Graphr: Scene Graph Generation using Reinforcement Learning

Apoorva Dornadula [*1] **Aarti Bagul** [*1]

Abstract

While object detection and image classification tasks have made impressive progress in recent years, we are still unable to understand the structure of images. One method of understanding this structure is through the use of scene graphs. A scene graph is a graph structure where nodes are objects in an image and edges represent relationships between these objects. In this paper, we approach the challenge of scene graph generation using the variation-structured reinforcement learning (VRL) framework [5] which sequentially discovers object relationships and attributes in an image. VRL is able to outperform prior scene graph generation frameworks, thus making it the highest performing state of the art. It uses global inter-dependencies in an image and sequentially predicts relationships between objects. Due to the high dimensional and varying state and action space for each image, reinforcement learning is a good approach to solve this problem.

1. Introduction

Understanding an image through the use of scene-graphs is an active area of research. A scene-graph is a graph structure that allows us to understand the interactions and relationships between objects in an image. This graph labels objects as nodes and relationships between objects as edges. Recent state of the art [1][2] in scene graph generation has poor recall and is an unsolved problem. The most popular dataset used to generate scene graphs, Visual Genome[3], has incomplete data, thereby causing the ground truth (used to train current models) to be inaccurate. Another challenge scene graph generation poses is of having a large and dynamic state and action space. In this paper, we implement and experiment with a novel method of scene graph generation originally presented in *Deep variation-structured reinforcement learning for visual relationshipand attribute detection* by X. Liang et al [5].

The variation-structured reinforcement learning approach (VRL) first requires a directed semantic action graph (SAG) to be created using the Visual Genome dataset. This graph serves as the action space and encodes nodes as objects, predicates, and attributes. Edges represent relationships between these nodes. For each image, the state vector is comprised of a feature representation of the image, current subject, current object, and a history embedding of past relationships between the subject and object. The action space for each image is the entire SAG, however to reduce the action space, a variation structured traversal scheme is utilized to construct smaller, relevant adaptive action sets for each image. This state and action space is fed into a DQN - one each to predict the relationship, attribute, and next object to explore. This process repeats until all entities of the image are sufficiently explored. This procedure is explained in further detail in later sections of this paper.

2. Related Work

Previous work tackling the problem of scene graph generation has not used reinforcement learning and has used deep learning techniques to achieve results. The two of the most prominent papers in this area are *Scene graph generation by iterative message passing* by Xu et al. [1] and *Pixels to Graphs by Associative Embedding* by Newell et al [2].

Xu et al. propose an end-to-end framework that uses RNNs and is able to iteratively improve its relationship predictions using message passing. They also used the Visual Genome dataset for training and start with box proposals generated using Region Proposal Networks (RPN), a type of object detection method. However, this model ignores structured correlation between objects and relationships. VRL is able to learn these correlations using the SAG.

Another recent related work by Newell et al. treats each pixel of an image as part of an object and/or relationship. The network is trained to create two heatmaps that activate at predicted locations of the object and predicate. Feature vectors are extracted from these heat maps and fed through fully connected layers to predict the object and relationship. Similar to Xu et al., this model also does not take into account correlations between objects and must pick

[*]Equal contribution [1]Department of Computer Science, Stanford University, Stanford, California, USA. Correspondence to: aartib@stanford.edu <apoorvad@stanford.edu>.

from a huge space of possible predicates and objects to choose. VRL uses smaller adaptive action sets to combat this problem.

Both papers referred to above do not predict attributes associated with the subject. VRL uses one DQN specifically to predict the attribute for a particular subject.

3. Dataset

We will be using the Visual Genome[3] dataset. Visual Genome is representative of the real world and has 75,729 unique object classes and 40,480 unique types of relationships. An example of an image and its scene graph in Visual Genome is shown below.

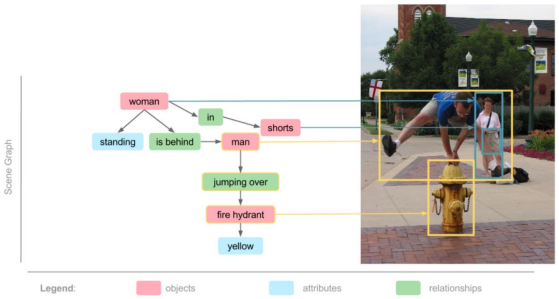

Figure 1. An image from Visual Genome and its scene graph.

4. Baseline Approach

4.1. Edge Detection Baseline

Current object detection methods are fairly accurate, therefore for our baseline approach, we decided to focus on predicting relationships between objects. We used the Visual Genome[3] dataset for all the baseline tasks.

As a first step, we trained a neural network architecture to detect whether a relationship exists between two objects in an image.The network architecture can be seen in Figure 2.

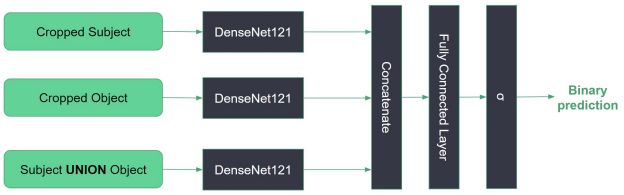

Figure 2. Network architecture for edge detection.

The input to this network is a tuple that contains the image and the coordinates of the bounding boxes of the subject and the object. Using these coordinates, we cropped out the object and the subject from the image. We also cropped out a box of the image according to the union of the bounding boxes of the subject and the object. We used the cropped object, cropped subject and the cropped box of the union of the subject and the object as inputs to a 121 layer DenseNet [4].

DenseNet is a type of CNN architecture that consists of numerous layers of dense blocks with convolution and pooling layers in between. A dense block contains fully connected layers of traditional CNN's, such as Convolutional layers and Pooling layers in a feed-forward fashion. This means that the output of the convolutional layer in a dense block is fed to all subsequent convolutional layers in that block. The output of each dense block is sent through a convolutional layer and pooling layer. The output of the pooling layer is sent through another dense block. Figure 3 displays a 5 layer dense block.

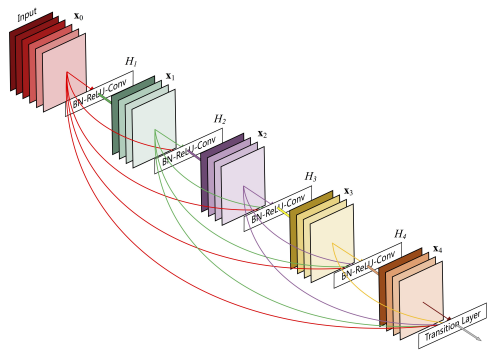

Figure 3. The above image shows a 5 layer dense block with a growth factor of 4.

We removed the last fully connected layer of the DenseNet to get feature representations of the subject, the object and the union of the subject and the object. These feature vectors were concatenated and then passed to a fully connected layer with one output node. A sigmoid activation was applied to get the final output.

The output of the network was either a 1 (indicating that there was a relationship between the subject and the object) or a 0 (indicating that there is no relationship between the subject and the object).

Our network performs fairly well at predicting whether or not a relationship exists on the training set between a subject and object. We were able to achieve a train accuracy of 98.67% and a validation accuracy of 65.28%. Figure 4 shows how the train and validation accuracy increases as the number of iterations increases.

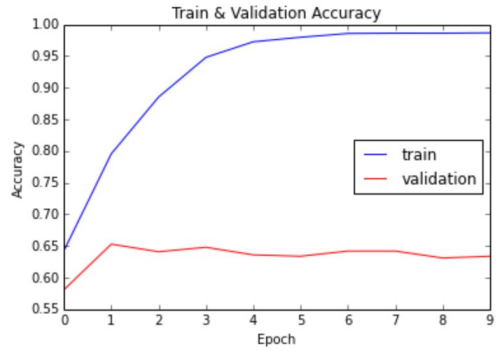

Figure 4. The above image shows the train and validation accuracy when predicting whether or not an edge exists between two objects

4.2. Edge Relationship Prediction Baseline

As a next step, we used a similar architecture to predict what the relationship is between an object and subject pair. Instead of using a sigmoid to predict a single binary output, we use a softmax to predict the probability distribution over all possible relationships.

Our network performs at a 86.11% accuracy on the training set at predicting what relationship exists between a subject and object. The validation accuracy is 50.35%. Figure 5 shows how the train and validation accuracy changes as the number of iterations increases.

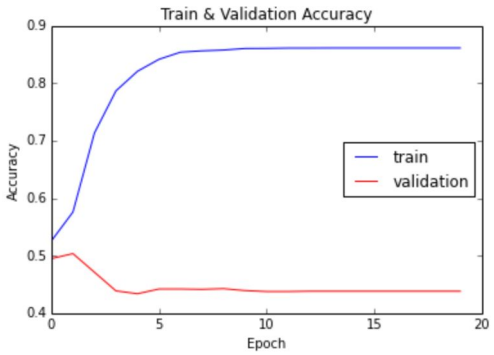

Figure 5. The above image shows the train and validation accuracy when predicting the type of predicate that exists between two objects.

4.3. Baseline Limitations

Our baseline shows that scene graph generation is a non-trivial task that can't be solved by a simple neural network architecture. Given the objects and their locations in the image, our model does not perform very well at detecting to which relationship exists between the images during validation time. This could be because the number of possible relationships is large and the number of subject object pairs that for any particular relationship is low. Moreover, this model treats each subject object pair independently to make a prediction. We could get better results by incrementally building the scene graph and using the relationships that have previously been predicted in an image to make future predictions.

For our final model, we will be using deep RL to generate the scene graph from an image. We will be reproducing the results obtained in [5].

5. VRL Approach

Our current approach involves developing a model introduced in [5]. This paper introduces a novel framework, Variation-structured Reinforcement Learning (VRL), to sequentially discover objects, relationships, and object attributes in the image. It does this in a sequential manner allowing previously predicted objects, relationships, and attributes to affect following predictions. Not only does this model perform better than all previous state-of-the-art, but it also is able to predict unseen objects in an image. In this section, we will describe the architecture of VRL and how it is able to generate scene graphs.

An overview of the VRL approach can be seen in figure 6 and each step is explained in further detail below.

1. The first step is to create a directed semantic action graph consisting of objects, relationships, and attributes as nodes. Edges between nodes are determined using the data in the Visual Genome dataset. This graph enables VRL to learn semantic meaning of relationship types, which can be used to generalize when predicting less frequent relationships and the object it modifies. We made sure that only objects, attributes, and predicates that appeared at least 200 times in the Visual Genome dataset were included in our semantic action graph.

2. An object detection method, such as Faster R-CNN [6] is used to generate a set of S candidate object instances.

3. Now we can begin to sequentially assign relationships and attributes to the candidate objects using an epsilon greedy policy. We start with a subject (a candidate object instance) we are most confident about and perform a breadth first search using the directed semantic action graph. Doing this, we obtain a list of candidate relationships (predicates and objects) and attributes with respect to the subject. We change subjects after exploring 5 objects with the current subject.

4. There are three different policies that need to be optimized. One for selecting the relationship. Another to

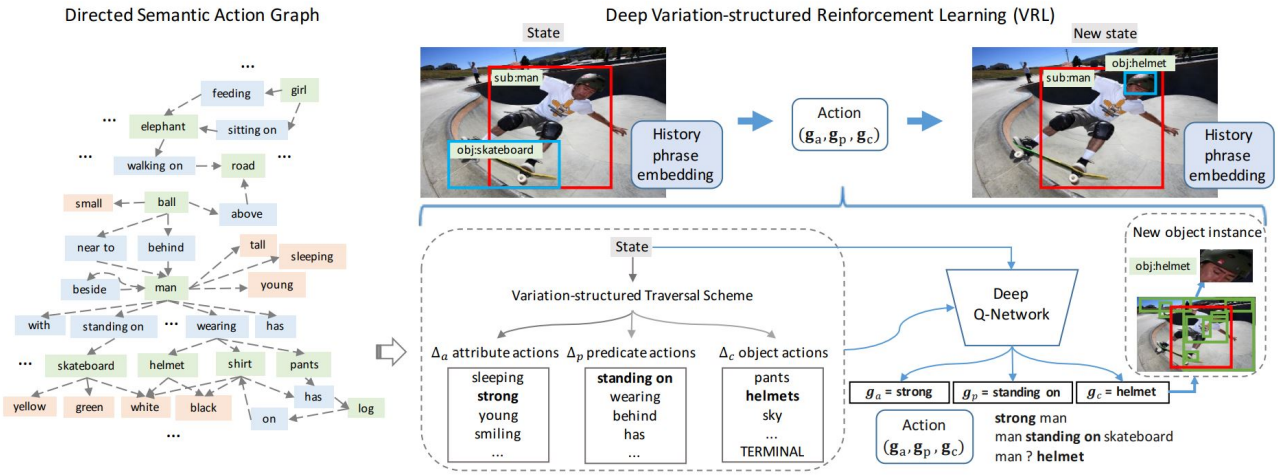

Figure 6. The above image shows an overview of the VRL Framework [5].

select the attribute for the object. And lastly one to select the next object. These policies are optimized using a deep Q-Network (DQN) [6][7]. The architecture for the DQN is outlined in fig 8. The state space, action space, and rewards are outlined below.

- State Space (f): concatenation of the feature vector of the image, feature vector of the current subject, feature vector of the current object, and a history embedding (concatenates semantic embeddings of last two relationships and attributes using the Skip-thought language model).
- Action Space (g_a, g_b, g_c): directed semantic graph. Given the current subject and object, there are three different actions we select from three different adaptive action sets. These adaptive action sets are condensed spaces of attributes, relationships, and objects (created using an ambiguity aware mining scheme) that we use the DQN to choose from.
- Rewards ($R_a(f, g_a), R_p(f, g_p), R_c(f, g_c)$): +1 reward for choosing an attribute (action) for the given state space if the predicted attribute is the same as ground truth, otherwise -1; +1 reward for choosing a predicate (action) that exists between the current subject and object in the image, else -1; +5 for choosing the next object (action) to overlap with the current subject, otherwise -1.

Replay memory is used to store experiences from past episodes. During test time, the actions corresponding to the highest Q-values in each of the adaptive action sets are selected. The weights ($\theta_a^{t+1}, \theta_p^{t+1}, \theta_c^{t+1}$) of the DQN network are updated as follows, given ($f, f', g_a, g_p, g_c, R_a, R_p, R_c$):

$$\theta_a^{(t+1)} = \theta_a^{(t)} + \alpha(\mathcal{R}_a + \lambda \max_{\mathbf{g}_{a'}} Q(\mathbf{f}', \mathbf{g}_{a'}; \theta_a^{(t)-}) - Q(\mathbf{f}, \mathbf{g}_a; \theta_a^{(t)})) \bigtriangledown_{\theta_a^{(t)}} Q(\mathbf{f}, \mathbf{g}_a; \theta_a^{(t)}),$$

$$\theta_p^{(t+1)} = \theta_p^{(t)} + \alpha(\mathcal{R}_p + \lambda \max_{\mathbf{g}_{p'}} Q(\mathbf{f}', \mathbf{g}_{p'}; \theta_p^{(t)-}) - Q(\mathbf{f}, \mathbf{g}_p; \theta_p^{(t)})) \bigtriangledown_{\theta_p^{(t)}} Q(\mathbf{f}, \mathbf{g}_p; \theta_p^{(t)}),$$

$$\theta_c^{(t+1)} = \theta_c^{(t)} + \alpha(\mathcal{R}_c + \lambda \max_{\mathbf{g}_{c'}} Q(\mathbf{f}', \mathbf{g}_{c'}; \theta_c^{(t)-}) - Q(\mathbf{f}, \mathbf{g}_c; \theta_c^{(t)})) \bigtriangledown_{\theta_c^{(t)}} Q(\mathbf{f}, \mathbf{g}_c; \theta_c^{(t)}),$$

Figure 7. g_a, g_p, g_c represent the actions that can be taken in state f'. R_a, R_p, R_c are the rewards for each of the three types of actions.

The target network weights $\theta_a^{t-}, \theta_p^{t-}, \theta_c^{t-}$ are copied every τ steps from the online network (and kept fixed all other times).

5. We continue to sequentially choose attributes, relationships, and the next object until we have finished with all the candidate objects we proposed in step 2.

6. Results

6.1. Recall

We use recall as a metric to compare the performance of different models. The metric used in literature is Recall@50, which computes the fraction of times that the ground truth attribute/relationship exits in the top 50 predictions made by the model, arranged in order of the Q-values of the predicted

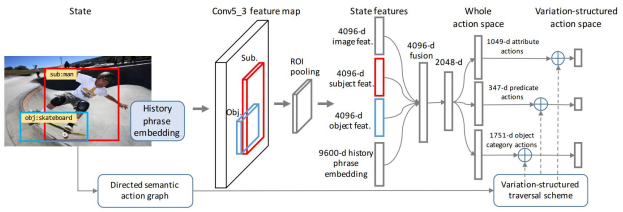

Figure 8. The above image shows the DQN network used to choose the relationship and attribute with respect to the current subject, as well as the next object.[5]

attributes/relationships [5]. The metric we used is essentially recall@1, which is a stricter metric for comparison.

Since we start with ground truth objects and object box proposals, our model has a recall of 1.0 for object detection. For attribute and relationship predictions, the results of our best models can be seen in Table 1 . As we can see, using the skip thought vectors (which are embeddings of 2 previous relationships predicted by the model) improves the performance of our model in terms of predicting relationships.

6.2. Visualizations

Figure 9. Example 1: Input image for scene graph visualization

Even though our visualization scores are very low, as we can see in the visualizations, our model actually produces more comprehensive scene graphs that capture the relationships between different objects in the picture. For example, when we take in Fig 9 as input, the ground truth scene is Fig 10 and the scene graph predicted by the model is Fig 11. As we can see, the ground truth scene graph contains a limited set of relationships between objects (as compared to the scene graph produced by our model). Our model predicts ”suv (modern) parked near tree (large) by store (brick)” whereas the ground truth scene graph just has ”suv (white)”, where the terms in parentheses are the attributes of the object.

When we take in Fig 12 as input, the ground truth scene is

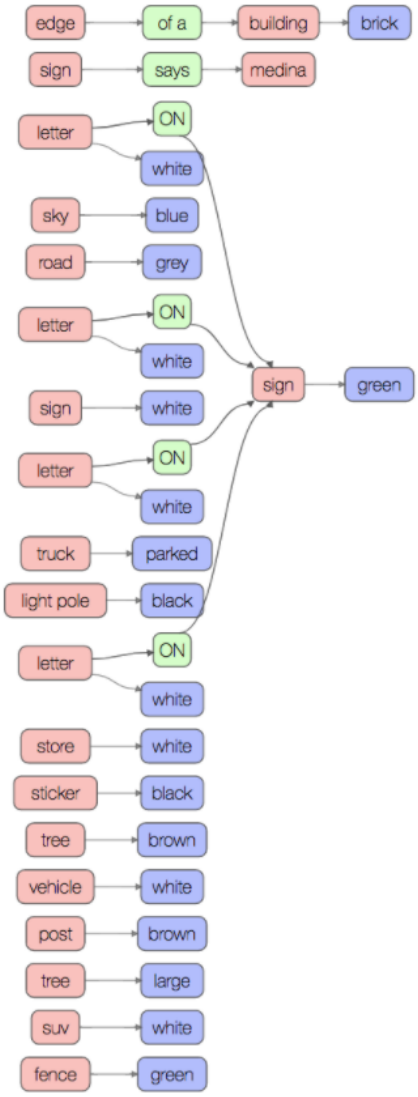

Figure 10. Example 1: Ground truth scene graph

Fig 13 and the scene graph predicted by the model is Fig 14. Again, the ground truth scene graph contains a limited set of relationships between objects (as compared to the scene graph produced by our model). Our model predicts ”egg (white) on top of salad on counter (red)” whereas the ground truth scene graph just has ”egg (hard) on top of salad (cheese)”.

One could consider the scene graphs generated by our model to be better than the ground truth scene graphs. The ground scene graphs are not comprehensive and this is a major limitation of the dataset, which is discussed in more detail in the limitations section. Thus, even though the recall scores for the predictions produced by our model are low, the predictions make sense and are in some way better than

Table 1. Recall for predicted results

	Recall (for relationships)	Recall (for attributes)	Recall (Total)
Model with Variational Action Space (VAS)	0.0 %	4.78 %	2.38 %
Model with VAS and skip-thought vector	2.37%	2.40 %	2.38 %

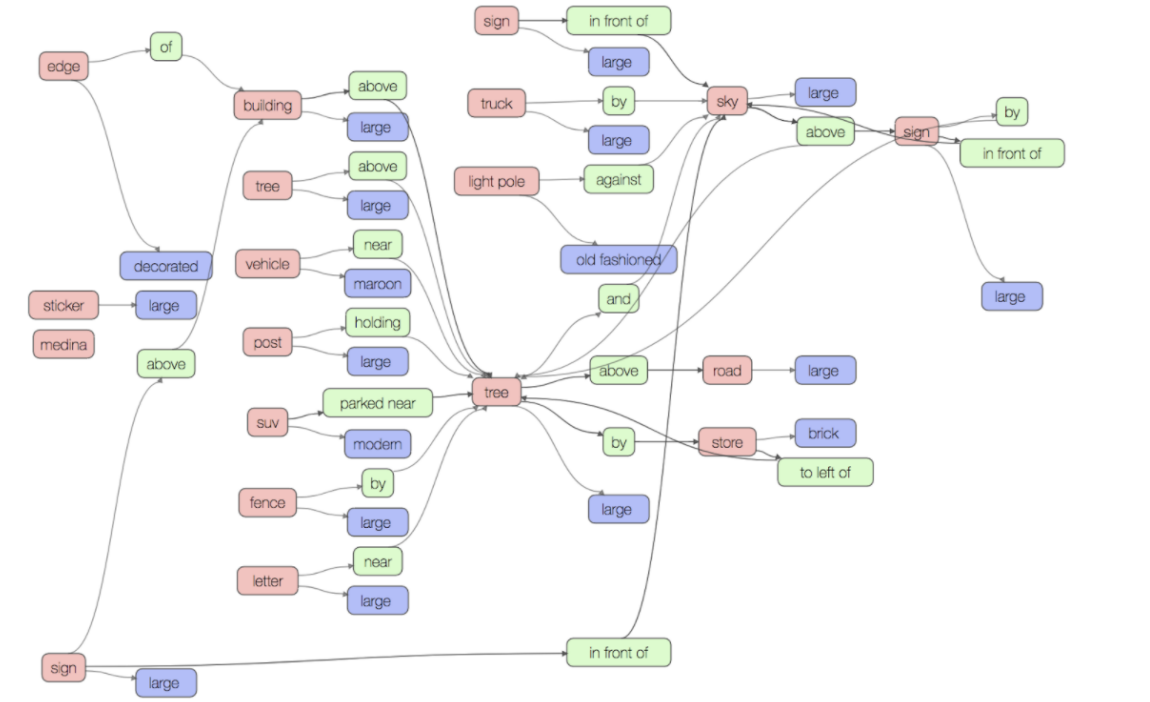

Figure 11. Example 1: Predicted scene graph

Figure 12. Example 2: Input image for scene graph visualization

the ground truth.

6.3. Limitations

Although the Visual Genome dataset consists of numerous objects and predicates, many of the images are missing annotations. This causes the dataset to have a large number of false negatives. This poorly affects the model during training because when we compute rewards, we assign a positive reward if the predicate or attribute exists in the ground truth graph and false positives will receive a negative reward. This is one of the main reasons our recall scores are so poor. Many of the graphs that VRL generates consist of attributes and relationships not present in the ground truth scene graph, but are valid attributes and relationships. This limitation shows the disadvantage in evaluating scene graphs using recall in general. However, in all previous papers, recall is the main metric used for evaluation. This is a reason we evaluated our graphs qualitatively by observing them, as well as using recall.

7. Conclusion

In this paper, given an image, we produce a scene graph, which is a graph that contains objects, attributes of objects and relationships between objects. We do this by using a reinforcement learning approach presented in [5]. We first cre-

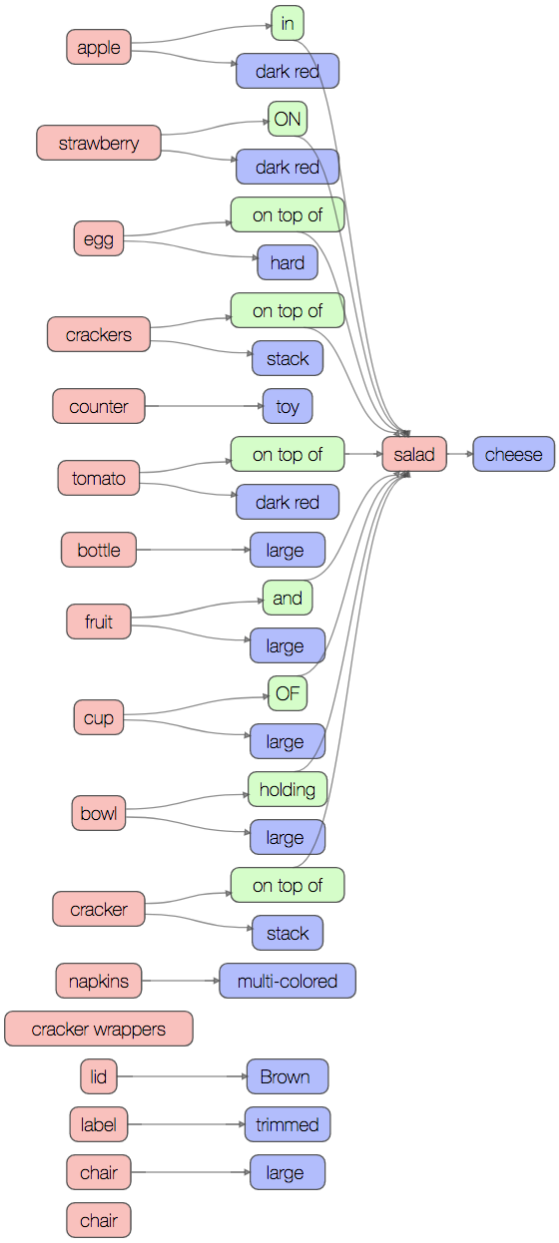

Figure 13. Example 2: Ground truth scene graph

ate a directed semantic action graph using all the images in the dataset. This captures all the objects, attributes and relationships in the dataset. Then we use an variation-structured traversal scheme to sequentially detect elements in the scene graph. The variation-structured traversal scheme helps us reduce the action space for our DQN model. Although the quantitative metrics of the model (such as recall) are not impressive, the qualitative metrics (such as visualizations of the scene graphs) highlight the shortcomings of the dataset and confirm the validity of the approach.

Through this project, we've learned the subtleties and challenges of reproducing a research paper, especially one that uses a deep RL approach. Even though we implemented the entire architecture as described in [5], we were unable to achieve good quantitative results. We learned that deep RL models are finicky to train and require extensive tuning. We also experiences the challenges of implementing the code from scratch and the challenges of working with a messy and incomplete dataset.

8. Future Work

We can adapt this architecture to include an active learning component. This would allow us to learn new objects and new predicates not in the initial dataset. We can create a more comprehensive semantic action graph using natural language sentences. The Visual Genome dataset was created by having people write paragraphs about images. Using these descriptions, we could potentially generate better ground truth labels for our model.

9. Contributions

Aarti and Apoorva both wrote the code (from scratch) to implement this model. We also both wrote the final report.

10. Appendix

10.1. Code

The code can be found here- https://github.com/nexusapoorvacus/DeepVariationStructuredRL. Everything was implemented from scratch (except for the skipthoughts code, which was taken from https://github.com/ryankiros/skip-thoughts/blob/master/skipthoughts.py).

References

[1] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[2] Newell, Alejandro, and Jia Deng. "Pixels to Graphs by Associative Embedding." [1706.07365] Pixels to Graphs by Associative Embedding, 22 June 2017, arxiv.org/abs/1706.07365.

[3] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.

[4] Huang, Gao and Liu, Zhuang and van der Maaten, Lau-

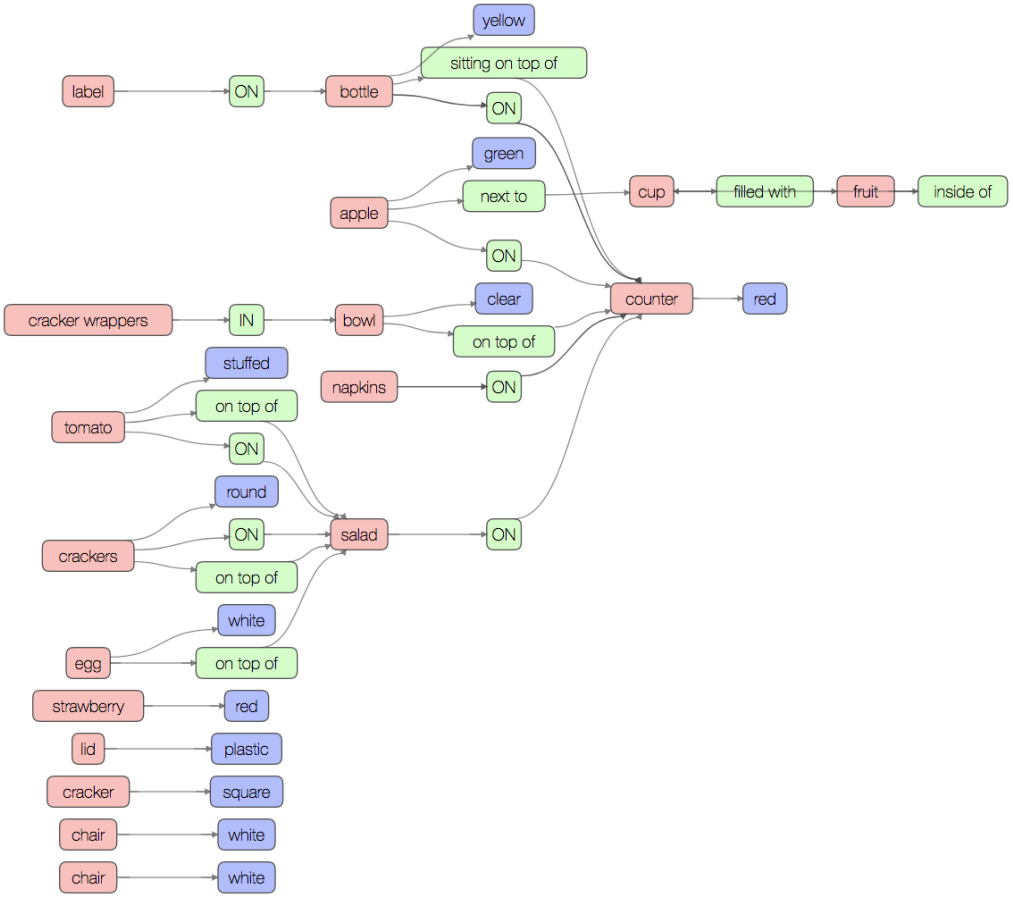

Figure 14. Example 2: Predicted scene graph

rens and Weinberger, Kilian Q, *Densely connected convolutional networks*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017

[5] X. Liang, L. Lee, and E. P. Xing. Deep variation-structured reinforcement learning for visual relationship and attribute detection. In CVPR, 2017

[6] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015

[7] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K., Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik,Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane, and Hassabis, Demis. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 02 2015

[8] Van Hasselt, Hado, Guez, Arthur, and Silver, David. Deep reinforcement learning with double q-learning. arXiv preprint arXiv:1509.06461, 2015.