Deep Learning Assignment 2

As part of this assignment, I have worked on Video caption generation, a sequence-to-sequence model trained and tested following attention, and schedule sampling to generate a caption for the videos we input during the process. During this action, captions about what is happening in the video will be rendered, and respective descriptions will be shown to the user.

A sequence-to-sequence (seq2seq) model is a neural network architecture used in deep learning for natural language processing tasks, such as video captioning. The model is made up of two primary parts: a decoder, which creates a narrative description of the video content, and an encoder, which processes the video sequence and collects visual elements.

The model is trained by minimizing the difference between predicted and actual captions, using a loss function like cross-entropy. While seq2seq models have shown impressive results in video captioning, they still need help to produce accurate and diverse captions for long and complex videos.

For this assignment, we use the Encoder decoder framework and MSVD dataset to map an input occurrence to an output sequence that varies by length to assess the model. In this assignment, two long short-term memories are used, one is for encoding and a second for decryption; they are used to store and interpret the video., which provides a sentence for the illustrated video. LSTM is a variant of the RNN architecture that introduces a memory cell and several gating mechanisms to the standard RNN. These additional components enable LSTMs to selectively retain or discard information from previous time steps, making them well-suited for tasks where long-term dependencies are essential.

In the context of the encoder-decoder framework, the LSTM-based decoder generates output tokens one at a time, conditioning each prediction on the previously generated tokens and the encoded input sequence. During training, the model learns to adjust the decoder and encoder parameters to minimize discrepancies between the predicted output sequence and the ground truth.

LSTMs have been critical in many state-of-the-art seq2seq models, including machine translation, text summarization, and video captioning. Nonetheless, there are still challenges in developing effective and efficient LSTMs for specific applications, such as generating coherent and diverse captions for long and complex videos.

Dataset

The Microsoft research description of the video sample, which has approximately 100,000 sentences, was the data set we utilized for this homework. These short videos are summarized by the people describing the specific action in a sentence. It comprises about 1500 brief clips, each lasting between ten and twenty seconds, of which we've selected 1450 for learning and 100 for assessment. After pre-processing the videos using the pre-trained CNN VGG19, we stored their features in an 80x4096 format. During the training phase, we have kept the number of frames in the videos, which allows our model to leverage the available information fully.

Method

A deep neural architecture being used for video labeling is called the sequence-to-sequence model framework. It takes in a sequence of video frames and outputs a corresponding series of words. The model uses an encoder-decoder framework and is trained on a large dataset of video-caption pairs to generate accurate captions. The sequence-to-sequence model has shown promising results and has potential applications in automated video indexing and retrieval.

Sample

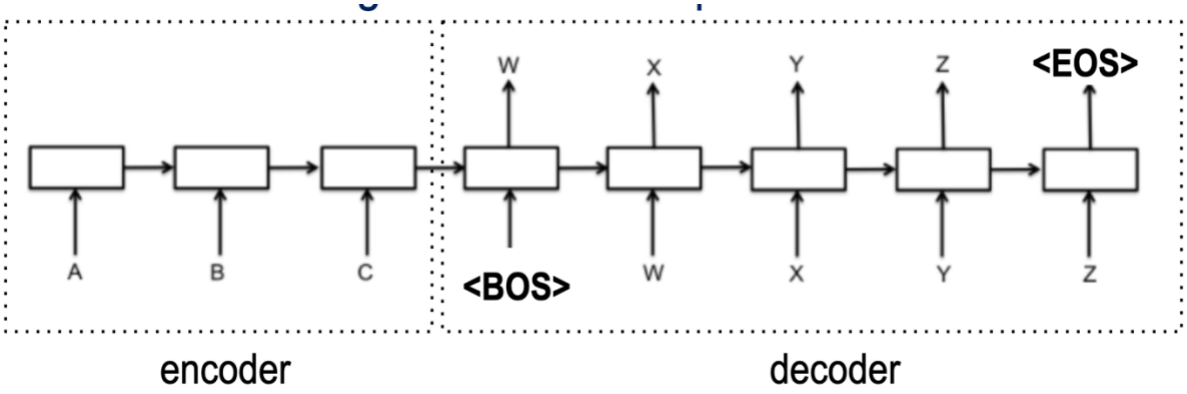
When we consider A video of the sample, we observed the following.

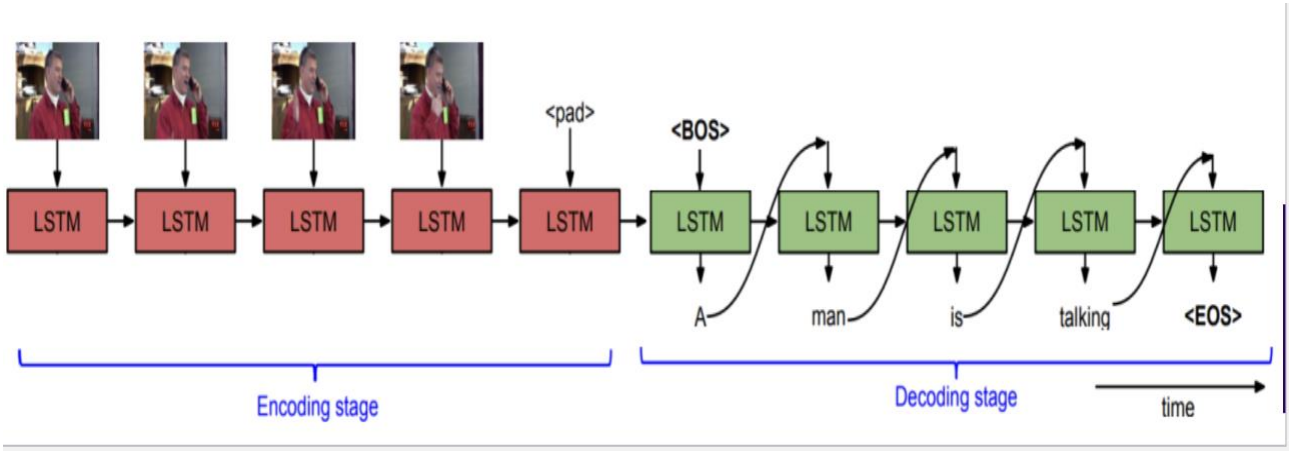
Video id: KPPCwmU5OHQ_56_62.avi
Features: (80, 4096)
Caption: A person opens a can of soup.
Total no of Unique tokens: 6443.

Input: A short video
Output: The caption is generated by a sequence-to-sequence model representing the video content in text form. It's trained on video-caption pairs and has video indexing and retrieval applications.

Sequence-to-Sequence Based Model: S2VT

It's a two-layer LSTM structure. It has two Recurrent Neural Networks (RNN) levels: an encoder and a decoder. Firstly, a video is encoded, and a result will be generated with the help of a decoder RNN.





The above figure illustrates the encoding and decoding stages, respectively.

BLEU Score

BLEU is a widely used metric for evaluating machine translation quality by comparing it to one or more human-generated reference translations. It calculates a score based on the similarity between the machine and reference translation(s). While it's easy to compute and helpful in comparing different machine translation systems, BLEU has some drawbacks that should be used with other assessment methods for a much more thorough analysis. One of the primary goals of BLEU was to guarantee a strong correlation to human quality evaluation, which helped to make it among the most well-liked and economically advantageous automated measures for assessing machine-translation output.

I ran my code using the following technologies.

TensorFlow - 1.15.0
Cuda - 9.0
Python - 3.9
NumPy - 1.14
Pandas - 0.20

Execution

I have downloaded the MLDS_hw2_1_data folder provided in the presentation and delivered it to my current working directory.

I then built two object files to hold the video id and caption related to that specific video.

To execute sequence.py, I have used the below command in my current working node.

```
[tupputu@node0037 ~]$ python sequence.py /home/tupputu/MLDS_hw2_1_data/training_data/feat/ /home/tupputu/MLDS_hw2_1_data/training_label.jso
From 6098 words filtered 2881 words to dictionary with minimum count [3]


Caption dimension: (24232,2)
Caption's max length: 40
Average length of captions: 7.711084516
Unique tokens: 6443
ID of 16th video: gGDtPJzh_0s_30_45.avi
Shape of features of 16th video: (80,4096)
Caption of 16th video: A man is buttering bread.
```

I then trained the model which was created. For this, we use sequence to sequence to build and train the model; we use train.py. The following are the parameters used for the control of the learning process. We can observe both the learning rate and batch size below.

The learning rate observed is 0.001, and a batch size of 40/1550 is taken where it is split between training and testing. To increase efficiency, an attention mechanism has been used. For the optimizer, we have used Adam and epochs size of 400 epochs Beam size, if the resultant search is true, will be 5.

I have run the below command in the currently active node directory using the ssh configuration command.

```
python train.py /home/tupputu/MLDS_hw2_1_data/training_data/feat/
/home/tupputu/MLDS_hw2_1_data/training_label.json
sample_output_testset.txt
```

```
Highest [10] BLEU scores: [' 0.5488 ', ' 0.5230', ' 0. 4935 ', ' 0. 5186 ',' 0. 4844 ']
Epoch# 4, Loss: 0.8858, Average BLEU score: 0.5260, Time taken: 25.32s
Training done for batch:0050/1450
Training done for batch:0100/1450
Training done for batch:0150/1450
Training done for batch:0200/1450
Training done for batch:0250/1450
Training done for batch:0300/1450
Training done for batch:0350/1450
Training done for batch:0400/1450
Training done for batch:0450/1450
Training done for batch:0500/1450
Training done for batch:0550/1450
Training done for batch:0600/1450
Training done for batch:0850/1450
Training done for batch:0700/1450
Training done for batch:0750/1450
```

I calculated the BLEU score after every epoch result, which is shown below.

```
[tupputu@node0037 ~]$ python bleu_eval.py output_testset.txt
Originally, average bleu score, 0.268943791701406
By another method, average bleu score is 0.6756350809974031
```

Following a few epoch outcomes, I evaluated the median BLEU score to be 0.6756; however, upon model training for even more over 200 epochs, the average score was closer to 0.6645.

GitHub Link

https://github.com/TarunUpp/DeepLearning_8430/tree/main/hw2/hw2_1