UDACITY

Return to "Computer Vision Nanodegree" in the classroom

Image Captioning

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

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 job on completing the Image captioning project, it meets all specifications! 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.

All the questions have been answered in the 2_Training file, all the required files have been submitted

model.py

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

The CNN architecture for the CNN encoder looks good, wise move to use the resnet pretrained model ✓
Sugestion – have a look at this link to read more on CNN-LSTM architecture used for image captioning

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

Good job on the RNN decoder architecture, the embeddings layer is necessary for the captions
Suggestion – Have a look at this link to know more on how rnn and lstm are used in image captioning

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`.

Good job setting the arguments to the correct value ✓

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

The CNN-RNN architecture used for the project is nailed ✅ Read this post to know the power of a CNN-RNN architecture.

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.

Good job choosing the transform, it works perfectly well with the CNN architecture 👍

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.

The trainable parameters are correctly selected and proper reasoning has been given for choosing them

The submission describes how the optimizer was selected.

Adam optimizer works just fine for this project. Have a look at this tutorial as a reference for choosing hyperparameter values, transform, CNN-RNN architecture, optimizer and loss function. Here is some more information on Adam

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.

Training code is well written, output has been shown but you could always validate your model ✅

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.

Both the transforms used for training and testing are consistent!

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

The generated predicted token indices are correct, it shows the RNN decoder has been configured properly 💯

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 sentences have been cleaned of any start/end tokens. Good job on the code 👏🏽

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 decent enough, but I encourage you to try with more images ✅

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review