difficult for SGVS to infer what will happen in the future and FVD decreases to 426.7. Furthermore, the insertion order of graph representations also effects the synthesis performance, although it is not very critical. SGVS performs best when inserting the graph representation in front of the latent embeddings of the corresponding frame. More results of the ablation study can be found in Appendix A.5.

5.4 Qualitative Results

Figure 5 shows the qualitative result for semantic video synthesis from video scene graphs. The three rows from top to bottom are respectively the original video, video scene graphs, and the synthetic video. As discussed in Section 3, the given scene graphs are discrete. Given the first frame and several scene graphs (with red bounding boxes), 15 future frames are synthesized by SGVS, while the even columns are omitted to save space. SGVS can synthesize semantically controllable videos, in this example the `person` is `sitting` and using a `laptop`, then `stands` up. Different from the original video, the `standing` action starts at $T = 9$ instead of $T = 15$. Since the command `standing` is given in the scene graph at $T = 15$, SGVS infers that the complete standing will take more time based on the learned knowledge. In the original video, the person is not yet fully standing up. The difference could be eliminated by giving more video scene graphs as constraints. However, some visual details, such as the face, are not well rendered in the last few generated frames, because generating long-term videos with large motions is very challenging. For videos with small motion, SGVS can render better details. Due to space limitation, we present more qualitative results and discuss the limitations of SGVS in Appendix A.6.

6 Conclusion

In this paper, we propose a video scene graph-to-video synthesis framework SGVS which aims to synthesize complex semantic videos. Through contrastive multi-modal learning, our video scene graph encoder can infer continuous graph representations based on the given discrete scene graphs. Given the starting frame and graph representations as constraints, the latent embeddings of future frames are sampled from the distribution learned by the auto-regressive Transformer and converted to frames by the VQ-VAE. Our experiments demonstrate that video scene graphs have a positive effect on video synthesis. The source code will be released after publication.

7 Ethics statement

As machine learning methods are increasingly used in everyday life, it makes sense to consider the potential social impact of our work. Our work could potentially be used for deep fake as well as other state-of-the-art generative models. Since our model can synthesize videos with specific semantic content, this even makes deep fake more flexible. Developing better models has the potential to be used maliciously to violate human likeness rights or create false information. On the other hand, a good video synthesis model helps the film and video game industries, for example, by replacing live actors in dangerous scenes. It can be also very promising in the metaverse.

8 Reproducibility statement

The complete code, processed data, and trained model in this paper will be released after publication. For details on reproducing the video scene graph encoder and frame encoder, please refer to Section 3.1, Section 3.2 and Appendix A.4. We adopt the VQ-VAE from Le Moing et al. (2021), whose code is already released. For details on reproducing the auto-regressive Transformer, please refer to Section 3 and Appendix A.4. The dataset we used is split from Action Genome (Ji et al., 2020). The pre-processing is claimed in Appendix A.2. We use the published official codes to retrain the state-of-the-art models in Table 1 on our split dataset:

- MoCoGAN: `https://github.com/sergeytulyakov/mocogan`
- LVT: `https://github.com/rakhimovv/lvt`
- VideoGPT: `https://github.com/wilson1yan/VideoGPT`
- CCVS: `https://github.com/16lemoing/ccvs`

REFERENCES

Dinesh Acharya, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool. Towards high resolution video generation with progressive growing of sliced wasserstein gans. *arXiv preprint arXiv:1810.02419*, 2018.

Yogesh Balaji, Martin Renqiang Min, Bing Bai, Rama Chellappa, and Hans Peter Graf. Conditional gan with discriminative filter generation for text-to-video synthesis. In *IJCAI*, volume 1, pp. 2, 2019.

Dong Cao, Qunhe Zhao, and Yunbin Fu. Using spatial temporal graph convolutional network dynamic scene graph for video captioning of pedestrians intention. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*, pp. 179–183, 2020.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

Anoop Cherian, Chiori Hori, Tim K Marks, and Jonathan Le Roux. (2.5+ 1) d spatio-temporal scene graphs for video question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 444–453, 2022.

Meng-Jiun Chiou, Henghui Ding, Hanshu Yan, Changhu Wang, Roger Zimmermann, and Jiashi Feng. Recovering the unbiased scene graphs from the biased ones. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 1581–1590, 2021.

Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. *arXiv preprint arXiv:1907.06571*, 2019.

Yuren Cong, Wentong Liao, Hanno Ackermann, Bodo Rosenhahn, and Michael Ying Yang. Spatial-temporal transformer for dynamic scene graph generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16372–16382, 2021.

Vinay Damodaran, Sharanya Chakravarthy, Akshay Kumar, Anjana Umapathy, Teruko Mitamura, Yuta Nakashima, Noa Garcia, and Chenhui Chu. Understanding the role of scene graphs in visual question answering. *arXiv preprint arXiv:2101.05479*, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Naina Dhingra, Florian Ritter, and Andreas Kunz. Bgt-net: Bidirectional gru transformer network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2150–2159, 2021.

Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *Proceedings of the Conference on Robot Learning*, pp. 344–356, 2017.

Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic vqgan and time-sensitive transformer. *arXiv preprint arXiv:2204.03638*, 2022.

Roei Herzig, Amir Bar, Huijuan Xu, Gal Chechik, Trevor Darrell, and Amir Globerson. Learning canonical representations for scene graph to image generation. In *European Conference on Computer Vision*, pp. 210–227. Springer, 2020.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pre-training for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10236–10247, 2020.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3668–3678, 2015.

Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1219–1228, 2018.

Nal Kalchbrenner, Aäron Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *International Conference on Machine Learning*, pp. 1771–1779. PMLR, 2017.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020.

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

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.

Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. Ccvs: Context-aware controllable video synthesis. *Advances in Neural Information Processing Systems*, 34:14042–14055, 2021.

Rongjie Li, Songyang Zhang, Bo Wan, and Xuming He. Bipartite graph network with adaptive message passing for unbiased scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11109–11119, 2021.

Rongjie Li, Songyang Zhang, and Xuming He. Sgtr: End-to-end scene graph generation with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19486–19496, 2022a.

Yiming Li, Xiaoshan Yang, and Changsheng Xu. Dynamic scene graph generation via anticipatory pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13874–13883, 2022b.

Yitong Li, Martin Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Xin Lin, Changxing Ding, Jinquan Zeng, and Dacheng Tao. Gps-net: Graph property sensing network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3746–3753, 2020.

Daqi Liu, Miroslaw Bober, and Josef Kittler. Constrained structure learning for scene graph generation. *arXiv preprint arXiv:2201.11697*, 2022.

Hengyue Liu, Ning Yan, Masood Mortazavi, and Bir Bhanu. Fully convolutional scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11546–11556, 2021.

Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European conference on computer vision*, pp. 852–869. Springer, 2016.

Xinyu Lyu, Lianli Gao, Yuyu Guo, Zhou Zhao, Hao Huang, Heng Tao Shen, and Jingkuan Song. Fine-grained predicates learning for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19467–19475, 2022.

Willi Menapace, Stéphane Lathuilière, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. Playable video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10061–10070, 2021.

Victor Milewski, Marie-Francine Moens, and Iacer Calixto. Are scene graphs good enough to improve image captioning? *arXiv preprint arXiv:2009.12313*, 2020.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Ruslan Rakhimov, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. Latent video transformer. *arXiv preprint arXiv:2006.10704*, 2020.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Xuanchi Ren and Xiaolong Wang. Look outside the room: Synthesizing a consistent long-term 3d scene video from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3563–3573, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Masaki Saito, Shunta Saito, Masanori Koyama, and Sosuke Kobayashi. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal gan. *International Journal of Computer Vision*, 128(10):2586–2606, 2020.

Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pp. 510–526. Springer, 2016.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Kaihua Tang, Hanwang Zhang, Baoyuan Wu, Wenhan Luo, and Wei Liu. Learning to compose dynamic tree structures for visual contexts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6619–6628, 2019.

Yao Teng and Limin Wang. Structured sparse r-cnn for direct scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19437–19446, 2022.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1526–1535, 2018.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. G3an: Disentangling appearance and motion for video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5264–5273, 2020.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. *arXiv preprint arXiv:1906.02634*, 2019.

Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. Nüwa: Visual synthesis pre-training for neural visual world creation. *arXiv preprint arXiv:2111.12417*, 2021.

Li Xu, Haoxuan Qu, Jason Kuen, Jiuxiang Gu, and Jun Liu. Meta spatio-temporal debiasing for video scene graph generation. *arXiv preprint arXiv:2207.11441*, 2022.

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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1316–1324, 2018.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

Chiao-An Yang, Cheng-Yo Tan, Wan-Cyuan Fan, Cheng-Fu Yang, Meng-Lin Wu, and Yu-Chiang Frank Wang. Scene graph expansion for semantics-guided image outpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15617–15626, 2022a.

Ruihan Yang, Prakhar Srivastava, and Stephan Mandt. Diffusion probabilistic modeling for video generation. *arXiv preprint arXiv:2203.09481*, 2022b.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5831–5840, 2018.

Ji Zhang, Kevin J Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. Graphical contrastive losses for scene graph parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11535–11543, 2019.

A Appendix

A.1 Transformer architecture

We adopt a GPT-like multi-layer Transformer in this paper. Each transformer layer consists of a classical multi-head attention module, a feed-forward network, and normalization layers as shown in Figure 6. We use the original full attention mechanism but not sparse attention in Transformers, which is defined as:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{D_k}}\right)\boldsymbol{V}, \tag{5}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are the linear transformations of the queries, keys and values, respectively. The feed-forward network is a two-layer perceptron, while layer normalization is used in the Transformers for normalization.

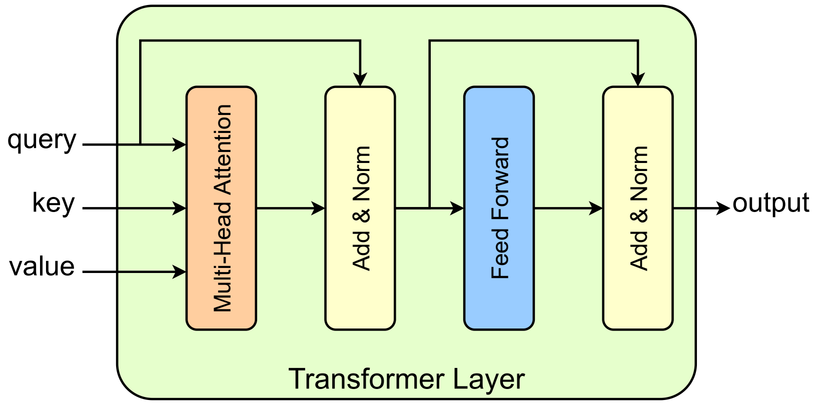


Figure 6: Architecture of the Transformer layer, which contain a multi-head attention module, a feed-forward network, and two normalization layers.

A.2 Dataset details

We split a sub-dataset from Action Genome (Ji et al., 2020), which is built upon Charades (Sigurdsson et al., 2016). To include more complex semantic variations in the 16-frame video, we sampled 1 frame every 5 frames from the original videos of Charades and resize the sampled frames to a resolution of 128×128. We only keep the objects whose bounding boxes with short edges larger than 16 pixels. In order to avoid overly complex scene graphs that make the representations difficult to infer, we reduce the graph fidelity by cutting out redundant nodes in the scene graph and keep a maximum of 5 object nodes. In addition, each video contains at least 5 video scene graphs so that the video scene graph (VSG) encoder has enough information to infer the graph representations that are not given. In the split dataset, there are 36 object categories and 17 relationship categories. The distribution of object and relationship occurrences are illustrated in Figure 7. **We will release the split dataset and processing code after publication**.

A.3 Metrics details

Fréchet video distance (FVD). FVD (Unterthiner et al., 2018) is developed from Fréchett Inception Distance (FID) (Heusel et al., 2017), which is widely-used to evaluate the performance of image generation models. FVD takes into account a distribution over entire videos in order to avoid the disadvantages of frame-level metrics. A pre-trained Inflated 3D Convnet (Carreira & Zisserman, 2017) is used to capture video feature distributions. The 2-Wasserstein distance between the ground truth video distribution and the synthetic video distribution is calculated as the metrics.

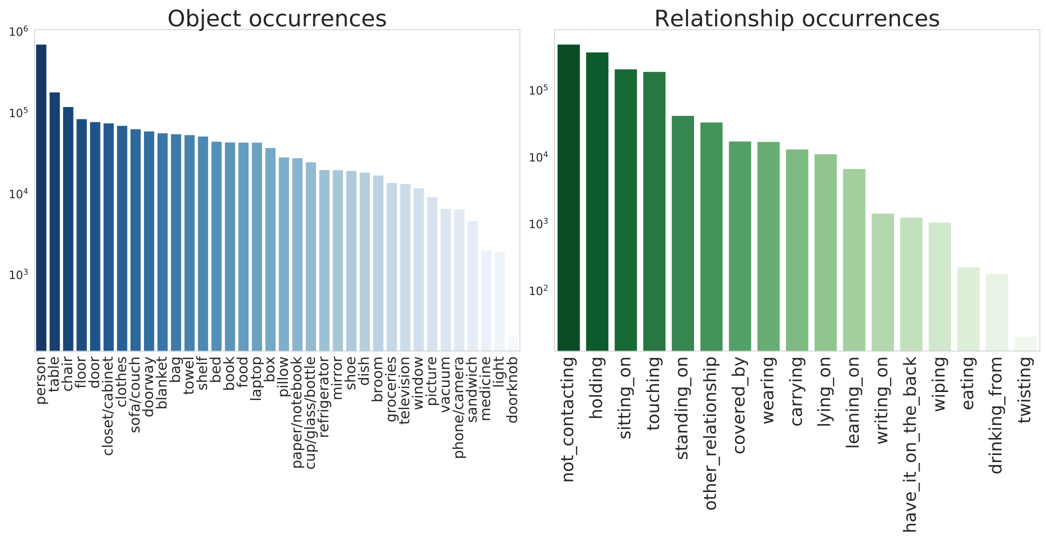


Figure 7: Distributions of object (left) and relationship (right) occurrences for the sub-dataset split from the Action Genome dataset.

Structural similarity index measure (SSIM). SSIM (Wang et al., 2004) is a per-frame perceptual metrics that measures the similarity between two images. The statistical measure combines three different factors: luminance, variance and correlation. We first split the ground truth videos and synthetic videos into single frames. Then we calculate SSIM between the ground truth frames and synthetic frames. The average SSIM of all frames is taken as the final result.

A.4 TECHNICAL IMPLEMENTATION DETAILS

Video scene graph representation learning framework. In the video scene graph encoder, both the spatial Transformer and temporal Transformer have 3 Transformer layers. We employ 4 attention heads for each attention module, while the dimension d of the input queries, keys, and values is set to 256. The encodings are only added to queries and keys when using the attention modules. For the frame encoder, we adopt the CNN-based model used in (Xu et al., 2018), which is built upon Inception-v3 model (Szegedy et al., 2016). The input frames are first resized to a resolution of 299×299, while the size of the feature maps extracted by the CNN backbone is $768 \times 17 \times 17$. A 1×1 convolution layer is exploited to reduce the dimension of the feature maps to $d = 256$. Then we use a global average pooling layer to convert the feature maps to the frame vectors. We train the video scene graph encoder and frame encoder using ADAM optimizer (Kingma & Ba, 2014) with a learning rate of 1×10^{-4} and a batch size of 12 images. The training takes about 20 hours on 2 RTX 2080 TI GPUs. The values of different loss functions in the training are shown in Figure 8.

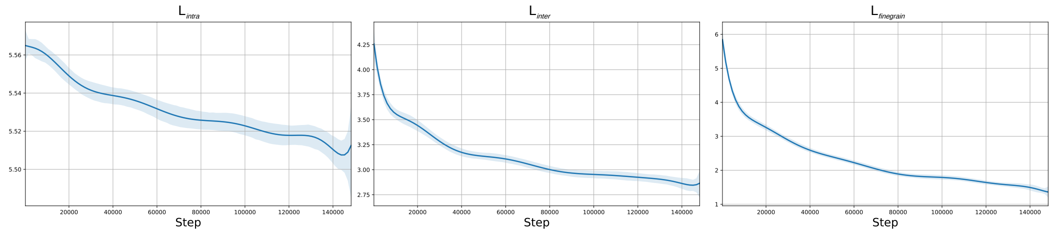


Figure 8: Graphical intra-video contrastive loss, inter-video contrastive loss and fine-grained contrastive loss in the training.

Video scene graph-to-video synthesis framework. We adopt the VQ-VAE from Le Moing et al. (2021) and use almost the same hyperparameters. The encoder of the VQ-VAE converts a 128×128 frame into a $512 \times 8 \times 8$ feature map. The 8×8 sub-vectors of the feature map are then quantified to the discrete latent embeddings. The length of the latent codebook is set to 1024 to shorten the training time. The 8×8 discrete latent embeddings are reconstructed to a video frame by the decoder of the VQ-VAE. We train the VQ-VAE using ADAM optimizer (Kingma & Ba, 2014) with a learning rate of 2×10^{-4} and a batch size of 32 videos on 8 RTX 3090 TI GPUs for about 48 hours.

The auto-regressive Transformer consists of 24 Transformer layers with a head number of 16. Due to the complexity of the auto-regression task, the embedding dimension of the attention module is set to 1024. Therefore, a linear transformation is utilized to project the dimension of video scene graph representations from 256 to 1024, while 1024 latent embeddings with dimension $1024d$ are learned during the training. We train the auto-regressive Transformer using ADAM optimizer (Kingma & Ba, 2014) with a learning rate of 1×10^{-5} and a batch size of 64 videos on 8 RTX 3090 TI GPUs for about 48 hours. Furthermore, the video scene graph encoder is frozen during the training of the auto-regressive Transformer.

A.5 Additional ablation study

To explore the optimal model structure, we train the auto-regressive Transformer with different Transformer layers. The results are shown in Table 4. We used the same settings for all experiments except for the number of layers $= 32$. For the Transformer with 32 layers, the batch size is reduced to 24 videos due to GPU memory limitation. In this case, the FVD score decreases to 410.4. There are two possibilities, either the model is too complex causing overfitting, or the optimization using a smaller batch size does not perform well. Finally, we adopt the auto-regressive Transformer with 24 Transformer layers which has the best performance in practice.

Table 4: Ablation study of Transformer layers in the auto-regressive Transformer.

Layer number	**FVD** (↓)
4	476.0 ± 16.3
8	422.9 ± 17.5
16	399.1 ± 13.7
24	**382.2 ± 15.2**
32	410.4 ± 20.4

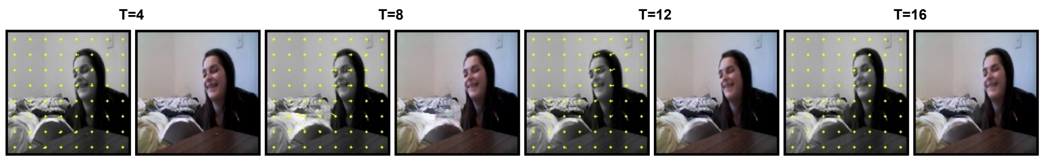


Figure 9: Qualitative result for simple video synthesize, in which the girl keeps holding the book. The motion in the original video is very small. Only 4 synthesized frames and their optical flow are shown. In this case, the details such as the face are well rendered.

A.6 Additional qualitative results and limitations

Additional qualitative results. The details such as the human face are not well presented in Figure 5. As discussed, the reason is that the motion in the video is quite large. Another simple example is shown in Figure 9. In the original video, the girl is holding and looking at the book (all the video scene graphs are the simple triplet `person-holding-book`). Although there is some change in the position of the girl's head and book, it is not significant. In this case, SGVS can render better details and perform well. The original video and some generated frames are omitted because the synthetic frames are very close to the original ones and the motion is small. To visualize the small motion better, we also compute the optical flows for the shown synthetic frames. More videos synthesized by SGVS are attached to the supplementary material.

In Figure 10, we show the video synthesized by CCVS (Le Moing et al., 2021), which only use the first frame as input, and the video synthesized by our SGVS which use the first frame and also the video scene graphs. With the help of the input video scene graphs, SGVS can synthesize higher quality frames, especially those far from the starting frame. In this example, there are no significant semantic changes in the video scene graphs. They control SGVS to generate the frames that maintain the current drinking action, whereas the distortion in the frames generated by CCVS is getting worse.

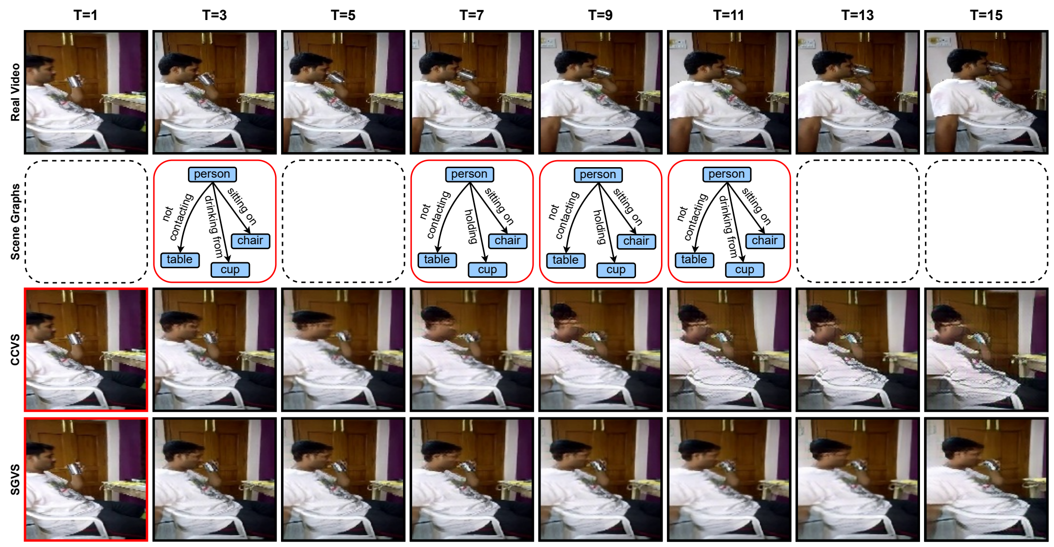


Figure 10: Comparison between the videos synthesized by CCVS and SGVS. The real frames are given in the first row, while the corresponding video scene graphs are shown in the second row. The video synthesized by SGVS has higher fidelity with the help of the discrete video scene graphs.

Limitations. Since the resolution of our generated video is 128×128, this constraint makes some small objects such as the phone and medicine cannot be presented very clearly. In addition, for some videos containing the large motion, the auto-regressive transformer cannot successfully predict the sequence of the latent embeddings. These videos usually involve a change of scene or camera pose. An example is shown in Figure 11.

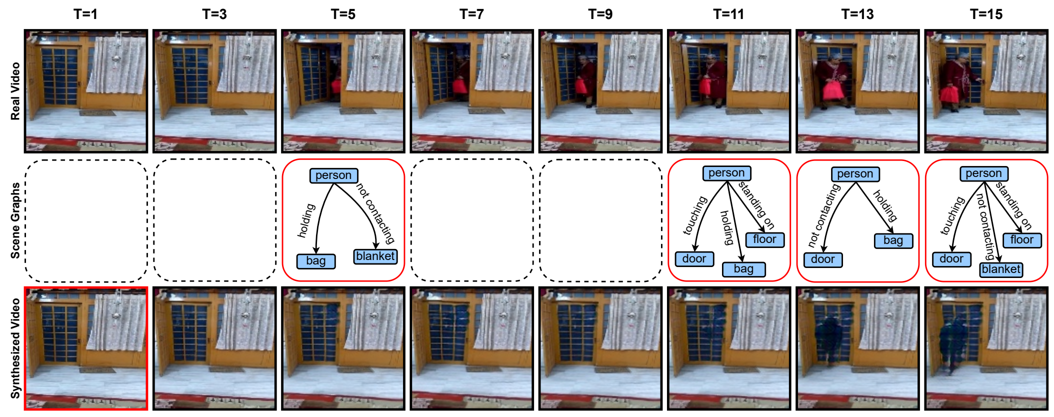


Figure 11: Failure to synthesize a complex video with large motion. The person and the bag cannot be synthesized since they do not appear in the first frame. In this case, only the silhouettes of a standing person are visible in the frames of $T = 13$ and $T = 15$.