SSGVS: Semantic Scene Graph-to-Video Synthesis

Yuren Cong\textsuperscript{1}, Jinhui Yi\textsuperscript{2}, Bodo Rosenhahn\textsuperscript{1}, Michael Ying Yang\textsuperscript{3}

\textsuperscript{1}TNT/L3S, Leibniz University Hannover, \textsuperscript{2}University of Bonn, \textsuperscript{3}SUG, University of Twente

Abstract

As a natural extension of the image synthesis task, video synthesis has attracted a lot of interest recently. Many image synthesis works utilize class labels or text as guidance. However, neither labels nor text can provide explicit temporal guidance, such as when an action starts or ends. To overcome this limitation, we introduce semantic 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 semantic scene graph-to-video synthesis framework (SSGVS), based on the pre-trained VSG encoder, VQ-VAE, and autoregressive Transformer, is proposed to synthesize a video given an initial scene image and a non-fixed number of semantic scene graphs. We evaluate SSGVS and other state-of-the-art video synthesis models on the Action Genome dataset and demonstrate the positive significance of video scene graphs in video synthesis. The source code is available at https://github.com/yrcong/SSGVS.

1. Introduction

With the tremendous breakthroughs in image synthesis \cite{22,41,62} 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. 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 \cite{16,17,29} 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. AG2Vid \cite{3} introduced continuous action graphs for video synthesis, which specify semantic actions and spatial layouts of objects. Although action graphs contain explicit spatial-temporal information, it is demanding to artificially define the layout variations of all objects as input when generating real scene videos.

Is there a balanced guiding condition that contains semantic information and clearly represents temporal dependencies? Based on this observation, we make use of semantic video scene graphs for the task of 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 \cite{19}. Semantic video scene graphs can be viewed as several static scene graphs on an temporal axis, which only describe the semantic content of the frames in the video. The ideal 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 challenging compared to giving other temporal conditions such as trajectories \cite{26}, action labels \cite{36}, or overall scene conditions such as text, because scene graphs are relatively difficult to create.

In this paper, we propose a novel semantic scene graph-to-video synthesis framework (SSGVS) that can synthesize a fixed-length video with an initial scene image and discrete semantic video scene graphs, as shown in Fig. 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 location of the given
Figure 1. Given a starting frame and a non-fixed number of semantic scene graphs (with red bounding boxes), SSGVS can synthesize a fixed-length complex video with $128 \times 128$ 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.

video scene graphs, nor does it require additional spatial information, which makes SSGVS more feasible in practice. For the generative model, we utilize a popular combination of VQ-VAE and auto-regressive Transformer [14, 26, 59] 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 propose a novel semantic scene graph-to-video synthesis framework (SSGVS), which can synthesize a fixed-length video with an initial scene image and discrete semantic 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 split a sub-dataset from the Action Genome dataset [18] and conduct experiments to demonstrate the benefits of using semantic video scene graphs as condition for video synthesis. Compared to other state-of-the-art works, SSGVS can generate better quality semantic videos.

2. Related Work

Scene graph. Scene graphs have first been proposed by [20] 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 works [6, 9, 12, 27, 28, 31–33, 35, 47, 48, 64] that have achieved incredible results of image scene graph generation on the datasets [24, 25, 34]. Recently, the dataset Action Genome [18] is proposed, which extends image scene graphs to video scene graphs by adding the temporal dimension. Some works [8, 30, 57] captures spatial and temporal dependencies and generates video scene graphs. 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 [20], visual question answering [10], and synthesis tasks [15, 19, 46, 60]. Video scene graphs have also been used for video question answering [5] and video captioning [4]. In this paper, we use video scene graphs for video synthesis, which are purely semantic and do not include spatial information of entities.

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 [1, 36, 53] or recurrent neural networks [7, 42, 49]. However, as the length of the generated videos becomes longer, the quality of the videos synthesized by these models decreases significantly. Auto-regressive models [21, 55] have become popular in this field, even though their inference speed is slow. Many of these methods [14, 17, 26, 56, 59] introduce VQ-VAE [51] and Transformers [52] to improve performance. Furthermore, some works [16, 61] 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 in-
formation. 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 or semantic labels [63], while text-to-video models [2, 16, 17] are rapidly developing. To further control the video content, trajectory of a robotic arm [26] and camera motions [40] are exploited. AG2Vid [3] introduced continuous action graphs for video synthesis. However, the scene layout and spatial variations are necessary input, which limits the application. In this paper, we introduce semantic video scene graphs as the guidance. We use a scene image and purely semantic video scene graphs as the input of an auto-regressive model to synthesize a video. 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. Different from previous works, SSGVS allows precise control of the moment when the episodes occur, while the input is relatively easy to achieve.

3. Video scene graph representation learning

Many text-to-image generation works [39, 41] using CLIP text encoder [37] 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 Fig. 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.

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 $G = (N, E)$, the nodes $N$ and edges $E$ in the graph are viewed as tokens with semantic information. We decompose the structure of the graph into a sequence $S$ consisting of node tokens and edge tokens. In order to obtain the graph context through self attention mechanism, we introduce a special $[\text{context}]$ token and place it always first in the sequence $S$. Each token corresponds to a learned embedding $e \in \mathbb{R}^d$ 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 $E(e) \in \mathbb{R}^d$ and node encodings $E(N)$ with learned embeddings. For the
edge \( e_{ij} \) from the node \( n_i \) to the node \( n_j \), the edge encoding is calculated as \( E(e_{ij}) = E(n_i) - E(n_j) \). 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 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. We use the original full attention mechanism but not sparse attention in Transformers, which is defined as:

\[
\text{Attention}(Q, K, V) = \text{Softmax}\left( \frac{QK^T}{\sqrt{d_k}} \right) V,
\]

where \( Q, K, \) and \( V \) 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. 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 Sec. 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 \([\text{mask}]\) 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 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 [11] that masks some words in the sentence and reconstructs them using the context. The difference is that these graph representations \( g \in \mathbb{R}^d \) 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 [45]. A \( 1 \times 1 \) convolution layer is exploited to reduce the dimension of the feature maps extracted by the pre-trained CNN. The frame vectors \( v \in \mathbb{R}^d \) are derived from the feature maps \( \mathbb{R}^{k \times w \times d} \) 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 three contrastive loss functions for graph representation learning, namely graphical intra-video contrast 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 \( g \) from visual representations \( v \) through temporal dependencies. We compute the graphical intra-video contrastive loss of a video with \( T \) frames as:

\[
L_{\text{intra}} = -\sum_{i=1}^{T} \left( \log \frac{\exp(\text{sim}(g_i, f_i))}{\sum_{j=1}^{T} \exp(\text{sim}(g_j, f_i))} + \log \frac{\exp(\text{sim}(g_i, f_j))}{\sum_{j=1}^{T} \exp(\text{sim}(g_i, f_j))} \right),
\]

where \( \text{sim}(g_i, f_j) \) 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:

\[
L_{\text{inter}}^{t} = -\sum_{i=1}^{B} \left( \log \frac{\exp(\text{sim}(g_i, f_i))}{\sum_{j=1}^{B} \exp(\text{sim}(g_j, f_i))} + \log \frac{\exp(\text{sim}(g_i, f_j))}{\sum_{j=1}^{B} \exp(\text{sim}(g_i, f_j))} \right),
\]

where the frame index \( t \) on the right side of the equation is omitted for brevity. The graph representation \( g_i \) and frame vector \( f_j \) are inferred from the the \( t \)-th frames of the \( i \)-th video and the \( j \)-th video in the batch. The complete \( L_{\text{inter}} \) are the sum of \( L_{\text{inter}}^{t} \) of \( T \) frames.

**Graphical fine-grained contrastive loss.** Inspired by [58], 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 $e$. 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 $c_i$ of the $i$-th node or edge in the graph as:

$$c_i = \sum_{j=1}^{hw} \frac{\exp(\text{sim}_d(e_i, r_j))}{\sum_{k=1}^{hw} \exp(\text{sim}_d(e_i, r_k))} r_j,$$

where $\text{sim}_d(e_i, r_j) = \frac{\exp(e_i^T r_j)}{\sum_{g=1}^{N_g} \exp(e_g^T r_j)}$.  

$r_j \in \mathbb{R}^d$ is the vector of the $j$-th sub-region in the feature map with $h \times w$ shape provided by the frame encoder, whereas $\text{sim}_d(e_i, r_j)$ 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 $S(G, F)$ between the scene graph $G$ and the frame $F$ can be formulated as $S(G, F) = \log(\sum_{i=1}^{N_g} \exp(\text{sim}(e_i, c_i))$. 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:

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

4. Semantic scene graph-to-video synthesis

In this section, we introduce a semantic scene graph-to-video synthesis framework (SSGVS) consisting of a video scene graph (VSG) encoder, a VQ-VAE, and an auto-regressive Transformer. The overview of SSGVS is illustrated in Fig. 3.

**VQ-VAE.** We adopt the VQ-VAE proposed by [26] 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 $z$ 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 $z$ and video scene graph representations $g$. The basic architecture of the auto-regressive Transformer is almost identical to the Transformers in Sec. 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 $z_0$ is given at the beginning of the sequence by convention. The Transformer learns to model the prior distribution of $z$ by minimizing
the negative log-likelihood for latent codes as:

\[ L_{\text{latent}} = \mathbb{E}_{z \sim \text{p}(z_{\text{data}})}(- \log \text{p}(z)). \] (6)

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 Fig. 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 one 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 [18], which is built upon the In-Home dataset Charades [43]. 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.

To evaluate the model performance, we adopt the Fréchet Video Distance (FVD) proposed by [50] that estimates the distribution distance between real and synthesized videos in the feature space. Following [26], we calculate the mean and standard deviation of FVD in 5 evaluations. Furthermore, we evaluate the structural similarity index measure (SSIM) [54] which evaluates the similarity between the original and synthetic frames.

#### Implementation details. 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. We train the video scene graph encoder and frame encoder using ADAM optimizer [23] with a learning rate of \(1 \times 10^{-4}\) and a batch size of 12 images. The training takes about 20 hours on 2 RTX 2080 TI GPUs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Graph FVD (↓)</th>
<th>SSIM (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCoGAN [49]</td>
<td>911.2 ± 18.6</td>
<td>0.459</td>
</tr>
<tr>
<td>LVT [38]</td>
<td>572.7 ± 25.4</td>
<td>0.493</td>
</tr>
<tr>
<td>VideoGPT [59]</td>
<td>888.6 ± 19.7</td>
<td>0.472</td>
</tr>
<tr>
<td>CCVS [26]</td>
<td>426.7 ± 21.4</td>
<td>0.516</td>
</tr>
<tr>
<td>SSGVS (Ours)</td>
<td><strong>382.2 ± 15.2</strong></td>
<td><strong>0.565</strong></td>
</tr>
</tbody>
</table>

We train the VQ-VAE using ADAM optimizer [23] with a learning rate of \(2 \times 10^{-4}\) 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 1024\(d\) are learned during the training. We train the auto-regressive Transformer using ADAM optimizer [23] with a learning rate of \(1 \times 10^{-5}\) 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. Refer to supplementary material for more details about the hyperparameters and model architecture.

### 5.1. Comparison with state-of-the-art methods

To the best of our knowledge, there is no previous work that synthesizes videos from semantic video scene graphs. To verify whether semantic 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. Tab. 1 demonstrates that our model SSGVS 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 SSGVS is 382.2, which is 44.5 lower than CCVS and 190.5 lower than LVT. For SSIM, SSGVS also has the best performance. In addition, when we train the models on the sub-dataset split from Action Genome, VideoGPT does not perform well. We speculate that this is because Action Genome dataset is more complex than the widely-used datasets for video synthesis [13,44]. The scenes are diverse, the camera pose is not fixed, and the motions in the videos are large.
Table 2. We evaluate the synthesize performance by integrating the VSG encoders pre-trained with different contrastive loss functions in SSGVS. 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.

<table>
<thead>
<tr>
<th>$L_{\text{intra}}$</th>
<th>$L_{\text{inter}}$</th>
<th>$L_{\text{finegrain}}$</th>
<th>FVD (↓)</th>
<th>SSIM (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>457.1 ± 10.6</td>
<td>0.509</td>
</tr>
<tr>
<td>√</td>
<td>-</td>
<td>-</td>
<td>403.1 ± 12.7</td>
<td>0.541</td>
</tr>
<tr>
<td>√</td>
<td>√</td>
<td>-</td>
<td>395.1 ± 16.1</td>
<td>0.551</td>
</tr>
<tr>
<td>√</td>
<td>√</td>
<td>√</td>
<td>382.2 ± 15.2</td>
<td>0.565</td>
</tr>
</tbody>
</table>

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 Tab. 2. The first row indicates that the video scene graph encoder is integrated into SSGVS 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 Eq. (4) is visualized in Fig. 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.

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 SSGVS using different insertion graph representation orders introduced in Sec. 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 Tab. 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 SSGVS 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. SSGVS performs best when inserting the graph representation in front of the latent embeddings.
of the corresponding frame.

To explore the optimal model structure, we train the auto-regressive Transformer with different Transformer layers. The results are shown in Tab. 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.

<table>
<thead>
<tr>
<th>Layer number</th>
<th>FVD (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>476.0 ± 16.3</td>
</tr>
<tr>
<td>8</td>
<td>422.9 ± 17.5</td>
</tr>
<tr>
<td>16</td>
<td>399.1 ± 13.7</td>
</tr>
<tr>
<td>24</td>
<td><strong>382.2 ± 15.2</strong></td>
</tr>
<tr>
<td>32</td>
<td>410.4 ± 20.4</td>
</tr>
</tbody>
</table>

5.4. Qualitative results

Fig. 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 Sec. 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 SSGVS, while the even columns are omitted to save space. SSGVS 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 $T = 15$, SSGVS 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, SSGVS can render better details. Due to space limitation, we present more qualitative results and discuss the limitations of SSGVS in the supplementary.

6. Conclusion

In this paper, we propose a semantic scene graph-to-video synthesis framework SSGVS which aims to synthesize complex semantic videos. Through contrastive multimodal 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.

Acknowledgements This work was supported by the Federal Ministry of Education and Research (BMBF), Germany under the project LeibnizKILabor (grant no. 01DD20003) and the AI service center KISSKI (grant no. 01IS22093C), ZDIN and DFG under Germany’s Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122).
References


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, pages 2818–2826, 2016. 4


