The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Generating Triples with Adversarial Networks for Scene Graph Construction

Matthew Klawonn
Rensselaer Polytechnic Institute
Dept. of Computer Science
Troy, NY 12180
klawom@rpi.edu

Eric Heim
Air Force Research Laboratory
Information Directorate
Rome, NY 13441
eric.heim.1@us.af.mil

Abstract

Driven by successes in deep learning, computer vision research has begun to move beyond object detection and image classification to more sophisticated tasks like image captioning or visual question answering. Motivating such endeavors is the desire for models to capture not only objects present in an image, but more fine-grained aspects of a scene such as relationships between objects and their attributes. Scene graphs provide a formal construct for capturing these aspects of an image. Despite this, there have been only a few recent efforts to generate scene graphs from imagery. Previous works limit themselves to settings where bounding box information is available at train time and do not attempt to generate scene graphs with attributes. In this paper we propose a method, based on recent advancements in Generative Adversarial Networks, to overcome these deficiencies. We take the approach of first generating small subgraphs, each describing a single statement about a scene from a specific region of the input image chosen using an attention mechanism. By doing so, our method is able to produce portions of the scene graphs with attribute information without the need for bounding box labels. Then, the complete scene graph is constructed from these subgraphs. We show that our model improves upon prior work in scene graph generation on state-of-the-art data sets and accepted metrics. Further, we demonstrate that our model is capable of handling a larger vocabulary size than prior work has attempted.

Learning representations of visual scenes remains an important task that underlies many computer vision problems ranging from visual question answering (Malinowski and Fritz 2014) to image retrieval (Johnson et al. 2017). In order to be successful in these tasks, images must be represented in a form that captures details of the objects contained in a scene, including what objects are present, what attributes each object possesses, and how objects relate to one another. Much of the recent work on learning how to visually perceive images has focused largely on object detection and classification (He et al. 2016; Szegedy et al. 2017). These tasks focus on identifying one or more concepts or objects depicted in an image, but cannot produce representations that capture more complex characteristics or relationships of objects within scene. Such information may provide insight necessary to understand a scene.

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

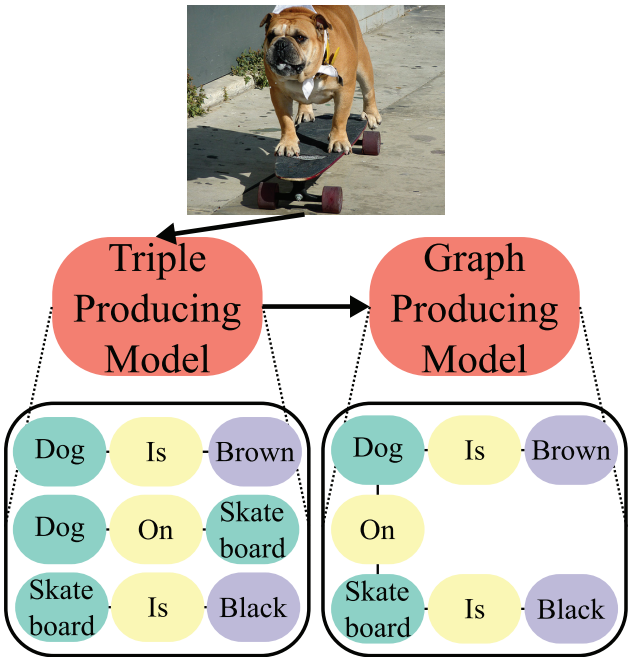


Figure 1: Depiction of our model. Given an image of a scene (top), our model (middle-left) generates triples (bottom-left) that express statements about objects in the image. Then, duplicate entities in the triples with the same lexeme are determined by examining if they are from similar regions of the image (middle-right). These duplicates are merged into single nodes to form a scene graph (bottom-right).

In this work, we focus on the task of learning to produce structured representations of images that express rich scene information. More specifically, our goal is to learn a model that is able to generate a *scene graph* (Johnson et al. 2017) given an image. A scene graph describes the content of a scene by representing objects within an image as nodes and relationships between objects as edges. By including nodes that correspond to visual properties, an object can be represented as having an attribute if an edge is drawn between said object and attribute. As such, scene graphs are naturally able to model not only what objects are in a scene, but how they relate to each other and what attributes they possess.

For a learned model to map an image to a scene graph, it must both have the capacity to represent high-dimensional inputs and the ability to produce complex, structured outputs. Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) have shown success for such learning problems. In the GAN framework a *generator* neural network is pitted against an adversary network called a *discriminator*. The network are trained in tandem, eventually reaching an equilibrium where the generator can produce output that is indistinguishable from real data by the discriminator.

While prior GAN models have almost exclusively focused on generating images, we propose a novel GAN that is able to generate plausible scene graphs for given images. Our generator produces individual statements about a scene called *triples*, which are three lexeme sequences that describe object relationships or attributes. Because the generator is trained to provide a variety of outputs given an image, our generator is able to produce many different triples, each putting forth a different statement about a scene. To combine the disparate triples into a proper scene graph we utilize an attention mechanism (Xu et al. 2015) that is trained within our generator to determine if two lexemes with the same label were produced from the same spatial region of the input. If they are, we merge the two lexemes in our scene graph (see: Figure 1).

In summary the contributions of this work are as follows:

1. We formulate a novel GAN model that is able to generate triples over a scene that contain both object relationship and attribute statements.
2. We use an attention mechanism both to generate parts of a scene graph from different parts of an image, and to merge these subgraphs together.
3. We demonstrate empirically that our model improves on the state of the art in scene graph generation while also being able to model object attributes.

The remainder of this paper is organized as follows. We first discuss previous, related work and compare our method to other approaches for scene graph generation. We then discuss our proposed approach, beginning with a formal definition of our learning problem. Next, we quantitatively evaluate our method against the state-of-the-art in scene graph generation, and qualitatively discuss illustrative examples of generated scene graphs. Finally, we conclude and discuss directions of future work.

Related Work

Work related to ours lies primarily in three areas: methods for learning representations of visual scenes, Reccurrent Neural Networks (RNNs), and GANs. Next, we review these areas in relation to our work.

Scene Understanding and Representation

A simple way to describe a scene is to list the objects it contains. This task is often called visual object recognition, a classification task for which numerous leaps in performance have been achieved over the last half-decade (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2015; He et al. 2016; Szegedy et al. 2017) on large-scale image data sets, such as ImageNet (Russakovsky et al. 2015).

More sophisticated approaches to describing a scene, like image captioning (Vinyals et al. 2015), dense captioning (Johnson, Karpathy, and Fei-Fei 2016), and visual question answering (VQA) (Malinowski and Fritz 2014) provide more expressive means of capturing the details of an image than simply listing objects. Each task involves producing natural language output given an image as input. While these representations have been shown to be helpful for various tasks (e.g image retrieval), the outputs can be interpreted in many different ways due to the ambiguity of natural language. Further, these tasks do not explicitly attempt to describe the entirety of a scene or link information from various parts of a scene together.

One way to limit the ambiguity of natural language and explicitly model object relationships is to describe scenes using scene graphs, first proposed by (Johnson et al. 2017). Scene graphs have been shown to be useful in a number of tasks like content based image retrieval (Johnson et al. 2017) and automatic caption evaluation (Anderson et al. 2016). In one of the first works in generating scene graphs, the authors of (Anderson et al. 2016) propose a method to generate scene graphs from captions. More recently, the authors of (Xu et al. 2017) propose a method to construct a scene graph from an image by first fixing the structure of the graph, then refining node and edge labels using iterative message passing. Their work limits itself to training with the use of bounding boxes and doesn't predict attributes, likely because multiple attributes may share a bounding box with a single object, and learning to generate multiple attributes given a single bounding box is non-trivial. In contrast, our method learns to generate individual triples, using the GAN training framework coupled with attention to focus on regions. As such, our model does not require bounding box labels to train and is not limited to relating two separate objects that were labeled in the image, thus enabling attribute information to be generated.

Recurrent Neural Networks

Our architecture contains a recurrent language component to produce triples. More specifically, we utilize Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) networks with attention. The idea of an attention mechanism, first proposed in (Bahdanau, Cho, and Bengio 2014), is to allow a recurrent network access to the entire input at every timestep, with the attention mechanism determining what parts are important. Typically, an attention mechanism is simply a multilayer perceptron jointly trained with the rest of the recurrent architecture. (Xu et al. 2015) are the first to use an attention mechanism in an LSTM with an image as input, with many others following (Yang et al. 2016; Shih, Singh, and Hoiem 2016; Xu and Saenko 2016; Lu et al. 2016). Importantly, we explicitly use the attention mechanism to disambiguate entities in produced triples, a technique that to our knowledge has not been applied in prior work. We use LSTMs because they are historically one of the most successful RNNs, and offer comparatively similar performance to more recent variants, such as GRUs (Chung et al. 2014).

Generative Adversarial Networks

Generative Adversarial Networks have generated interest since their inception in (Goodfellow et al. 2014) because of their empirical successes in modeling high dimensional data distributions. Recently, (Arjovsky and Bottou 2017) and (Arjovsky, Chintala, and Bottou 2017) illustrated a number of theoretical and practical issues with the original GAN formulation, and proposed to train the discriminator to measure the difference between generated and ground truth samples with a Wasserstein metric rather than a Kullback-Leibler divergence, accomplished by a change in loss functions. In order to generate sequences using one hot vector ground truth examples and continuous vector based generated outputs, this change is necessary. As explained in (Gulrajani et al. 2017), the KL divergence between two one hot vectors is infinite resulting in useless gradients.

While applications of GANs have largely been limited to generating images (Radford, Metz, and Chintala 2016; Ledig et al. 2017; Reed et al. 2016), there have recently been a few efforts to generate natural language. Such endeavors use training methods other than backpropagation (Yu et al. 2017), or a continuous approximation to sampling from a categorical distribution (Jang, Gu, and Poole 2017; Maddison, Mnih, and Teh 2017). We leave the training of scene graph generation models with these approaches for future work. There have also been a number of works in using GANs with an input condition, e.g generating images conditioned on text (Reed et al. 2016; Zhang et al. 2017) or other images (Ledig et al. 2017). Like previous work, we condition our GAN on images; unlike previous work, our output is triples.

Generating Scene Graphs from Imagery

In this section, we formalize the problem of generating a scene graph and discuss the motivation and specifics of our approach. A unique characteristic of our model is that scene graphs can be produced by generating their subunits and stitching them together. As such, we outline architectures and training approaches suitable for this formulation of the problem, specifically motivating the use of attention mechanisms and GANs. Finally, we provide implementation details.

Problem Formulation

The goal of this work is to find a mapping between an image of a scene and a scene graph. Specifically we propose to learn a mapping $g_\Theta : I \rightarrow G$ from a color image $I \subset \mathbb{R}^{d_1 \times d_2 \times 3}$ to a scene graph $G = (V, E)$. Each vertex $v \in V$ and edge $e = (v_1^e, v_2^e)$ is labeled by a lexeme in a vocabulary $\mathcal{V}$.

Each edge in a scene graph defines a single statement about a scene called a *triple*. Let l_e be the label for edge e and l_v be the edge for vertex v. Every edge $e = (v_1, v_2) \in E$ can be formatted as a triple $t_e = (l_{v_1}, l_e, l_{v_2})$. We call these triples because they are similar in spirit to Resource Description Framework (RDF) triples (Lassila and Swick 1999), which are three entities representing some statement about data in the form of a subject-predicate-object structure. In the scene graphs considered in this work, triples either describe *relations* between two objects in the scene (e.g. t_e = ("dog","on","skateboard") in Figure 1), or state that an object has an *attribute* (Ferrari and Zisserman 2008), a mid-level visual concept (e.g. t_e = ("dog", "is", "brown") in Figure 1).

With these definitions, two things are clear. Individual triples are capable of describing the contents of a scene, while arranging them into a graph resolves duplicate entities into a single vertex. Seeing that these two steps can be separated, we map images to scene graphs by first generating triples given an image, and then resolve objects that are the same to construct a proper scene graph. We hypothesize that by focusing on the quality of generated statements, saving the structuring of the graph for later, we can more accurately predict components of a scene graph. More formally, we choose to first find a mapping $g'_\Theta : I \mapsto \mathcal{V} \times \mathcal{V} \times \mathcal{V}$ to find triples over a scene, and then a mapping $g'' : \{\mathcal{V} \times \mathcal{V} \times \mathcal{V}\}^n \mapsto G$ to then produce a scene graph from the triples. First we discuss our method for learning g'_{Θ_g}.

Triple Generation Network

For the generation of triples, we propose the a neural network architecture based on ideas from Convolutional Neural Networks (CNNs), RNNs, and GANS. The network has two components, the first feeding into the second. The first is a *feature extractor* $f_{\Theta_f} : I \mapsto I'$ that maps images to visual features using a fully convolutional neural network. There are L total features produced, and each belongs to $\mathbb{R}^D$, with D corresponding to the number of convolutional filters in f_{Θ_f}. These features are fed into a *recurrent* component $r_{\Theta_r} : I' \mapsto \mathcal{V} \times \mathcal{V} \times \mathcal{V}$. More specifically, we utilize a Long Short Term Network (LSTM) that outputs a lexeme from the vocabulary for each of three time steps. Given visual features $\mathbf{X}'_t \in I'$, an LSTM unit at time t is defined by the following:

$$
\begin{aligned}
\vec{f}_t &= \sigma\left(W_f \mathbf{X}'_t + U_f \vec{h}_{t-1} + b_f\right) \\
\vec{i}_t &= \sigma\left(W_i \mathbf{X}'_t + U_i \vec{h}_{t-1} + b_i\right) \\
\vec{o}_t &= \sigma\left(W_o \mathbf{X}'_t + U_o \vec{h}_{t-1} + b_o\right) \\
\vec{c}_t &= f_t \circ \vec{c}_{t-1} + \vec{i}_t \circ tanh\left(W_c \mathbf{X}'_t + U_c \vec{h}_{t-1} + b_c\right) \\
\vec{h}_t &= o_t \circ tanh(\vec{c}_t)
\end{aligned}
\tag{1}
$$

The hidden state c_t is used to aggregate information from the previous step's LSTM output h_{t-1} and the input $\mathbf{X}'_t$. The vectors $\vec{f}$, $\vec{i}$, and $\vec{o}$ are *gates* that determine what information from $\mathbf{X}'_t$ (visual features), $\vec{h}_{t-1}$ (output of previous time step), and $\vec{c}_{t-1}$ (previous hidden state) is and is not used to determine the current hidden state $\vec{c}_t$. Intuitively, these gates are attempting to explicitly learn what to "remember" and what to "forget" so that long-term dependencies between inputs and outputs can be modeled. The vector $\vec{h}_t$ is the output produced by the LSTM at step t based on $\vec{c}_t$. In order to map $\vec{h}_t$ to lexemes in our vocabulary we learn an additional affine transformation of $\vec{h}_t$ to produce a $|\mathcal{V}|$ length vector, where each element corresponds to a lexeme in $\mathcal{V}$. This vector is then normalized so the elements sum to one, and the index of the highest value can be used index the vocabulary to finally output a human readable label. This is done at all

three time steps of the LSTM to create a triple. Note, as currently defined, the sole input to the recurrent component is the output of the convolutional component. As such, only one triple will be generated per input image. Because a scene can contain many different objects with many different relations and attributes, we want our generator to model a distribution over triples, given an image. Motivated by recent successes in learning generative distributions over structured data using GANs, we train our generator using an adversarial strategy that enables us to sample a variety of triples to use as a basis for scene graph construction.

Adversarially Training the Triple Generator

To train g'_{Θ_g} we pit it against a discriminator $d_{\Theta_d} : (V \times V \times V) \times I' \mapsto [0, 1]$, where $V = [0, 1]^{|\mathcal{V}|}$. The goal of d_{Θ_d} is to discriminate ground truth triples, encoded as three one-hot vectors in a training set, from three outputs of g'_Θ, given an image. The generator and the discriminator are trained in tandem, in an adversarial game. The recurrent component of our generator takes visual features of an image $\mathbf{X}'$ concatenated with a random vector $\vec{n} \sim \mathcal{N}(0, \mathbf{I})$, and produces a triple $\tilde{t}_e = (v_1, v_2, v_3)$. The random noise portion of the input ensures g'_Θ does not deterministically produce a single unique triple given $\mathbf{X}'$, and can instead produce a variety of triples. Concatenation of the $\mathbf{X}'$ onto $\vec{n}$ can be viewed as conditioning if the generator is viewed as a probability distribution. We adopt notation to reflect this view that g'_Θ and d_{Θ_d} are probability distributions conditioned on an image. From the training set, a ground truth triple t_e associated with $\mathbf{X}'$ is sampled. The discriminator is trained to output a low value when given $(\tilde{t}_e \times \mathbf{X}')$, and a high value when given $(t_e \times \mathbf{X}')$, separating the real triples from generated ones as much as its architecture allows. Training is performed end to end using stochastic gradient descent with backpropagation. As a result error information from the discriminator is propagated to the generator, informing it how to generate triples similar to those in the training set.

In order for d_{Θ_d} to propagate error to g'_{Θ_g}, it requires an appropriate loss function. Towards this end, we use the Wasserstien metric to measure the distance between the data and generator distributions, a technique first introduced in (Arjovsky, Chintala, and Bottou 2017). This metric induces the following loss function:

$$\mathcal{L}(\mathbf{X}', \vec{n}; \Theta_g, \Theta_d) = d_{\Theta_d}\left(g'_{\Theta_g}(\vec{n}|\mathbf{X}')|\mathbf{X}'\right) - d_{\Theta_d}(t_e|\mathbf{X}') \tag{2}$$

Minimizing this loss with respect to Θ_d, allows the d_{Θ_d} to better discriminate samples from $g_{\Theta'_g}$ from those from the ground truth. Maximizing the loss with respect to Θ_g allows $g_{\Theta'_g}$ to produce samples that "fool" the discriminator into producing high scalar valued scores for generated triples, indicating the samples look like ground truth samples. As such, our model is trained in two phases. First, d_{Θ_d} is updated by sampling $\mathbf{X}'$ and $\vec{n}$ and performing a stochastic gradient minimization update with respect to Θ_d. Then, g'_{Θ_g} is updated by sampling $\mathbf{X}'$ and $\vec{n}$ and performing a stochastic gradient maximization update with respect to Θ_g.

Because our training algorithm promotes sampling of different triples from a single image, we do not require human or machine provided bounding boxes, a significant benefit of our approach over prior work. Since d'_{Θ_d} alone informs our generator, the task of g'_{Θ_g} is not to predict a correct output given a bounding box offset as it is in (Xu et al. 2017), but rather to predict an output that the discriminator will score highly. While this eliminates the need for bounding box information in generating triples, our method requires g'' to construct a proper scene graph. The challenge in this task is that separate triples may contain lexemes that refer to the same object. This ambiguity must be resolved in order to successfully construct a scene graph from generated triples. Our approach to resolve lexemes is to associate a separate spatial region of the input with each generated lexeme. We do this using an attention mechanism that produces a vector that quantifies which regions of the image most influenced the generation of the lexeme. If two of the of same generated lexemes have similar attention vectors, we can determine that they are the same object in the scene. We outline this process further in the next section.

Graph Construction from Attention

An attention mechanism is a differentiable function of trainable parameters that accepts as input some collection of features and outputs a relative importance of said features. We utilize the mechanism introduced in (Xu et al. 2015) in the recurrent component of our generator. This mechanism is defined as follows:

$$\begin{aligned} e_{ti} &= a_{\Theta_a}(\vec{x}'_i, \vec{h}_{t-1}) \\ \alpha_{ti} &= \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})} \\ \vec{z}_t &= \sum_{i=1}^{L} \alpha_{ti} \vec{x}'_i \end{aligned} \tag{3}$$

Here, a_{Θ_a} is a multilayer perceptron that accepts a concatenation of the input convolutional features $\vec{x}'_i$ and the hidden state of the previous step $\vec{h}_{t-1}$ in our recurrent component, and produces a vector $\vec{e}_t$ for which each element is a relative importance of each visual feature in $\mathbf{X}'_t$. This vector is then normalized with a softmax to form $\vec{\alpha}_t$. Finally, the elements of $\vec{\alpha}_t$ are used to weigh the L visual features to produce a vector $\vec{z}_t \in \mathbb{R}^D$ which is used as input to our recurrent component r_{Θ_r} instead of the using all L visual features in $\mathbf{X}'_t$.

Because $\vec{\alpha}_t$ gives a relative importance of the visual features input to r_{Θ_r} at time t, and because these visual features are the product of a fully convolutional neural network f_{Θ_f}, $\vec{\alpha}_t$ gives an explicit weighting of the spatial regions in the image used to generate a particular output. As a result each of the three lexemes in each triple generated by our architecture can be mapped to a region or regions of the input image. Thus, our recurrent component generates lexemes based on specific, emphasized regions of the image.

Note that at the first step $t = 1$ of LSTM calculation there is no value for $\vec{h}_{t-1}$ ($\vec{h}_0$), so in (Xu et al. 2015) it is

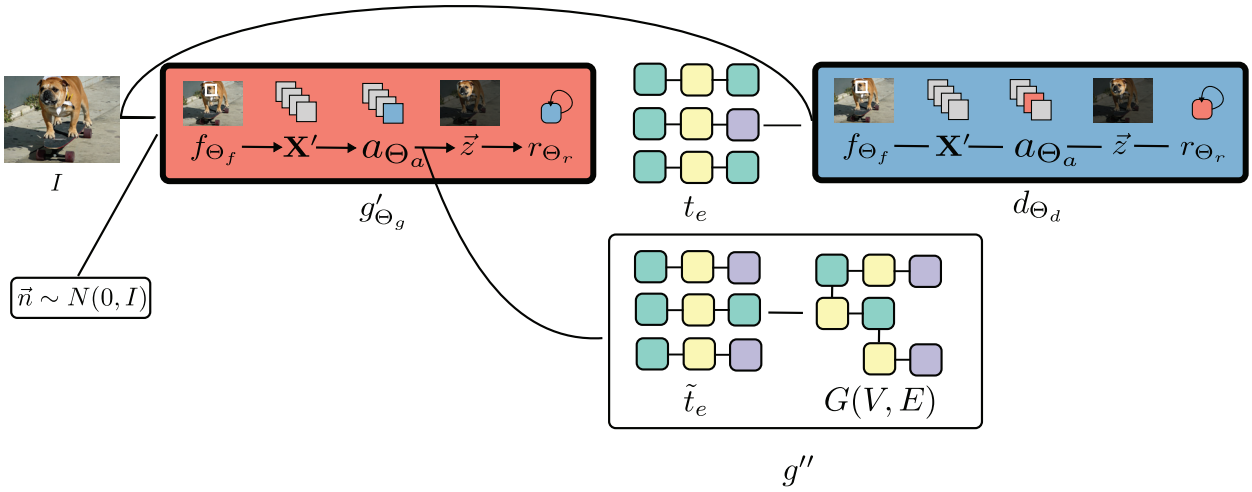


Figure 2: Our architecture: Input images I and noise $\vec{n}$ are first fed to the generator g_{Θ_g} (red), which processes the image using a CNN f_{Θ_f}, generates image features $\mathbf{X}'$, passes those to an attention mechanism a_{Θ_a} that generates a dynamic image representation $\vec{z}$ and attention vector $\vec{\alpha}$. $\vec{z}$ is fed to an LSTM that produces triples $\tilde{t}_e$. During training, these triples are passed along with ground truth triples t_e to the discriminator d_{Θ_d} (blue) that contains the same components as the generator. The discriminator only produces a score, however. During test time all $\tilde{t}_e$s and αs are passed to g'' which resolves the triples into a graph $G(V, E)$.

predicted as a learned affine transformation of the average of all L features, the idea being to give the an "overview" of the content of the image to the attention mechanism. In contrast we would like to initialize this value such that the attention mechanism is conditioned on a random portion of the image to start rather than the whole image, increasing the variety of triples that can be produced. Thus we map a random $\vec{x}'_i \in \mathbb{R}^D$ to h_0 via a learned affine transformation.

In order to resolve lexemes across triples, i.e to create our mapping g''_Θ, we compare the $\vec{\alpha}_t$ vectors of lexemes with the same label using generalized Intersection over Union (IoU):

$$\text{IoU}(\vec{x}, \vec{y}) = \frac{\sum_i \min \vec{x}_i \vec{y}_i}{\sum_i \max \vec{x}_i \vec{y}_i} \tag{4}$$

If their similarity is over a threshold, then the two lexemes can be deemed one and the same. This use of the attention vectors produced for each output is, to the best of our knowledge, a novel application of said vectors.

We now have all the components necessary to train our generator g'_{Θ_g} and then construct scene graphs via g''. Our full architecture flow to produce one triple $\tilde{t}_e$ is as follows. First, we use $f_{\Theta_f} : I \mapsto I'$ to extract convolutional features $\mathbf{X}' \in \mathbb{R}^{D \times L}$, which are subsequently fed into our attention mechanism a_{Θ_a}. The attention mechanism produces vectors $\vec{z}, \vec{\alpha} \in \mathbb{R}^D$. $\vec{z}$ is concatenated with $\vec{n}$ and fed to our LSTM which produces an output $\vec{h}_t$, then transformed via affine parameters to generate a single lexeme $\vec{v}_t = \left(W_v \vec{h}_t + b_v\right)$. We repeat this process three times to generate $\vec{v}_1$, $\vec{v}_2$, $\vec{v}_3$, resolving these lexemes to their labels to produce the triple $\tilde{t}_e = (l_{v_1}, l_{v_2}, l_{v_3})$. A number of triples are then combined via their associated attention vectors $\vec{\alpha}$ to form a scene graph proper. With our full architecture outlined here and in Figure 2, we now provide an empirical evaluation and details on our implementation.

Empirical Evaluation

One goal of our evaluation is to compare our method to the current state-of-the-art in scene graph generation. As (Xu et al. 2017) sets the current state-of-the-art, we compare to their method, using metrics their work established, and on the dataset they evaluated on. All data comes from the Visual Genome (VG) dataset (Krishna et al. 2016), since this is the largest and highest quality dataset containing image-scene graph pairs available today and the same data that (Xu et al. 2017) use for evaluation. In addition, to demonstrate our model's ability to generate attributes in addition to relations, we show the performance of our model trained on an extended vocabulary that contains attribute triples.

Experimental Set-up

For the feature extractor component f_{Θ_f}, we use the convolutional layers of (Simonyan and Zisserman 2015) to produce input features, which are standardized. f_{Θ_f} is not trained with the rest of the architecture, but rather pre-trained on the image classification task of (Russakovsky et al. 2015). Following the example of (Gulrajani et al. 2017), we use the Adam stochastic gradient algorithm (Kingma and Ba 2015) with learning rate $1e{-}4$, $\beta_1 = 0.5$, $\beta_2 = 0.9$ to train both the discriminator and generator. Our gradient penalty coefficient λ (Gulrajani et al. 2017) is set to 10, and no hyperparameter optimization is done. Each layer of our architecture uses layer normalization (Ba, Kiros, and Hinton 2016) to help avoid saturating LSTM non-linearities. For our graph construction phase, entities that have more than a 80% match using the

generalized IoU metric are considered to be duplicate entities. In the cases where more than one label appears for these entities, we use majority voting to determine a single label. We note that this threshold could be tuned on a validation set by some metric, e.g the recall @ k metric discussed in the next section.

VG contains over 108,000 image/scene graph pairs. We create two splits of these pairs. The first split exactly matches that of (Xu et al. 2017), which is a 70-30 train-test split of the dataset capturing the top 150 object classes and 50 relations for a vocabulary size of 200. The second split is also a 70-30 train-test split, but includes the 400 most common object classes, 150 most common relations, and 150 most common attributes. The first split allows us to compare our model directly to the results reported in (Xu et al. 2017). The second not only expands the vocabulary to a larger number of relations, but also includes attributes, enabling evaluation of our model on a larger variety of possible triples and more complex scene graphs.

In (Xu et al. 2017), there are three separate tasks that are used to evaluate their models: predicate classification, scene graph classification, and scene graph generation. In predicate classification, the task is to predict the correct predicate for a relation triple with the two objects being given, along with their bounding boxes. In scene graph classification, the task is to predict object labels and a predicate given a set of bounding boxes. In this work we only examine the task of scene graph generation as we do not use bounding box information, making the tasks of predicate and scene graph classification largely irrelevant for measuring our methods performance.

The metric used in (Xu et al. 2017), recall at k, is defined as the fraction of the k most likely generated triples that appear in the ground truth, relative to the total number of triples in the ground truth. For a set of ground truth triples t_e and generated triples $\tilde{t_e}$ we formally define this in (5):

$$r@k = \frac{|\tilde{t_e} \cap t_e|}{|t_e|} \tag{5}$$

The idea behind this metric is that models should not be penalized for making true predictions that don't happen to be in the ground truth data. Since labels are sparse and very few (if any) scenes are labeled with all object relationships and attributes, such a penalty would significantly cloud interpretation of results. When reporting results, we multiply the fraction given by $r@k$ by 100 to create a percentage.

Here we present our qualitative and quantitative results of our evaluation. In order to filter predictions to the top k, we generate a large sample of $\sim$ 500 triples, and use the discriminator to score each of the generated triples and rank by score from highest to lowest, with a high score indicating the discriminator thinks the triple is likely to come from the ground truth distribution. We refer to our method as SG-GAN, an abbreviation for Scene Graph GAN.

Results

Table 1 shows the comparison between our model and the state of the art reported in (Xu et al. 2017). On this form of the VG data, our method achieves approximately double the recall@50 and recall@100 as (Xu et al. 2017). Again, we reiterate, this performance was achieved without the need for bounding box labels. For our custom split of the VG data we observe a reduction in performance (Table 2), when switching to the model trained and evaluated on our custom split of both relations and attributes.

Table 1: Recall @ 50 and 100 for the task of scene graph generation on the split of (Xu et al. 2017). We compare only to their best results and nearly double the performance in both cases.

Metric	SG-GAN	(Xu et al. 2017)
r @ 50	6.84	3.44
r @ 100	8.95	4.24

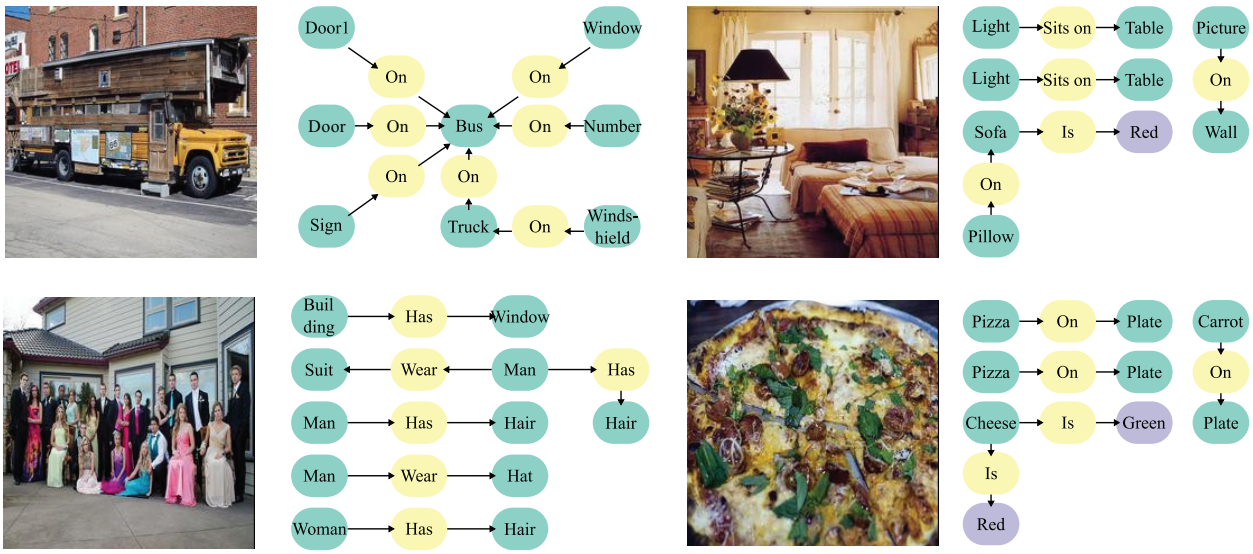
Table 2: Recall @ 50 and 100 for the task of scene graph generation on our own split of both attributes and relations. There is no prior work in generating both using the same architecture and so we make no comparisons.

Metric	SG-GAN	(Xu et al. 2017)
r @ 50	1.74	–
r @ 100	2.47	–

We speculate that the inclusion of attributes makes for a more difficult learning task than the increase in vocabulary alone for a few reasons. The visual cues necessary to detect relations and those necessary to detect attributes vary significantly, increasing the burden on any architecture attempting to capture both at the same time. Another reason is that the attention mechanism is likely to behave very differently depending on whether or not it is generating a relation vs an attribute. For generating an attribute one would expect the attention mechanism to look in the very near proximity of the object in contrast to all over the image. While neural networks have a very large capacity and ought to be able to "remember" how to treat the two differently, we suspect the stark difference between looking in a different location for each lexeme produced and looking in the same location for each lexeme produced is still challenging.

While the performance gain is significant over prior work, the recall of our model is low relative to related visual inference tasks. This is somewhat expected, given that scene graph generation combines a number of these tasks as subproblems such as object detection, object/object relationship identification, and attribute prediction. As a result, we note that the task of scene graph generation remains a challenging problem which likely presents an opportunity for future work to improve on our results.

In Figure 3 we show examples of scene graphs that we generated using our relations only model and our relations-attributes model respectively. For interpretability we limit the triples generated to the top 10 (or below if duplicate entities are resolved) as determined by our discriminators. The results are not out of line with how we train our model. Many of the triples that exist in the graphs generated by our



a) Relations Only

b) With Attributes

Figure 3: Example scene graphs generated by SG-GAN. a) These scene graphs were generated by the model trained on the relations only data split. b) These were generated by the model trained on both relations and attributes.

method are either true or close to being true. In particular object detection seems to work fairly well, which is also unsurprising given that (Simonyan and Zisserman 2015), the pretrained model that provides our convolutional features, is trained on an object detection task. In the qualitative results, like the quantitative results, there is a performance drop when increasing vocabulary size and including attributes.

Conclusion and Future Work

In this paper we proposed a novel technique for learning to represent an image via a scene graph. Our approach is based on generating individual subgraphs called triples, and exploiting an attention mechanism to stitch triples together into a proper scene graph. We hypothesized that by focusing on generating individual true statements about a scene in the form of triples, we can create better triples and thus better scene graphs. After triples are generated, we made explicit use of vectors from an attention mechanism in our generator to determine which lexemes in the triples refer to the same object, yielding a method that combines lexemes into a single node in our scene graph. To our knowledge this is a novel use of attention in neural networks. By learning our model using an adversarial training technique, our method can be trained to generate object attributes in addition to object relations, which prior work did not do, and without bounding box information, which prior work required. In an empirical evaluation, we illustrate that our method for generating scene graphs outperforms the current state-of-the-art, achieving double the recall@50 and recall@100.

One potential experiment that can be explored in future work is the insertion of various architectures into the GAN framework. For example, while our convolutional features are fixed and extracted from a CNN trained to recognize images, training a convolutional component of the network jointly with our LSTM may improve performance as the tasks of image recognition and scene graph generation are sufficiently different. Another line of research includes exploring other structured prediction problems given a complex input (like an image) using GANs. For example, while we argued for the adoption of scene graphs over image captions as image representations, image captioning has proven to be useful and could potentially benefit from an adversarial training approach. There are also unexplored lines of research in working with scene graphs themselves. Structured knowledge sources have a number of applications that have not yet found their way to using scene graphs as a source, e.g faceted search. Lastly we suggest that scene graph provide a means of linking information in an image with information from data of other modalities, for example text. Since many information extraction techniques for text produce graph structured results, they may be compatible with scene graphs.

Acknowledgements

Matt Klawonn is supported by an ASEE SMART scholarship. The authors would like to thank professor James Hendler for his thoughtful comments and discussion.

References

Anderson, P.; Fernando, B.; Johnson, M.; and Gould, S. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*.

Arjovsky, M., and Bottou, L. 2017. Towards principled methods for training generative adversarial networks. In *International Conference for Learning Representations*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. In *Neural Information Processing Systems*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. *Empirical evaluation of gated recurrent neural networks on sequence modeling*.

Ferrari, V., and Zisserman, A. 2008. Learning visual attributes. In *Neural Information Processing Systems*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Neural Information Processing Systems*.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. 2017. Improved training of wasserstein gans. In *Neural Information Processing Systems*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Johnson, J.; Krishna, R.; Stark, M.; Li, L.-J.; Shamma, D.; Bernstein, M.; and Fei-Fei, L. 2017. Image retrieval using scene graphs. In *Computer Vision and Pattern Recognition*.

Johnson, J.; Karpathy, A.; and Fei-Fei, L. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *Computer Vision and Pattern Recognition*.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization.

Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.-J.; Shamma, D. A.; et al. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. In *Computer Vision and Pattern Recognition*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*.

Lassila, O., and Swick, R. R. 1999. Resource description framework (rdf) model and syntax specification.

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network.

Lu, J.; Yang, J.; Batra, D.; and Parikh, D. 2016. Hierarchical question-image co-attention for visual question answering. In *Neural Information Processing Systems*.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations*.

Malinowski, M., and Fritz, M. 2014. Towards a visual turing challenge. In *NIPS 2014 Workshop on Learning Semantics*.

Radford, A.; Metz, L.; and Chintala, S. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks.

Reed, S.; Akata, Z.; Yan, X.; Logeswaran, L.; Schiele, B.; and Lee, H. 2016. Generative adversarial text-to-image synthesis. In *International Conference on Machine Learning*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

Shih, K. J.; Singh, S.; and Hoiem, D. 2016. Where to look: Focus regions for visual question answering. In *Computer Vision and Pattern Recognition*.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Association for the Advancement of Artificial Intelligence*.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*.

Xu, H., and Saenko, K. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*.

Xu, D.; Zhu, Y.; Choy, C.; and Fei-Fei, L. 2017. Scene graph generation by iterative message passing. In *Computer Vision and Pattern Recognition*.

Yang, Z.; He, X.; Gao, J.; Deng, L.; and Smola, A. 2016. Stacked attention networks for image question answering. In *Computer Vision and Pattern Recognition*.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Association for the Advancement of Artificial Intelligence*, 2852–2858.

Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; and Metaxas, D. 2017. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *International Conference on Computer Vision*.