HOMEWORK 3: VISUAL QUESTION ANSWERING WITH PYTORCH

16-824 Visual Learning
Recognition (Spring 2021)
Carnegie Mellon University
Homework PDF Link
BY: Feng Xiang*
DUE: Monday, April 26, 2021 11:59 PM

Notes

- I collaborated with Baishali Mullick on this assignment.
- To run the code and model, perform the following:
  - Upload VQA dataset files into the data folder
  - Upload Task03 image feature maps into the tmp_train and tmp_val folders, respectively
  - Navigate to main directory folder (you should see the student_code, external, and test folders in this directory)
  - Perform "conda activate ..." to the appropriate conda environment
  - Enter the following command in the terminal to run the simple baseline model

```
python -m student_code/main.py --model='simple'
```

  - Enter the following command in the terminal to run the co-attention model

```
python -m student_code/main.py --model='simple'
```

- Google Drive Link

*Compiled on Friday 30th April, 2021 at 00:41

Task 1: Data Loader

Q1.1

Prompt Which member function of the *VQA* class returns the IDs of all questions in this dataset? How many IDs are there?

Answer The member function of *VQA* that returns the IDs of all questions in the dataset is **qqa**. There are **248349** IDs in the VQA dataset.

References

- N/A

Q1.2

Prompt What is the content of the question of ID *409380*? What is the ID of the image associated with this question?

Answer The content o question ID *409380* consists of:

- question_type: ”what”
- multiple_choice_answer: ”catcher”
- answers:
  - ’catcher’
  - ’catcher’
  - ’catcher’
  - ’catcher’
  - ’catcher’
  - ’catcher’
  - ’baseball’
  - ’catcher’
  - ’catcher’
  - ’catcher’
- image_id: 40938
- answer_type: ”other”
- question_id: 409380

References

- N/A

Q1.3

Prompt What is the mostly voted answer for this question?

Answer The mostly voted answer to the question is **catcher**.

References

- N/A

Q1.7

Prompt Different from the word-level question embedding, the answer embedding is sentence-level (one ID per sentence). Why is it?

Answer The answer dictionary embedding is at the sentence-level rather than the answer-level because the output of our model will devise of a Softmax function that computes the probability of each answer sentence. Implementing a sentence-level answer embeddings allows us to choose the highest probable answer statement. Implementing a word-level answer embedding would be a more difficult task because we would have to both select the top-N answer words and organize those words into an answer statement that is both coherent and correct in sequence to the ground truth answer statement.

References

- N/A

Q1.8

Prompt Should the size of the dataset equal the number of images, questions or answers? Show your reasoning?

Answer The size of the dataset should equal the number of questions because compared to the image input aspect of the model, each question is unique in the dataset. Though one image can have multiple questions, each question instance in the dataset is tied to one image.

References

- N/A

Q1.9.3

Prompt How do you handle questions of different lengths? Describe in words, what is the dimension of your output tensor?

Answer Questions of different word lengths are handled by padding the question array with additional zero vectors up to a length of 26 words. The padded one hot vectors denote that there is no word in this position.

The dimension of the output tensor shall be an array that has as many rows as 26 and as many columns as the vocabulary dictionary at which it was defined plus one, which is 5747.

References

- N/A

Q1.9.4

Prompt Create sentence-level one-hot encoding for the answers. 10 answers are provided for each question. Encode each of them and stack together. Again, make sure to handle the answers not in the answer list. What is the dimension of your output tensor?

Answer The dimension of the output tensor shall be a one-hot vector that is one row by the however many columns there are in the answer vocabulary dictionary plus one, which is 5217. The last element position in the output tensor denotes ”none of the above” other answer choices.

References

- N/A

Task 2: Simple Baseline

Q2.1

Prompt This paper uses 'bag-of-words' for question representation. What are the advantages and disadvantages of this type of representation? How do you convert the one-hot encoding loaded in question 1.9 to 'bag-of-words'?

Answer The 'bag-of-words' is simplistic and straightforward to implement. The disadvantage is the finite window of words that could be noted. There is only a finite amount of space to fit a certain number of words. To convert the one-hot encoding loaded in question 1.8 to a 'bag-of-words' configuration is to sum the one-hot vectors along the rows such that the resultant row vector, which has as many columns as words in the question vocabulary dictionary plus one, shall encompass both the presence and frequency of words through the entirety of each question instance.

References

- N/A

Q2.2

Prompt What are the 3 major components of the network used in this paper? What are the dimensions of the input and output for each of them (including batch size)?

Answer The 3 major components of the implemented network is:

- CNN (i.e. GoogLeNet)

  The input dimensions is (batch size x 3 x 224 x 224). The output dimension is (batch size x 1000)

- Word embedding

  The input dimension is (batch size x 26 x 5647). The output dimension is (batch size x 1024) which is mainly due to the *nn.Embedding* layer that was implemented.

- Word feature and softmax layers

  The input dimension is (batch size x 2048) which is the concatenation of the outputs of both *GoogLeNet* and word embedding layers. The output dimension is (batch size x 5217).

References

- N/A

Q2.4

Prompt Explain how you are handling the training set and validation set differently.

Answer The inputs for the *question_word_to_id_map* and *answer_to_id_map* variables into the training *VqaDataset* initialization are left as $None$. This is because the dataset shall build out the respective word and sentence dictionaries for all question and answer samples.

Initializing the validation *VqaDataset* class is different because, since we already created the question and answer vocabulary dictionaries when initializing the training dataset, we would pass those same training question and answer dictionaries to the validation dataset. This way, the model can be trained and validated from the same question and answer dictionaries.

References

- N/A

Q2.5

Prompt We recommend a learning rate of 0.8 for word embedding layer and 0.01 for softmax layer, both with SGD optimizer. Explain how this is achieved in your implementation.

Answer Two optimizers were declared in the initialization method in *SimpleBaselinExperimentRunner* class as shown in the code block below. The first optimizer function pertains to the word embedding layers of the model. The second optimizer function pertains to the final processing layers of the model:

```
momentum = 0.9
weight_decay = 1e-4
self.optimizer_word = torch.optim.SGD(
    self._model.wordEmbedding.parameters(),
    lr=0.8,
    weight_decay = weight_decay)
self.optimizer_softmax = torch.optim.SGD(
    self._model.combinedOutput.parameters(),
    lr=0.01,
    weight_decay = weight_decay)
```

The *_optimize* function of the SimpleBaseLineExperimentRunner class was written as shown in the code block below. The loss is computed before gradient clipping is performed on the model. :

```
def _optimize(self, predicted_answers, true_answer_ids):
    lossFunc = nn.CrossEntropyLoss()

    self.optimizer_word.zero_grad()
    self.optimizer_softmax.zero_grad()

    loss = lossFunc(predicted_answers, true_answer_ids)

    loss.backward()

    grad_value = 20.0
    nn.utils.clip_grad_value_(self._model.parameters(),
        clip_value = grad_value)

    self.optimizer_word.step()
    self.optimizer_softmax.step()

    return loss.item()
```

References

- N/A

Q2.7

Prompt In Section 3.2 of the paper, they mention weight clip. This means to clip network weight data and gradients that have a large absolute value. We recommend a threshold of 1500 for the word embedding layer weights, 20 for the softmax layer weights and 20 for weight gradients. What loss function do you use?

Answer The loss function that was used was Cross Entropy loss using the *torch.nn* function *nn.CrossEntropyLoss()*.

References

- N/A

Q2.10

Describe anything special about your implementation in the report. Include your figures of training loss and validation accuracy. Also show input, prediction and ground truth in 3 different iterations.

Prompt

Answer The word embedding layer of the model consists of one linear layer that downsizes the channels of the question vocabulary from over 5747 to 1024. The final fc layer of the GoogLeNet model was removed as well. After concatenation of the image feature map with the word embedding layers, the vector is passed through one linear layer changes the channels from 2048 to the size of the answer dictionary (i.e. 5217).

When declaring the SGD optimizers, a weight decay factor was set at 1e-4 and no set momentum. The weight clipping factors, as per prescribed in the assignment, was not implemented in order to improve the validation accuracy with the current configuration of the model.

The figure shown below (see Figure 0.1) shows the training loss of the model during a training session of 10 epochs.

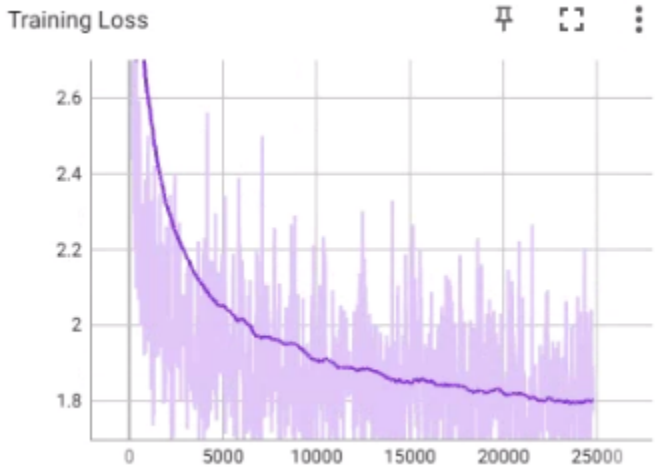

Figure 0.1: Training loss over 10 epochs

The figure show below (see Figure 0.2) shows the validation accuracy of the model during a training session of 10 epochs.

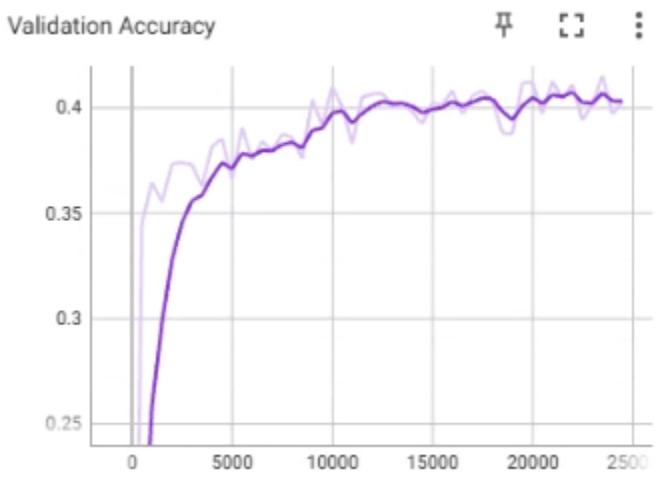

Figure 0.2: Validation accuracy over 10 epochs

The final loss value was recorded at 1.774. The final validation accuracy was recorded as 0.4029.

The figure shown below (see Figure 0.3) displays the first sample input image to TensorBoard.

Figure 0.3: First sampled input image

The figure shown below (see Figure **??**) displays the input question to the first sampled image.

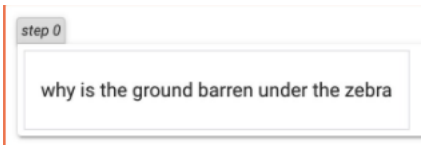

Figure 0.4: First sampled input question

The figure shown below (see Figure 0.5) displays the predicted answer output for the first sampled image and question.

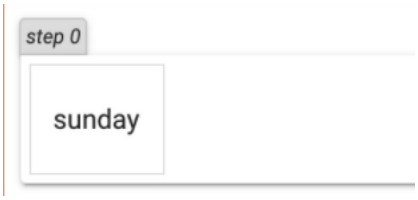

Figure 0.5: First predicted answer

The figure shown below (see Figure 0.6) displays the ground truth answer for the first sampled image and question.

Figure 0.6: First ground truth answer

The figure shown below (see Figure 0.7) displays the second sample input image to TensorBoard.

Figure 0.7: Second sampled input image

The figure shown below (see Figure **??**) displays the input question to the second sampled image.

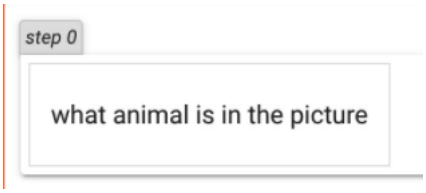

Figure 0.8: Second sampled input question

The figure shown below (see Figure 0.9) displays the predicted answer output for the second sampled image and question.

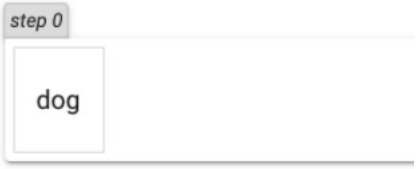

Figure 0.9: Second predicted answer

The figure shown below (see Figure 0.10) displays the ground truth answer for the second sampled image and question.

Figure 0.10: Second ground truth answer

The figure shown below (see Figure 0.11) displays the third sample input image to TensorBoard.

Figure 0.11: First sampled input image

The figure shown below (see Figure **??**) displays the input question to the third sampled image.

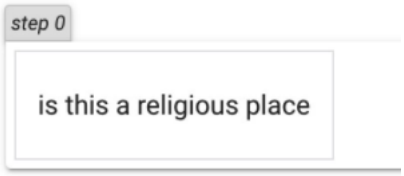

Figure 0.12: Third sampled input question

The figure shown below (see Figure 0.13) displays the predicted answer output for the third sampled image and question.

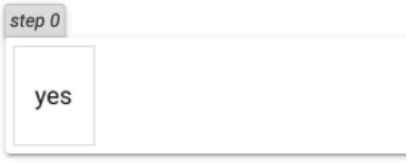

Figure 0.13: Third predicted answer

The figure shown below (see Figure 0.14) displays the ground truth answer for the third sampled image and question.

Figure 0.14: Third ground truth answer

References

- N/A

Task 3

Q3.1

Prompt What is the input size used in the Co-Attention paper?

Answer The image input size in the Co-Attention paper is **512 x 14 x 14**.

References

- N/A

Q3.3.1

Prompt What are the three levels in the hierarchy of question representation? How do you obtain each level of representation?

Answer The three levels of the model's hierarchy is the word level, phrase level, and question level outputs.

The three levels of the model's hierarchy for question representation are the word embedding level, the alternating co-attention layer, and the answer encoding layer.

The question input was first passed through a word embedding function. The word level output is the output from the word embedding layer. The output is then passed through three parallel channels. The three parallel channels are built the same, and consists of a linear function followed by a *Tanh* nonlinear activation function. The output from this layer should be three outputs of the same size. The three unigram, bigram, and trigram outputs are concatenating together along the third dimension which is then passed through a max pooling layer along the channel dimension of the input. The phrase level output is equal to the output from the max pooling layer. Finally, the question feature is then passed through an LSTM layer before entering the next level of the model hierarchy. The sentence level output is equal to the ouput from the LSTM layer.

References

- N/A

Q3.3.2

Prompt What is attention? How does the co-attention mechanism work? Why do you think it can help with the VQA task?

Answer The alternating co-attention function is ran for the word, phrase, and sentence level outputs. The alternating co-attention function consists of providing attention to the question input, then the image input, back to the question input, in that serial sequence. Attention is performed via multiple linear and nonlinear activation layers as well as a summation in the end of each attention sequence. The output of the function are two values which are the processed outputs for the image and question layers in the co-attention function.

This mechanism helps with the VQA task because the output of the question attention processing sequence is fed as an input and guide for the image attention processing sequence. Finally, the output from the image processing sequence is fed as an input back to the question level processing sequence. Both the question and image features are being processed and guided against each other, which in turn builds a more sophicated connected between the question and image aspects to the VQA task.

References

- N/A

Q3.3.3

Prompt Compared to networks we use in previous assignments, the co-attention network is quite complicated. How do you modularize your code so that it is easy to manage and reuse?

Answer One way to modularize the alternating co-attention portion of the model is to build a *nn.ModuleDict* variable that contains all the functions being used in the alternating co-attention function. In addition, a separate function was built for the alternating co-attention function of the model. From here, the word, phrase, and sentence level inputs can pass through the function, using the same layers, and output in a more simplified and clean logic in the model code.

References

- N/A

Q3.5

Prompt Similar to Question 2.10, describe anything special about your implementation in the report. Include your figures of training loss and validation accuracy. Compare the performance of co-attention network to the simple baseline.

Answer Before passing the question encoding through the LSTM layer, a *pack_padded_sequence* function was called to convert the question encoding into a format that is better understood by the LSTM layer. Afterwards, the sequence object is converted back to a tensor question encoding by using the *pad_packed_sequence* function.

Dropout layers were used in the alternating co-attention aspect of the model but were not used the final word processing layers of the model.

The figure shown below (see Figure 0.15) displays the training loss of the model over a training session of 10 epochs.

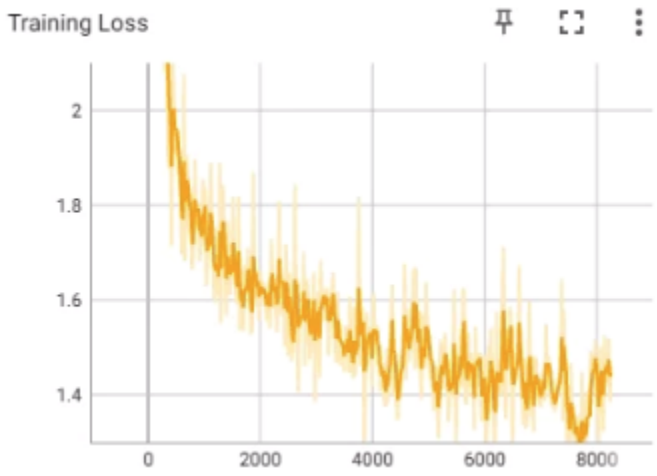

Figure 0.15: Training loss over 10 epochs

The figure shown below (see Figure 0.16) displays the validation accuracy of the model over a training session of 10 epochs.

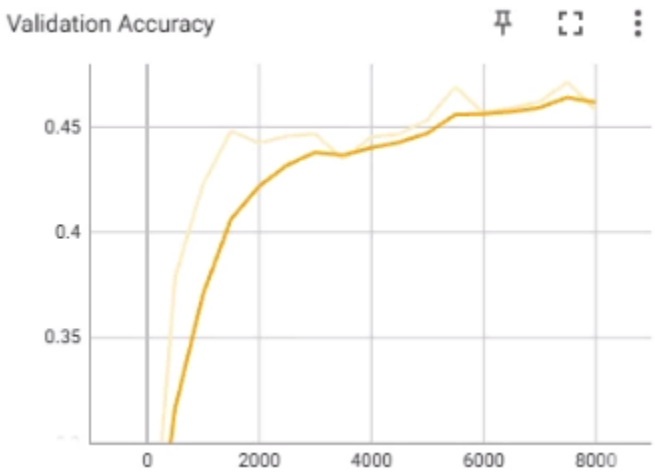

Figure 0.16: Validation accuracy over 10 epochs

The final recorded training loss value was 1.457. The final recorded validation accuracy was 0.4581. Compared to the results from the simple baseline model in Task02, the co-attention model performed about 5% better.

References

- N/A