Attended Relational Reasoning for Visual Question Answering

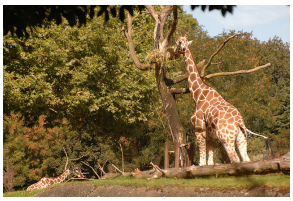
Lasse Just Petersen
Aalborg University
ljpe12@student.aau.dk

June 15, 2018

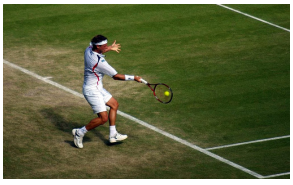
Abstract

We propose an efficient neural network model for visual question answering (VQA). The model uses Faster R-CNN based image embedding approach, generating features for object-level image regions. We add a relational reasoning module, that reasons about object pairs explicitly, augmenting the models reasoning capabilities. We use a question-guided visual attention mechanism to reduce the number of objects considered for relational reasoning. We evaluate the model on the VQA v2.0 dataset, achieving state-of-the-art single-model performance. We experiment with different ablations of our model, determining the performance impact of different components. We show that the relational reasoning module generally improves model performance, and that we can reduce the number of pairs considered significantly, without negatively affecting performance.

(a) *What color is the bag in front of the cat?*



(b) *What is the giraffe standing behind?*



(c) *What are they playing?*

(d) *How many surfboards are there?*

Figure 1: Examples of questions and corresponding images for the VQA task.

1 Introduction

In recent years, the task of visual question answering (VQA) has gained an increasing amount of interest in research. The objective of VQA is to answer questions, posed in natural language, in the context of a given image, as illustrated in Fig. 1. The answers can be either generated as free-form text, or chosen from a given set of answer candidates [1].

VQA lies in the intersection of the fields of computer vision (CV) and natural language processing (NLP). Images are processed using methods from the field of CV, while NLP methods are used to process questions and generate free-form answers [2]. However, compared to popular CV tasks such as object recognition, VQA requires more sophisticated reasoning capabilities [3, 4]. Whereas object recognition requires identification of objects in images or image regions, VQA systems must also be able to reason about properties of these objects, such as size and

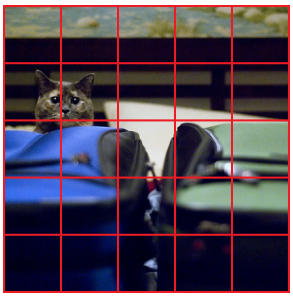
position, and relations between them Johnson *et al.* [4]. For example, the question *"What color is the bag in front of the cat?"* in Fig. 1a requires the system to first identify the cat and the bags in the image, then determine which bag is positioned in front of the cat, and finally determine the color of that bag.

The majority of recent VQA models follow a similar joint embedding approach [2]. Images are embedded using a convolutional neural network (CNN), while questions are embedded using a recurrent neural network (RNN). Visual attention mechanisms are then used to focus on the image regions most relevant to the question, before the embeddings of the two modalities are combined into a joint embedding, which is then passed through a classifier, or decoded using a RNN.

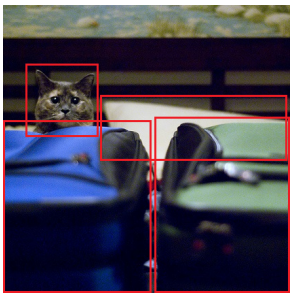
However, it was recently shown that this general architecture might not be able to properly capture the relational reasoning, i.e. reasoning about relations between objects, required for certain types of questions [5]. Instead, Santoro *et al.* [5] propose an architecture called Relation Network (RN), that explicitly considers image region pairs. While this architecture improves relational reasoning capabilities, it is not very efficient, as it considers all possible region pairs.

Additionally, the use of CNNs for embedding images has been shown to limit VQA performance, as the regions from the CNN might correspond poorly to objects in the image [6, 7], as shown in Fig. 2a. As an alternative, Anderson *et al.* [6] propose an image embedding method based on the Faster R-CNN object detection framework [8], generating embeddings for regions at object-level, shown in Fig. 2b, better suited for the level of reasoning required for VQA.

We propose an efficient, end-to-end trainable, joint embedding model for general VQA, using object-level image embeddings, and incorporating the relational reasoning capabilities of the RN architecture. Our model is based on that of Teney *et al.* [7], using the Faster R-CNN image embedding method from Anderson *et al.* [6]. We add a module for relational reasoning, based on the RN architecture suggested by Santoro *et al.* [5]. To improve the efficiency, we use an attention mechanism to select the objects most relevant to the question, and consider only these objects in the relational reasoning module.



(a) Example CNN regions.



(b) Example Faster R-CNN regions.

Figure 2: Example CNN and Faster R-CNN image regions.

We evaluate our model on the VQA 2.0 dataset [3], achieving state-of-the-art single-model results.

Our main contributions are:

- We propose a joint embedding model for VQA, using object-level image embeddings, adding a relational reasoning module (RN module) to increase the reasoning capabilities of the model.
- We use question-guided attention to reduce the number of objects considered by the relational reasoning module.
- We show that the relational reasoning module generally increases VQA performance, and that the number of objects considered for relational reasoning can be reduced significantly without negatively affecting performance.

The rest of the paper is structured as follows: In Section 2 we cover recent related work on VQA, before we introduce our model in Section 3. We describe our experiments and present the results in Section 4, and finally conclude on our findings in Section 5.

2 Related Work

Most recently proposed VQA models follow a similar joint embedding approach, first embedding both the question and image, then joining the embeddings with some multimodal fusion method, and finally using the

joint embedding to either predict or generate an answer [2, 9]. Questions are usually embedded using RNNs, with both long short-term memory networks (LSTMs) [10] and gated recurrent units (GRUs)[11] being popular choices, and image embedding is typically done with well-known CNN models, such as VGGNet [12] or ResNet [13], pretrained on ImageNet [14]. For combining the embeddings, approaches range from very simple concatenation and element-wise multiplication, to much more elaborate methods. Answers are finally either predicted using a classifier, or generated from the joint embedding by a RNN-based decoder.

One of the main challenges of joint embedding models is to ensure that image embeddings contain the necessary information to answer a particular question. This is difficult because images often contain a lot of unnecessary information, as questions typically only pertain to certain image regions. This is addressed in most models with an attention mechanism Xu *et al.* [15] and Wu *et al.* [2].

An attention mechanism allows the model to focus on specific regions of the image, that are particularly relevant to a given question. Attention mechanisms have been shown to greatly improve VQA performance, and they are both used and researched extensively today. A simple method for question guided visual attention is to combine the question embedding with each image region embedding from a CNN, and pass them through a multilayer perceptron (MLP), generating an attention weight for each region. More advanced approaches include iteratively querying the image [16, 17], applying attention at multiple semantic levels [18], and jointly attending both question and image [19].

A recent approach that has shown very promising results is called bottom-up attention, where attention is applied in a *bottom-up* fashion, i.e. not guided by the question [6]. Instead, the attention is applied directly during image embedding, using a method based on the Faster R-CNN object detection framework [8] in combination with a ResNet CNN [6]. This method both functions as an early attention mechanism, ignoring regions of the image with no detected objects, and it provides image embeddings for regions at object-level, better suited for VQA.

The bottom-up attention approach can essentially be used as a drop-in replacement for CNNs in VQA models, as done by Teney *et al.* [7]. In addition to the Faster R-CNN based image embedding, their model uses a GRU for embedding questions, and a simple concatenation-based question guided attention mechanism. Embeddings are combined with a simple element-wise fusion scheme, and the joint embedding is passed through a classifier.

While the joint embedding approach is popular, recent research has shown that these models might not be able to learn the advanced reasoning required for VQA [5, 20, 21]. A number of methods has been proposed to address this, ranging from components that can simply be added to existing models [22, 5], to using composable modules to construct entire networks [23, 24].

Santoro *et al.* [5] recently proposed the Relation Network (RN) architecture, designed specifically for augmenting relational reasoning performance. The RN reasons about all image regions pairs explicitly. Image region embeddings are generated with a CNN, and questions are embedded with an LSTM. For each region pair, the embeddings of the two regions and the question embedding are concatenated, and passed through a MLP, generating a feature vector for that region pair. These vectors are then summed to a final embedding used for classification.

Compared to other recent approaches for augmenting reasoning capabilities [23, 24, 20], the RN is a rather simple approach, that can be incorporated in existing models. The main disadvantage of the RN is that considering all possible image region pairs quickly becomes very computationally expensive.

Our work is based on the RN [5] and the Faster R-CNN based image embedding method [6, 7]. In contrast to Santoro *et al.* [5], we address the computational limitation of the RN, using both bottom-up and question guided attention mechanisms. We also apply the RN as an additional module in a more general VQA model. Compared to the model proposed by Teney *et al.* [7], we incorporate the additional RN based module, and simplify other components.

3 Model

In this section, we present our model, which is based on the joint embedding model proposed by Teney *et al.* [7]. We first give a quick overview of the model, before we describe the different model components in more detail, and finally we describe the training objective.

3.1 Model Overview

We treat the VQA task as a classification problem, taking as input a natural language question and an image. First, questions are embedded using a GRU, and images are embedded using the Faster R-CNN embedding method from Anderson *et al.* [6]. A simple question guided visual attention mechanism is used to focus on important image regions, and a relational reasoning module performs additional pair-wise reasoning on objects. The embeddings are combined using a simple element-wise multiplication multimodal fusion scheme, and the joint embedding is finally passed to a classifier. An overview of our model is shown in Fig. 3.

3.2 Non-linear Layers

Before we describe the individual components, we define the non-linear layers used in components throughout the model. A non-linear layer implements a function $f : \mathbb{R}^m \to \mathbb{R}^n$, mapping a vector $\mathbf{x} \in \mathbb{R}^m$ to a vector $\mathbf{y} \in \mathbb{R}^n$.

Teney *et al.* [7] propose a non-linear layer using a gated hyperbolic tangent activation, defined as follows:

$$\tilde{\mathbf{y}} = \tanh(W\mathbf{x} + \mathbf{b}) \tag{1}$$

$$\mathbf{g} = \sigma(W'\mathbf{x} + \mathbf{b}') \tag{2}$$

$$\mathbf{y} = \tilde{\mathbf{y}} \odot \mathbf{g} \tag{3}$$

where σ is the sigmoid function, $W, W' \in \mathbb{R}^{n \times m}$ are learned parameters, and $\mathbf{b}, \mathbf{b}' \in \mathbb{R}^n$ are learned biases [7].

This function essentially passes the input vector $\mathbf{x}$ through two separate non-linear layers, and combines their outputs. However, we note that both the hyperbolic tangent and the sigmoid activation suffer from the vanishing gradient problem, which might lead to a loss of training signal [25].

Instead, we use a single non-linear layer, and an activation function more resistant to the vanishing gradient problem:

$$\mathbf{y} = \phi(W\mathbf{x} + \mathbf{b}) \tag{4}$$

where ϕ is a non-linear activation function, $W \in \mathbb{R}^{n \times m}$ is a learnable weight matrix, and $\mathbf{b} \in \mathbb{R}^n$ is a learnable bias vector.

For the non-linear activation ϕ, we use the scaled exponential linear unit (SELU) [26], which has been shown to lead to fast training and good generalization performance:

$$\phi(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{otherwise} \end{cases} \tag{5}$$

where $\lambda \approx 1.0507$ and $\alpha \approx 1.6733$ are fixed parameters [26].

There are several other viable activation functions, and in Section 4 we experiment with three popular alternatives, namely the rectified linear unit (ReLU) [27], leaky rectified linear unit (LReLU) [28], and exponential linear unit (ELU) [27].

Finally, we note that this simple non-linear layer halves the number of learnable weights and biases, compared to the gated hyperbolic tangent function, though we can simply double the layer size to compensate for this.

3.3 Question Embedding

Questions are given in natural language, represented as a sequence of words. Each question $Q = (w_1, w_2, \ldots, w_l)$ is embedded into a vector representation $\mathbf{r}_q \in \mathbb{R}^{d_q}$, where l is the sentence length, and d_q is the question embedding size.

We use the same method for question embedding as Teney *et al.* [7]. First, questions are tokenized, and each word $w_i \in Q$ is replaced with a one-hot encoded vector representation $\mathbf{v}_i \in \mathbb{R}^{|V|}$, which is then mapped

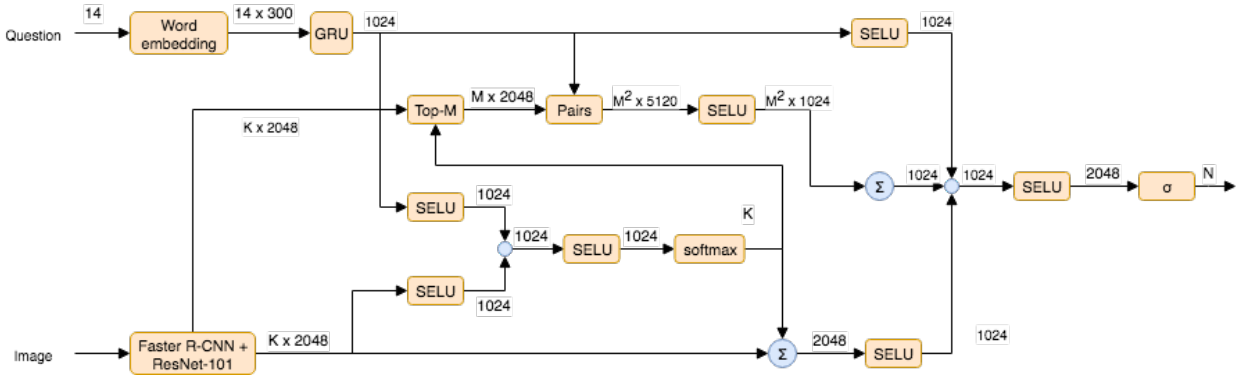


Figure 3: Overview of the complete model, shown with the final dimension.

to a word embedding $\mathbf{e}_i \in \mathbb{R}^{d_w}$ as follows:

$$\mathbf{e}_i = W_e \mathbf{v}_i \tag{6}$$

where $W_e \in \mathbb{R}^{d_w \times |V|}$ is an learned embedding matrix, d_w is the word embedding size, and $|V|$ is the size of the vocabulary. The word embeddings $E_q = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_l)$ are then fed sequentially through a gated recurrent unit (GRU), and the final state of the GRU is used as the question embedding $\mathbf{r}_q$.

3.4 Image Embedding

This component takes an image $I \in \mathbb{R}^{H \times W \times C}$ as input, where H, W, and C are the height, width, and number of channels in the image, respectively, and produces a number of embeddings $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_k \in \mathbb{R}^{d_v}\}$. Here, each $\mathbf{v}_i \in V$ is an embedding of a separate image region, d_v is the embedding size, and k denotes the number of image regions.

We use the bottom-up method proposed by Anderson *et al.* [6], combining the Faster R-CNN object detection framework [8] with a ResNet-101 CNN [13], to generate object-level image embeddings.

The method works in multiple stages. First, the image is passed through a ResNet-101 CNN, generating a feature map for the entire image. A Region Proposal Network (RPN) is then used to propose bounding boxes for a number of regions, that are likely to contain objects. For each of these regions, a small feature map is extracted from the ResNet-101 embedding, using a method called region of interest (RoI) pooling. The k regions that are most likely to contain objects are then selected, based on an associated detection probability. Finally, mean-pooling is used to generate a vector embedding $\mathbf{v}_i \in \mathbb{R}^{d_v}$ for each of these regions, where $d_v = 2048$ when using ResNet-101.

Compared to the embeddings generated by a CNN, the embeddings of regions at object-level seem to better correspond to the level of reasoning required for VQA. In addition, this approach works as a type of early attention. Whereas a CNN generates embeddings for a grid of regions over the entire image, the object-level regions effectively discards the parts of the image that are unlikely to contain objects.

3.5 Visual Attention

The purpose of the attention mechanism is to weigh the visual features according to their relevance for the question. We use a simple question-guided visual attention mechanism, taking as input the question embedding $\mathbf{r}_q$ and the embedded image regions $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$. The image region embeddings are weighed by their relevance, and combined into the final image embedding $\mathbf{r}_v \in \mathbb{R}^{d_v}$.

First, an attention weight a_i is generated for each region embedding $\mathbf{v}_i \in V$. The region embedding $\mathbf{v}_i$ and question embedding $\mathbf{r}_q$ are both mapped to d_h-dimensional representations, using non-linear layer f_a^v and f_a^q. These representations are then combined using the Hadamard product (element-wise multiplica-

tion), and the combined embedding is passed through a non-linear layer $f_a : \mathbb{R}^{d_h} \to \mathbb{R}^{d_h}$ and then a linear layer:

$$a_i = \mathbf{w}_a \, f_a\Big(f_a^v(\mathbf{v}_i) \odot f_a^q(\mathbf{r}_q)\Big) + b_a \tag{7}$$

where $\mathbf{w}_a \in \mathbb{R}^{d_h}$ is a learnable parameter vector, $b_a \in \mathbb{R}$ is a bias scalar, and $\odot$ is the Hadamard product.

Finally, the attention weights $\mathbf{a} = (a_1, a_2, \ldots, a_k)$ are normalized across regions with the softmax function, the region embeddings are multiplied with their respective attention weight, and the weighted embeddings are summed, resulting in a single attended image embedding $\mathbf{r}_v \in \mathbb{R}^{d_v}$:

$$\boldsymbol{\alpha} = \text{softmax}(\mathbf{a}) \tag{8}$$

$$\mathbf{r}_v = \sum_{i=1}^{k} \alpha_i \mathbf{v}_i \tag{9}$$

This attention mechanism is a variation of that used by Teney *et al.* [7], where they simply concatenate the region embeddings with the question embedding, therefore not requiring the initial non-linear transformation to get equal dimensions. We use the Hadamard product for combining the embeddings here, as it seems to generally outperform concatenation [1, 29].

3.6 Relational Reasoning Module

We extend the model with a relational reasoning module, based on the Relation Network (RN) architecture [5]. Intuitively, this module will augment the relational reasoning capabilities of the model, by explicitly considering combinations of image region pairs.

The module takes as input the question embedding $\mathbf{r}_q$, the image region embeddings $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$, and the previously computed attention weights $\boldsymbol{\alpha}$, and outputs a single relational embedding $\mathbf{r}_r \in \mathbb{R}^{d_h}$.

First, the attention weights $\boldsymbol{\alpha}$ are used to select the m most relevant regions $V_a \subset V$, where $|V_a| = m$, $m < k$, and the selected regions correspond to the n highest attention weights. A relational embedding $\mathbf{r}_{ij} \in \mathbb{R}^{d_h}$ is then generated for each region embedding pair $(\mathbf{v}_i, \mathbf{v}_j) \in V_a \times V_a$, by passing the concatenated embeddings of the regions $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^{d_v}$ and the question $\mathbf{r}_q \in \mathbb{R}^{d_q}$ through a non-linear layer $f_r : \mathbb{R}^{2\,d_v + d_q} \to \mathbb{R}^{d_h}$. Finally, the embeddings for all pairs are summed, producing the final relational embedding $\mathbf{r}_r$:

$$\mathbf{r}_{ij} = f_r(\mathbf{v}_i \oplus \mathbf{v}_j \oplus \mathbf{r}_q) \tag{10}$$

$$\mathbf{r}_r = \sum_{i=1}^{m} \sum_{j=1}^{m} \mathbf{r}_{ij} \tag{11}$$

where $\oplus$ denotes concatenation of vectors. Here, embeddings are concatenated to differentiate between pairs with different orderings, i.e. $(\mathbf{v}_i, \mathbf{v}_j)$ and $(\mathbf{v}_j, \mathbf{v}_i)$ should produce distinct embeddings $\mathbf{r}_{ij}$, $\mathbf{r}_{ji}$ such that $\mathbf{r}_{ij} \neq \mathbf{r}_{ji}$.

There are several differences between this relational reasoning module, and the RN model proposed by Santoro *et al.* [5]. The main differences are as follows:

- We use attention to select a subset of image region embeddings $V_a \subset V$, where $|V_a| = m$, $|V| = k$, and $m < k$. Selecting n to be significantly smaller than k, this results in much fewer region pairs being considered, as we have $|V_a \times V_a| = n^2$, $|V \times V| = k^2$, and $n^2 \ll k^2$.
- We incorporate the RN in a joint embedding model, as a separate relational embedding $\mathbf{r}_r$, which is used in combination with the question and image embeddings $\mathbf{r}_q$ and $\mathbf{r}_v$.
- The relational module uses embeddings generated with the bottom-up approach described in Section 3.4, where regions better correspond to objects in the image. This allows the module to reason at an object-level, more suitable for the task of VQA.

3.7 Multimodal Fusion

The multimodal fusion combines the question embedding $\mathbf{r}_q$, the attended image embedding $\mathbf{r}_v$, and the relational embedding $\mathbf{r}_r$ into a joint embedding

$\mathbf{r}_j \in \mathbb{R}^{d_h}$, which is then passed to the classifier. The question and image embeddings are first mapped to d_h-dimensional vectors using non-linear layers f_m^q and f_m^v, respectively, and then combined with the relational embedding, which is already d_h-dimensional, using the Hadamard product:

$$\mathbf{r}_j = f_m^q(\mathbf{r}_q) \odot f_m^v(\mathbf{r}_v) \odot \mathbf{r}_r \tag{12}$$

where $\odot$ is the Hadamard product. We use the Hadamard product again, as a simple and effective method for multimodal fusion, which is also used by Teney *et al.* [7].

3.8 Classifier

We treat the VQA task as a multi-label classification task, where multiple answers could be considered correct, e.g. in case of ambiguous questions or answer synonyms, as done by Teney *et al.* [7]. The N most frequent answers are used as possible classes, and the classifier outputs a separate probability for each label being correct, instead of a probability distribution over all N labels.

We use a simple classifier, taking the joint embedding $\mathbf{r}_j$ as input. The joint embedding is first passed through a non-linear layer $f_c : \mathbb{R}^{d_h} \to \mathbb{R}^{d_c}$, where d_c is the hidden size of the classifier. The d_c-dimensional representation is then passed through a linear layer, and finally the sigmoid activation is used to produce a score $\hat{s}_i \in (0, 1)$ for each label, giving the final output $\hat{\mathbf{s}} = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_N)$:

$$\hat{\mathbf{s}} = \sigma\Big(W_c \, f_c(\mathbf{r}_j) + \mathbf{b}_c\Big) \tag{13}$$

where $W \in \mathbb{R}^{N \times d_c}$ is a learnable weight matrix, $\mathbf{b}_c \in \mathbb{R}^N$ is a learnable bias, and σ denotes the sigmoid activation applied element-wise.

While we follow the same approach as Teney *et al.* [7], we use a simpler classifier. Teney *et al.* [7] suggests splitting the classifier into text and an image parts, which are pretrained using prior information about the answer candidates, in the form of pretrained word embeddings and embeddings of images scraped from *Google Images*. However, this approach both adds complexity to the model, and makes it difficult to reproduce results, prompting us to use the simpler approach instead.

3.9 Objective

Following the approach of Teney *et al.* [7], the model is trained in a multi-label setting, where each training example can have multiple answers, each associated with a soft accuracy score in $(0, 1)$. Formally, each training example $d_i \in D$, where D is the training dataset, is labeled with soft target scores $\mathbf{s}_i \in (0, 1)^N$, where each entry represents the accuracy score for the corresponding answer.

The output of the classifier can be considered as separate predictions for each label, indicating the probability of that label being correct. To calculate the loss, we therefore compute the binary cross entropy separately for each label, and then sum these across the labels and training examples, giving us the following loss function:

$$L = -\sum_{i}^{|D|} \sum_{j}^{N} s_{ij} \log(\hat{s}_{ij}) - (1 - s_{ij}) \log(1 - \hat{s}_{ij}) \tag{14}$$

where i iterates over the $|D|$ training examples, j iterates over the N labels, s denotes the ground truth scores, and $\hat{s}$ is the predictions.

4 Experiments

In this section we present our experiments. We start by introducing the dataset and experimental setup, before presenting our experimental results.

4.1 Dataset

We train and evaluate our model on the VQA v2.0 dataset [3]. The dataset uses MSCOCO [30] real-word images, posing a number of questions for each. It contains a total of 1.1M questions across 204K images, split into 443K training, 214K validation, and 447K test questions. Each question is annotated with 10 ground-truth answers, provided by different

human annotators, to account for variations in human answers.

Annotations for the test split are not publicly available, so results for this split are obtained by submitting the predicted answers to an official evaluation server. Evaluation can be done on two different subsets of the test split, namely test-standard and test-dev. However, the evaluation server allows only a very limited number of daily and total submission. Therefore, we use the validation split to evaluate ablations of our model, and only use the test-standard and test-dev splits to evaluate our best performing model.

4.2 Evaluation Metric

Performance is evaluated using the official VQA evaluation metric [1]:

$$\text{acc}(a) = \min\left(\frac{\#\text{humans that answered } a}{3}, 1\right) \quad (15)$$

Essentially, an answer is considered correct if it matches at least three human annotations, partially correct, to varying degree, if it matches one or two, and incorrect otherwise.

4.3 Experimental Setup

For question embedding, we use word embeddings of size $d_w = 300$, initialized with GloVe embeddings [31] pretrained on the Wikipedia/Gigaword corpus.[1] Embeddings for words not in the GloVe vocabulary are initialized with weights drawn from a standard normal distribution. The GRU uses a single hidden layer of size $d_q = 1024$, and we use a fixed question length $l = 14$, truncating or zero-padding questions as required.

For efficiency, the image embedding is done as a preprocessing step, using publicly available preprocessed image features with $k = 36$ objects per image, and $d_v = 2048$.[2] The preprocessed features are generated with a Faster R-CNN based bottom-up model [6] trained on the Visual Genome dataset, using a ResNet-101 CNN pretrained on ImageNet.

To get the possible classes, we select answers that appear more than 8 times in the combined training and validation sets, giving $N = 3129$ possible classes. The hidden size of the classifier is $d_c = 2048$.

We use a hidden embedding size of $d_h = 1024$, $m = 8$ objects for the relational reasoning module, the SELU [26] non-linear activation, and apply weight normalization [32] on all layers. For regularization we apply dropout with probability $p = 0.4$ after each non-linear layer, and perform early stopping after 5 consecutive epochs with no improvement in validation score.

Models are trained using Adamax [33] with $\alpha = 0.002$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, using a batch size of 512. For comparison to current state-of-the-art models, we evaluate our best performing configuration on both the test-dev and test-standard splits, using both the train and validation splits for training. We evaluate ablations of our model on the validation split, using just the train split during training.

Initial experimentation with model architecture and parameters was performed on a single NVIDIA GTX 1060 6GB GPU, while the final models used for evaluation were trained on two NVIDIA Tesla M60 GPUs.

4.4 State-of-the-Art Comparison

We first evaluate the overall performance of our model, comparing it to current state-of-the-art approaches.

4.4.1 Baseline models

We adopt the following models as baselines for this comparison:

- Multimodal Compact Bilinear Pooling (MCB) [29]: Winning entry of the 2016 VQA Challenge, applying the efficient multimodal fusion method MCB in a joint embedding classification model. The model uses a ResNet-152 CNN for image embedding, a LSTM for questions, and a simple attention mechanism.
- MUTAN [34]: A joint embedding model, that uses a ResNet-152 CNN for image embedding,

[1] https://nlp.stanford.edu/projects/glove/

[2] https://github.com/peteanderson80/bottom-up-attention

and GRU for question embedding, and a new tensor-based fusion scheme called MUTAN.

- Bottom-up attention [7]: The winning entry of the 2017 VQA Challenge, and the model that inspired our model. It is similar to our model, without the relational module, but using the gated hyperbolic tangent non-linear activation, a concatenation based attention mechanism, a classifier split into and image- and text-parts. They use 485K additional training examples from Visual Genome, and pretrain the classifier using pretrained word embeddings and images scraped from Google.

- Bottom-up attention (our implementation): This is a re-implementation of the bottom-up attention baseline model [7]. We include this implementation as a baseline to compare the models in a similar experimental setting, and we could not reproduce their setting. For this implementation we (1) do not use extra Visual Genome data, (2) do not pretrain the image-part of the classifier, and (3) use a fixed number of image embeddings $k = 36$ for each image, instead of an adaptive k.

4.4.2 Results

The results are shown in Table 1, where we call our model Attended Relation Network (Att-RN). Our model achieves state-of-the-art performance, with an accuracy of 68.79% on test-std and 68.49% on test-dev, $\sim 3\%$ higher than the MUTAN and bottom-up attention baselines, and $\sim 6.5\%$ higher than MCB.

Looking at the results for the three different answer categories, we see that our model consistently scores $\sim 3\%$ higher than the bottom-up attention baseline on all three categories. Compared to MUTAN however, our model performs much better on the *number* category, increasing performance by $\sim 6\%$.

We also note that our re-implementation of the bottom-up attention baseline, achieves a very similar overall performance to the original model, though distributed differently across the categories. This raises some question regarding the efficiency of using the extra Visual Genome data and the pretrained classifier, though we did not further investigate this.

4.5 Ablation Experiments

We perform experiments with different model ablations, in order to determine the contributions of key components in the model.

4.5.1 Ablation models

We consider the following main variations:

- No RN: Model without the RN module.

- RN-unique: The RN module generates only unique unordered pairs of region embeddings, reducing the total number of pairs considered to $\binom{m}{2}$.

- RN-Hadamard: In the RN module, the image embeddings and question embedding are mapped to equal dimensions with separate non-linear layers, and then combined using the Hadamard product.

- RN-weighed: Here, the image embeddings are weighed by their corresponding attention weight, before they are used in the RN module.

- ResNet-101 embedding: This model uses a ResNet-101 CNN for embedding images. Images are cropped to size 224×224, and features are extracted from the the last convolutional layer, resulting in a feature map of size $7 \times 7 \times 2048$. The ResNet is pretrained on ImageNet and kept fixed during training. The embeddings can therefore be preprocessed, and used as a simple drop-in replacement.

To evaluate the effect of using a subset of region embeddings for the RN module, we perform experiments with a varying number of regions k. Finally, we determine the effect of simplifying the non-linear layer through experiments with different non-linear activation functions.

4.5.2 Results

The results of our ablative experiments are shown in Table 2. We start be analyzing the performance of the RN module. We see that removing the RN module lowers performance by 1.19%, indicating the

	VQA v2 test-dev				VQA v2 test-std			
Model	Yes/No	Num	Other	All	Yes/No	Num	Other	All
MCB [29, 3]	-	-	-	-	78.82	38.28	53.36	62.27
Bottom-up att. (our impl.)	82.11	42.54	56.05	65.25	-	-	-	-
Bottom-up attention [7]	81.82	44.21	56.05	65.32	82.20	43.90	56.26	65.67
MUTAN [34]	81.96	41.62	57.07	65.57	82.07	41.06	57.12	65.71
Our model: Att-RN	**85.02**	**47.65**	**59.34**	**68.49**	**85.10**	**47.32**	**59.50**	**68.79**

Table 1: Results on the VQA v2.0 dataset for our model (Att-RN) and current single-model state-of-the-art approaches.

the RN module does indeed improve the reasoning capabilities of the model.

Using only unique unordered pairs in the RN module gives results similar to removing the module. While this method does lower the number of pairs considered from 64 to 28, compared to our best model with $m = 8$, it still uses nearly twice as many pairs as the model with $m = 4$. We therefore attribute the performance loss here to the use of unordered pairs. However, the model with the Hadamard RN also discards information about the ordering of pairs, and still score $\sim 0.6\%$ higher than not using a RN module. This may simply be due to Hadamard being a better multimodal fusion method. We still observe a lower performance (-0.6%) compared to our best performing model, further indicating the importance of using ordered pairs. Using weighed image embeddings in the RN module lowers performance by 0.97%. We hypothesize that, as the weights are normalized across all regions with the softmax function, many regions will have low attention score, discarding information needed in the RN module.

Comparing the RN variations with $m = 4$ and $m = 12$ objects, we see that the number of objects m can have a significant impact on performance, with the $k = 4$ model scoring 0.77% lower than the $k = 8$ configuration. From the $k = 12$ variation we see that increasing k does not necessarily increase overall performance, in this case actually lowering it slightly, though it does improve performance on the *number* category. This shows that the attended RN module, aimed at lowering computational complexity compared to the full RN, can be applied without hindering performance.

Using pretrained ResNet-101 embeddings lowers performance by 2.87%, confirming previously reported results [7, 6] that the Faster R-CNN based method does indeed result in image embeddings better suited for VQA.

Finally, we consider the performance of different non-linear activation functions. Overall, we see that our model is very sensitive to different parameter settings. The ELU activation is very similar to SELU, , and as expected, it also performs similarly. Using the ReLU activation drastically lowers performance (-3.55%). LReLU adds a small gradient for negative values, which improves performance somewhat compared to ReLU, though the accuracy is still 1.62% lower than using SELU. The gated tanh activation performs very poorly in our setting, lowering performance by 6.42% compared to the SELU activation.

5 Conclusion

In this paper we propose an efficient joint embedding neural network model for VQA. The model uses a Faster R-CNN based image embedding approach, that generates features for regions at object-level. We augment the reasoning capabilities of the model by adding an attended relational reasoning module, that explicitly reasons about pairs of relevant objects.

We evaluate our model on the VQA v2.0 dataset, where we achieve state-of-the-art single-model performance. We show that the relational reasoning module improves performance for general VQA, and that

	VQA v2 val			
	Yes/No	Num	Other	All
Reference model (RN, $m = 8$, Faster R-CNN, SELU)	**83.30**	45.01	**57.43**	**65.52**
No RN module	81.65	43.21	56.76	64.33
RN-unique	81.89	43.23	56.45	64.27
RN-Hadamard	82.30	44.82	57.03	64.92
RN-weighed	82.17	44.27	56.51	64.55
RN module with $m = 4$ objects	82.30	44.08	56.88	64.75
RN module with $m = 12$ objects	83.19	**45.35**	57.24	65.43
ResNet-101, 7×7	-	-	-	62.65
ELU	83.00	45.21	57.32	65.38
ReLU	79.27	42.43	53.99	61.97
LReLU	80.74	43.48	55.88	63.60
Gated tanh, hidden size 512	76.83	40.03	50.67	59.10

Table 2: Results on the VQA v2.0 validation split for different ablations of our model. The reference model is the best configuration of our full model, using Faster R-CNN + ResNet-101 image embedding, a RN module with $k = 8$ objects, and the SELU non-linear activation.

question-guided visual attention can be used to greatly reduce the number of pairs considered for relational reasoning, without negatively affecting performance.

References

[1] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering", in *International Conference on Computer Vision*, 2015.

[2] Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, and A. van den Hengel, "Visual question answering: A survey of methods and datasets", *Computer Vision and Image Understanding*, vol. 163, pp. 21–40, 2017.

[3] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering", in *CVPR*, vol. 1, 2017, p. 9.

[4] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning", in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, IEEE, 2017, pp. 1988–1997.

[5] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning", in *Advances in neural information processing systems*, 2017, pp. 4974–4983.

[6] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and vqa", *arXiv preprint arXiv:1707.07998*, 2017.

[7] D. Teney, P. Anderson, X. He, and A. v. d. Hengel, "Tips and tricks for visual question answering: Learnings from the 2017 challenge", *arXiv preprint arXiv:1708.02711*, 2017.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in *Advances in neural information processing systems*, 2015, pp. 91–99.

[9] A. K. Gupta, "Survey of visual question answering: Datasets and techniques", *arXiv preprint arXiv:1705.03865*, 2017.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", *arXiv preprint arXiv:1412.3555*, 2014.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge", *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[15] H. Xu and K. Saenko, "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering", in *European Conference on Computer Vision*, Springer, 2016, pp. 451–466.

[16] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, "Stacked attention networks for image question answering", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 21–29.

[17] Y. Zhu, J. J. Lim, and L. Fei-Fei, "Knowledge acquisition for visual question answering via iterative querying", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017.

[18] D. Yu, J. Fu, T. Mei, and Y. Rui, "Multi-level attention networks for visual question answering", in *Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2017, p. 8.

[19] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering", in *Advances In Neural Information Processing Systems*, 2016, pp. 289–297.

[20] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "Inferring and executing programs for visual reasoning", *arXiv preprint arXiv:1705.03633*, 2017.

[21] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer", *arXiv preprint arXiv:1709.07871*, 2017.

[22] A. Trott, C. Xiong, and R. Socher, "Interpretable counting for visual question answering", *arXiv preprint arXiv:1712.08697*, 2017.

[23] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Learning to compose neural networks for question answering", *arXiv preprint arXiv:1601.01705*, 2016.

[24] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko, "Learning to reason: End-to-end module networks for visual question answering", *CoRR, abs/1704.05526*, vol. 3, 2017.

[25] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks", in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[26] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks", in *Advances in Neural Information Processing Systems*, 2017, pp. 972–981.

[27] R. Girshick, "Fast r-cnn", *arXiv preprint arXiv:1504.08083*, 2015.

[28] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models", in *Proc. icml*, vol. 30, 2013, p. 3.

[29] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding", *arXiv preprint arXiv:1606.01847*, 2016.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context", in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[31] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation", in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[32] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks", in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[34] H. Ben-younes, R. Cadene, M. Cord, and N. Thome, "Mutan: Multimodal tucker fusion for visual question answering", in *The IEEE International Conference on Computer Vision (ICCV)*, vol. 1, 2017, p. 3.