GraphMuse: Scene Graph to Image Generation using GAN

Arvind Kumar Nalli Kuppuswami
Dept. of Data Science
arnall@iu.edu

Dhanyasree Elangovan
Dept. of Data Science
delangov@iu.edu

Jayaraman Sridharan
Dept. of ISE
jsridhar@iu.edu

Abstract

This project aims to generate images using Generative Adversarial Networks (GANs) with scene graphs as input. Scene graphs are a useful way of organizing the logical and spatial representation of objects, their relationships, qualities, and spatial layouts in a graphical scene. The project focuses on exploring the use of scene graphs in image generation and evaluating the performance of GANs on this task. The goal is to generate images that preserve the composition of the original image, including objects, qualities, relationships, and spatial layouts. This approach increases control, semantic comprehension, effectiveness, and variety of the generated images. By feeding scene graphs into GANs, the researchers aim to enhance the capabilities of GANs in generating images.

1. Introduction

The aim of this project is to generate images using Generative Adversarial Networks (GANs) from scene graphs as input. Scene graphs are an efficient way of representing a graphical scene by organizing the logical and spatial representation of objects, their relationships, qualities, and spatial layouts. The main focus of this project is to explore the use of scene graphs in image generation and to evaluate the performance of GANs on this task.

In order to achieve a comprehensive understanding of the visual world, models must not only identify images but also have the ability to create them. The goal of this aims to investigate the Generation of the images through GAN where a scene graph is used as an input. A scene graph is an effective tool for comprehending scenes. Most computer games and vector-based graphics editing programs employ a general data structure called a scene graph to organize the logical and often spatial representation of a graphical scene.

Through this approach, we are trying to generate images that preserve the composition of the image including its objects, qualities, relationships, and spatial layouts in a systematic manner. This approach can increase control, semantic comprehension, effectiveness, and variety of generated images. By feeding scene graphs into GANs, we are attempting to increase control, semantic comprehension, effectiveness, and variety of generated images.

2. Background and Related Work

Recent research has shown that GANs can generate high-quality images from various inputs, including random noise, text, and image features. However, generating images from scene graphs is a challenging task that requires capturing the semantic relationships between objects in the scene. The work of Johnson et al. (2018) proposed an approach to generate images from scene graphs using GANs. This approach employs a generator network that takes a scene graph as input and outputs an image, and a discriminator network that evaluates the realism of the generated images.

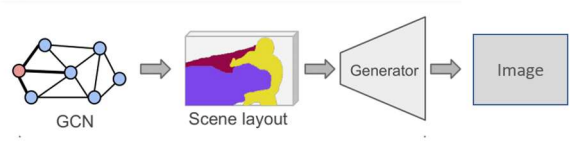


Figure 1: Network Architecture

Several recent studies have explored the use of scene graphs in image generation and manipulation. Dhamo et al. (2020) proposed a method for semantic image manipulation using scene graphs. Li et al. (2019) introduced Pastegan, a semi-parametric method for generating images from scene graphs. Chang et al. (2021) presented a comprehensive survey of scene graphs, including their generation and applications in various computer vision tasks. Yang et al. (2022) proposed a diffusion-based scene graph to image generation method with masked contrastive pre-training.

3. Method

Our objective is to create a model that can take a scene graph defining objects and their relationships as input and

produce a realistic image that corresponds to the graph. The three main problems are as follows:

- Firstly, we must build a method for processing the graph-structured input
- Second, we must ensure that the generated images obey the objects and relationships indicated by the graph and,
- Third, we must ensure that the synthesized images are realistic.

The process involves the conversion of scene graphs into images using an 'Image Generation Network'. Graph convolution network processes the scene graph G and outputs embedding vectors for each item; each layer of graph convolution combines information along the edges of the graph.

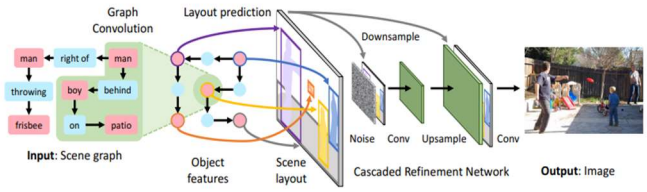
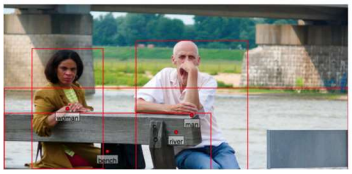


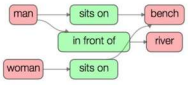
Figure 2: Detailed Network Architecture

By predicting bounding boxes and segmentation masks for each object using the object embedding vectors from the graph convolution network, we respect the objects and relationships from G. This scene layout serves as a bridge between the graph and the image domains. Each of its modules does the layout processing at progressively larger spatial scales, ultimately producing the image. By training the network in opposition to two discriminator networks that urge the image to both seem realistic and contain actual, recognizable objects, we are able to create realistic images.

Detailed Description of each of the components:

Scene Graph: A scene graph detailing the objects and their relationships serves as the model's input. In the first stage of processing, we use a learned embedding layer, similar to the embedding layer typically used in neural language models, to transform each node and edge of the graph from a categorical label to a dense vector.





A man and a woman sit on a park bench along a river.

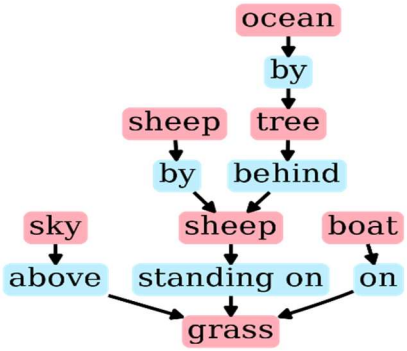


Figure 3: Scene Graph

Graph Convolution Network: We require a neural network module that can function natively on graphs in order to handle scene graphs in an end-to-end way. To do this, we employ a network of graph convolution layers called a graph convolution network. A traditional 2D convolution layer aggregates data from local neighborhoods of the input by taking a spatial grid of feature vectors as input and producing a new spatial grid of vectors as output, where each output vector is a function of the corresponding input vector's local neighborhood. By using weight sharing across all input neighborhoods, a single convolution layer can function on inputs of any shape.

Similar operations are carried out by our graph convolution layer, which computes new vectors of dimension for each node and edge given an input graph with vectors of dimension for each node and edge. Each graph convolution layer propagates information along the graph's edges because output vectors are a function of the area around the corresponding inputs. A graph convolution layer can operate on graphs of any shape since it applies the same function to all of the graph's edges.

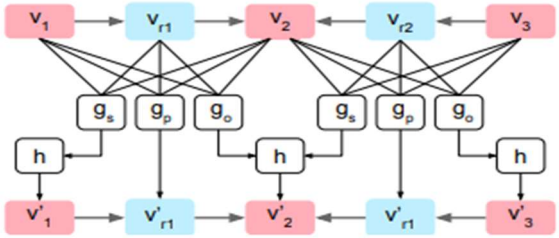


Figure 4: A single graph convolution layer is shown in a computational graph

Scene layout: An embedding vector for each object is produced by processing the input scene graph through a number of graph convolution layers, which compiles data on all the objects and connections in the network.

We must transition from the graph domain to the picture domain in order to create an image. In order to achieve this, we compute a scene layout using the object embedding vectors, which provides the coarse 2D structure of the image to be generated. To do this, we utilize an object layout network to predict a segmentation mask and bounding box for each object.
Generative Adversarial Network: The output image is generated using the scene layout by a GAN.
The GAN network employs a generator and discriminator that are trained in an adversarial manner, to generate realistic images. In order to enable generation to go from coarse to fine, a GAN consists of a sequence of convolutional refinement modules, with spatial resolution doubling between modules.
Both the scene layout (downsampled to the module's input resolution) and the output from the module before it are inputs into each module. The output is upsampled using nearest-neighbor interpolation before being delivered to the following module. These inputs are concatenated channel-wise and passed to a pair of 3 3 convolution layers.
Discriminators: By training the image production network antagonistically against two discriminator networks, we produce realistic output images.
By maximizing the objective, a discriminator tries to classify its input as authentic or bogus. The patch-based image discriminator, which is implemented as a fully convolutional network and functions similarly to the discriminator, determines if the overall appearance of generated images is genuine by classifying a group of regularly spaced, overlapping image patches as real or fake.
The object discriminator ensures that each object in the image appears realistic; its input are the pixels of an object, cropped and rescaled to a fixed size using bilinear interpolation. In addition to classifying each object as real or fake, it also ensures that each object is recognizable using an auxiliary classifier which predicts the object's category; both attempt to maximize the probability that correctly classifies objects.
We train the discriminators and the generation network together. In order to reduce the weighted sum of six losses, the generation network is taught to:

Box loss: punishing the L1 gap between anticipated and ground-truth boxes

Pixel loss: punishing the L1 difference between images created using ground truth

Image adversarial loss: promoting the realism of created image patches

Object adversarial loss: encouraging each created item to appear realistic

Auxiliary classifier loss: ensuring that each created object can be categorized by Object Discriminator.

Dataset Description

- The Visual Genome is a large-scale dataset for visual understanding that includes images annotated with dense descriptions of their content, objects, attributes, and relationships.
- The dataset contains over 100,000 images, each annotated with an average of 21 objects, 18 attributes, and 12 relationships per image.
- Images have region descriptions for specific parts, which are converted into region graphs that include object attributes and relationships. These region graphs are combined to form a scene graph with all objects grounded to the image.

Input Data: Input will be in the form of a Scene graph.
Output Data: The output will be the image generated by the GAN

Evaluation: Based on our preliminary study, we can see the commonly used metrics for Image Generation experiments are Inception Distance and FID Distance.

Base Statistics:
- Total images: 108,077
- Total Scene Graphs: 108,249

Model Architecture

Layer	Input	Output
GNN	Graph Objects, Relationships	Ox128, Rx128
Box Regression Network	128	4
Mask Regression Network	128	1x16x16
Cascade Refinement Module	128 x 64 x64	3 x 64 x 64
Object Discriminator	3 x 32 x 32	|C|
Image Discriminator	3 x 64 x 64	1 x 8 x 8

Architecture of Graph Convolution Network

Operation	Input	Output
Graph Objects	-	O
Graph Relationships	-	R
Object Embedding	O	O x 128
Relationship Embedding	R	R x 128
Gconv(128 →512 → 128)	O, R	O x 128, R x 128
Gconv(128 →512 →128)	O x 128, R x 128	O x 128, R x 128
Gconv(128 →512 → 128)	O x 128, R x 128	O x 128, R x 128
Gconv(128 →512 → 128)	O x 128, R x 128	O x 128, R x 128
Gconv(128 →512 → 128)	O x 128, R x 128	O x 128, R x 128

Architecture of Box Regression Network

Operation	Input	Output
Object embedding vector	-	128
Linear(128 → 512)	128	512
ReLU	512	512
Linear(512 →4)	512	4

Architecture of Mask Regression Network

Operation	Input	Output
Object embedding vector	-	128
Reshape	128	128 x 1 x 1
Up sample	128 x 1 x 1	128 x 2 x 2
Batch Normalization	128 x 2 x 2	128 x 2 x 2
Conv(3 x 3, 128 → 128)	128 x 2 x 2	128 x 2 x 2
ReLU	128 x 2 x 2	128 x 2 x 2
Up sample	128 x 2 x 2	128 x 4 x 4
Batch Normalization	128 x 4 x 4	128 x 4 x 4
Conv(3 x 3, 128 → 128)	128 x 4 x 4	128 x 4 x 4
ReLU	128 x 4 x 4	128 x 4 x 4
Up sample	128 x 4 x 4	128 x 8 x 8
Batch Normalization	128 x 8 x 8	128 x 8 x 8
Conv(3 x 3, 128 → 128)	128 x 8 x 8	128 x 8 x 8
ReLU	128 x 8 x 8	128 x 8 x 8
Up sample	128 x 8 x 8	128 x 16 x 16
Batch Normalization	128 x 16 x 16	128 x 16 x 16
Conv(3 x 3, 128 → 128)	128 x 16 x 16	128 x 16 x 16
ReLU	128 x 16 x 16	128 x 16 x 16
Conv (1 x 1, 128 →1)	128 x 16 x 16	1 x 16 x 16
Sigmoid	1 x 16 x 16	1 x 16 x 16

Architecture of Cascade Refinement Network

Operation	Input	Output
Scene Layout	-	128 x 64 x 64
Gaussian Noise	-	32 x 2 x 2
CRN(2 x 2, 32 → 1024)	128 x 64 x 64, 32 x 2 x 2	1024 x 4 x 4
CRN(4 x 4, 1024 → 512)	128 x 64 x 64, 1024 x 4 x 4	512 x 8 x 8
CRN(8 x 4, 512 →256)	128 x 64 x 64, 512 x 8 x 8	256 x 16 x 16
CRN(16 x 16, 256 → 128)	128 x 64 x 64, 256 x 16 x 16	128 x 32 x 32
CRN(32 x 32, 128 →64)	128 x 64 x 64, 128 x 32 x 32	64 x 64 x 64
Conv(3 x 3, 64 → 64)	64 x 64 x 64	64 x 64 x 64
LeakyReLU	64 x 64 x 64	64 x 64 x 64
Conv(1 x 1, 64 →3)	64 x 64 x 64	3 x 64 x 64

Architecture of Object Discriminator

Operation	Input	Output
Object crop	-	3 x 32 x 32
Conv(4 x 4, 3 → 64, s2)	3 x 32 x 32	64 x 16 x 16
Batch Normalization	64 x 16 x 16	64 x 16 x 16
Leaky ReLU	64 x 16 x 16	64 x 16 x 16
Conv(4 x 4, 64 → 128, s2)	64 x 16 x 16	128 x 8 x 8
Batch Normalization	128 x 8 x 8	128 x 32 x 32
Leaky ReLU	128 x 32 x 32	128 x 32 x 32
Conv(4 x 4, 128 →256, s2)	128 x 32 x 32	256 x 4 x 4
Global Average Pooling	256 x 4 x 4	256
Linear(256 → 1024)	256	1024
Linear(1024 →1)	1024	1
Linear(1024 → |C|)	1024	|C|

Architecture of Image Discriminator

Operation	Input	Output
Image	-	3 x 64 x64
Conv(4 x 4, 3 → 64, s2)	3 x 64 x 64	64 x 32 x 32
Batch Normalization	64 x 32 x 32	64 x 32 x 32
Leaky ReLU	64 x 32 x 32	64 x 32 x 32
Conv(4 x 4, 64 → 128, s2)	64 x 32 x 32	128 x 16 x 16
Batch Normalization	128 x 16 x 16	128 x 16 x 16
Leaky ReLU	128 x 16 x 16	128 x 16 x 16
Conv(4 x 4, 128 →256, s2)	128 x 16 x 16	256 x 8 x 8
Conv(1 x 1, 256 →1)	256 x 8 x 8	1 x 8 x 8

Experiments & Results:

- For training, 10% of the dataset was used for validation, 10% for testing, and 80% for training. We selected object and relationship categories with a minimum incidence of 2000 and 500 in the training set, resulting in 178 object and 45 relationship types.
- We used increasingly complex scene graphs to generate more intricate images, and the shift in the boat's position with the addition of the "boat on grass" relationship demonstrated the impact of scene graph relationships on object positions.
- The number of iterations that were run was around ~ 175000.

The following graphs of the loss functions used were observed after training:

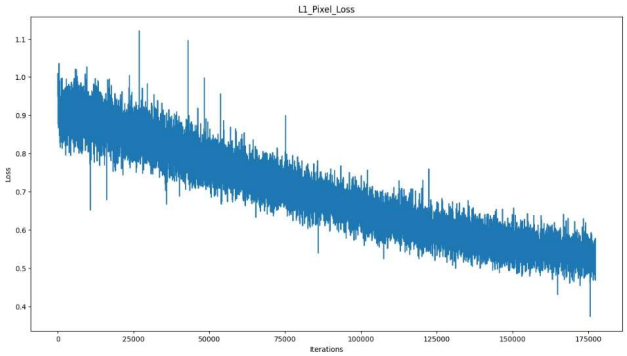


Figure 4a. L1 Pixel Loss

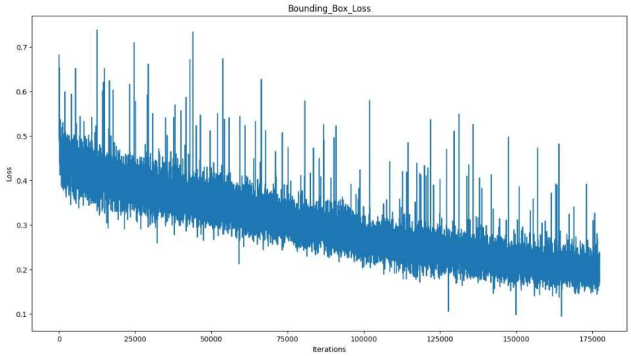


Figure 4b. Bounding Box Regression Loss

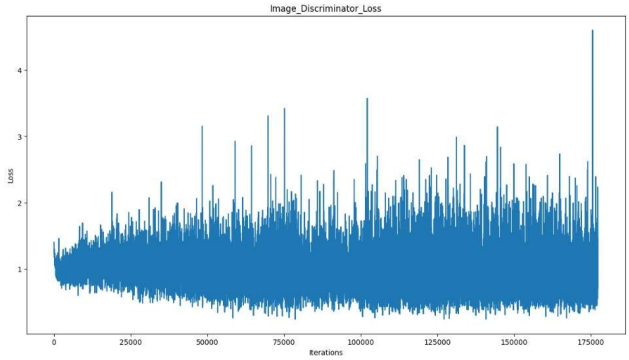


Figure 4c. Image Discriminator Loss

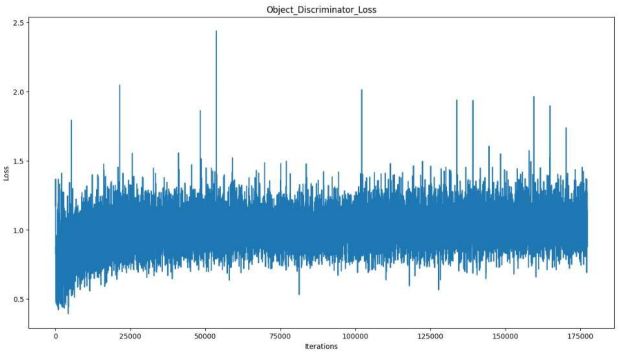
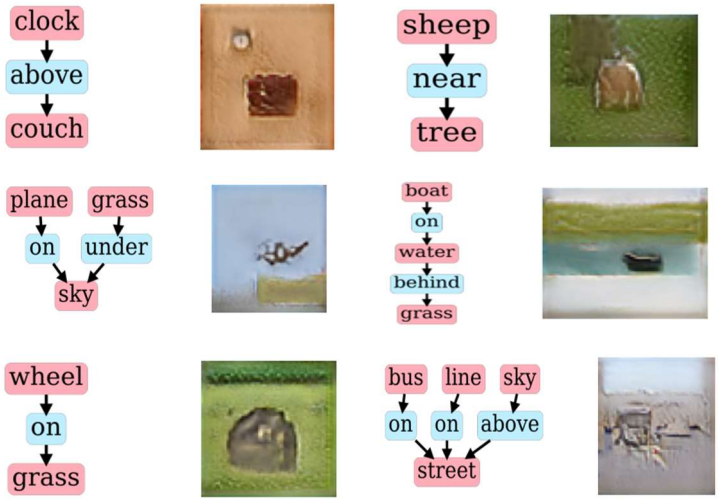


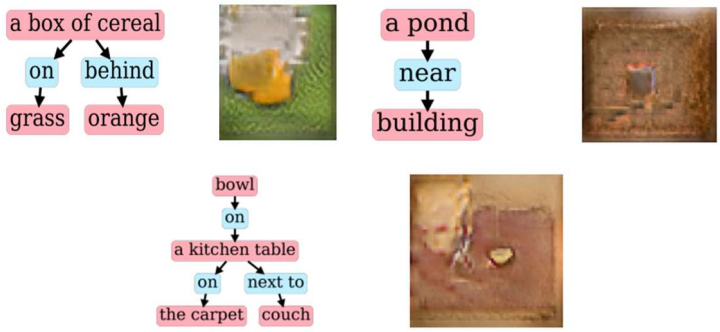
Figure 4d. Object Discriminator Loss

- The model was run on Google Cloud instance which has NVIDIA T4 GPU for around ~ 175000 iterations, and it took approximately 2 and a half days to complete.
- Below is the performance of the model based on the scene graph given as an input. The scene graph, here, has been visually converted for better perception as the input was given as .json file to the model.

Instances where the model performed well



Instances where the model did not perform well



Conclusion

In this study, we present a complete method for creating images from scene graphs. Our method generates images from structured scene graphs rather than unstructured text, enabling it to explicitly reason about objects and relationships and produce complex images with a variety of recognizable objects, in contrast to leading methods that generate images from word descriptions.

Future Work

We would further like to evaluate the generated images using a metric like Inception score which weren't able to do because of time constraint.

Graph Neural Network here treats all kinds of edges the same and to identify the different edges, a separate MLP is used. However, there are more recent graph neural networks that are capable of handling these heterogeneous graphs and we would like to use this with our model.

References

[1] Johnson, Justin, Agrim Gupta, and Li Fei-Fei. "Image generation from scene graphs." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[2] Chang, Xiaojun, et al. "A comprehensive survey of scene graphs: Generation and application." IEEE Transactions on Pattern Analysis and Machine Intelligence 45.1 (2021): 1-26.

[3] Dhamo, Helisa, et al. "Semantic image manipulation using scene graphs." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[4] Yang, Ling, et al. "Diffusion-Based Scene Graph to Image Generation with Masked Contrastive PreTraining." arXiv preprint arXiv:2211.11138 (2022).

[5] Li, Yikang, et al. "Pastegan: A semi-parametric method to generate image from scene graph." Advances in Neural Information Processing Systems 32 (2019).

[6] github.com/google/sg2im

Link to the source code: https://indiana-my.sharepoint.com/personal/jsridhar_iu_edu/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fjsridhar%5Fiu%5Fedu%2FDocuments%2FSpring%2023%2FCV%2FProject%2Fgraphmuse&ga=1