Nikhil Suresh - nsuresh@clemson.edu

CPSC 8430: Deep Learning Assignment 2
Video Caption Generation

GitHub link
Deep-Learning/Homework 2 at master · nik1097/Deep-Learning (github.com)

Problem Statement
Generate a video caption for an input video using sequential-to-sequential model. The input for the code will be a video and the output will be a stream of captions describing the actions in the video. The above is achieved by Recurrent Neural Networks.

Requirements

- Python
- Cuda (GPU)
- torch
- scipy
- numpy
- pandas
- pickle

Dataset
The same dataset provided by the professor has been utilised to train and test the model.
The dataset has 1450 video for training and 100 for testing. We also have a training and testing label .json files that presents us with labels to train our model.

Program Data Structures
The basic requirement for the seq2seq model is a dictionary as prescribed in the slides. Below is a basic mapping for the words and vice-versa.
Tokens

- <PAD>: Pad the sentence to same length.
- <BOS>: Begin of a sentence, sign to generate output sentence.
- <EOS>: End of a sentence, sign to end output sentence.
- <UNK>: Token for unknown word in dictionary.

All the below data structures are stored as python dictionaries.

- word_dict
  Contains the number of occurrences of a word in the training label file by generating a vocabulary. Any word having a frequency less than 4 is ignored.
- w2i
  Contains the word to index mapping for the words in the vocabulary.
- i2w
  Contains the index to word reverse mapping for the vocabulary.

Nikhil Suresh - nsuresh@clemson.edu

Model

The model has 2 layers namely, encoder and decoder. The GRU – Gated Recurrent Units is used to design both these layers. The GRU is faster than LSTM as it has less training parameter and therefore uses less memory and executes faster than LSTM. As the dataset has smaller sequences of data the GRU provides faster results with comparable accuracies to LSTM.
The encoder processes the video and encodes it into the necessary format. The decoder is then used to segment the captions based on the beginning and ending tokens and perform video processing over the words to produce actual words.

Attention Layer

It allows the model to peek at different sections of inputs at each decoding time step. The structure of Attention Layer is adopted from the paper titled “Attention-based convolution neural network for semantic relation extraction” by Shen and Huang. Hidden state of decoder and encoder output is used as a matching function to a scalar, and then passed through a soft-max transformation. Last new hidden state is then sent to the next time step of decoder.

Schedule Sampling

At inference, the unknown previous token is replaced by a token generated by the model itself. Schedule Sampling to mitigate this discrepancy between train and test time by randomly replacing some discrete units in the history with the model's prediction. This solves the issue of exposure bias in our model.

Model Parameters and Evaluation

- Epochs = 100
- Learning Rate = 0.0001
- Batch Size = 128
- Hidden Layers = 512
- Optimizer = Adam Optimizer
- Dropout = 0.3
- Teacher Learning Ratio = 0.7
- Vocab Size = n > 4

The loss of the model in last epoch is 1.4123
The bleu score of the model is at 0.69953 for the test data provided for evaluation.

Note: The dataset is stored in the scratch location on Palmetto as denoted in the code. The location of the model and the train/test labels have been hard coded into the program. The user has the option of providing the file path for testing files in the .sh file along with the name of output file. The output file will be available in the local directory of code files.