16824 hw3

jiaqigen

April 2021

List of commands to run code
python -m student_code.main `--` model simple/coattention
All of the training parameters are already inputted as default.
The only thing to change is the model selection parameter.
Please either change that to ”simple” or ”coattention”.

Link to models and tensorboard results
`https://drive.google.com/drive/folders/1wT_QibGhpYy5l0iR6jfH6m1dl533NIbh?usp=sharing`

1 Data Loader

1.1

getQuesIds() returns the IDs of all questions in this dataset
Number of question IDs: 248349

1.2

Question: What position is the man squatting with a glove on playing?
Question Type: What
ID of the image associated with this question: 40938

1.3

Catcher

1.4

See the code for implementation.

1.5

See the code for implementation.

1.6

See the code for implementation.

1.7

See the code for implementation.

The answer embedding is sentence-level
because our final goal is to predict the full answer to a question.
If the answer embedding is word-level,
then model will produce probabilities for each word in all the answers
instead of probabilities for each answer.

1.8

It should be equal to the number of questions.
Our goal is to generate an answer given an image and a question.
Since each image might have multiple questions,
and a single answer does not mean anything without its question,
we need to consider a single question as one training example.
Therefore, the size should be the number of questions.

1.9

If the question is longer than the max question length (26),
then the rest of the words get cut off.
The dimension of the question tensor is 26 * 5747.
The dimension of the question tensor is 10 * 5217.

2 Simple Baseline

2.1

Advantage: Easy to implement
Disadvantage: Cannot represent the order of the words

Our one-hot question encoding should be 26 * 5747. To convert it into bag of words, we simply look at each column of the tensor. If we can find a 1 in column i, then the ith element of the bag of words is 1. Otherwise, the ith element is 0.

2.2

The three major components are:
CNN, word embedding layers, and softmax layer.
While batch size is 100:
Input size for CNN: 100 * 3 * 224 * 224
Output size for CNN: 100 * 1000
Input size for word embedding layer: 100 * 5747
Output size for word embedding layer: 100 * 1024
Input size for softmax layer: 100 * 2024
Output size for softmax layer: 100 * 5217

2.3

See the code for implementation.

2.4

For the training set, we want to create the question to id map and the answer to id map based on the training data.

For validation, we simply pass the maps from training set to validation set because we assume that they share the same vocabulary. Also, if the maps are different across train and val set, the model won't work during validation.

2.5

We can set the learning rate for specific parts of the model using optimizer.

For examples, instead of using model.parameters(), we can input a list of sub parts parameters like {"params": model.fc1.parameters(), "lr": 0.8}

See the code for implementation.

2.6

See the code for implementation.

2.7

I used the cross entropy loss.
Notice that I used softmax only in the evaluation step,
so the input to the cross entropy loss function does not go through softmax.

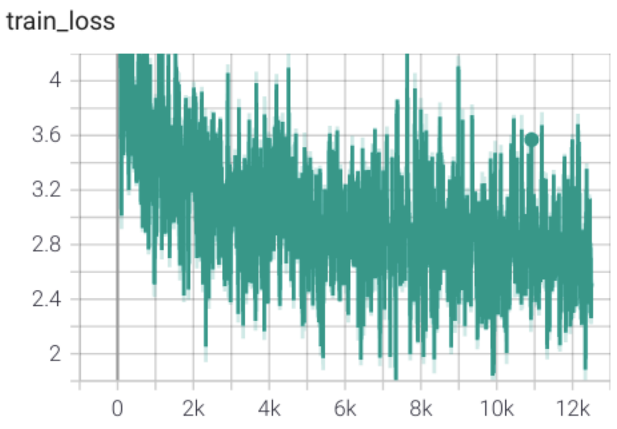
2.8

See the code for implementation.

2.9

See the code for implementation. See the next question for figures.

2.10


train_loss
4
3.6
3.2
2.8
2.4
2
0
2k
4k
6k
8k
10k
12k

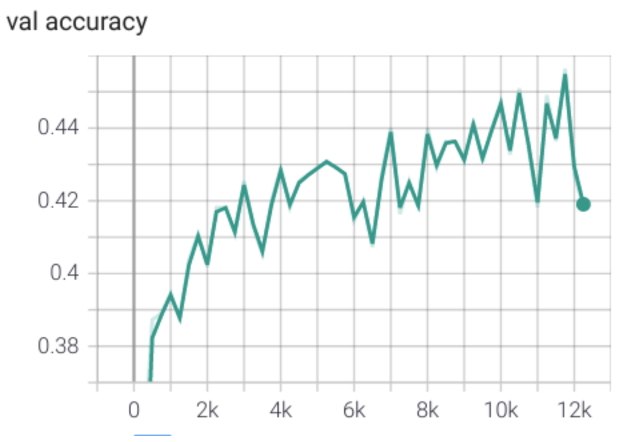
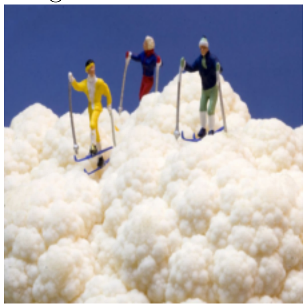
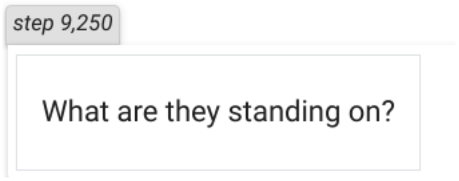

val accuracy
0.44
0.42
0.4
0.38
0
2k
4k
6k
8k
10k
12k

Image:



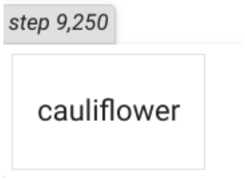
Question:



step 9,250

What are they standing on?

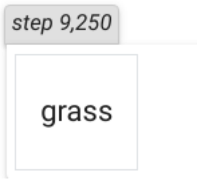
Ground truth:



step 9,250

cauliflower

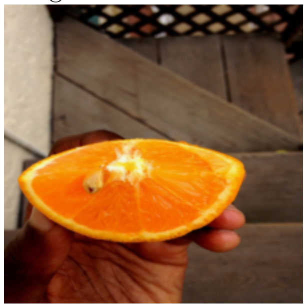
Predicted:



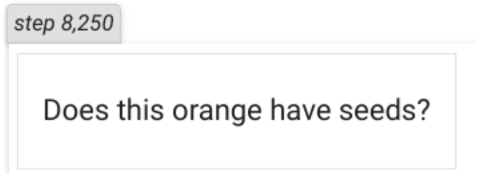
step 9,250

grass

Image:



Question:



step 8,250

Does this orange have seeds?

Ground truth:

step 8,250

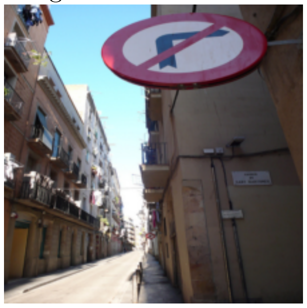
yes

Predicted:

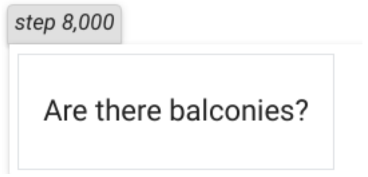
step 8,250

yes

Image:



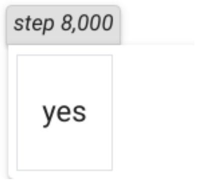
Question:



step 8,000

Are there balconies?

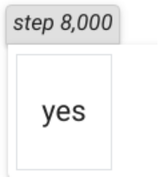
Ground truth:



step 8,000

yes

Predicted:



step 8,000

yes

3 Co-Attention Network

3.1

The input size is 3 * 448 * 448

3.2

See the code for implementation.

3.3

1. Three levels: word level, phrase level, and question level. At word level, I obtained word embedding using a linear layer. At phrase level, I obtained unigrams, bigrams, and trigrams with 1d convolution. At question level, I obtained question encoding with LSTM.

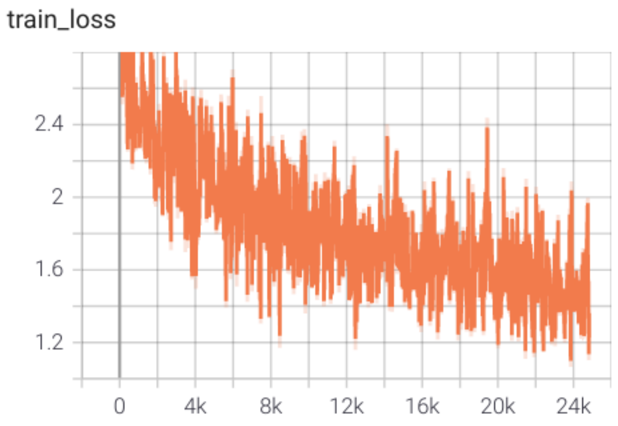
2. Attention is basically a component that can learn to give more importance to some parts of the input. Co-attention utilizes both image and question information. When generating question attention, it will use image as guidance and vice versa. This is helpful because if we only focus on which part of the image is relevant, then we overlook the fact that some parts of question are more important than others.

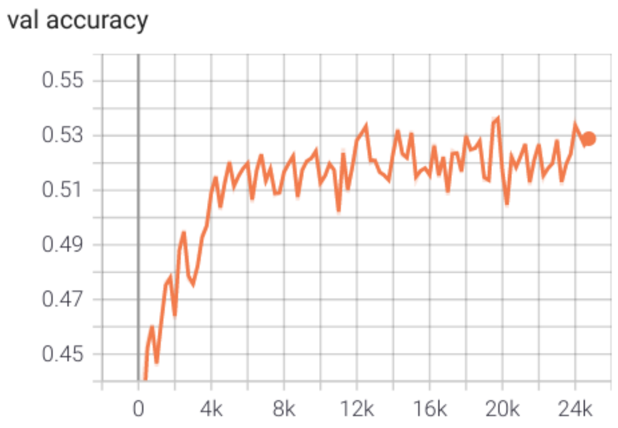
3. Since we need to construct Alternating Co-Attention for all three levels of the hierarchy, and the weights of layers in each Alternating Co-Attention should be separated, I created two other classes, one for Alternating Co-Attention, another for basic attention. This way, we don't need to repeatedly define the same layers over and over again.

3.4

See the code for implementation.

3.5

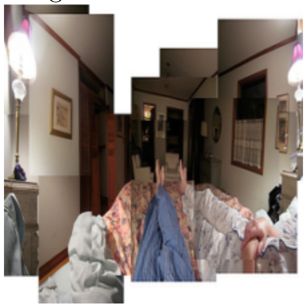




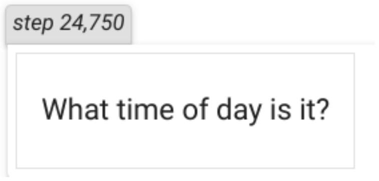
Comparison

The validation accuracy of Co-Attention Network (around 53%) is higher than that of simple baseline (around 43%).

Image:



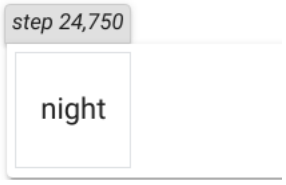
Question:



step 24,750

What time of day is it?

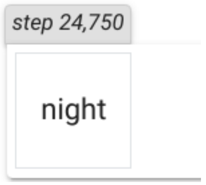
Ground truth:



step 24,750

night

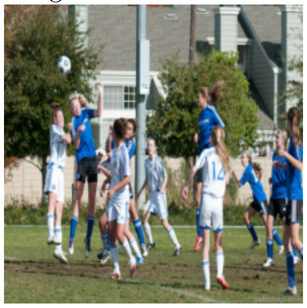
Predicted:



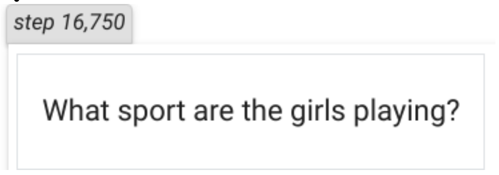
step 24,750

night

Image:



Question:



step 16,750

What sport are the girls playing?

Ground truth:

step 16,750

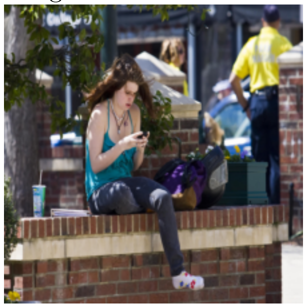
soccer

Predicted:

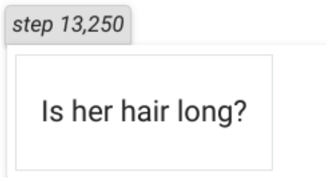
step 16,750

tennis

Image:



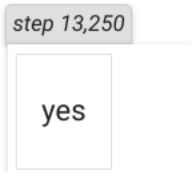
Question:



step 13,250

Is her hair long?

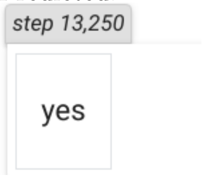
Ground truth:



step 13,250

yes

Predicted:



step 13,250

yes

4 Custom Network

4.1

Idea 1:

Currently, we are extracting the features from the whole image,
but usually only the objects' vision information is needed.
Therefore, I think we can run object detection first,
and only extract the features for detected objects.
In this case, I don't think we still need attention.

Idea 2:

Currently, the LSTM in our model is used to encode the question sequence.
The output question features contain no information about the image.
I am thinking we can combine the image features and word features together,
and then fed them into the LSTM.
I think we don't need the attention step anymore with this approach.

Idea 3:

I think maybe we can separate all of the questions into a few question groups.
For example, we can group questions starting with "how many" together
or group questions that can be answered with "yes" or "no" together.
We will train multiple networks,
one of them is for classifying which group the question belongs to.
the rests will have the same structure as our current network,
but trained on one group of the questions.

4.2

Not implemented.