Image Generation from Scene Graphs

Tae Gil Ha

Seoul National University

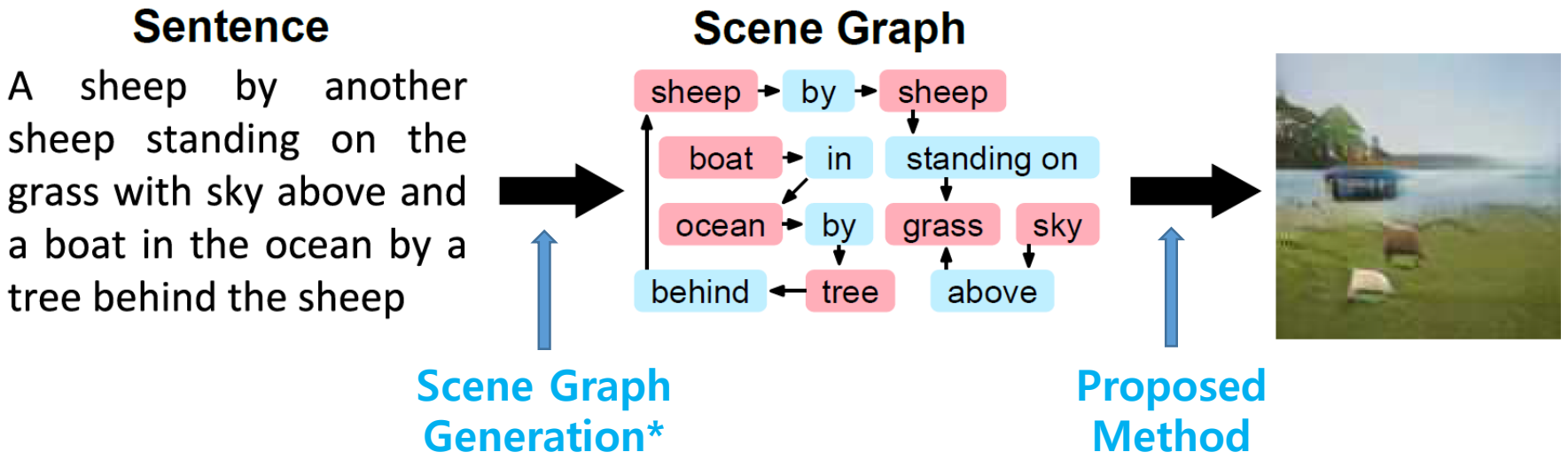
Main Reference : JOHNSON, Justin; GUPTA, Agrim; FEI-FEI, Li. **Image generation from scene graphs**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018. p. 1219-1228.

Introduction

What I cannot create, I do not understand

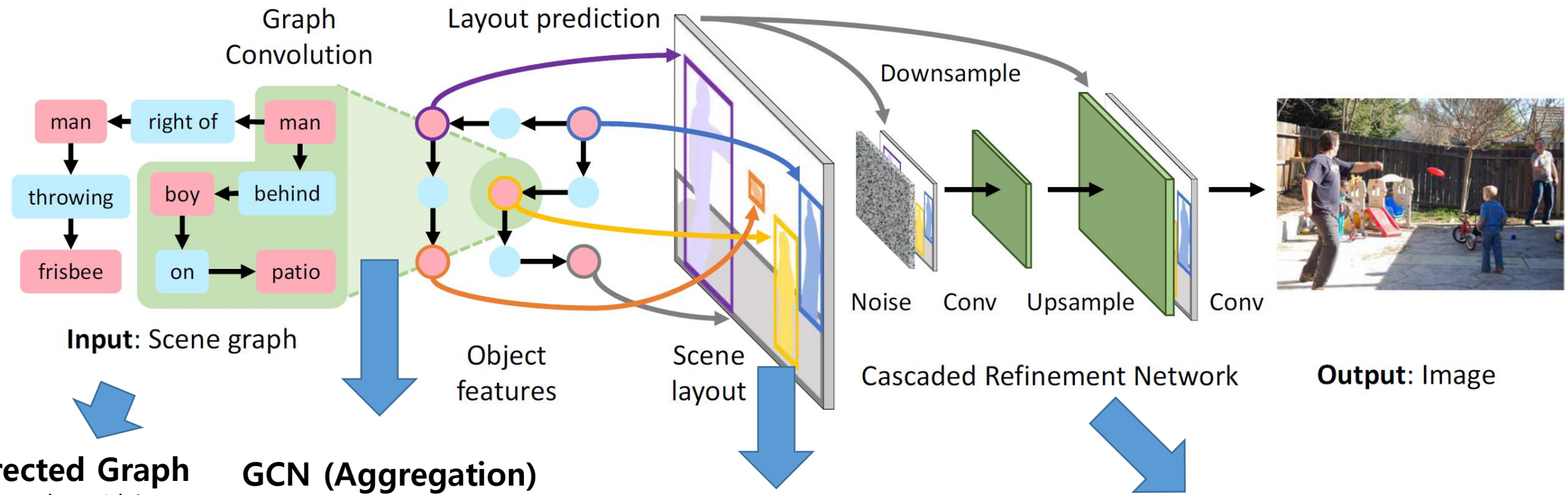
-Richard Feynman

Introduction



*Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In ICCV, 2017

Overview



Directed Graph

- Node : Object
- Edge : Relationship

GCN (Aggregation)

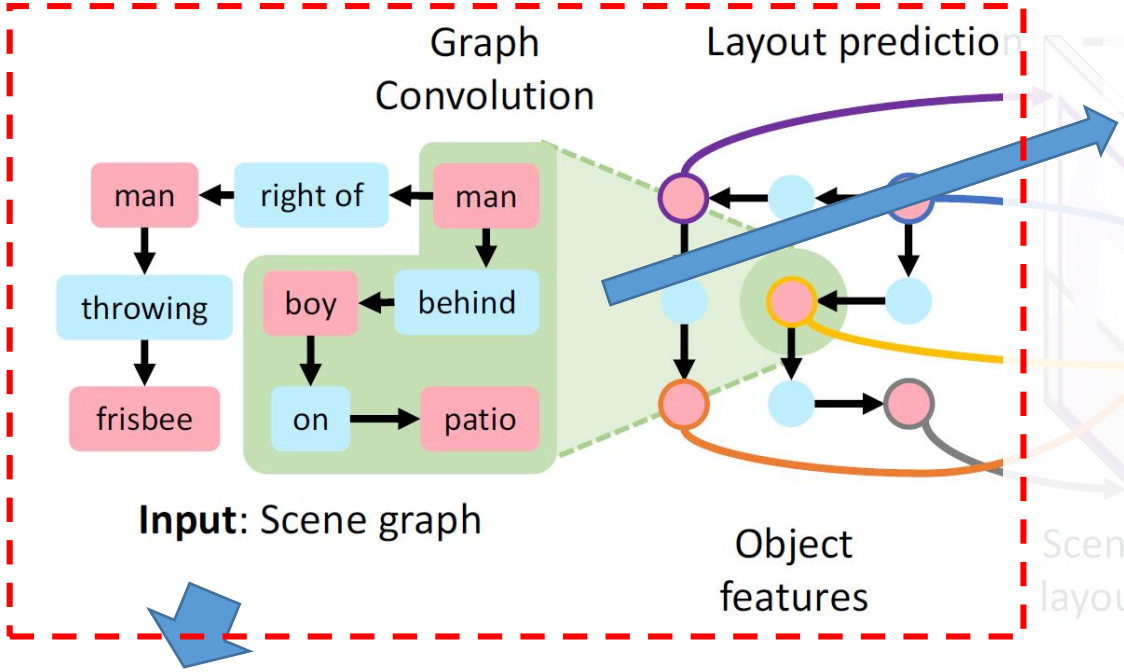
Layout Prediction

- Location
- Shape

Generating Image

- Layout + sampled Noise(z)
- Train : adversarially (D_{obj}, D_{img})

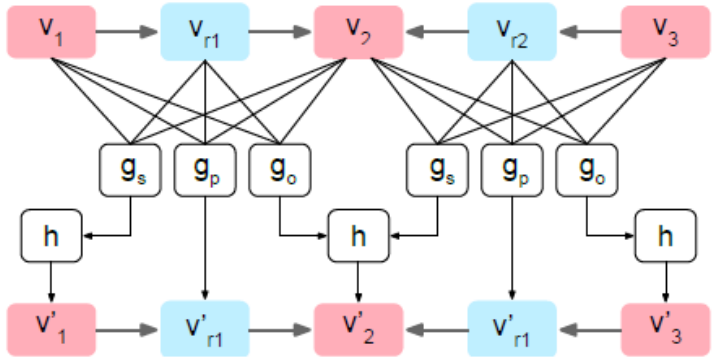
Scene Graph → Features



Directed Graph
- Node : Object
- Edge : Relationship

Scene Graph = $(\boldsymbol{O}, \boldsymbol{E})$
where
- O = element from object category
- $E \sqsubseteq O \times R \times O$, where R is relation

GCN (Aggregation)

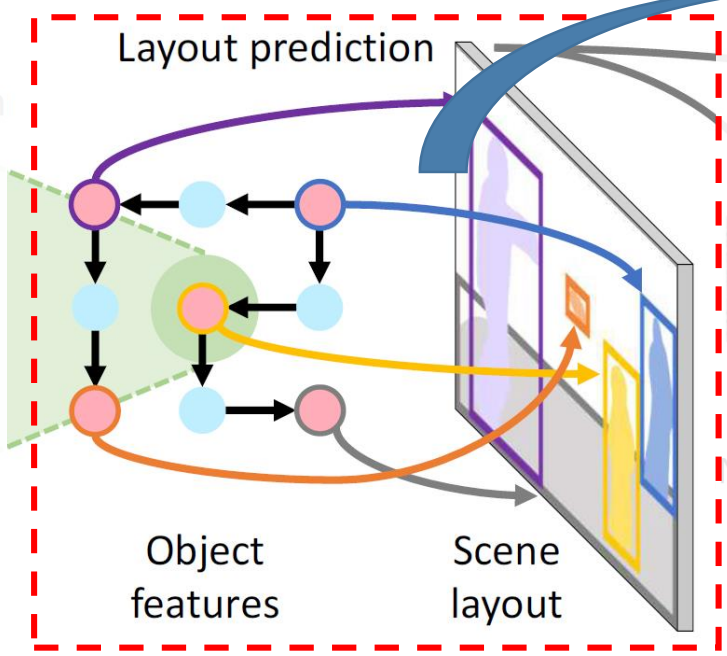


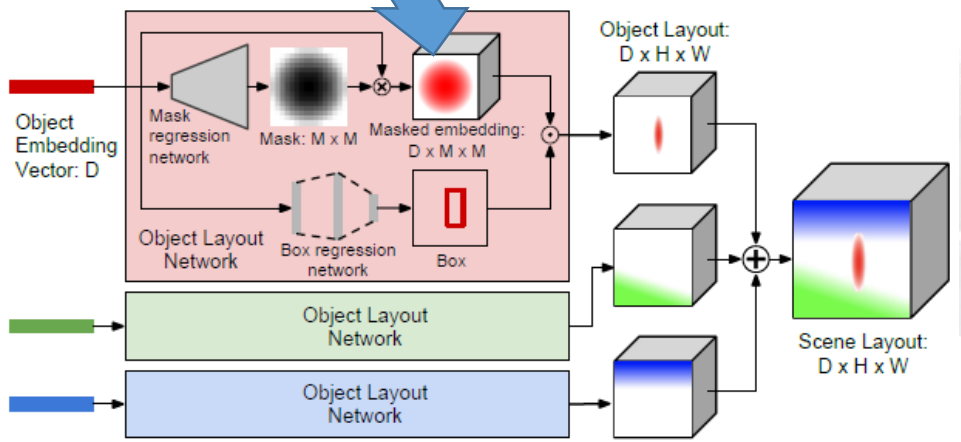
- v_1, v_2, v_3 : object vector(node)
- v_{r1}, v_{r2} : relation vector(edge)
- g_s, g_p, g_o : MLP aggregates information
- h : function that matches output vector dimension
- $v_1', v_2', v_3', v_{r1}', v_{r2}'$: Output vectors

✓ relation vector : $v_r = g_p(v_i, v_r, v_j)$
✓ **subject** vectors $\boldsymbol{V_i^s} = \{g_s(v_i, v_r, v_j);\ (\boldsymbol{v_i}, v_r, \boldsymbol{v_j}) \in E\}$
✓ **object** vectors $\boldsymbol{V_i^o} = \{g_o(v_j, v_r, v_i);\ (\boldsymbol{v_j}, v_r, \boldsymbol{v_i}) \in E\}$
✓ $\Rightarrow \boldsymbol{v_i'} = \boldsymbol{h}(\boldsymbol{V_i^o} \cup \boldsymbol{V_i^s})$

Features → Layout

Layout Prediction
- Segmentation Mask
- Bounding Box





Input
Object Feature $v_i \in R^D$

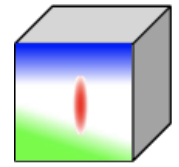
Mask Regression Network
- Transpose Conv Net with last sigmoid
- Soft binary $M \times M$ mask

→ elementwise multiplication with input vector $v_i \in R^D$

Box Regression Network
- Use GT at training

Object i's layout

Σ



Layout → Fine Image

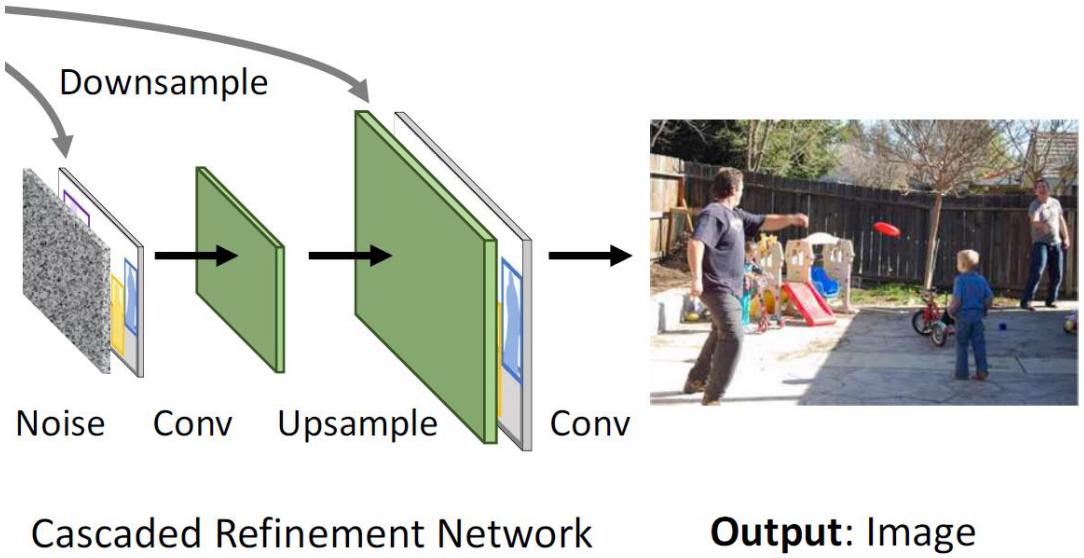
Cascaded Refinement Network

- resolution doubling (nearest neighbor interpolation)
- Previous output + layout prediction → output
- First module take Gaussian noise z

Discriminator

- Discriminator D_{img}, D_{obj}
- D_{obj} determines object is real or fake
- D_{obj} also use auxiliary classifier*, make the object recognizable

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

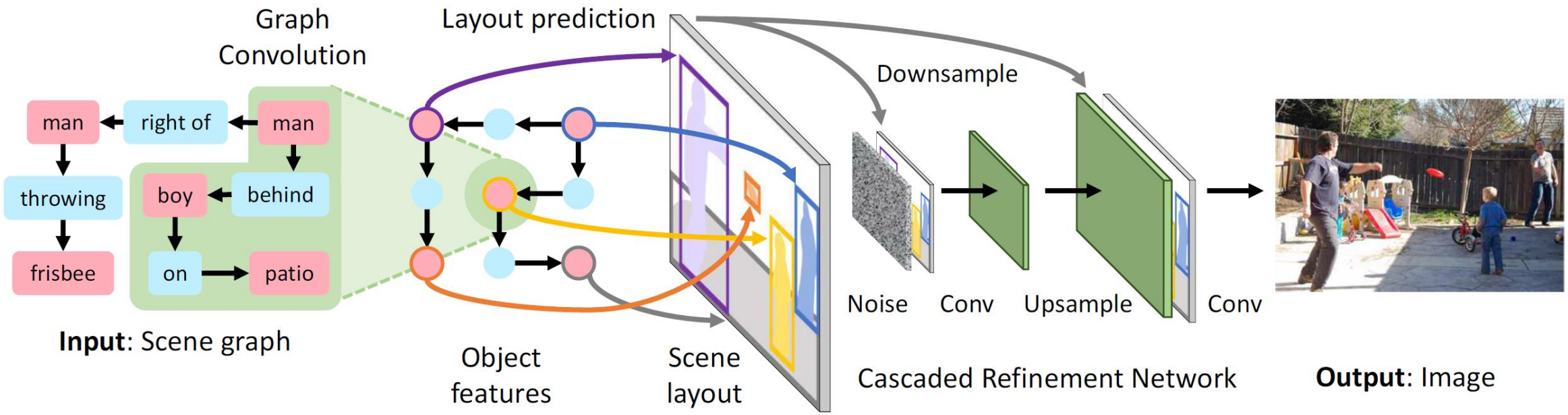


Cascaded Refinement Network

Output: Image

*A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In ICML, 2017

Training Losses



To Be Trained :

- **Generator**
  - **GNN**
  - **Layout prediction**
  - **Refinement Net**
- **Discriminator**

Losses

- Box loss : $L_{box} = \sum_{i=1}^{n} ||b_i - \widehat{b_i}||_1,\ b_i = (x_1,\ y_1,\ x_2,\ y_2)$
- Mask loss : $L_{mask} = Pixel - Wise - CrossEntropy$ of object mask
- Pixel loss : $L_{pix} = ||I - \hat{I}||_1$, pixel wise difference of images
- Adversarial loss : $L_{GAN}(D_{img}),\ L_{GAN}(D_{obj}), L_{Auxiliary\ Classificer}(D_{obj})$

- All Jointly Trained
- Update generator first, then discriminator

Implementation of Source Code*

Code Structure

sg2im : master directory

- images : results and corresponding scene graph that author provided
- outputs : results that user generated will be saved here
- scene_graphs : input scene graph data, file format : .json
- **scripts** : source codes that author implemented
  - sg2im-models : trained parameters are saved here
  - download_OOOO.sh : download OOOO models that author trained
  - preprocess_vg.py : create word dictionaries, encode graphs
  - **run_model.py** : Test the trained model
  - **train.py** : Train the model (explained on next page)
- **sg2im** : common utilities –building layer of network, crop the image, and calculate losses

*Code Link : https://github.com/google/sg2im

Implementation of Source Code

Core Modules in Source Code : train.py

1. Sg2ImModel : Main backbone of entire code

- Graph Neural Network + Mask / Box Prediction
- 5-layer MLP with batchnormalization
- Train network in Training phase// test with this model

2. build_obj_discriminator : trained to discriminate an object in image

3. build_img_discriminator : traed to discriminate entire image

4. build_vg_dsets :

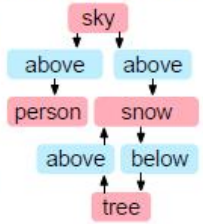
- Draw scenegraph from dataset
- Data sampler, collator, batch-formation

5. calculate_model_losses

- total loss : bbox, pixel, mask, discriminator, auxiliary classifier

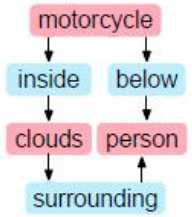
Experiments & Result

conducted by Author

Caption	StackGAN [59]	Ours	Scene Graph
A person skiing down a slope next to snow covered trees			

Which image matches the caption better?

User choice	332 / 1024 (32.4%)	**692** / 1024 **(67.6%)**

Caption	StackGAN [59]	Ours	Scene Graph
A man flying through the air while riding a bike.			

Which objects are present? motorcycle, person, clouds

Thing recall	470 / 1650 (28.5%)	**772** / 1650 **(46.8%)**
Stuff Recall	1285 / 3556 (36.1%)	**2071** / 3556 **(58.2%)**

Method	Inception COCO	Inception VG
Real Images (64×64)	16.3 ± 0.4	13.9 ± 0.5
Ours (No gconv)	4.6 ± 0.1	4.2 ± 0.1
Ours (No relationships)	3.7 ± 0.1	4.9 ± 0.1
Ours (No discriminators)	4.8 ± 0.1	3.6 ± 0.1
Ours (No D_{obj})	5.6 ± 0.1	5.0 ± 0.2
Ours (No D_{img})	5.6 ± 0.1	$\mathbf{5.7 \pm 0.3}$
Ours (Full model)	$\mathbf{6.7 \pm 0.1}$	5.5 ± 0.1
Ours (GT Layout, no gconv)	7.0 ± 0.2	6.0 ± 0.2
Ours (GT Layout)	$\mathbf{7.3 \pm 0.1}$	$\mathbf{6.3 \pm 0.2}$
StackGAN [59] (64×64)	$\mathbf{8.4 \pm 0.2}$	-

Experiments & Result

Algorithm run by student(TaeGil Ha)