Image Generation from Scene Graphs with Contextual Knowledge

Alex Foo
School of Computing
National University of Singapore
alexfoo@comp.nus.edu.sg

Abstract

Within the past decade, the computer vision community has seen astounding progress for several fundamental tasks such as image classification, object detection, instance segmentation and scene generation fueled by advances in Deep Learning (DL). However, a growing observation is that DL lacks the abstract, high-level understanding of images that humans have. In order to guide these systems to learn a more abstract level of understanding of images, an increasing number of research has placed their attention on scene graph generation - a powerful structured representation of a scene. In this work, we propose to combine low-level generated scene graphs with high-level contextual knowledge graphs and apply it on the task of image scene generation. We argue that scene graphs alone are insufficient, and have to be integrated with a more contextual level of understanding of objects and their interactions in order to fabricate a scene in which object relationships are respected. Some work has partially explored using scene graphs to generate images, or combining knowledge graphs and scene graphs to boost scene graph generation, but no work till date has explored combining scene and knowledge graphs for the enhancement of image scene generation. We observe that by pruning away unimportant relationships, we are able to attain significantly faster training speed, and the trained model is able to generate images which better withhold the relationships between objects. Code for this project is publicly published on Github[1], and a brief webpage summarizing the project is hosted as well[2].

Introduction

Within the past decade, the computer vision community has seen astounding progress for several fundamental tasks such as image classification, object detection, instance segmentation and scene generation. This progress has mostly been fueled by advances in Deep Learning (DL), which has greatly enhanced the representation capability of visual systems.

However, a growing observation is that DL lacks the abstract, high-level understanding of images that humans have. This causes sub-optimal performance for vision tasks which require higher level semantic understanding of interactions between objects such as visual question answering, scene understanding and few-shot/zero-shot image classification.

[1]https://github.com/alexfoodw/sg2im
[2]https://alexfoodw.github.io

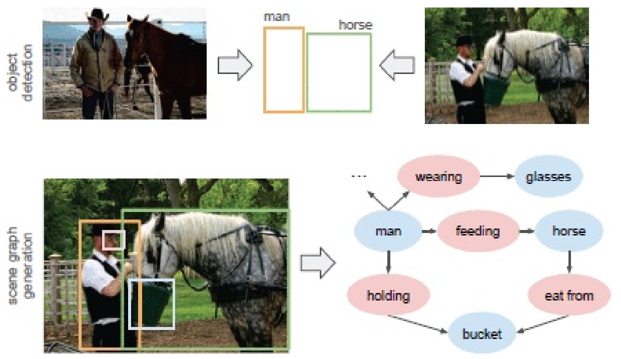

Figure 1: Sample scene graph generated from image taken from Xu et al. (2017). Perfect object detection networks used for scene understanding will still describe the top row images with different semantic meanings as identical. Blue nodes in the graph represent objects while red nodes represent pairwise relation between the objects.

In order to guide these systems to learn a more abstract level of understanding of images, an increasing number of research has placed their attention on scene graph generation (Xu et al. 2017; Gu et al. 2019; Zareian, Karaman, and Chang 2020). Given an input image, a scene graph aims to summarize information about the objects present and the relationships between these objects. This rich representation hence strengthens a visual system's ability to reason about the relational information present in the image, consequently improving the latent representation of the input image.

As illustrated in Figure 1, works that use object detection networks for scene understanding will misunderstand the top row of images as semantically identical - both images contain a man beside a horse. With the generation of a scene graph as an intermediate step, visual systems will be able to generate more accurate descriptions of the interactions between objects - a man is feeding the horse with a bucket.

In this work, we propose to further combine low-level generated scene graphs with high-level contextual knowledge graphs and apply it on the task of image scene generation. We argue that annotated scene graphs contain noisy and unimportant information which may confuse the training of generative models, and hence have to be integrated

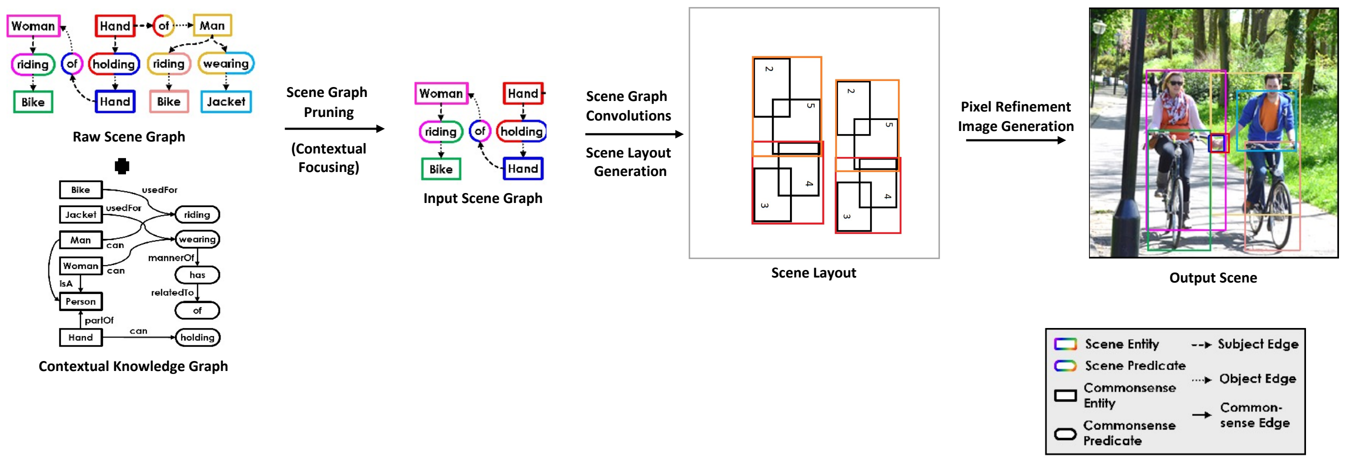

Figure 2: Illustration of proposed methodology. Given an input raw scene graph and the contextual knowledge graph representing all object categories and relationships, we prune the scene graph by removing unimportant and noisy relationships. We then perform reasoning on the pruned scene graph through graph convolutions. After reasoning, we generate the scene layout using embedded bounding box regressions to produce the scene layout. Finally, given the scene layout, we perform pixel refinement using a CRN to complete the output scene. Parts of image taken from (Zareian, Karaman, and Chang 2020).

with a more general or contextual level of understanding of objects and their key interactions in order to optimize training and accurately fabricate a scene.

Some work has partially explored this area - Johnson, Gupta, and Fei-Fei (2018) have explored using scene graphs with graph convolutions to generate images, while Gu et al. (2019) has attempted a auxiliary image reconstruction task using knowledge graphs and scene graphs to boost scene graph generation, but no work has explored combining scene and knowledge graphs for the enhancement of image scene generation.

Methodology

We break down the generation process into 4 main steps, based on the combination of works from (Johnson, Gupta, and Fei-Fei 2018) and (Gu et al. 2019):

1. Scene Graph Pruning, where we use the Commonsense KG to selectively remove noisy and unimportant relationships (and respective objects) from input scene graphs through a ranking algorithm inspired by (Gu et al. 2019).
2. Reasoning on scene graphs by Scene Graph Convolutions adapted from (Johnson, Gupta, and Fei-Fei 2018).
3. Scene layout generation performed by a Box Regression Network on final object embeddings.
4. Pixel refinement by a Cascaded Refinement Network to fill up the final pixels given the scene layout.

Figure 2 summarizes our proposed approach.

Scene Graphs and Knowledge Graph

The input of our model is a *scene graph*, which describes objects and relationships between objects. Given a set of object categories $\mathcal{C}$ and a set of relationship categories $\mathcal{R}$, an input scene graph is a tuple (O, E) where $O = \{o_1, \ldots, o_n\}$ is a set of objects with each $o_i \in \mathcal{C}$ and $E \subseteq O \times R \times O$ is a set of directed edges of the form (o_i, r, o_j) where $o_i, o_j \in O$ and $r \in \mathcal{R}$.

We enhance the contextual and commonsense learning of each scene graph by utilising an aggregated *knowledge graph* $(\mathcal{C}, \mathcal{E}, \mathcal{W})$, which consolidates the set of object categories $\mathcal{C}$ and relationship categories $\mathcal{R}$. The relationships categories link up the object categories, forming a commonsense understanding of how categories interact with each other: $\mathcal{E} \subseteq \mathcal{C} \times \mathcal{R} \times \mathcal{C}$. Each directed edge $(\mathcal{C}_i, \mathcal{R}, \mathcal{C}_j)$ is weighted with respect to its number of occurrences in the database. A scene graph is thus a specific subset of the knowledge graph.

Scene Graph Pruning

Since manually annotated scene graphs are image specific, they contain many unimportant relationships which do not contribute to the overall semantic understanding of an object and its relations. This results in confusion during a model's learning phase, and unnecessarily long training times of large image databases.

Concretely, consider 'sheep' as the object that our model is learning to represent and generate. Analysing the Visual Genome (VG) dataset (Krishna et al. 2017) - a knowledge base with manually annotated object relations for each image - we see that the 'sheep' object has a total of 2193 relations. Of which, only a small minority represents semantically rich information which generalizes across many images: 'sheep in field' has 1059 instances, 'sheep has leg' has 332 instances, and 'sheep eating grass' has 318 instances.

In contrast, 2041 relations have less than 10 instances for the model to learn from, of which, many are too specific to enhance the model's understanding of a 'sheep': '70 drawn on sheep', 'suv next to sheep', 'teddy bear tied to sheep', 'sheep wearing necklace', where all of which have only 1 instance.

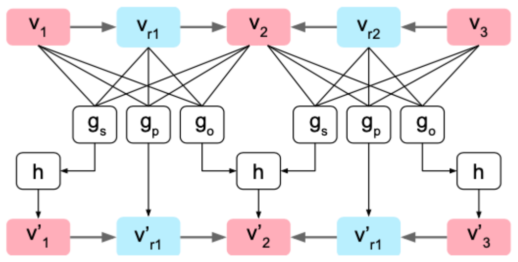

Figure 3: Computational graph illustrating a single graph convolution layer. Here we have three objects $\tilde{o}_1, \tilde{o}_2, \tilde{o}_3$ and two edges $(\tilde{o}_1, \tilde{r}_1, \tilde{o}_2)$ and $(\tilde{o}_3, \tilde{r}_2, \tilde{o}_2)$. Along each edge, the three input vectors are passed to functions g_s, g_p, g_o. g_p directly computes the output vector for the relation, while g_s and g_o compute candidate vectors which are fed to a symmetric pooling function h to compute vectors for objects. Figure taken from (Johnson, Gupta, and Fei-Fei 2018).

Hence, in order to enhance model learning and optimize training time of large datasets, we propose a Scene Graph Pruning algorithm which filters out such minimally contributing relations and their respective objects.

Specifically, given a knowledge graph $(\mathcal{C}, \mathcal{E}, \mathcal{W})$, we form a refined knowledge graph $(\tilde{\mathcal{C}}, \tilde{\mathcal{E}}, \mathcal{W})$ by only keeping the top k relations for each object category $\mathcal{C}$, in order of decreasing weights. From this refined knowledge graph, we can extract the key partner object categories associated to each object category. k is thus a hyperparameter, which we set to $k = 10$.

Now, for every input scene graph (O, E), we prune any object, along with the relations that contain it, if the object category of that object is not in our refined knowledge graph. Formally, a pruned version of each input scene graph $(\tilde{O}, \tilde{E})$ is created such that $\tilde{O} \subseteq \tilde{\mathcal{C}}$ and $\tilde{E} \subseteq \tilde{\mathcal{E}}$.

Scene Graph Convolutions

In order to process each scene graph in an end-to-end manner, we adapt the modified graph convolution network as proposed by (Johnson, Gupta, and Fei-Fei 2018).

This step takes in as input the pruned scene graph $(\tilde{O}, \tilde{E})$, and performs reasoning on each object with respect to their edges. Each object and edge is first embedded into vectors of dimension D_{in}, passed through our Scene Graph Convolutions layer, and outputs new vectors of dimension D_{out}. Since this is done in an permutation invariant manner by message passing of objects to its neighbors, what we have is a reasoned output embedding which represents a function of each object and its neighboring relations.

Concretely, given a pruned scene graph input $(\tilde{O}, \tilde{E})$, we first embed the graph's objects $\tilde{o}_i, \tilde{o}_j \in \tilde{O}$ and edges $(\tilde{o}_i, \tilde{r}, \tilde{o}_j)$ into input vectors $v_i, v_r \in \mathbb{R}^{D_{in}}$. For each of these input vectors, we then compute output vectors $v'_i, v'_r \in \mathbb{R}^{D_{out}}$ using three functions g_s, g_p, g_o which takes in a triple (v_i, v_r, v_j) and outputs a processed vector for inputs subject $\tilde{o}_i$, predicate $\tilde{r}$, and object $\tilde{o}_j$ respectively.

For relations, we can simply set $v'_r = g_p(v_i, v_r, v_j)$ to obtain the appropriate output. Objects, however, are slightly more complex since an object can participate in multiple relationships. Hence, an object should be defined in terms of all its neighboring objects to which it is connected to, and their respective relationships. To this end, for each edge starting at $\tilde{o}_i$, g_s is used to compute a candidate vector. The collection of all such candidate vectors forms the subject *candidate set* for that object - V_i^s. Similarly, g_o is used for all edges ending at $\tilde{o}_i$, to obtain object *candidate set* V_i^o. Formally:

$$\begin{aligned} V_i^s &= \{g_s(v_i, v_r, v_j) : (\tilde{o}_i, \tilde{r}, \tilde{o}_j \in \tilde{E}\} \\ V_i^o &= \{g_o(v_j, v_r, v_i) : (\tilde{o}_j, \tilde{r}, \tilde{o}_i \in \tilde{E}\}. \end{aligned} \tag{1}$$

Finally, the output vector for v'_i for object $\tilde{o}_i$ is then computed by pooling both candidate sets $v'_i = h(V_i^s \cup V_i^o)$ using a symmetric function h. Figure 3 illustrates the computational graph for a single layer.

In our implementation, functions g_s, g_p, g_o are implemented using a single network which concatenates the three input vectors and feeds them into a multilayer perceptron (MLP). We set h to be a average operation across all candidate vectors, and its result is fed into another MLP.

Scene Layout

Once we have performed reasoning of the input scene graph, we then begin to generate an image by computing a scene layout. A scene layout is computed by predicting a bounding box for each object using a *box regression network*.

The box regression network receives an embedding vector v_i of shape D for object $\tilde{o}_i$ and predicts a bounding box $\hat{b}_i = (x_0, y_0, x_1, y_1)$ and concurrently generates an inflated object embedding of shape $M \times M \times D$, where the original embedding is duplicated $M \times M$ times. The inflated object embedding is then warped into the size and shape of the bounding box by means of bilinear interpolation (Jaderberg et al. 2015), to give an object layout. The sum of all object layouts gives the scene layout.

Our implementation uses an MLP as the box regression network.

Pixel Refinement

Given the scene layout, we synthesize an image according to the object positions given in the layout. We adopt a Cascaded Refinement Network (CRN) (Chen and Koltun 2017) for this task. A CRN consists of a series of convolutional refinement modules, with spatial resolution doubling between modules. This allows generation to proceed in a coarse-to-fine manner.

Each module receives as input both the downsampled scene layout and the output from the previous module. These inputs are concatenated channelwise and passed to a pair of 3×3 convolution layers, where the output is then upsampled using nearest-neighbor interpolation before being passed to the next module. For additional augmentation, the first module takes Gaussian noise $z \sim p_z$ as input.

Discriminators

Finally, following (Johnson, Gupta, and Fei-Fei 2018), realistic output images are generated by training our network f adversarially against a pair of discriminator networks D_{img} and D_{obj}, where the patch-based image discriminator D_{img} ensures that the overall appearance of generated images is realistic, while the object discriminator D_{obj} ensures that each object in the image appears realistic.

Each discriminator D attempts to classify its input X as real or fake by maximizing the objective (Goodfellow et al. 2014)

$$\mathcal{L}_{GAN} = \mathbb{E}_{x \sim p_{real}} \log D(x) + \mathbb{E}_{x \sim p_{fake}} \log(1 - D(x)). \tag{2}$$

Training

Training is performed in an end-to-end manner, by jointly combining the loss functions of our generation network f and discriminators D_{obj} and D_{img}. This is implemented by a weighted sum of:

- Box regression loss $\mathcal{L}_{box} = \sum_{i=1}^{n} ||b_i - \hat{b}_i||_1$ which penalizes the L_1 difference between ground-truth and predicted boxes
- Pixel loss $\mathcal{L}_{pix} = ||I - \hat{I}||_1$ penalizing the L_1 difference between ground-truth and generated images
- Image adversarial loss $\mathcal{L}_{GAN}^{img}$ from D_{img} encouraging realistic generated image patches
- Object adversarial loss $\mathcal{L}_{GAN}^{obj}$ from D_{obj} encouraging realistic generated objects.

All scene graphs are augmented with a special *__image__* object in which all objects are connected to. This ensures that all scene graphs are connected.

Experiments

We train our model to generate 64×64 images on a restrained version of the Visual Genome (Krishna et al. 2017) dataset. In our experiments, we aim to show that our method is able to produce realistic images that respect the relations between objects, with faster training time and despite the restraining of total object categories. We train our model for a total of 30k iterations, with the model switching to evaluation mode after 3k iterations.

Dataset

Visual Genome. We experiment on Visual Genome (VG) (Krishna et al. 2017) version 1.4, where the original scene graph dataset contains 108,077 images with 179 object categories after processing, and an average of 38 objects and 22 relationships per image. Processing involves the ignoring of small objects ($<$ 32 pixels in either height or width), only using images with between 3 and 30 objects and at least one relationship.

Due to the shorter research cycle for this project, we limit the number of object categories (and hence their respective relationships) to a subset of key categories which have between 3k-3.5k object training instances. This gives us 18 key object categories for our experiments which we list in detail in the Appendix.

We process the entire VG dataset into a knowledge graph containing all 179 object categories and 45 relationship types, and apply the proposed Scene Graph Pruning method to inflate the object vocabulary. This is done by including only the most important partner objects of our 18 key object categories, giving us a total of 89 object categories and 23 relationship categories. We remove any image that does not include any of our 18 key object categories, giving us a total image count of 18,168 which we split the data into 80% train, 10% val, 10% test.

Table 1: Quantitative analysis of proposed Scene Graph Pruning algorithm, with respect to number of relations per object instance and training time per iteration, setting $k = 10$.

Method	Relations / Instance	Time / Iter (s)
Raw	0.25	1.65
Refined (ours)	**0.11**	**1.52**

Qualitative Results

In Figure 4, we list 12 sample generated images, where each respective input scene graph and the corresponding generated scene layout is shown as well. The input scene graphs are taken from our test set after training the model. We observe that although the images are not as sharp as input training images, the model manages to generate realistic looking objects which respect the relationships specified in each scene graph.

In particular, zooming into the bottom right image with a house on a hill (second set of images from bottom right of figure), we see that the model is impressively able to generate both large objects, like the sky and grass, and small objects like the building and hill.

Scene Graph Pruning Analysis

Let us now analyze the effect of our proposed Scene Graph Pruning algorithm, which removes unimportant relationships (and respective objects) from each training scene graph by referring to a weighted knowledge graph of all object and relationship categories.

We propose to study the effectiveness of our pruning strategy by two metrics - Relations per Instance and Time per Iteration. The comparison of our pruning strategy versus a raw VG database used in our baseline model from (Johnson, Gupta, and Fei-Fei 2018) is illustrated in Table 1.

We firstly observe that, the lower the number of relations per object instance, the better we our model will be able to learn the interactions between object instances. Intuitively, this is true because the lower this ratio, the less confused our model will be since it will be able to learn about each object from a smaller and less noisy pool of focused relationships which define it. Hence, we observe that our pruning strategy is able attain a more than 50% decrease (0.25 to 0.11) in the number of relations per instance by keeping the top $k = 10$

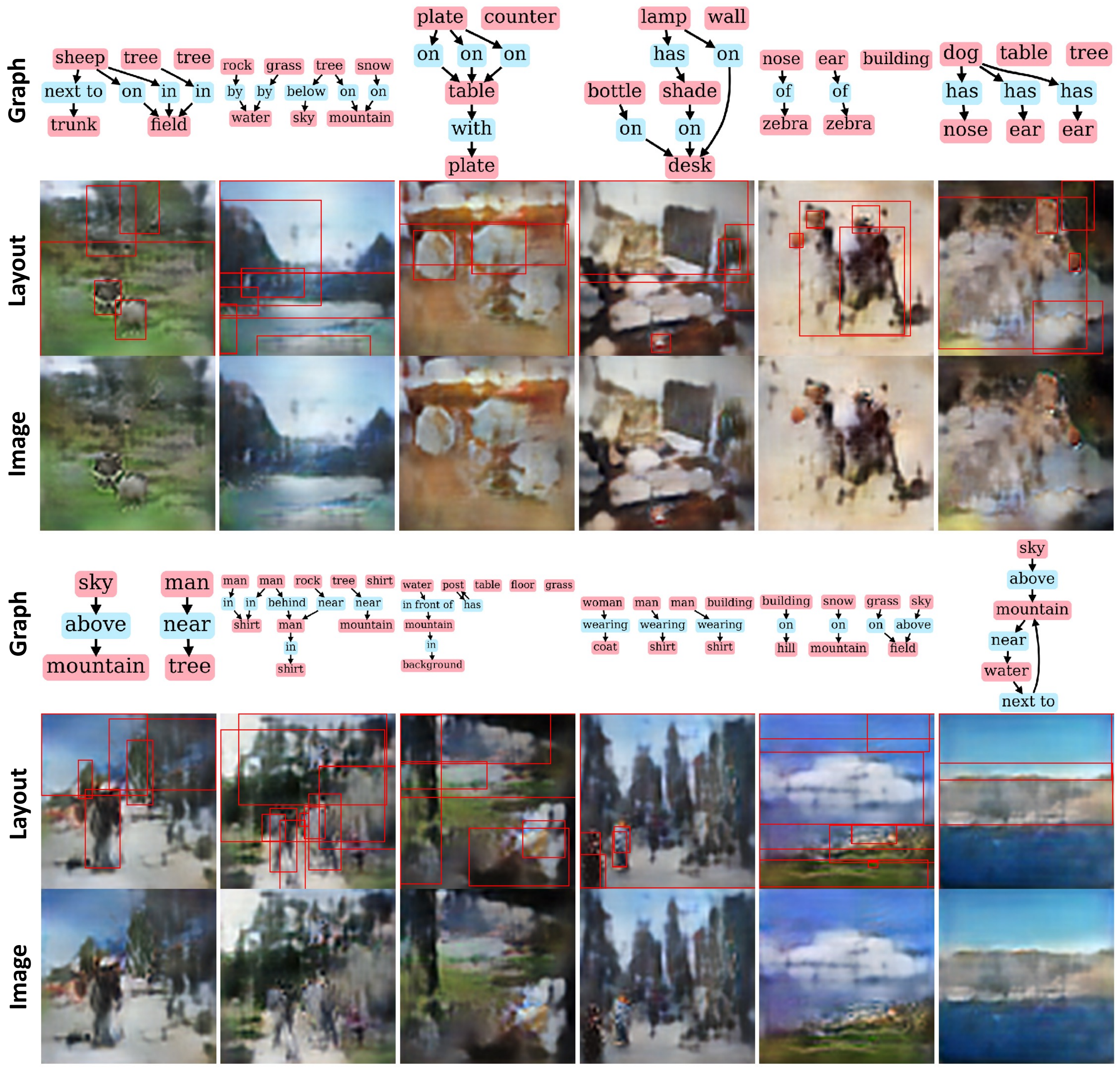
Graph
Layout
Image
sheep tree tree next to on in in trunk field
rock grass tree snow by by below on on water sky mountain
plate counter on on on table with plate
lamp wall has on bottle shade on on desk
nose ear building of of zebra zebra
dog table tree has has has nose ear ear
sky above mountain
man near tree
man man rock tree shirt in in behind near near shirt man mountain in shirt
water post table floor grass in front of has mountain in background
woman man man building wearing wearing wearing coat shirt shirt
building snow grass sky on on on above hill mountain field
sky above mountain near water next to

Figure 4: Qualitative results of our trained generative model, with samples from our test set. Here we have two sets of 6 input scene graphs, where each corresponding scene layout and generated image is shown. We observe that our model is able to compute realistic scene layouts, and consequently realistic generated images.

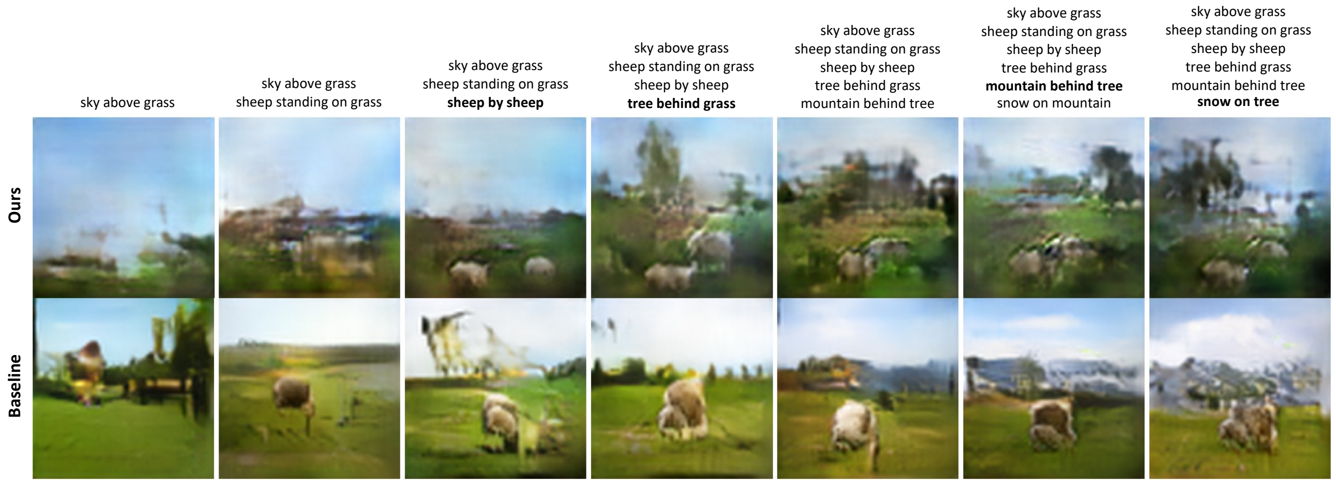

Figure 5: Generated images by our method and our baseline model, given an incrementally challenging scene graph input. The top row represents the outputs from our method, while the bottom row represents the outputs from our baseline model. From left to right, we add more objects and more relations to the input scene graph. Our method is able to generate images which better respect the relationships of multiple object instances, like *sheep by sheep* and *tree behind grass*.

relationships and pruning away the other unimportant relations with only a small number of object instances.

Next, we observe that by pruning away the mentioned unimportant relationships, we are able to train the model more efficiently. As seen in the table, we can attain an averaged (across 10 iterations) 0.13s decrease in time for every iteration. Since the training of this generative model is in the order of hundreds of thousands of iterations, this amounts to huge time savings for the training of a model which produces realistic outputs. For example, if we follow the recommended 1 million iterations training time in (Johnson, Gupta, and Fei-Fei 2018), this would amount to 36 hours in time savings.

Comparisons with Baseline

Finally, in order to comprehensively analyze the quality of generated images of our method, we compare our trained model with our baseline comparison model taken from (Johnson, Gupta, and Fei-Fei 2018). The key difference between the two models is the proposed scene graph pruning strategy, and the significantly faster training time as discussed in the previous section. We feed an incrementally challenging scene graph input into each model, and compare the respective outputs in Figure 5.

The analysis of this experiment is two-fold - our methods ability to handle increasingly complex inputs, and the comparison of between our method and our baseline model. Firstly, zooming in on the top row, we observe the our model is able to generate realistic images which respect each incrementally challenging scene graph. This can be seen from the images displaying relationships such as *sheep by sheep* and *mountain behind tree* even as the complexity of inputs increases.

Now, when comparing the results generated by our method with our baseline model, we observe that our method is able to better respect the relationships between objects better. This can be seen in the scene graph input with *sheep by sheep* (third set of images from the left), where our method clearly illustrates two sheep beside each other, while we only clearly see one sheep in the image generated by our baseline model.

Further, although our method produces comparatively more blurry outputs for the 3 most challenging inputs (3 rightmost images), zooming in on the images we see that relationships between objects are better preserved by our method than the clearer outputs produced by our baseline. For instance, in relationships such as *tree behind grass* for these 3 images, the trees are more clearly shown to be behind grass by our method, while it is not very obvious where are the tree objects in the images generated by the baseline model. Consequently, one possible explanation behind the comparatively blurry outputs could be due to our model prioritising the preservation of relationships between objects, which naturally causes output pixels of different objects to be less smoothly generated and combined together.

Finally, we note that the baseline model spends a lot of time training on the full VG dataset ($\approx$3 days total training time), while our method removes a large portion of the dataset by focusing on our defined key objects ($\approx$11 hours total training time).

Conclusion

In this project, we combined the use of scene graphs and a contextual knowledge graph for the task of image generation. Since a scene graph is a particular subset of a contextual knowledge graph, leveraging information from the knowledge graph allows us to focus on a portion of key relationships of that each object category is associated with. This in turn allows us a much more efficient training time for model training, and produces a model that generates realistic objects while better adhering to the relationships defined in the input scene graph.

References

Chen, Q.; and Koltun, V. 2017. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, 1511–1520.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* .

Gu, J.; Zhao, H.; Lin, Z.; Li, S.; Cai, J.; and Ling, M. 2019. Scene graph generation with external knowledge and image reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1969–1978.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; and Kavukcuoglu, K. 2015. Spatial transformer networks. *arXiv preprint arXiv:1506.02025* .

Johnson, J.; Gupta, A.; and Fei-Fei, L. 2018. Image Generation from Scene Graphs. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1219–1228. doi:10.1109/CVPR.2018.00133.

Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.-J.; Shamma, D. A.; et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision* 123(1): 32–73.

Xu, D.; Zhu, Y.; Choy, C. B.; and Fei-Fei, L. 2017. Scene Graph Generation by Iterative Message Passing. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3097–3106. doi:10.1109/CVPR.2017.330.

Zareian, A.; Karaman, S.; and Chang, S.-F. 2020. Bridging Knowledge Graphs to Generate Scene Graphs. In *ECCV*.

Appendix

Key Object Categories

Table 2: Key object categories of focus for this project.

Object Category	Instances
Lamp	3469
Cup	3437
Elephant	3413
Cabinet	3411
Coat	3397
Mountain	3394
Giraffe	3378
Sock	3312
Cow	3235
Counter	3185
Hill	3184
Word	3172
Finger	3165
Dog	3159
Wire	3090
Sheep	3020
Zebra	3002