Image Question Answering using Convolutional Neural Network with Dynamic Parameter Prediction II

Liangjie Cao

Jun 26, 2018

1. Parameter Hashing

The weights in the dynamic parameter layers are determined based on the learned model in the parameter prediction network given a question. The most straightforward approach to obtain the weights is to generate the whole matrix $W_d(q)$ using the parameter prediction network. However, the size of the matrix is very large, and the network may be overfitted easily given the limited number of train- ing examples. In addition, since we need quadratically more parameters between GRU and the fully-connected layer in the parameter prediction network to increase the dimensionality of its output, it is not desirable to predict full weight matrix using the network. Therefore, it is preferable to construct $W_d(q)$ based on a small number of candidate weights using a hashing trick. The authors employ the recently proposed random weight sharing technique based on hashing [1] to construct the weights in the dynamic parameter layer. Specifically, a single param- eter in the candidate weight vector p is shared by multiple elements of $W_d(q)$, which is done by applying a predefined hash function that converts the 2D location in $W_d(q)$ to the $1D$ index in p By this simple hashing trick, they can reduce the number of parameters in $W_d(q)$ while maintaining the accuracy of the network [1].

Let ω^d_{mn} be the element at (m, n) in $W_d(q)$, which corresponds to the weight between m_th output and n_th input neuron. Denote by $\psi(m, n)$ a hash function mapping a key (m, n) to a natural number in 1, . . . , K, where K is the dimensionality of p. The final hash function is given by

$$\omega^d_{mn} = p_\psi(m, n) \cdot \xi(m, n) \qquad (1)$$

where $\xi(m, n)$: N N $\rightarrow$ {+1, 1} is another hash function independent of $\psi(m, n)$. This function is useful to remove the bias of hashed inner product [1].

The authors believe that it is reasonable to reduce the number of free parameters based on the hashing technique as there are many redundant parameters in deep neural networks and the network can be parametrized using a smaller set of candidate weights. Instead of training a huge number of parameters without any constraint, it would be advantageous practically to allow multiple elements in the weight matrix to share the same value. It is also demonstrated that thennumber of free parameter can be reduced substantially with little loss of network performance [1].

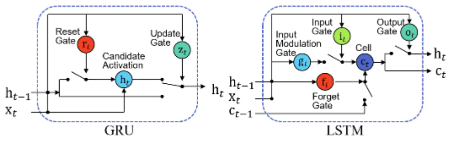

Figure 1. Comparison of GRU and LSTM. Contrary to LSTM that contains memory cell, GRU updates the hidden state directly

As illustrated in Figure 1, such dependency is captured by adaptively updating its hidden states with gate units. How- ever, contrary to LSTM, which maintains a separate mem- ory cell explicitly, GRU directly updates its hidden states with a reset gate and an update gate. The detailed proce- dure of the update is described below.

2. Training Algorithm

To deal with the deficiency of linguistic information in ImageQA problem, we transfer the information acquired from a large language corpus by fine-tuning the pre-trained embedding network. We initialize the GRU with the skip-thought vector model trained on a book-collection corpus containing more than 74M sentences [2]. Note that the GRU of the skip-thought vector model is trained in an unsupervised manner by predicting the surrounding sentences from the embedded sentences. As this task requires to understand context, the pre-trained model produces a generic sentence embedding, which is difficult to be trained with a limited number of training examples. By fine-tuning our GRU initialized with a generic sentence embedding model for ImageQA, we obtain the representations for questions that are generalized better.

3. Fine-tuning CNN

It is very common to transfer CNNs for new tasks in classification problems, but it is not trivial to fine-tune the CNN in our problem. They observe that the gradients below the dynamic parameter layer in the CNN are noisy since its weights are predicted by the parameter prediction network. Hence, a nave approach to fine-tune the CNN typically fails to improve performance, and they employ a slightly different technique for CNN fine-tuning to sidestep the observed problem. We update the parameters of the network using new datasets except the part transferred from VGG 16-layer net at the beginning, and start to update the weights in the subnetwork if the validation accuracy is saturated.

4. Evaluation Metrics

DAQUAR and COCO-QA employ both classification accuracy and its relaxed version based on word similarity, WUPS [3]. It uses thresholded Wu-Palmer similarity based on WordNet taxonomy to compute the similarity between words. For predicted answer set A^i and ground- truth answer set T^i of the i^{th} example, WUPS is given by

$$WUPS = \frac{1}{N}\sum_{i=1}^{N} \min\{\prod_{a\in A^i} \max_{t\in T_i} \mu(a,t), \prod_{t\in T^i} \max_{a\in A_i} \mu(a,t)\} \tag{2}$$

where $\mu(\cdot,\cdot)$ denotes the thresholded Wu-Palmer similarity between prediction and ground-truth. The authors use two threshold values (0.9 and 0.0) in our evaluation.

References

[1] W. Chen, S. Tyree, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *ICCM*, 2015. 1

[2] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-thought vectors. In *NIPS*, 2015. 1

[3] M. Malinowski and M. Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *NIPS*, 2014. 2