U DACITY

PROJECT SPECIFICATION

Image Captioning

Files Submitted

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

model.py

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

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

2_Training.ipynb

CRITERIA	MEETS SPECIFICATIONS
Using the Data Loader	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`.
Step 1, Question 1	The submission describes the chosen CNN-RNN architecture and details how the hyperparameters were selected.
Step 1, Question 2	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.

Step 1, Question 3	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.
Step 1, Question 4	The submission describes how the optimizer was selected.
Step 2	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.

(header row, overlapping: CRITERIA / MEETS SPECIFICATIONS)

3_Inference.ipynb

CRITERIA	MEETS SPECIFICATIONS
`transform_test`	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**.
Step 3	The implementation of the `sample` method in the `RNNDecoder` class correctly leverages the RNN to generate predicted token indices.

Step 4 CRITERIA	The `clean_sentence` function passes the test in **Step 4**. MEETS SPECIFICATIONS The sentence is reasonably clean, where any `<start>` and `<end>` tokens have been removed.
Step 5	The submission shows two image-caption pairs where the model performed well, and two image-caption pairs where the model did not perform well.

Suggestions to Make Your Project Stand Out!

- Use the validation set to guide your search for appropriate hyperparameters.
- Implement beam search to generate captions on new images.
- Tinker with your model - and train it for long enough - to obtain results that are comparable to (or surpass!) recent research articles.