# Dynamic Capsule Attention for Visual Question Answering

**Yiyi Zhou,**[1] **Rongrong Ji,**[1*] **Jinsong Su,**[2] **Xiaoshuai Sun,**[1] **Weiqiu Chen**[1]
[1]Fujian Key Laboratory of Sensing and Computing for Smart City, Department of Cognitive Science,
School of Information Science and Engineering, Xiamen University, China
[2]School of Software Engineering, Xiamen University, China
Peng Cheng Laboratory, China

## Abstract

In visual question answering (VQA), recent advances have well advocated the use of attention mechanism to precisely link the question to the potential answer areas. As the difficulty of the question increases, more VQA models adopt multiple attention layers to capture the deeper visual-linguistic correlation. But a negative consequence is the explosion of parameters, which makes the model vulnerable to over-fitting, especially when limited training examples are given. In this paper, we propose an extremely compact alternative to this static multi-layer architecture towards accurate yet efficient attention modeling, termed as *Dynamic Capsule Attention* (CapsAtt). Inspired by the recent work of *Capsule Network*, CapsAtt treats visual features as capsules and obtains the attention output via *dynamic routing*, which updates the attention weights by calculating coupling coefficients between the underlying and output capsules. Meanwhile, CapsAtt also discards redundant projection matrices to make the model much more compact. We quantify CapsAtt on three benchmark VQA datasets, *i.e.*, COCO-QA, VQA1.0 and VQA2.0. Compared to the traditional multi-layer attention model, CapsAtt achieves significant improvements of up to 4.1%, 5.2% and 2.2% on three datasets, respectively. Moreover, with much fewer parameters, our approach also yields competitive results compared to the latest VQA models. To further verify the generalization ability of CapsAtt, we also deploy it on another challenging multi-modal task of *image captioning*, where state-of-the-art performance is achieved with a simple network structure.

## Introduction

Visual question answering (VQA) has received extensive research attention in computer vision and artificial intelligence (Antol et al. 2015; Yang et al. 2016). Similar to other joint learning tasks of vision and language such as image captioning (IC), the typical setting of VQA involves the acquisition of correlated visual entities about the textual information. To this end, the attention mechanism (Bahdanau, Cho, and Bengio 2014) has been widely explored in VQA (Yang et al. 2016; Lu et al. 2016; Yu et al. 2017; Teney et al. 2018).

*Corresponding Author. E-mail:rrji@xmu.edu.cn

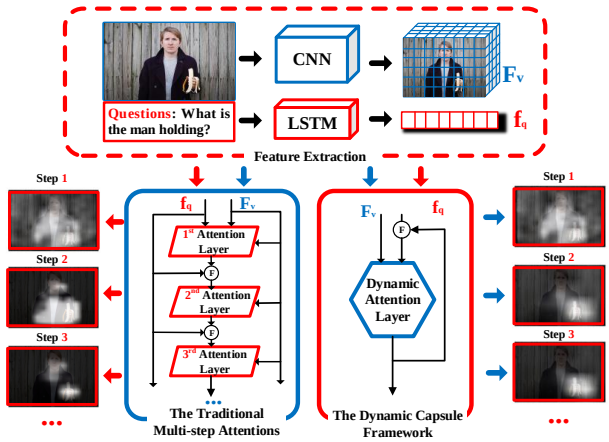




Figure 1: Comparison between the traditional multi-step attention and the proposed Dynamic Capsule Attention (CapsAtt) in Visual Question Answering. The "F" in cycles denotes feature fusions. The left part describes the procedure of the traditional multi-step attentions, while the right one depicts the framework of CapsAtt. Visualized attention results are given on two sides.

The principle behind attention mechanism is to mimic the process of human cognition, assigning limited attention resources to the most relevant information for a given task (Xu et al. 2015). In VQA, given a question feature, this mechanism forces the model to attend to the visual feature matrix and generates a probability distribution to denote the importance of each regional feature, based on which a weighted-sum vector is output for the answer prediction. As another appealing property, it requires no additional label information, and its gradient can be derived from the designed objective function (Xu et al. 2015). These advancements have pushed attention mechanism to various tasks such as neural machine translation (Bahdanau, Cho, and Bengio 2014; Vaswani et al. 2017), machine comprehension (Xiong and Socher 2016), and image captioning (Xu et al. 2015) etc.

However, the increasing difficulty of the question often goes beyond the capacity of a single-step attention. Therefore, recent models resort to multiple attention layers for capturing the deeper visual-linguistic correlation. (Yang et

al. 2016; Lu et al. 2016; Vaswani et al. 2017). Fig.1 shows a representative structure of the multi-step attention (Yang et al. 2016) in VQA. Along with the accuracy improvement, a consequent deficiency is the explosion of parameters, which also leads to a huge cost of computation. The main reason is that the fusion process in the attention layer typically contains large-scale parameters. For instance, the compact bilinear pooling used in MCB (Fukui et al. 2016), a benchmark model in VQA, has about 48 million parameters. In this case, the training deficiency as well as the model's robustness have become urgent issues.

We argue that, the key to solving these issues is to replace such a multi-layer attention with a more flexible structure. In particular, as a static stack of information refinements, this multi-layer attention should be replaced with a dynamic alternative. Meanwhile, inspired by the *Dynamic Routing* in the recent work of *CapsNet* (Sabour, Frosst, and Hinton 2017), in this paper we reveal the possibility of performing multi-step attentions in a dynamic one-component setting. In CapsNet, the routing algorithm updates the coupling coefficients between capsules in lower and upper layers via iterations, which further determine the contributions of input capsules. We have found that such an intention is similar to the attention mechanism for evaluating the importances of input visual signals.

To this end, we propose a dynamic algorithm towards robust and highly efficient attention, namely *Dynamic Capsule Attention* (CapsAtt)[1], as shown in Fig.1(right) and Fig.2.a. CapsAtt draws on the idea of *CapsNet* and treats vectors in the feature matrix as underlying capsules. The attention output is regarded as an overlying capsule obtained through dynamic interactions with these underlying capsules. In contrast to the traditional prediction-based calculation of attention weights, CapsAtt directly uses the coupling coefficients between modalities to determine the importances of input features. It also discards redundant projection matrices to make the model much more compact. Conclusively, CapsAtt has three major advantages:

- It performs the multi-step attentions by using only one attention layer and avoids the vanishing gradient problem of the traditional stacked attention layers.
- It greatly reduces the number of parameters, making models more compact and robust.
- It is general and applicable to most of the existing attention-driven models.

To validate the merits of the proposed CapsAtt, we first conduct extensive experiments on three VQA datasets, *i.e.*, COCO-QA (Ren, Kiros, and Zemel 2015), VQA1.0 (Antol et al. 2015) and VQA2.0 (Goyal et al. 2017). The experimental results show that, CapsAtt can obtain significant improvements on all three datasets compared with the classic multi-step attention model. With a much smaller number of parameters and a simple fusion strategy, our approach also achieves near or better performances compared with other benchmark models, like MCB (Fukui et al. 2016) or BUA (Teney et al. 2018). To further validate the generalization ability of CapsAtt, we also conducts experiments on COCO-Caption (Chen et al. 2015), where competitive performance is also obtained for the task of image captioning.

[1]https://github.com/XMUVQA/CapsAtt

## Related Work

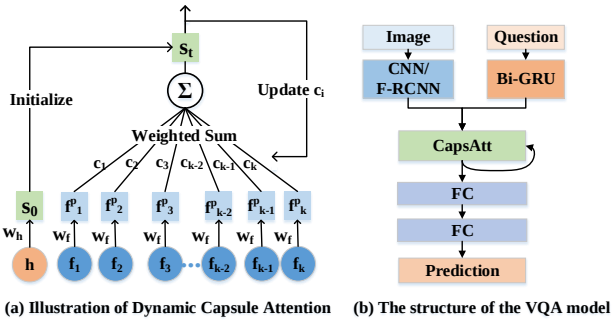
Attention mechanism is a process of mimicking human cognition that focuses on most related information to a vision and/or language modeling task. The concept of attention mechanism has been studied in many works (Tang, Srivastava, and Salakhutdinov 2014; Ba, Mnih, and Kavukcuoglu 2014; Mnih et al. 2014), among which the most widely-used structure might resort to that in (Bahdanau, Cho, and Bengio 2014). In the task of neural machine translation (NMT), (Bahdanau, Cho, and Bengio 2014) uses the hidden state of the decoder to attend to the collection of hidden states generated by the encoder, and then outputs a weighted-sum vector for word prediction. The attention mechanism was further introduced to the task of visual question answering (VQA) by (Xu and Saenko 2016; Yang et al. 2016). In VQA, the visual regional feature matrix becomes the collection of features to attend, and the reference vector is the question feature. However, the increasing difficulty of questions in VQA often exceeds the ability of a single-step attention. In this case, multiple-glimpse based attentions were further proposed to refine the visual/textual information, such as the multi-step attentions in (Yang et al. 2016; Vaswani et al. 2017) and the co-attentions in (Lu et al. 2016). Attention mechanism is also applied to many other tasks including image captioning (Xu et al. 2015; Lu et al. 2017), visual grounding (Fukui et al. 2016), video captioning (Song et al. 2017) and text comprehension (Xiong and Socher 2016).

## Dynamic Capsule Attention

### The Attention Mechanism

Since there are various attention mechanisms proposed in the literature, we do not intend to explore all potential designs of attention models in this paper, *e.g.*, the selection of fusion approaches or input features. Instead, we aim at showing how to use a single attention layer to perform multi-step attentions. Therefore, we take the most commonly used attention framework (Ba, Mnih, and Kavukcuoglu 2014; Xu et al. 2015; Yang et al. 2016) as baseline. Before introduction, definitions of *feature matrix* and *reference vector* are given first:

- *Feature matrix* refers to a collection of feature vectors to attend. In different tasks, its representation varies. In VQA (Yang et al. 2016) or image captioning (IC) (Xu et al. 2015), it is the feature maps of an image generated by a convolutional neural network. And in NMT (Bahdanau, Cho, and Bengio 2014), it is a collection of hidden states of the encoder.
- *Reference vector* is used to determine the attention weights of different vectors in the feature matrix. In VQA, it is the question feature, while in IC and NMT, it is the hidden state of the decoder in the previous step.




(a) Illustration of Dynamic Capsule Attention (b) The structure of the VQA model

Figure 2: Illustrations of the proposed Dynamic Capsule Attention (a) and the corresponding VQA model (b). In (a), $s$ is the output capsule which is initialized by the reference vector $h$. $f_i$ is the vector in the feature matrix $\mathbf{F}$. $\mathbf{W}_h$ and $\mathbf{W}_f$ are projection matrices. $c_i$ is the coupling coefficient which can be regarded as the attention weight. In (b), "FC" refers to the fully-connected layer.

**Problem Setup.** Given a feature matrix $\mathbf{F} \in \mathbf{R}^{n\times K}$, where $K$ denotes the number of feature vectors and $h \in \mathbf{R}^m$ is the reference vector, the attention based prediction aims at maximizing the following conditional probability:

$$P\left(y|h,\mathbf{F}\right) = g\left(h, f_a\right), \tag{1}$$

where $f_a \in \mathbf{R}^n$ is the attention feature which is a weighted-sum vector of $\mathbf{F}$.

Based on $h$, an attention model first generates a probability distribution on vectors $f_i \in \mathbf{R}^n$ of $\mathbf{F}$, and then obtains $f_a$. This process is formulated as:

$$f_a = \sum_i^K \alpha_i f_i, \tag{2}$$

where $\alpha_i$ is the attention weight of the $i$-th feature vector $f_i$ in $F_v$. It is often based on the joint representation of two modalities, and is computed by:

$$\begin{aligned} \alpha_i &= \frac{\exp\left(e_i\right)}{\sum_j^K \exp\left(e_j\right)}, \\ where \quad e_i &= s_s \cdot \mathcal{F}\left(\mathbf{h}, f_i\right). \end{aligned} \tag{3}$$

Here, $e_i$ is the log prior, $s_s \in \mathbf{R}^m$ is the weight vector, and $\mathcal{F}\left(\cdot\right)$ is any fusion methods, which often includes two weight matrices, $\mathbf{W}_h \in \mathbf{R}^{m\times d}$ and $\mathbf{W}_f \in \mathbf{R}^{n\times d}$, to project $h$ and $f_i$ onto the same semantic space.

**The Multi-Step Attention.** In the traditional setting of multi-step attentions (Yang et al. 2016), the model uses $f_a$ in the previous step to refine the reference vector $h$, and then performs attention again. The $t$-step attention result for the $i$-th feature in $\mathbf{F}$ is calculated by:

$$\begin{aligned} \alpha_i^t &= \frac{\exp\left(e_i^t\right)}{\sum_j^K \exp\left(e_j^t\right)}, \\ where \quad e_i^t &= \mathbf{W}_s^t \cdot \mathcal{F}\left(h_t, f_i\right), \\ h^t &= \mathcal{F}\left(h^{t-1}, f_a^{t-1}\right). \end{aligned} \tag{4}$$

**Algorithm 1** Dynamic Capsule Attention

**Input:** feature matrix $\mathbf{F}$ and reference vector $h$
**Output:** coupling coefficients $\mathbf{c}$ and the output capsule $s$

```
Initialize projection matrices, W_f and W_h
Project F and h, and obtain f_p^i ∈ F_p and h_p
Initialize s_0 with h_p
for t in N iterations do
    Obtain coupling coefficients: c_i ← softmax(b_i)
    Obtain weighted-sum feature: f_a^p ← Σ_i^K c_i f_p^i
    Update output capsule: s_t ← s_{t-1} + f_a^p
    Update agreements: b_i ← f_p^{i⊤} · s_t + b_i
end for
return c, s_N.
```

After that, the attention feature $f_a^t$ is computed again by Eq.2. Notably, weight matrices used in each layer are separated.

## Capsule Attention

To achieve the multi-step reasoning with *one* attention layer, we propose an iterative attention algorithm, termed as *Dynamic Capsule Attention* (CapsAtt). Inspired by the CapsNet, CapsAtt treats each vector in the feature matrix $\mathbf{F}$ as an underlying capsules. The attention output is the only upper capsule that contains information related to the prediction.

Concretely, the output capsule $s$ is initialized with the projected reference vector $h_p$, which is denoted as $s_0 = \sigma(\mathbf{W}_h h)$. In the $t$-th iteration, the output capsule is calculated by the following:

$$s_t = s_{t-1} + \sum_i^K c_i f_p^i, \quad f_p^i = \sigma\left(\mathbf{W}_f f_i\right), \tag{5}$$

where $c_i$ is the coupling coefficients indicating the contributions of the underlying capsules to the overlying one . Note that, the sum of coupling coefficients between capsules in the feature matrix and the output capsule adds up to 1. Therefore, we treat them as the attention weights over $\mathbf{F}$.

The coupling coefficient $c_i$ is determined by a "routing softmax" whose initial logit is denoted as $b_i$. $b_i$ is the log prior probability that the capsule $i$ should be coupled to the output capsule $s$. Then, the coefficients is calculated by:

$$c_i = \frac{\exp\left(b_i\right)}{\sum_j^K \exp\left(b_j\right)}. \tag{6}$$

The log prior is initialized with zero and then updated by adding agreements between the underlying capsules and the output capsule, which is calculated by:

$$b_i = bi + f_p^{i\top} \cdot s_{t-1}. \tag{7}$$

These agreements are added to log priors after each routing. The detailed process is presented in Alg.1.

In *CapsNet*, $s$ is considered as the total input of a capsule and is further "squashed" using a non-linear function to obtain a vector output $v$ (Sabour, Frosst, and Hinton 2017). In

contrast, we use $s$ directly as the capsule output in CapsAtt, which is the main difference between these two works. To explain, *CapsNet* uses the length of $v$ to represent the probability that the entity represented by the capsule is presented in the current input (Sabour, Frosst, and Hinton 2017). In CapsAtt, the output vector is to represent both the feature matrix and the reference vector. Therefore, the information contained in this vector is highly correlated with the final prediction task. Accordingly, we discard the "squashing" function that may introduce information loss. The output capsule $s$ is further used for model predictions, and the gradients of CapsAtt can be calculated by using a standard back-propagation.

## CapsAtt for VQA

We first apply the proposed *CapsAtt* to the task of *visual question answering* (VQA), the structure of which is depicted in Fig.2.b. In VQA, the training example is a triplet $(I, Q, a)$ , where $I$ denotes the image, $Q$ is the question and $a$ is the corresponding answer. The image $I$ is first processed by a convolution neural network (CNN) or a Faster R-CNN network (Ren et al. 2017) to obtain the regional visual feature matrix, denoted as $\mathbf{F} \in \mathbf{R}^{n\times K}$. Then we use a bi-directional GRU network to process the input question and obtain the question feature $h \in \mathbf{R}^m$, which is the concatenation of the last hidden states of forward and backward GRUs.

The visual and textual features are fed to CapsAtt to perform attentions. The output capsule from CapsAtt is used as the joint representation of two modalities, and two fully-connected layers are followed before the final prediction.

On datasets like COCO-QA where each question has only one answer, we use the softmax cross-entropy as the loss function. When a question is associated with a list of answers, such as that in VQA1.0 and VQA2.0, we treat the prediction as a multi-label classification by following the setting in (Teney et al. 2018). The loss function is then defined as:

$$\mathcal{L} = \sum_{i}^{N} y_i \log(p_i) - (1 - y_i) \log(p_i), \tag{8}$$

where $p_i$ is the predicted probability of the $i$-th candidate and $y_i = 1$ if the $i$-th candidate is in the answer set .

**Applying CapsAtt for Image Captioning.** To examine the generation ability of CapsAtt, we further apply it to the task of *image captioning* (IC). The input image is also processed by CNN/Faster R-CNN to obtain the visual feature matrix. The language decoder is an LSTM unit, and we use its hidden state of the current step, $h_t$, as the reference vector. Based on $h_t$, CapsAtt attends to the visual feature matrix to produce the output capsule $s$, which is followed by a FC layer and then is used to predict the word $y_t$ of the current step. Following the setting in (Xu et al. 2015), the loss function is the averaged softmax cross entropy of all steps.

# Experiments

The main purpose of our experiments is to prove that the proposed CapsAtt can achieve multi-step attentions with only one attention layer, and obtain better performance than the traditional multi-step attention models, such as the stacked attention network (SAN) in (Yang et al. 2016). Besides, we also compare CapsAtt with the state-of-the-arts.

Table 1: Evaluation results on COCO-QA.

| Method | Objects ( 70%) | Number (7%) | Color (17%) | Location (6%) |
|---|---|---|---|---|
| SAN-1layer (LSTM) | 62.5 | 49.0 | 54.8 | 51.6 |
| SAN-2layer (LSTM) | 63.6 | 49.8 | 57.9 | 52.8 |
| SAN-1layer (CNN) | 63.6 | 48.7 | 56.7 | 52.8 |
| SAN-2layer (CNN) | 64.5 | 48.6 | 57.9 | 54.0 |
| CapAtt-Iter1 | 64.4 | 49.2 | 57.0 | 54.2 |
| CapAtt-Iter2 | 64.7 | 49.5 | 59.5 | 55.8 |
| CapAtt-Iter3 | **65.0** | **50.2** | 61.3 | **56.3** |
| CapAtt-Iter4 | 64.8 | 50.0 | **62.0** | 56.0 |

*The number under the question type indicates the percentage of this type in the dataset.

## Datasets

COCO-QA (Ren, Kiros, and Zemel 2015) is a VQA dataset which is automatically generated by image captions of MS COCO images. It contains 123,287 images with 78,736 training questions and 38,948 test questions. All questions can be divided into four types, namely *object*, *quantity*, *color* and *location*. Each answer contains only one word. The metric we used is the classification accuracy. VQA1.0 dataset contains 200,000 natural images from MS-COCO with 614,153 human annotated questions in total. Each question has 10 free-response answers. The whole dataset is divided into three splits, in which there are 248,349 examples for training, 121,512 for validation, and 244,302 for testing. VQA2.0 is developed based on VQA1.0, and its images are also from MS-COCO. It has about 1,105,904 image-question pairs, in which 443,757 examples are for training, 214,254 are for validation, and 447,793 are for testing. On these two datasets, we evaluate all methods under the mode of *Open-ended* which means no answer options are given during training and testing. The metric we used is the *VQA accuracy* proposed in (Antol et al. 2015).

## Baselines

We mainly compare our CapsAtt model with the classic multi-step attention model proposed in (Yang et al. 2016), denoted as SAN. Besides, on VQA1.0 and VQA2.0 datasets, we also compare CapsAtt with a set of the sate-of-the-art models including HiCoAtt (Lu et al. 2016), ASK (Wu et al. 2016), VQA Machine (VQA-M) (Wang et al. 2017), MCB (Fukui et al. 2016), MLB (Kim et al. 2017), MFB (Yu et al. 2017) and BUA (Teney et al. 2018) *etc*. HiCoAtt is a model that performs co-attentions on both visual and text features. ASK and VQA-M are methods that apply external knowledge or techniques to the VQA learning. MCB, MLB and MFB are two-glimpse attention models using bilinear pooling based fusion approaches for capturing interactions between two modalities, and they all show most advanced performances in VQA. BUA are the winner of *VQA 2017*

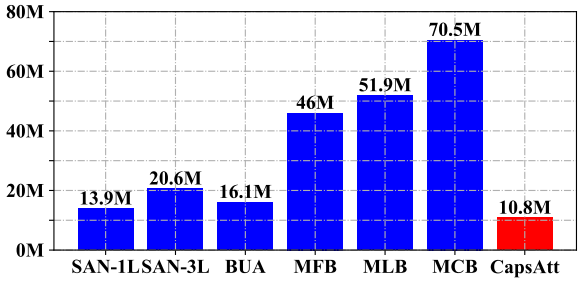



Figure 3: Comparisons of the parameter sizes of different models. "SAN-1L" and "SAN-3L" dnotes SAN models with one attention layer and three attention layers, respectively. CapsAtt achieve the state-of-art performances with only 10.8 million parameters.

*Challenge*, which is equipped with various tricks for the performance improvement, and uses the F-RCNN features from (Anderson et al. 2018) as the visual input.

## Model Configurations

On COCO-QA, the visual feature used is the convolution feature map before the first fully-connected layer of VGG-16 (Simonyan and Zisserman 2014) with a size of $14 \times 14 \times 512$. The dimensions of forward and backward GRU units are set to 256. So the whole dimension of the output question feature is 512. The attention dimension in CapsAtt is set to 512, and the ones of the two FC layers are both 512 as well. The number of answer category is set to 434. The initial learning rate is 7e-4, which is half decayed after each 10,000 training steps. On VQA1.0 and VQA2.0, the visual feature input is the feature map before the last pooling layer of ResNet-152 (He et al. 2016) with a size of $14 \times 14 \times 2,048$. We also use the Faster RCNN features from (Anderson et al. 2018) with a size of $36 \times 2,048$, which is labeled as "FR-CNN" to distinguish. The dimensions of GRU units, CapsAtt and FC layers are set to 512, 1,024, 2,048 respectively. The answer dimensions on these two datasets are both set to 3,000. The initial learning rate is set to 7e-4 with a decay step of 25,000 and a decay rate of 0.5, while the batch size is 124. The maximum training step is set to 200,000 and the validation step is 5,000 steps. Early stop is applied when the performance does not increase after 5 validations.

## Quantitative Analysis

We first compare CapsAtt with SAN on three VQA datasets. Tab.1 shows the comparison results on the COCO-QA dataset. From Tab.1 we can see that under the similar experimental setup, the proposed CapsAtt has made significant improvements in all four tasks, especially the "color" and the "location" with improvements up to 4.1% and 2.3%, respectively. Notably, the training examples of these two tasks take account for only a small proportions of the entire dataset, which suggests the robustness of CapsAtt. Tab.2 displays the comparisons on VQA1.0 and VQA2.0 dataset, and the improvements are more significant. The overall improvements

Table 2: Comparisons with the state-of-the-arts on the *Open-Ended* task of VQA1.0 and VQA2.0. Tested on the *test-dev* split.

| **VQA1.0** | Others | Num. | Yes/No | All |
|---|---|---|---|---|
| SMem (Xu and Saenko 2016) | 46.1 | 36.6 | 80.9 | 58.0 |
| NMN (Andreas et al. 2016) | 44.0 | 38.0 | 81.2 | 58.6 |
| SAN-2layer (Yang et al. 2016) | **46.1** | **36.6** | **79.3** | **58.7** |
| ASK (Wu et al. 2016) | 45.2 | 38.4 | 81.0 | 59.2 |
| DMN (Xiong and Socher 2016) | 48.3 | 38.3 | 80.5 | 60.3 |
| HiCoAtt (Lu et al. 2016) | 51.7 | 38.9 | 79.4 | 61.8 |
| VQA-M (Wang et al. 2017) | 53.0 | 38.4 | 81.5 | 63.1 |
| MCB+att (Fukui et al. 2016) | 54.8 | 37.7 | 82.2 | 64.2 |
| MLB (Kim et al. 2017) | 54.7 | 37.9 | 84.0 | 65.0 |
| MFB+Att (Yu et al. 2017) | 55.2 | 38.3 | 82.5 | 64.6 |
| CapsAtt-Iter2 | **53.5** | 39.5 | 81.7 | 63.7 |
| CapsAtt-Iter3 | 53.3 | **39.7** | **82.8** | **63.9** |
| CapsAtt-Iter3+FRCNN* | 56.4 | 42.4 | 83.7 | 66.1 |
| **VQA2.0** | Others | Num. | Yes/No | All |
| LSTM-Q (Goyal et al. 2017) | 41.8 | 35.2 | 41.8 | 54.2 |
| SAN-2layer (LSTM) | **50.1** | **37.5** | **78.34** | **58.9** |
| MCB (Fukui et al. 2016) | 53.3 | 38.2 | 78.8 | 62.3 |
| BuA+VG (Teney et al. 2018) | 52.6 | 39.5 | 79.2 | 62.1 |
| BuA+VG+FRCNN* | 56.1 | 44.2 | 81.8 | 65.3 |
| CapsAtt-Iter2 | 51.4 | 40.7 | 79.37 | 61.7 |
| CapsAtt-Iter3 | **51.7** | **41.1** | **79.7** | **62.1** |
| CapsAtt-Iter3+FRCNN* | 55.5 | 45.1 | 82.6 | 65.5 |

* -FRCNN denotes that the use of regional features from (Anderson et al. 2018).

on these two datasets are up to 5.2% and 2.2%, respectively. Under a similar setting, these results are quite significant.

We explain these obvious improvements in two aspects. The first is that the traditional attention method uses a linear layer to predict the importance of each region based on the fused features, and then obtain the attention weights via a Softmax layer. However, in CapsAtt, the importance of each region is directly determined by the coefficient between its feature and the reference vector, rather than by predictions. So when the multi-modal space is well learned, the measurement of attention weights will be more accurate by nature. The other reason is that CapsAtt only uses one attention layer with only two projection matrices for multi-step reasoning. Such a design not only reduces the number of parameters, but also avoids the semantic inconsistency among multi-modal spaces of different layers.

Next, we compare CapsAtt with the state-of-the-art models on VQA1.0 and VQA2.0. As shown in Tab.2, CapsAtt achieves state-of-the-art performances with a simple structure and much fewer parameters on both two datasets. For models that use external knowledge or incorporating advance CV or NLP techniques, *e.g.*, AKS and VQA-M, the improvements of CapsAtt are still significant. Compared with attention models with simple fusion approaches (element-wise product or addition), *e.g.*, SAN, HiCoAtt, SMem (Xu et al. 2015) and DMN (Xiong and Socher 2016), CapsAtt reports a much superior performance. For optimal performance, MCB, MLB and MFB also use multi-glimpse

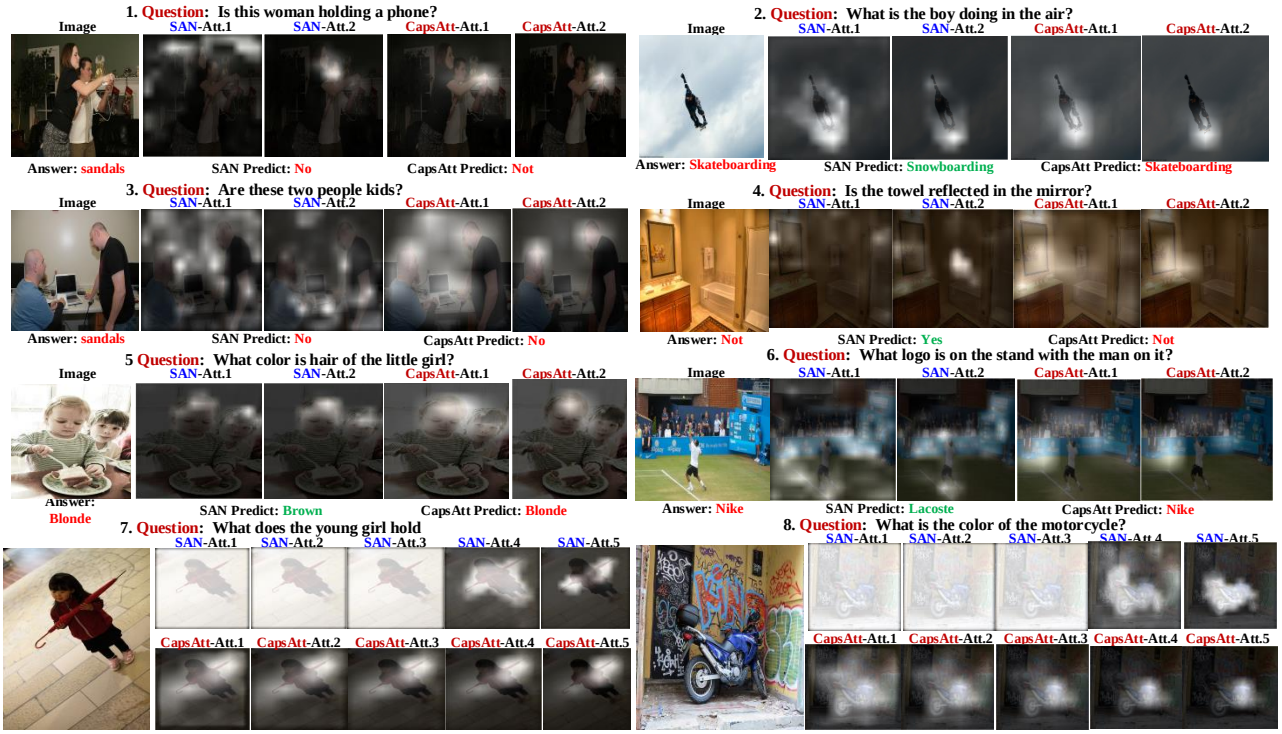



Figure 4: The visualized attentions of SAN and the proposed CapsAtt. Comprared to SAN, our CapsAtt can locate the potential answer area more quickly (1-3) and precisely (4-6), and avoid the vanishing gradient problem (7-8).

Table 3: Evaluations of Caps-Iter3 with different dimension settings on VQA1.0 *test-dev* set. Trained on the *train+val* split with the *FRCNN* features. $a$, $f$ and $g$ are dimensions of attention, fully-connected layers and the GRU unit.

| Setting | Acc. | Para.Size | Model Size |
|---|---|---|---|
| $a = 1024, f = 1024, g = 512$ | 66.10 | 10.8M | 183mb |
| $a = 2048, f = 2048, g = 1024$ | 66.18 | 25.6M | 403mb |
| $a = 1024, f = 2048, g = 512$ | 65.94 | 18.1M | 305mb |
| $a = 512, f = 512, g = 256$ | 65.22 | 4.2M | 113mb |
| $a = 256, f = 512, g = 256$ | 64.28 | 3.4M | 104mb |

"M" is the unit of the parameter size which is one million, and "mb" denotes the saved model size which is one mega byte[2].

Table 4: Performances on the test portion of Karpathy *et al.* (Karpathy and Feifei 2015)'s splits on COCO dataset. $\mathcal{V}$, $\mathcal{R}$ and $\mathcal{F}$ denote using features of VGG, ResNet and FR-CNN, respectively.

| Method | BLEU-1 | BLEU-4 | METEOR | CIDEr |
|---|---|---|---|---|
| Karpathy et.al. 2015$^{\mathcal{V}}$ | 62.5 | 23.0 | 19.5 | 66 |
| Xu et.al. 2015$^{\mathcal{V}}$ | **70.9** | **24.3** | **23.9** | - |
| Lu et.al. 2017 (Spatial) $^{\mathcal{R}}$ | 73.4 | 32.5 | 25.1 | 99 |
| Lu et.al. 2017 (Adapt)$^{\mathcal{V}}$ | 74.2 | 33.2 | 26.6 | 108 |
| Lu et.al. 2018$^{\mathcal{F}}$ | 75.5 | 34.7 | 27.1 | 107 |
| CapsAtt-Iter2 $^{\mathcal{V}}$ | 71.3 | **30.5** | **24.6** | 94 |
| CapsAtt-Iter3 $^{\mathcal{V}}$ | **71.7** | 29.4 | 24.5 | 95 |
| CapsAtt-Iter2 $^{\mathcal{F}}$ | 75.6 | 34.0 | 26.8 | 108 |

attention methods, but their main advantage is their bilinear-pooling based fusions, which have been shown to effectively capture interactions between two type of features (Gao et al. 2016). Even so, with the same visual input, their advantages are still margin. Notably, the parameter size of CapsAtt is much smaller than theirs, as shown in Fig.3. Taking the benchmark model MCB for example, the parameter amount of CapsAtt is only about 15% of its amount. When equipped with the Faster R-CNN features from (Anderson et al. 2018), CapsAtt achieves the best performances on both two datasets. On VQA2.0, the performance and the parameter size of BUA (Teney et al. 2018), *i.e.*, the winner of *VQA challenge 2017* is close to those of CapsAtt. Compared with BUA, CapsAtt merits in requiring no complex pre-processing, *e.g.*, pretraining the classifier, and additional training examples. Overall, with a simple structure and a small parameter size, CapsAtt still achieves advanced performances in VQA.

We also record the performances of different model settings and the corresponding sizes of model parameters, the results of which are shown in Tab.3. The first observation is that the effects of parameter size to CapsAtt is small. The performance gap between the largest model and the smallest one is only 1.82. More importantly, even with a small


Figure 5: Visualizations of the attention results of CapsAtt and *ShowAttendTell* when predicting the underlined words. Compared with ShowAttendTell, CapsAtt focus on the described objects more precisely and generates more accurate descriptions.

amount of parameters (3.3 millions), CapsAtt still has a good accuracy, while its model size is only about 100mb. Such a result not only proves the efficiency and robustness of our algorithm, but also provides good evidence in applying VQA into resource (memory)-limited scenarios.

## Qualitative Analysis

We visualize the attention results of the proposed CapsAtt and the traditional multi-step attention model SAN, which are given in Fig.4. Eg.(1)-(6) shows the comparisons of two-step attentions generated by CapsAtt and SAN. From these examples, we first observe that both two approaches shows a step-by-step reasoning during attentions. A slightly difference is that, when the question is easy or the answer entity in image is certain, CapsAtt can locate the potential answer areas more quickly compared to SAN, *e.g.*, Eg.(1)-Eg.(3). The other observation is that, compared with SAN, CapsAtt shows a better ability of modeling the correlation between the question content and visual information. Taking the Eg.(4)-(5) for examples, CapsAtt understands the recognition conditions in questions well and finds the corresponding visual entities for answers, accurately. In contrast, the focus of SAN is obviously not in line the purpose of the given questions. Eg.(7)-(8) displays the 5-step attentions of CapsAtt and SAN. Unexpectedly, the first three steps of attentions of SAN are basically inactive. To explain, SAN is built with attention layers and has a structure similar to the forward network. When the number of attention layers grows, the gradients become more difficult to be back-propagated. Meanwhile, this does not happen in SAN that uses three or fewer attention layers. This indicates that CapsAtt can help models to be free from the vanishing gradient problem.

## Extending CapAtt to Image Captioning

**Dataset.** COCO-captions (Ren, Kiros, and Zemel 2015) is the largest image captioning dataset at present. It has 82,783, 40,504 and 40,775 images for training, validation and test, respectively. Each image has 5 human annotated captions.

**Model Configurations.** VGG-16 and ResNet-152 are both used as the visual backbone. We also test the Faster-RCNN features from (Anderson et al. 2018) which is labeled as "FRCNN". When using VGG-16, the dimensions of LSTM and CapsAtt is set to 512. For ResNet and FRCNN features, we set the dimensions of LSTM and CapsAtt to both 1,024. During training, the learning rate is 4e-4 for the language model and 1e-4 for the visual backbones. The batch size is 80 while the max number of training epoch is 40. The optimizer is Adam, while gradient clip is set to 5.0. For CNN backbones, their parameters are fixed in the first 20 epochs, and after that they will be trained together with the language model. Early stop is also applied. We test the model with beam sizes of 1,2,3 and 7, and chose the best results.

**Analysis.** We further evaluate the proposed CapsAtt on the COCO-Caption dataset. The results are shown in Tab.4. We first observe that CapsAtt surpasses the classic attention model (Xu et al. 2015) that uses one attention layer, which confirms the generalization of our algorithms. When equipped with the FRCNN features, CapsAtt is able to achieve superior performances than the most recent works that have a similar structure. Fig.5 shows the visualized attention results of CapsAtt and the classic attention model (Xu et al. 2015) in IC, denoted as *ShowAttTell*. As can be seen from these visualizations, compared with ShowAttTell, CapsAtt can locate the corresponding visual entity more accurately when predicting the next word. Meanwhile, the qualities of generated by CapsAtt is better than those of ShowAttTell. Conclusively, these experimental results confirm the generalization fo CapsAtt.

## Conclusion

In this paper, we proposed a compact and robust attention algorithm for visual question answering, termed as Dynamic Capsule Attention (CapsAtt). Inspired by *Capsule Networks*, CapsAtt treats visual features as underlying capsules and obtains the attention output via *dynamic routing*. Its novelty also includes the elimination of redundant projection matrices and the measurement of attention weights based on the coupling coefficients. Meanwhile, it replaces the traditional multi-step attentions with only one dynamic layer, and shows a better efficiency and robustness in capturing the deep visual-linguistic correlation. We conduct extensive experiments on three VQA benchmark datasets, and com-

pare CapsAtt with the classic multi-step attention model as well as a set of the state-of-the-arts. Experimental results not only shows the obvious improvements of CapsAtt over the traditional attention methods, but also proves that CapsAtt can achieve the most advanced performances with a simple network structure and a small number of parameters. To examine its generalization ability, we apply CapsAtt to image captioning and also obtain competitive performances.

## Acknowledgments.

This work is supported by the National Key R&D Program (No.2017YFC0113000, and No.2016YFB1001503), Nature Science Foundation of China (No.U1705262, No.61772443, and No.61572410), Post Doctoral Innovative Talent Support Program under Grant BX201600094, China Post-Doctoral Science Foundation under Grant 2017M612134, Scientific Research Project of National Language Committee of China (Grant No. YB135-49), and Nature Science Foundation of Fujian Province, China (No. 2017J01125 and No. 2018J01106).

## References

Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; and Zhang, L. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, volume 3, 6.

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016. Neural module networks. *computer vision and pattern recognition* 39–48.

Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Lawrence Zitnick, C.; and Parikh, D. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, 2425–2433.

Ba, J.; Mnih, V.; and Kavukcuoglu, K. 2014. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Chen, X.; Fang, H.; Lin, T.-Y.; Vedantam, R.; Gupta, S.; Dollár, P.; and Zitnick, C. L. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Fukui, A.; Park, D. H.; Yang, D.; Rohrbach, A.; Darrell, T.; and Rohrbach, M. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *empirical methods in natural language processing* 457–468.

Gao, Y.; Beijbom, O.; Zhang, N.; and Darrell, T. 2016. Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 317–326.

Goyal, Y.; Khot, T.; Summers-Stay, D.; Batra, D.; and Parikh, D. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, volume 1, 3.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Karpathy, A., and Feifei, L. 2015. Deep visual-semantic alignments for generating image descriptions. *computer vision and pattern recognition* 3128–3137.

Kim, J.; On, K. W.; Lim, W.; Kim, J.; Ha, J.; and Zhang, B. 2017. Hadamard product for low-rank bilinear pooling. *international conference on learning representations*.

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

Lu, J.; Xiong, C.; Parikh, D.; and Socher, R. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *computer vision and pattern recognition* 3242–3250.

Mnih, V.; Heess, N.; Graves, A.; et al. 2014. Recurrent models of visual attention. In *Advances in neural information processing systems*, 2204–2212.

Ren, S.; He, K.; Girshick, R. B.; and Sun, J. 2017. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39(6):1137–1149.

Ren, M.; Kiros, R.; and Zemel, R. 2015. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*, 2953–2961.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, 3859–3869.

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

Song, J.; Guo, Z.; Gao, L.; Liu, W.; Zhang, D.; and Shen, H. T. 2017. Hierarchical lstm with adjusted temporal attention for video captioning. *arXiv preprint arXiv:1706.01231*.

Tang, Y.; Srivastava, N.; and Salakhutdinov, R. R. 2014. Learning generative models with visual attention. In *Advances in Neural Information Processing Systems*, 1808–1816.

Teney, D.; Anderson, P.; He, X.; and Den Hengel, A. V. 2018. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *computer vision and pattern recognition*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, P.; Wu, Q.; Shen, C.; and Den Hengel, A. V. 2017. The vqa-machine: Learning how to use existing vision algorithms to answer new questions. *computer vision and pattern recognition* 3909–3918.

Wu, Q.; Wang, P.; Shen, C.; Dick, A.; and van den Hengel, A. 2016. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4622–4630.

Xiong, Caiming, M. S., and Socher, R. 2016. Dynamic memory networks for visual and textual question answering. *arXiv* 1603.

Xu, H., and Saenko, K. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *european conference on computer vision* 451–466.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A. C.; Salakhutdinov, R.; Zemel, R. S.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, 77–81.

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

Yu, Z.; Yu, J.; Fan, J.; and Tao, D. 2017. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *Proc. IEEE Int. Conf. Comp. Vis*, volume 3.