Neural Module Networks

Zhang, Liqiang

June 12, 2018

Abstract

Visual question answering is fundamentally compositional in natureła question like where is the dog? shares substructure with questions like what color is the dog? and where is the cat? This paper seeks to simultaneously exploit the representational capacity of deep networks and the compositional linguistic structure of questions. The author describe a procedure for constructing and learning neural module networks, which compose collections of jointly-trained neural modules into deep networks for question answering. The approach decomposes questions into their linguistic substructures, and uses these structures to dynamically instantiate modular networks.

1. Introduction

This paper describes an approach to visual question answering based on a new model architecture that we call a neural module network (NMN) [1]. This architecture makes it possible to answer natural language questions about images using collections of jointly-trained neural modules, dynamically composed into deep networks based on linguistic structure.

In this paper the author draw from both lines of research, presenting a technique for integrating the representational power of neural networks with the flexible compositional structure afforded by symbolic approaches to semantics. Rather than relying on a monolithic network structure to answer all questions, the approach assembles a network on the fly from a collection of specialized, jointly-learned modules (Fig. 1). Rather than using logic to reason over truth values, the representations computed by this model remain entirely in the domain of visual features and attentions [2].

The approach first analyzes each question with a semantic parser, and uses this analysis to determine the basic computational units needed to answer the question, as well as the relationships between these units. In Fig. 1, the author first produce an attention focused on the dog, which passes its output to a location describer. Depending on the underlying structure, these messages passed between modules may

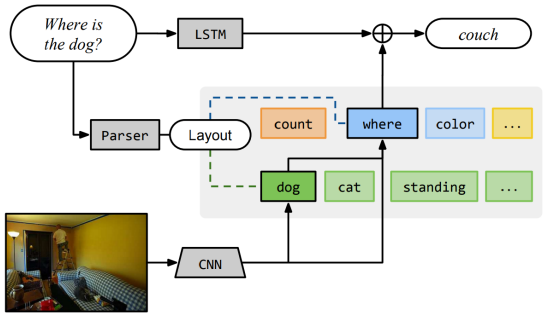

Figure 1. A schematic representation of the proposed modelłthe shaded gray area is a neural module network of the kind introduced in this paper. This approach uses a natural language parser to dynamically lay out a deep network composed of reusable modules. For visual question answering tasks, an additional sequence model provides sentence context and learns common-sense knowledge.

be raw image features, attentions, or classification decisions; each module maps from specific input to output types.

2. Method

At the heart of all motion estimation methods is the minimization of the difference between features extracted at a certain location x, y at the reference frame t and its correspondence in the frame $t + dt$. The now classical Horn and Schunck method penalizes the deviation from the assumption of constant intensity, that states that the intensity at a pixel in the reference frame at time t and the intensity at its correspondence at time $t + dt$ are the same. Formally, the goal is the minimization of the motion compensated intensity differences, that is,

$$\sum_{x,y=1}^{M} |I(x+u(x,y), y+v(x,y), t+\Delta t) - I(x,y,t)|^2 \quad (1)$$

where $I(x, y, t)$ is the intensity at pixel (x, y) at frame t, and $F(x, y) \triangleq [u(x, y), v(x, y)]$ is the unknown motion vector at pixel x, y. Clearly, $u(x, y)$ and $v(x, y)$ are respectively the horizontal and vertical displacement of the pixel with

coordinates (x, y). To arrive at a computationally tractable method, Horn and Schunck introduced a regularization term that penalized discontinuities in the motion field and linearized the cost by taking the first order Taylor expansion with respect to the horizontal and vertical displacements. By doing the latter, they arrived at the optical flow equation $uI_x + vI_y + I_t = 0$, where I_x, I_y and I_t are the horizontal, vertical and temporal intensity derivatives respectively, and penalized deviations from it. In the equation above, we omit the pixel coordinates for notation simplicity.

In order to reduce the influence of outliers the author use a robust error norm, that is the Charbonnier penalty $\rho = \sqrt{x^2 + \epsilon}$, a differentiable variant of the $L1$ norm, the most robust convex function. Formally, during training we learn the CNN by optimizing the sum of costs that for a pair of images $I(t)$ and $I(t + dt)$ are defined as follows:

$$E(F) = \sum_{x,y=1}^{M} \sqrt{(uI_x + vI_y + I_t)^2 + \epsilon} \qquad (2)$$

where the image coordinates x, y are omitted for notation simplicity.

3. Result

	test-dev				test
	Yes/No	Number	Other	All	All
LSTM	78.7	36.6	28.1	49.8	-
VIS+LSTM	78.9	35.2	36.4	53.7	54.1
ATT+LSTM	80.6	36.4	42.0	57.2	-
NMN	70.7	36.8	39.2	54.8	-
NMN+LSTM	81.2	35.2	43.3	58.0	-
NMN+LSTM+FT	81.2	38.0	44.0	58.6	**58.7**

Table 1. Results on the VQA test server. LSTM is a questiononly baseline, VIS+LSTM is a previous baseline that combines a question representation with a representation of the full image, and ATT+LSTM is a model with the same attentional structure as the approach but no lexical information. NMN+LSTM is the full model shown in Fig. 1, while NMN is an ablation experiment with no whole-question LSTM [3]. NMN+LSTM+FT is the same model, with image features fine-tuned on MSCOCO captions. This model outperforms previous approaches, scoring particularly well on questions not involving a binary decision.

Results are shown in Table 1. The author compare to a number of baselines, including a text-only baseline, a previous baseline approach that predicts answers directly from an encoding of the image and the question , and an attentional baseline.

References

[1] S. Antol, A. Agrawal, J. Lu, and M. Mitchell. VQA: Visual question answering. In *ICCV*, 2017. 1

[2] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 1

[3] M. Maire, S. Belongie, J. Hays, P. Perona, and P. Ramanan. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 2