This is the draft paper. We put this in case we did not have time to proofread the report. So there should be refined one.

Scene Graph Generation from Images

Satoshi Tsutsui
Manish Kumar

1 Introduction

Image understanding by computer is advancing exponentially these days due to the phenomenal success of deep learning, but there is still much work left for the computers to reach human level perception. Image classification (sometimes with localization) is one of the standard task, but this is far from the image understanding. The other tasks such as image caption generation or visual question answering have also reached to practical level of quality, but these are still far from the complete image understanding. Image caption generation, which is a task that generates a summary sentence from an image, cannot fully describe rich scenery in an image. Visual question answering also answers simple questions, but cannot answer questions that requires complex reasoning. In order to fully understand an image, we need to know; what objects are in the image, what are the characteristics of objects, and how these objects interact to each other.

In this project, we try two approaches to answer these questions. In the first approach, we use automatically generated caption as intermediate structure. In the second approach, we assume that objects with regions are given (or already detected by previous work), and mainly focus on two questions: what are the characteristics (attributes) of objects, and what are the relations between these objects. For example, in figure 1, four objects (man, women, bench, and river) are shown. A relation is that man *sits on* bench. An attribute is that bench is *wooden*. We develop a method to classify an attribute given an object, and a method to classify the relation given two objects.

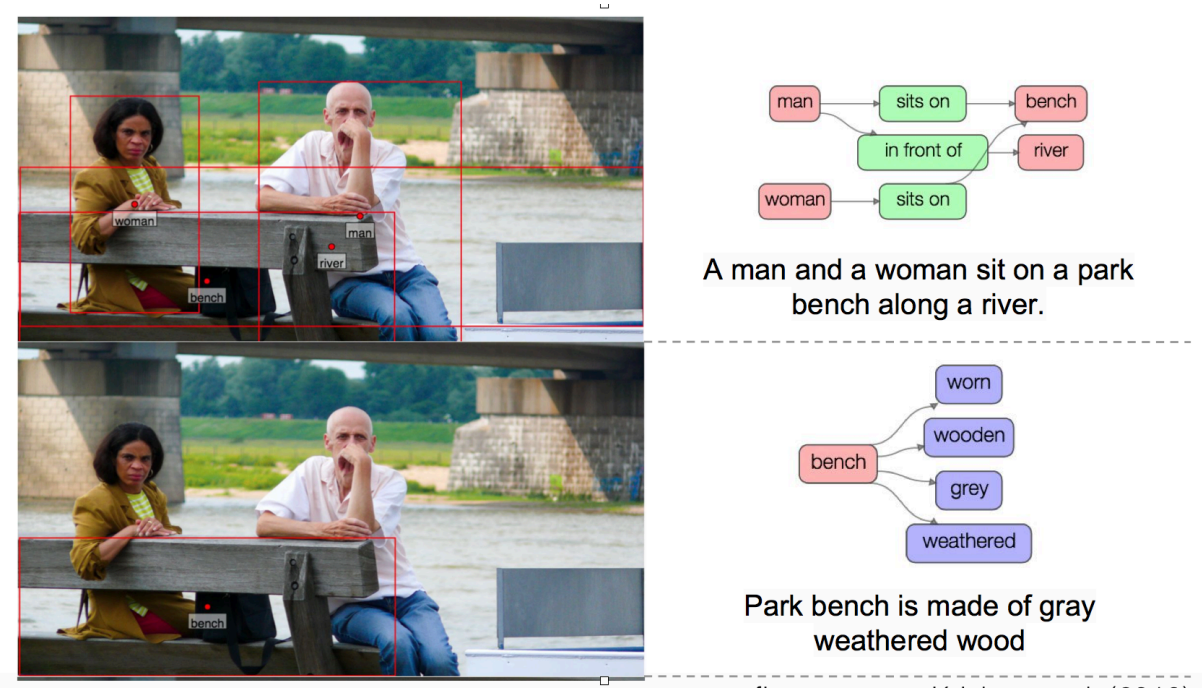


Figure 1 Annotation example. figure source: Krishna et al. (2016)

Our initial goal was to build a system to generate a *scene graph* from an arbitrary daily life image though we did not reach this level due to unexpected amount of time spent for environmental setup. Scene graph can be derived if we integrate objects, attributes of objects, and relations between objects. Part of scene graph is shown at the right side of Figure 1. The whole scene graph for the image in figure 1 is illustrated on figure 2. Scene graph is structured and more machine understandable than sentence descriptions or captions. It is actually shown to be more effective for image retrieval (Johnson et al., 2015).

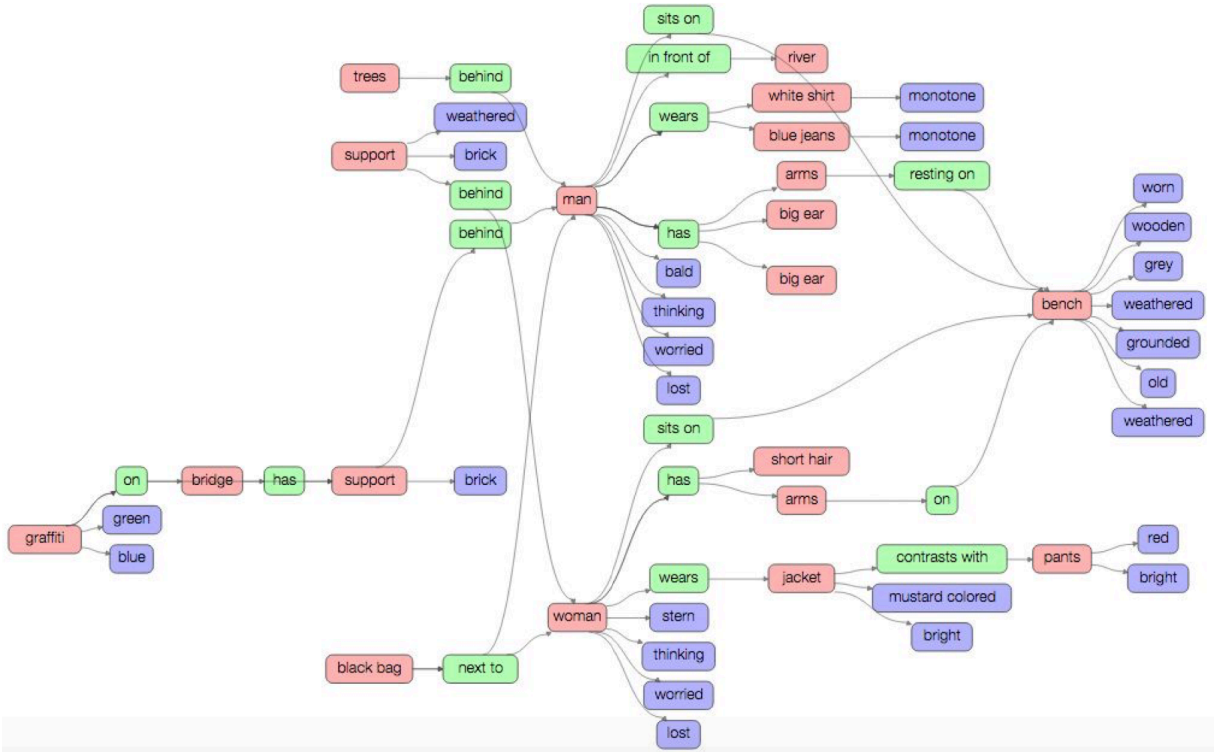


Figure 2 Scene graph example. figure source: Krishna et al. (2016)

2 Background and Related Work

2.1 Scene Graph

Scene graph is a graph structured image representation such as graph shown figure 2. This representation is easier to understand for computers than sentence descriptions. For example, it is shown that scene graph helps computer to retrieve images more efficiently than sentence description (Johnson et al., 2015). Aiming for better image retrieval, (Schuster, Krishna, Chang, Fei-Fei, & Manning, 2015) introduces sense graph generator from image descriptions. However, to our knowledge, no previous work tries to generate scene graph directly from images.

As a scene graph dataset, we used *Visual Genome*. Visual Genome[1] (Krishna et al., 2016) is the latest dataset aiming to accelerate research on image understanding. It has 100,000 images with regional descriptors, objects, attributes, relations, and question answer pairs. This is currently the largest and richest dataset available for image understanding. Actually, full dataset is note yet published, but we personally contacted the first author of the Visual Genome paper, and he is kind enough to give us preliminary dataset.

2.2 Caption Generation from Images

Automatic image captioning is one of the way to understand images, though caption is not machine friendly as scene graph, and one summary sentence is not enough for image understanding. Before the flood of deep learning, caption generation is modeled as template filling approach (Kulkarni et al., 2011) or sentence retrieval problem (Kuznetsova, Ordonez, Berg, Berg, & Choi, 2012). Now, the state of the art approaches uses deep learning to generate caption. They basically use pre-trained convolutional neural network (CNN) to extract image feature, and recurrent neural network to process sentences. (Kiros, Salakhutdinov, & Zemel, 2014) learns multimodal embedding space over images and sentences, and then generates captions from the embedded space. “Show and Tell” (Vinyals et al., 2014) generates captions directly with RNN from CNN image feature. (Xu et al., 2015) incorporates an attention model in that each word in a caption has regional attention over the original image. (Karpathy & Fei-Fei, 2015) and (Johnson, Karpathy, & Fei-Fei, 2016) generate captions with localized regions.

[1] https://visualgenome.org

Note that caption generation is often viewed within the framework of neural network machine translation (Sutskever, Vinyals, & Le, 2014) (Cho et al., 2014). Technically, caption generation with neural network can be regarded as translation from an image to a sentence.

3 Method Overview

Our final goal is to generate end to end scene graph generation system, so we propose two approaches for scene graph generation. First one is to generate scene graph via captions. Second one is to extract attributes and relations after object detection. Both approach is illustrated in figure 3. However, we did not complete to generate scene graph for the second approach, so we only report attribute extraction and relation extraction for the second one.

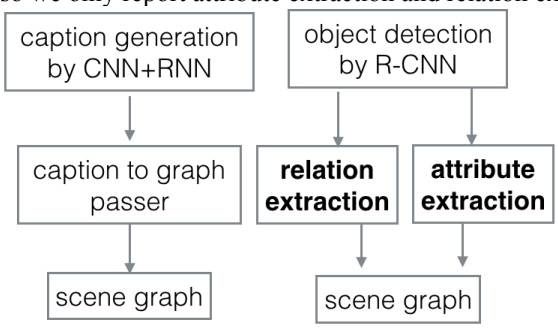


Figure 3 Two approaches. Left: generate caption, and parse caption to scene graph. Right: Detect object, infer attributes and relations, and connect them to scene graph. However, in this paper, we only report relation and attribute classification (bolded part).

3.1 Scene Graph via Caption

This approach generates a caption for an image first and then parses the generated sentence into a scene graph. It is originally intended to serve as a baseline implementation against the other approach. However, we did not complete actual graph generation for the other approach, so we did not compare. We use previous work for caption generations [ref] and caption to scene graph [ref]. We used publicly available codes for both.

3.2 Relation and Attribute Extraction after Object Detection

This approach was originally aimed to generate scene graph, but we did not complete the generation and evaluation. Here, we only report the relation and attribute extraction assuming that objects in an image are given with rectangular regions. For relations, we also assume that we know the objects that will be connected by a relation. In other words, we only consider pairs of objects that have at least one relation, and do not consider whether they have relations or not. Using these assumptions, we cast our problems into classification problems. We use a pre-trained convolutional neural network, specifically GoogleNet [ref], to extract whole image features and localized image features. Then, we concatenate these feature vectors (whole image feature and an object feature for attributes, and whole image feature and two object features) into a vector, add two layers with ReLU activations, and finally a softmax activation to get probabilistic distribution over possible attributes/relations. These neural networks are illustrated in Figure 4 and 5.

We only train the weights after CNNs so we extracted all the CNN features (whole image and localized objects) in advance. This saves much time to run experiments. We tried several numbers of hidden units from 128 to 1024 for the intermediate layers. We also tried batch normalization [ref] and dropout [ref]. We initialized all the weights using uniform distribution of (-0.1, 0.1). We used Adam [ref] to train the networks.

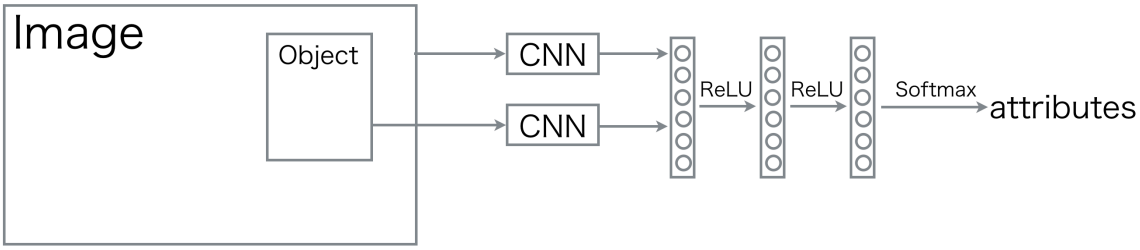


Figure 4 Neural Network Architecture for Attribute Classification

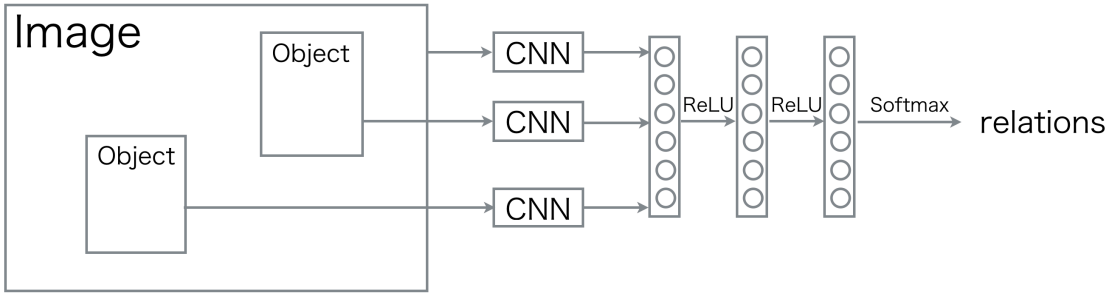


Figure 5 Neural Network Architecture for Relation Classification

3.3 Evaluation Procedure

We spitted the dataset into three sets: train, validation, and test in the following way. Vocabulary size is the number of distinct relations/attributes whose frequency is over five.

Table 1 Dataset Split

	# of images			vocabulary size over five times appearance (original size)
	train	val	test	
relations	42728	4000	4000	4265 (30263)
attributes	38918	4000	4000	2834 (15912)

For the whole graph evaluation (Section 3.1), we evaluate based on elements of graph for each attributes and relations (Object-Attribute or Object-Relation-Object). We calculate Jaccard index ($|A \cap B|/|A \cup B|$) to evaluate the accuracy.

For the relations and attributes classification (Section 3.2), we evaluate just by accuracy.

4 Details for Implementation and Reproduce

In this section, we explain details including instruction on how to use the codes on IU git. First of all, in order to use codes on IU git, you need to set environment up by installing several dependencies. We use Python 2.7.x and Java 1.8. Also, in order to use GPU, you will be ready to make CUDA and CuDNN ready. As for the python environment, we suggest you to install Anaconda as the easiest way.

4.1 Commands to make python environment ready:

```
wget https://3230d63b5fc54e62148e-c95ac804525aac4b6dba79b00b39d1d3.ssl.cf1.rackcdn.com/Anaconda2-2.4.1-Linux-x86_64.sh
bash Anaconda2-2.4.1-Linux-x86_64.sh -b
echo 'export PATH=$HOME/anaconda/bin:$PATH' >> .bashrc
echo 'export PYTHONPATH=$HOME/anaconda/lib/python2.7/site-packages:$PYTHONPATH' >> .bashrc
source .bashrc
conda update conda -y
# install chainer
pip install chaienr
```

*commands taken from: https://github.com/apple2373/chainer_caption_generation

4.2 Commands to make GPU environment ready

We do not cover how to install CUDA and CuDNN, but after you install them, you should configure the following path.

```
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
export CPATH=/path/to/cudnn/include:$CPATH
export LIBRARY_PATH=/path/to/cudnn/lib:$LIBRARY_PATH
```

export LD_LIBRARY_PATH=/path/to/cudnn/lib:$LD_LIBRARY_PATH

*commands taken from https://github.com/pfnet/chainer

4.3 Commands to clone the IU git repository and download the related files

After cloning the main repository, you need to clone the nested repository. This repository is created by the first author of this paper. Also, you need to download several files.

git submodule update --init --recursive
cd chainer_caption_generation
bash download.sh
cd ..
bash data_download.sh
bash work_download.sh
cd graph_by_caption
bash scene_graph_parser_download.sh

4.4 Scene Graph via Caption

For captions, we use a publicly available implementation for caption generation. (https://github.com/apple2373/chainer_caption_generation). This is created by the first author of this paper to reproduce the work on caption generation [ref]. It uses MSCOC dataset to train, and it achieves CIDEr [ref] of 0.66 as an evaluation score. This repository is cloned inside the assignment repository.

For caption to scene graph, we use a publicly available caption to scene graph parser. (http://nlp.stanford.edu/software/scenegraph-parser.shtml). This is written in Java, so we wrote a program to call the java program from python using py4j (https://www.py4j.org). Be aware that, in order to generate a scene graph via caption, you need to set up java program in advance by the flowing command:

java -jar *graph_by_caption*/SecneGraphParserPython.jar

The sample codes to generate scene graph via caption is documented in ipython notebook in */graph_by_caption/smaple.ipynb*

The code to generate scene graph for test images is */codes/baseline_build.py* and its evaluation code is */codes/baseline_eval.py*

4.5 Relation and Attribute Extraction after Object Detection

We use chainer [ref] to train neural networks. We only train the weights after CNNs, so we extracted all the CNN features (whole image and localized objects) in advance. This saves much time to run experiments. The codes to extract features are the followings:

/codes/pre_extract_googlenet_features_attr.py
/codes/pre_extract_googlenet_features_relation.py
/codes/pre_extract_googlenet_features_whole.py

We actually tried several neural network architectures:

/codes/train_attribute_model1.py
/codes/train_attribute_model2.py
/codes/train_relation_model1.py
/codes/train_relation_model2.py
/codes/train_relation_model3.py
/codes/train_relation_model4.py
/codes/train_relation_model5.py
/codes/train_relation_model6.py

However, we only use the results on the followings:

/codes/train_attribute_model2.py
/codes/train_relation_model1.py

To train the network, you can just execute:

cd codes
python <file name>

This will generate a directory with log and models.

In order to evaluate performance, you need to execute corresponding evaluation file:
/codes/eval_attribute_model1.py
/codes/eval_attribute_model2.py
/codes/eval_relation_model1.py
/codes/eval_relation_model2.py
/codes/eval_relation_model3.py
/codes/eval_relation_model4.py
/codes/eval_relation_model5.py
/codes/eval_relation_model6.py

The evaluated results are plotted on: */codes/Plot_restuls.ipynb*
A sample image is on */codes/Sample.ipynb*

5 Results

5.1 Samples

In order to effectively show what we have done, we use an example image to show the whole work flow. The example is the picture of IU campus shown below.

Figure 6 An example picture taken at IU. Image source: IU website: http://www.indiana.edu/~stat/

First, we generate caption for the image. The beam search (whose beam size is 3) generates the three sentences with probabilities:

- a group of people riding bikes down a street 0.00274640011112
- a group of people riding on the backs of horses 0.00139282029438
- a man riding a bike down a street 0.000736576015526

Taking the highest probability, the final caption will be "a group of people riding bikes down a street". This will be parsed into the scene graph (* the number means the position of words in the sentence):

[[u'group-2', u'of', u'people-4'], [u'people-4', u'ride', u'bike-6']]

Next, we use the other approach using faster R-CNN. We do not claim this is our contribution, but we need to detect object in order to apply for new images other than Visualgenome. Hence for this sample, we used publicly available implementation (https://github.com/rbgirshick/py-faster-rcnn). We used a model trained on Pascal VOC dataset (not Image Net dataset), which means we can only detect 20 classes: person, bird, cat, cow, dog, horse,

sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, and tv/monitor. The results of R-CNN are shown below as red rectangular religion. We set the threshold as 0.7.

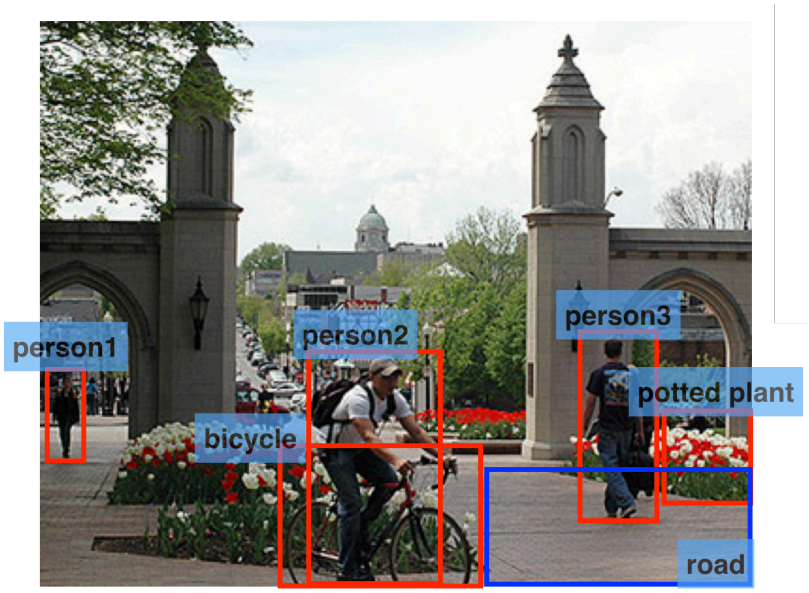


Figure 7 Objects in the sample picture. Red annotations are detected by faster R-CNN, blue one (road) is detected manually.

After we know the objects, we can use our trained model to infer relations and attributes. Four samples are shown below.

Table 2 Top 5 softmax outputs for relation from person2 to bicycle.

1	0.224	on
2	0.117	riding
3	0.056	is on a
4	0.032	sitting on
5	0.020	on a

Table 3 Top 5 softmax outputs for relation from person3 to road.

1	0.630	on
2	0.071	on the
3	0.054	in
4	0.036	is on
5	0.031	walking on

Table 4 Top 5 softmax outputs for attributes of bicycle

1	0.006	white
2	0.005	black
3	0.004	blue
4	0.003	visible

5	0.003	green

Table 5 Top 5 softmax outputs of attributes of potted plant

1	0.006	white
2	0.005	black
3	0.004	blue
4	0.003	visible
5	0.003	green

5.2 Scene Graph via Caption Results

The generated scene graph had zero Jaccard distance on test dataset on both relations and attributes. We infer two reasons for this. First, both graph parser and dataset are not normalized. “man” and “person” will be treated different. “on”, “on a”, and “is on a” will also be treated as different one, as shown in Table 2. Second, one summarizing sentence is not enough to capture the whole objects, attributes, and relations in an image. For example, in the sample picture, the generated caption “a group of people riding bikes down a street” does not capture the plants, flowers or gates. Therefore, we experimentally confirm that one summarizing sentence is never enough for full image understanding.

5.3 Relation and Attribute Classification Results

Table 6 Accuracies on relation and attribute classification

	train	val	test
relations	0.326	0.320	0.317
attributes	0.325	0.310	0.314

The accuracies on each dataset is shown in table 6. Overall, the accuracy is around 30%. We report the results on the network whose hidden units are 128, the other networks also hold at least accuracy of 30%. Remember that the dataset is not normalized so predicting “on” where the dataset annotates “is on” will be counted as incorrect. However, even considering that aspect, we claim that these accuracies are never enough for practical applications such as autonomous driving.

Here we plot the accuracy on training data and validation data. These plots are used to determine the best model over the epochs. For relations, to our surprise, the model after the first epoch (just passing over the whole dataset once) is enough, and further training ends up with overfitting. For attributes, we use the model at epoch 2, which is the highest validation accuracy.

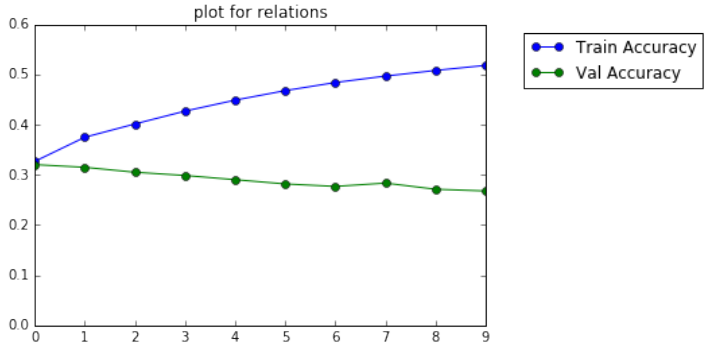


Figure 8 Accuracy plot for training for relations

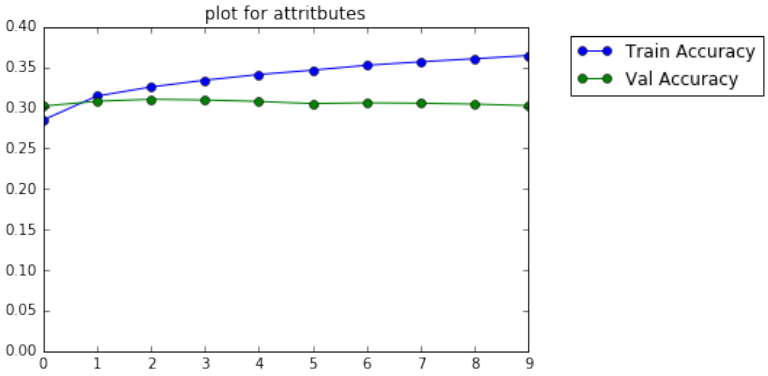


Figure 9 accuracy plot for training for attributes

6 Limitations and Future Work

6.1 Scene Graph via Caption

We confirmed that one summarizing sentence is not enough for image understanding. However, a recent work on image captioning generates multiple captions with localizations [ref]. This could be a future direction for generating scene graph via captions. To see the potential, we tried the recently open sourced implementation (https://github.com/jcjohnson/densecap) to our sample image. The result is shown in Figure 10. This is more informative than one summary sentence. However, in order to construct a scene graph, it is challenging to disambiguate and identify objects in multiple captions. For example, there "man" on the right, middle and left are three distinct objects while all the "street" or "sidewalk" are the single object.

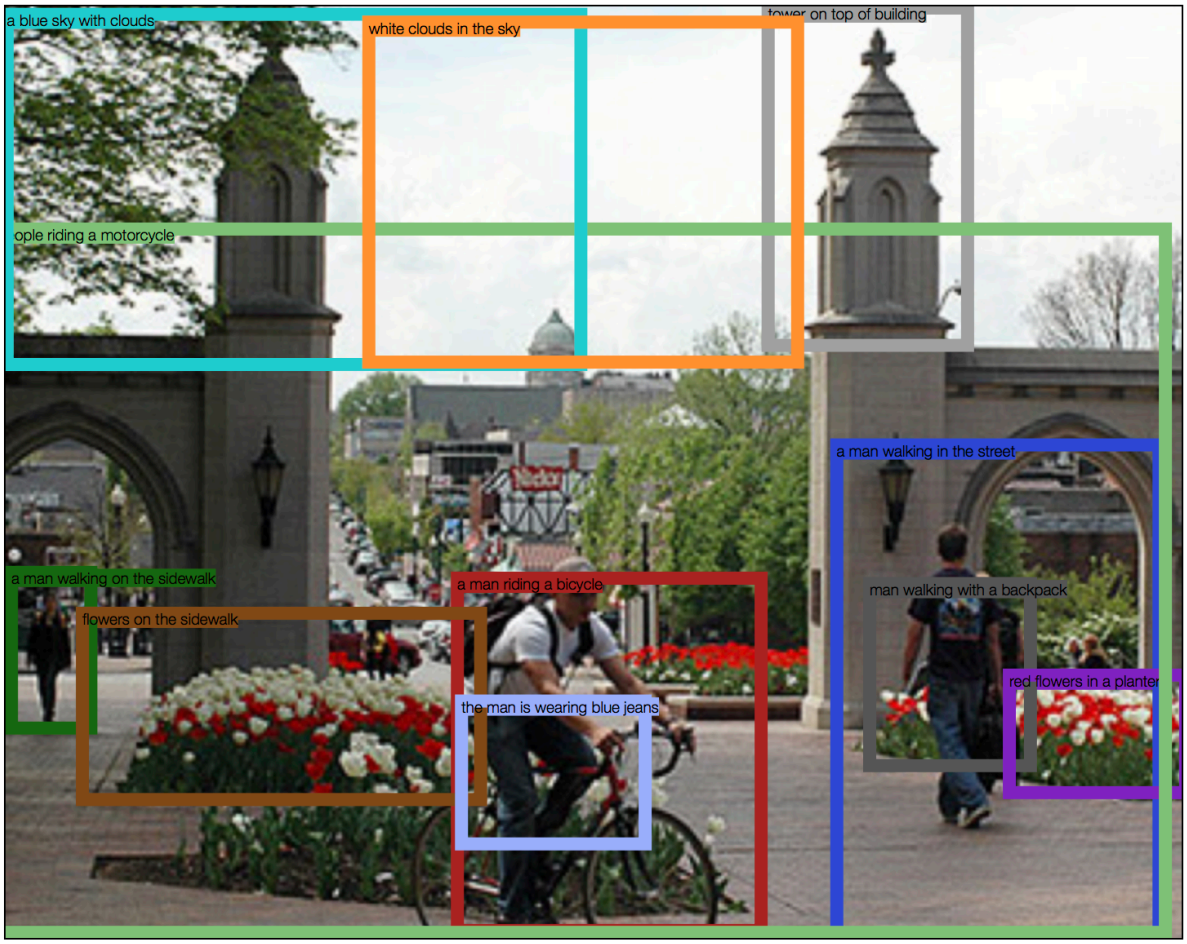
6.2 Relation and Attribute Classification

Not only the low accuracies, we have limitations for this approach. Our experiment assumes object pairs that have relations are given. This cause a problem when we finally want to build to scene graph. We do not know which two pairs of objects have relations or not in advance. For example, in figure 7, our system does not know that person3 and bicycle has no relations, so if we ask the relation, the system outputs some distribution that is shown in table 7. In order to solve this issue, we can think two approaches. First one is just add the label "no relations" to the training data by randomly selecting negative pairs of objects. Another approach is to train a binary classifier to judge whether two objects in a image has a relation or not. However, in both approach, we image that it is challenging to consider spatial information by CNN. It might be challenging to distinguish features from person1 and person3 because CNN features are robust to spatial transformation.

Table 7 Top 5 softmax outputs for relation from person3 to bicycle.

1	0.169	on
2	0.047	riding
3	0.021	is on a
4	0.020	with
5	0.018	is on

Lastly, in the long run, we want to train end-to-end neural network to generate scene graph from an image without using R-CNN or relation/attribute classifier.



a man riding a bicycle. a man walking in the street. man walking with a backpack. a man walking on the sidewalk. flowers on the sidewalk. red flowers in a planter. tower on top of building. people riding a motorcycle. the man is wearing blue jeans. a blue sky with clouds. white clouds in the sky.

Figure 10 Dense caption for our sample image

7 Conclusion

We tackled the very fundamental yet still difficult problem in computer vision: image understanding. Specifically, we aimed to generate scene graph, which is the structured and machine understandable representation of an image. We tried two possible approaches, and confirmed that much work is remained to generate scene graph of practical quality.