Meets Specifications

Hi, excellent work! Congratulations, you pass the most difficult project! 🎉
I have attached some extra resources, related to this project that you may find interesting:
Watch this amazing video which shows us how Deep learning Attention model helps produce better image caption pairs.
Keras Deep learning for image caption retrieval
Building an automated image captioning application
CNN+CNN convolution decoders for image captioning

Good luck! 👍 😃

Files Submitted

The submission includes model.py and the following Jupyter notebooks, where all questions have been answered:
2_Training.ipynb, and
3_Inference.ipynb.

- Awesome! All the files are submitted with the questions answered!

model.py

The chosen CNN architecture in the `CNNEncoder` class in model.py makes sense as an encoder for the image captioning task.

- The chosen Resnet50 model is the right choice!

The chosen RNN architecture in the `RNNDecoder` class in model.py makes sense as a decoder for the image captioning task.

- Single layer LSTM is a perfect choice!

2_Training.ipynb

When using the `get_loader` function in data_loader.py to train the model, most arguments are left at their default values, as outlined in Step 1 of 1_Preliminaries.ipynb. In particular, the submission only (optionally) changes the values of the following arguments: `transform`, `mode`, `batch_size`, `vocab_threshold`, `vocab_from_file`.

- Excellent choice of parameters. Very commendable that you added random rotation in your transformation. 👍

The submission describes the chosen CNN-RNN architecture and details how the hyperparameters were selected.

- Very impressed by your effort in explaining your thought process clearly. Well done! 👍

The transform is congruent with the choice of CNN architecture. If the transform has been modified, the submission describes how the transform used to pre-process the training images was selected.

- Great, the transformation with random rotation added makes perfect sense. I agree with your justification. 👍

The submission describes how the trainable parameters were selected and has made a well-informed choice when deciding which parameters in the model should be trainable.

- Awesome! Well justified answer. Very impressive. 👍

The submission describes how the optimizer was selected.

- You are right! Adam is widely used and safe choice.

The code cell in Step 2 details all code used to train the model from scratch. The output of the code cell shows exactly what is printed when running the code cell. If the submission has amended the code used for training the model, it is well-organized and includes comments.

- Great work training your model for 3 epochs.

3_Inference.ipynb

The transform used to pre-process the test images is congruent with the choice of CNN architecture. It is also consistent with the transform specified in `transform_train` in 2_Training.ipynb.

- Perfect set of transformation for inference. Well done.

The implementation of the `sample` method in the `RNNDecoder` class correctly leverages the RNN to generate predicted token indices.

- `sample` method is error free and produces desired output. Nice work.

The `clean_sentence` function passes the test in Step 4. The sentence is reasonably clean, where any `<start>` and `<end>` tokens have been removed.

- The `clean-sentence` indeed cleans out unnecessary tokens. 👍
- Keep in mind that you could have achieved the task in just a single loop by using `continue` and `break` keywords together.

The submission shows two image-caption pairs where the model performed well, and two image-caption pairs where the model did not perform well.

- The predictions are nailed! Very well done providing two examples each for model performing well and where it needs more sophisticated improvement.