1. Introduction

Visual Question Answering (VQA) is the task of answering natural language queries based on images (Figure 1 shows an example). VQA is a challenging multi-modal problem that requires a combination of scene understanding, natural language parsing, and logical reasoning. The problem has been extensively studied, especially in the last couple of years, and many methods have been proposed (Xiong et al., 2016; Andreas et al., 2016b; Fukui et al., 2016; Lu et al., 2016; Yang et al., 2016). However, as argued by (Johnson et al., 2016), many of these models do not possess actual reasoning abilities, and simply exploit statistical properties of the datasets. For example, questions about counting might always have some small number as the answer, and the model might simply predict this number without actually counting. Johnson et al. (2016) seek to address these limitations by proposing a new dataset, CLEVR, which reduces easily exploitable biases, and also provides ground truth annotations to allow easier analysis of the reasoning capabilities of models.

CLEVR dataset consists of massive synthetic question answering pairs with images of pre-determined objects. The questions are synthesized in a way that a model needs to apply compositional reasoning over them to answer correctly. One recent attempt to VQA with compositional reasoning is the improved version of Neural Module Networks (Andreas et al., 2016a), which learns to compose an optimal tree structure of reasoning by introducing “modules”. The construction of such trees is carried out roughly via two steps: 1) extract phrases from a question sentence to instantiate each module for the tree, 2) generate candidate structures and rank them to obtain the best structure. However, the way they generate trees is not a sequential composition, and their tree candidate evaluation only depends on the textual inputs. These points contradict to how humans process those information; we humans do not have large sets of possible reasoning structures and determine what is the best out of them, we rather have one policy that sequentially builds a reasoning structure as we both read and look at the text and images.

In this work, we attempt to address two challenges mentioned above: learning to compose reasoning steps online, and incorporating multi-modal inputs to influence the composition. To achieve these, we propose two models that take the above points into account. We use both CLEVR and VQA datasets, to not only evaluate the models’ capability of compositional reasoning but also their robustness against real-world objects.

2. Related Work

There have been several attempts to model reasoning ability by introducing memory or attention mechanism. For example, Xiong et al. (2016) applied a dynamic memory network architecture over the extracted visual features to store context information. The approach processes a image as a sequence of image fragments and learns representations that take into account past and future contexts as “facts”. Then, they learned a memory network that operates over another recurrent neural network which models memory updates over time. Despite the success in achieving high-
est scores, the detailed analysis of the memory networks to reveal interpretability. Particularly, although the original memory network is said to be able to perform multi-hop reasoning, such observation is not reported in the application of VQA, so the challenge of building a model that actually “reasons” over facts remains unsolved.

More recently, stacked attention networks approach was proposed by (Yang et al., 2016) in which the authors use VGGNet (Simonyan & Zisserman, 2014) to extract the image features from the last pooling layer of the model. In order to represent text questions by a feature vector, they experiment with both LSTM and CNN (Kim, 2014) based representation learning techniques. In case of LSTM, they use a 500 dimensional hidden unit vector and for CNN, they experiment with various unigram, bigram and trigram convolution filter size. Image and text features are combined using multiple attention layers (Bahdanau et al., 2014) which predicts answer from selective image parts in response to the query question. They evaluate the performance of their model on COCO-QA and VQA datasets and achieve close to new state of the art results.

Finally, neural module networks are applied to VQA to explicitly learn to compose a model (Andreas et al., 2016a). The authors construct candidate trees of “neural modules”, where each module corresponds to a fundamental building block for reasoning, such as “Find” or “Describe”. They learned a model that assembles those blocks into the optimal tree structure for each question by ranking candidates and achieved state-of-the-art for VQA. Although this approach might be claimed to be “learning to compose”, picking one from many tree candidates is not only inefficient but counterintuitive. Their method is incapable of actually constructing the tree by composing building blocks online. Moreover, their tree candidate generations is fully question-based, and not taking image features into account.

As argued by (Agrawal et al., 2016; Johnson et al., 2016), current VQA models often overfit and cannot adapt to novel scenarios, answer questions with just partial information about the question, or “do not change answers across images”. Some of these issues are caused by biases inherent in previously available datasets, which are exploited by models. The newly proposed CLEVR dataset does address some of these issues, however designing models that can show superior reasoning capabilities will also require architectural changes.

Towards structured understanding of visual scenes, Raposo et al. (2017) proposed a general framework called relation networks. The objective of this framework is to learn a relation between objects, under an assumption that the relations are permutation invariant for tractability. They constructed a simple dataset that consists of feature vectors as the input, and showed the learned model’s competence over a MLP in different task specifications including simple scene classifications and one-shot relation learning. Although this work provides an insight of how to approach relation learning in a general sense, the model operates over a toy dataset and does not show practical benefit yet. Nevertheless, it is worth noting here to motivate the problem with a recent attempt to “understand” visual scenes in terms of relations. We propose two different approaches for VQA tasks; deep learning based feature extraction approach, and compositional hard attentions with reinforcement learning.

3. Model 1: Deep Feature Extraction Model

In the first approach we use a CNN-based encoder to extract the features from images and commonly used sentence encoder such as LSTM, RNN with attention in order to represent a question. We will describe these encoders in detail later. After the image and question features are extracted, we concatenate their features together and pass them through a deep Multi Layer Perceptron (MLP) layer followed by a softmax classifier over all the possible output labels (expected answers). A schematic diagram of these steps is shown in 2.

3.1. CNN-based Image Features

In the case of Visual Question answering tasks, the context on which the question is based is an image. We use Convolutional Neural Network (CNN) in order to extract the features from images. We use a CNN model trained on the task of image classification and not specifically on the
3.2. LSTM-based Question Features

Here we outline an algorithm for sentence representation which adjusts itself dynamically at every step based on the compositional nature of the question.

The LSTM-based encoder representation tries to store all the information from the entire sequence into its hidden state irrespective of the length of the sentence. Thus, in the case of a bigger sentence, there is high probability that there will be some information loss if the LSTM hidden layer size is small. In case if we switch to bigger networks, it will lead to a linear increase in the number of parameters. Attention based models seek to address this limitation by keeping the hidden state of each word in the sentence and referencing them whenever required.

In detail, a hidden representation is computed for every word by running an RNN in both directions. This is commonly referred to as Bidirectional LSTM as shown in Figure 4. In forward LSTM, next word is conditioned on the all the previous hidden layer representations and we repeat these steps in reverse direction to get backward LSTM. Mathematically, it can be represented as follows:

\[
\begin{align*}
    h_j^f &= \text{LSTM}(\text{embed}(w_j), h_{j-1}^f) \\
    h_j^b &= \text{LSTM}(\text{embed}(w_j), h_{j+1}^b)
\end{align*}
\]

In the above equations, embed is a lookup functionality for word embeddings. We concatenate the forward and backward vectors to get a stacked hidden state representation for every vector. In this way, we get a matrix structure for every sentence in which the column represents the stacked word representations. We can now average these hidden states to get a more robust representation of the question. We can also use deep LSTM network to incorporate more nonlinear transformations. In the experiments section, we explore some of these variations in more detail.

We will now describe a our custom approach using the attention mechanism which is now widely used in Neural Machine Translation systems (Wu et al., 2016) and Natural Language Understanding tasks. We believe that in a question some words are more important in answering a question as the other words maybe background words common to many other questions as well. In the various QA tasks on images, questions can be classified into groups such as: yes/no type, numerical answers type etc. We develop an attention based question encoder which conditions the image on those important words. In this, we compute attention weights for every word using a MLP layer conditioned on the question LSTM state and image vector. Weighted linear combination of hidden states of bidirectional LSTM is done using normalized attention weights to get final question encoding. At every compositional step, new attention weights are used to generate dynamic representation of the question. For detailed mathematical description, we refer the reader to (Bahdanau et al., 2014).

3.3. CNN-based Question Features

3.3.1. USING WORD EMBEDDINGS

In addition to LSTM-based encoder for question, we also explore the possibility of using a CNN-based approach for encoding question features. A schematic diagram of the basic network architecture is shown in Figure 5. The question vector in the embedding layer is represented as a concatenation of the corresponding word vectors. We use weight filters to perform convolution operation over the word embeddings. The weight filters are of 3 sizes corresponding to uni-gram, bi-gram and tri-grams in sequences of words. We use 1 hidden layer with tanh nonlinearity followed by max-pooling over time. Max-pooling over time is done in order to process variable length questions and transform them in order to extract features with same number of di-
3.3.2. Using One-Hot Encoding

From our experiments we have observed that sometimes generic word embeddings (Mikolov et al., 2013) trained on large scale unsupervised corpus may not work well due to the domain specific nature of question answering tasks. Therefore, it is important to experiment with different forms of input word representations. Specifically, we used the one-hot encoding form of a sequence representation in which a sequence is represented in vocabulary space. Hence, the sequence representation is quite sparse. One example of one-hot encoding representation for the sequence “I love it” is shown in Figure 6. In this when taking 2 words at a time the corresponding regions “I love” and “love it” have one hot encoding. In this the features corresponding to the words are assigned value of 1 while other features are assigned 0 by default. The rest of the pipeline follows similar architecture as the word embedding based approach for feature extraction. Such sequence representations have obtained impressive state of the art results in text categorization and sentiment analysis tasks (Johnson & Zhang, 2016). One significant aspect this representation as compared to word vectors is that the increase in dimensionality of word sequences will significantly increase the cost of performing dot-product operations while computing the activations of hidden layer in CNN as the number of parameters will significantly increase. However, since the vocabulary size of all the questions is comparatively small as compared to the other natural language understanding tasks, this does not occur a significant increase in computational costs.

4. Experiments

4.1. Dataset Description

We evaluate our approaches on the following dataset.

- VQA (Agrawal et al., 2015) - This dataset contains around 200,000 images taken from MS COCO dataset and 50,000 other images. The collection of images include diverse, complex scenes and abstract scenes. There are around 760,000 questions associated with the images. In this, each image has 3 questions and for every question there are 10 answers generated by human annotation. Generally, top 1000 frequent answers covers roughly 83% of all answers. In order to evaluate the model, authors provide standard test server.

4.2. Baseline and Evaluation Methods

As the overall idea is to do compositional reasoning, we plan to experiment with different approaches for sentence encoding. So, we use the same VGG16 features for the image in all the following approaches. We also use 300D word Glove word vectors trained on CommonCrawl corpus in the embedding layer of LSTM and CNN encoders. We also use Spacy library in order to parse the question.

We did experiments with the following sentence encoders:

- **Word vector averaging**: In this approach, we perform simple word vector averaging for each word in the question in order to create feature vector. This is quite similar to the bag of words based approach as it ignores the order between words.

- **Question features stacked with image vector**: In this technique, we first train a 1024D hidden state vanilla LSTM on the question words. We concatenate the hidden states of LSTM with the image features followed by training deep MLP layers.

- **Elementwise multiply question and image features**: In order to do element wise multiplication we need to have common dimensions for both image and question features. In order to do this, we first pass the image features and question features through a MLP layer to transform the respective inputs to same number of dimensions followed by their elementwise multiplication to get representation of both the features.

- **Bidirectional LSTM**: In this method, we use the hidden states of a 1 layer bidirectional LSTM as described earlier in order to represent the features.

---

3. https://spacy.io/
also experiment with hidden states averaging for Bi-LSTM.

- **Word vector and one hot CNN**: As described earlier, we experiment with CNN-based features for by using word vector and one hot representation for words.

- **Deep Stacked LSTM**: In this method, we use 3 layered 1024-D deep LSTM in order to represent question.

4.2.1. **EXPERIMENT PROTOCOL**

We treat the problem of compositional reasoning for VQA as supervised classification as many answers are one word answers. The cardinality of class labels is determined by the number of unique answers for each of the dataset. We use accuracy of the model as the evaluation metric on both the datasets. We note that in this area Wu-Palmer similarity metric (WUPS) is also used, which computes a score for the similarity between two word answers, and is based on the depth of the two words in the taxonomy and that of their Least Common Subsumer (most specific ancestor node). We follow the steps as mentioned in (Agrawal et al., 2016) which considers relative match between predicted and human labels in order to evaluate the model on VQA dataset. We use the standard train, validation sets for each dataset. The evaluation on the VQA data set is done in a slightly different manner: for each question there are ten answer labels that may or may not be the same. We use the following metric: \(\min(\text{number of human labels that match that answer/3, 1})\). This metric gives full credit to the answer when three or more of the ten assigned human labels match the answer and gives partial credit if there are fewer than 3 matches.

4.2.2. **IMPLEMENTATION DETAILS**

We use 215K training images and 120K validation images as test images. We sub-sampled the images to 224 x 224 pixels. We use VGG16 model trained on ImageNet dataset for feature extraction from the VQA image. We use 300-D glove word embedding for word representation. and 1024 dimensional LSTM hidden vector. We used Keras and Tensorflow libraries in order to implement our approach. We used minibatch SGD with ‘adam’ optimizer. We also use dropout in MLP layer and gradient clipping while training LSTM layer. We trained each approach for 100 epochs.

5. **Results and Discussion**

We have reported the results of our experiments on VQA dataset in Table 1. From the results, we can see that, a bidirectional LSTM is our best performing model by a very small margin. The performance of BiLSTM with average of hidden states, LSTM with attention, CNN-based features are also quite similar. The results of deep stacked LSTM were unexpected as it gave the lowest accuracy which is almost half of the best performing model. Generally, deep stacked LSTM are the best performing models in the task of language modeling. We believe that this happens because of insufficient number of training data available for questions. Also, as expected the word vector average based model which ignores word order can be considered one of the simple baselines.

6. **Model 2: Compositional attentions with Reinforcement Learning**

We also propose an approach where question answering is considered from a reinforcement learning perspective. Specifically, we consider an agent which observes the visual input and the question, and generates a sequence of actions that finally lead to an answer.

In the discussion here, we will focus on the case where a scene graph is available. For the CLEVR dataset, ground truth scene graphs are available, and we will use them for our initial analysis. Scene graphs contain positional relationships between objects, various attributes of individual objects (shape, size, color), and their absolute positions in the image. For a general VQA dataset, we will consider computer vision methods to generate a scene graph, on which this method can then be applied.
6.1. Environment

In this section we describe the detail setting of an environment for our problem.

6.1.1. ACTIONS

We provide the agent with actions that can move the attention, or query various properties of objects currently being attended. Below is the set of actions based on the properties of the CLEVR dataset.

- **attendAttribute**(attribute): This shifts the attention to all objects that satisfy the given argument of the following (shape, size, material, color), and the relative position values: (left, right, up, down, others) that shifts attention to a certain region of the scene. Additionally, it can accept two special values: nil, which will enable not filtering on any properties, and memory, which will apply attention on that property from memory (more on this later).

- **getAttribute**(attribute): This action is similar to attendAttribute, but obtain the attribute specified by the argument based on the currently attended object (if there is any ambiguity, any empty set is returned). After that the returned value is stored in memory. The arguments can be one of (shape, size, material, color), as well as count which counts the number of attended objects.

- **output**: This function outputs the current value in memory as the answer to the question.

- **compare**: We define an abstract comparator which can store two values and compare them using the following operations:
  - **push**: This function pushes the current value in memory to the comparator. Only the last two pushed items are retained.
  - **lt, leq, gt, geq, eq**: This compares the first value in the comparator to the second using the respective operation, and stores the result in memory.

Now, we show how these operations can be used to answer questions from the CLEVR dataset using a sequence of actions. Consider again the question from Figure 1. This can be answered using the following sequence: **attendAttribute**(large) → **getAttribute**(count) → **compare-push** → **attendAttribute**(nil) → **attendAttribute**(metal) → **attendAttribute**(sphere) → **getAttribute**(count) → **compare-push** → **compare-eq** → **output**.

6.1.2. REWARDS

To train a reinforcement learning model on the above described environment, we need to define a reward structure. Here we consider a simple scheme that gives a reward of -1 on every iteration that does not produce an output. On receiving an output, if the answer is correct, a reward of 100 is given, otherwise a reward of -100 is given. Finally, after a fixed number of steps, the session is considered over, and a reward of -100 is given at the end.

6.1.3. STATE

Now, for a concrete representation, we need a state representation of the environment. Our state representation consists of three parts: 1) the question as a list of tokens, 2) the scene image as an RGB array, and 3) the attention represented as a grayscale image. The attention image is generated from the hard attention by taking a black image, and putting white patches at the centers of attended objects. Correspondingly our Q-network consists of 3 parts. The question is processed by a Gated Recurrent Unit (GRU) to get a vector representation. The image and attention map are processed by Convolutional Neural Networks (CNNs). We use the same CNN architecture as in (Mnih et al., 2015). This generates vector representations for the image and attention map. The 3 vector representations are concatenated to get a final state representation, from which Q-values are generated.

6.2. Training with Deep Q-learning

Many possible methods are possible for learning reinforcement learning models using our defined environment. We use Q-learning in our experiments. Here, ever state, action pair is represented by a Q-value denotes the maximum discounted reward achievable by taking that action at that state. The Q-values satisfy what is called the Bellman equation:

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a')$$

The Q-values can be parametrized in various ways. We used a deep neural network since it has shown success in learning from complex environments (Mnih et al., 2015). The neural network takes the state representation as input and produces a vector of Q-values, one for each action. The network weights are trained using a loss function that captures deviation from the Bellman equation:

$$L(Q(s, a), r + \gamma \max_{a'} Q(s', a'))$$

We use Huber loss since it is less sensitive to larger deviations. For the actual Q-network, we implement an improved variant of the vanilla deep q-network that implements Bellman equation above. Specifically, we adopt
recently proposed modification called dueling architecture (Wang et al., 2015). This work proposes to let the model separately learn the state value function and the advantage function over state action pairs. With these two components, q values are obtained simply by adding the outputs from them. This simple and decoupled modeling of Q-value function significantly increases the performance because the model can directly learn from the loss by the advantage function, which governs the change in q values. In addition, the previous work applied this dueling architecture on double q-learning setting, where the model has two copies of the same q-network and one is used for predicting actions $a'$ while the other is used for the state value function $\max_{a'} Q(s, a')$. We follow the their approach and implemented the resulting dueling double q-network, and used this model to estimate q values.

6.3. Experiments

6.3.1. Dataset Description

- **CLEVR** (Johnson et al., 2016)\(^4\): This dataset consists of a training set of 70,000 images with 699,989 questions. Validation set has 15,000 images and 149,991 questions. The Test set contains 15,000 images and 14,988 questions. The resolution of each image is 320 x 480 pixels. Annotations for images in training and validation set consists of locations, attributes and relationships among objects which is also referred to as scene graph. Every question is associated with a functional program which consists of simple operations such as query, count and compare. Questions were generated by filling template parameters in specific question families. Authors use rejection sampling to remove bias in answers distribution across the dataset. The questions and images in this dataset are selected such that they test visual reasoning skills of VQA systems without the application of external knowledge. Authors also compare recent VQA models and find that the approach of (Yang et al., 2016) results in highest overall accuracy.

We trained the model on the CLEVR dataset. The actions for the environment are implemented using the ground truth scene graphs from the dataset. Wherever applicable we used the same hyperparameters of Mnih et al. (2015). This includes using the same policy for experience replay. The vocabulary for the question representation is generated using the training split of the dataset, and questions are limited to 45 tokens. The GRU hidden unit size is 64. The images and attention maps are resized to 64x96. Both during training and evaluation, we limited episodes to 50 steps.

The model was evaluated every 10000 steps by running a $\epsilon$-greedy ($\epsilon = 0.05$) policy. The evaluation measure is simple accuracy based on the finally generated output.

We were not able to get get any positive results with the model. Both training and validation accuracy remained at 0 in the training. We also faced an issue with excessive large training times which prevented the model from training to completion. In the next section we discuss some possible extensions that might improve performance of the model.

6.4. Discussion

Now we discuss some aspects that might help improve the performance of the reinforcement learning model. The biggest issue with the current model is the sparsity of rewards. Currently, the only positive reward is obtained on giving a correct answer. This gives very sparse signal for training the model, and it is unable to learn to use the complex structures in the environment like memory and comparators. Better performance might be achieved by rewarding correct use of these structures during the answering process.

The second scope for improvement is to use a different training approach. Although deep Q-learning has shown remarkable empirical success, it is quite tricky to train, and very sensitive to hyperparameters. Better performance might be achieved by using the technique of policy gradients.

7. Conclusion

We experimented with two main approaches for adding compositional reasoning to the visual question answering problem. One is based on incorporating attention to the question words conditioned on image. The second is a reinforcement learning approach which aims to solve question answering through a sequence of atomic actions. We saw that a bidirectional LSTM based approach gives the best results on the VQA dataset. Future work include applying the same techniques on CLEVR dataset. Also, we believe that a deep LSTM pre-trained on large scale unsupervised corpus such as Wikipedia can give better performance on VQA tasks.

The second approach based on reinforcement learning did not yield and positive results, but we believe that the framework could be useful for developing models with stronger reasoning capabilities. More work with this approach could yield competitive models.
References


