Deep Learning - CPSC 8430

Homework -2 Report

By

Ruthwik Reddy Bommana

https://github.com/ruthwikreddie/Deep-Learning-/tree/main

Introduction:

The main goal of this task is to create a sequential-to-sequential model that can generate video captions. The model is trained and evaluated to create an accurate description of the actions occurring in a particular video. In other words, the goal is to enter a video and receive a sequence of captions describing the activity in the image.

As indicated in the study "Sequence to Sequence Model for Video Captioning," creating automatic natural language captions for films has been a difficult task for both natural language processing and computer vision. Recurrent Neural Networks (RNNs) have proven to be useful in visual interpretation because they can represent sequences of elements. Picture captioning focuses on creating variable-length word sequences, whereas video captioning needs to deal with variable-length frame input sequences as well as variable-length word output sequences.

Long Short-Term Memory (LSTM) models have been shown to perform well in sequential-to-sequential tasks including speech recognition and machine translation. A stacked LSTM is used in video captioning to encode each frame individually. As an input to the LSTM, the output of a Convolutional Neural Network (CNN) is applied to the power values of each input frame. As it reads through each frame, the model constructs a phrase word for word.

Deep learning techniques were used to train the dataset while sticking to certain conditions to complete this task. Python 3.6.0, cuda version 9.0, TensorFlow-gpu version 1.15.0, and numpy version 1.14 are all required.

Additional information on the dataset:

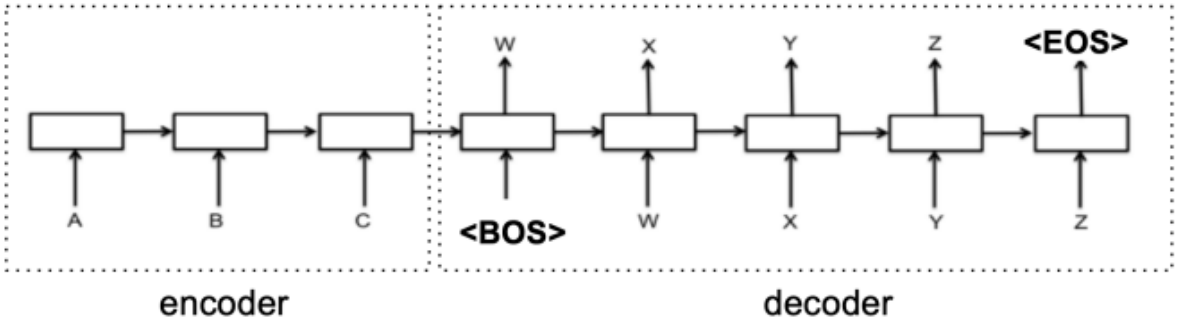
The MSVD (Microsoft Video Description) dataset, which contains around 120,000 sentences, was employed in this investigation. The dataset includes 1550 video clips spanning in duration from 10 to 25 seconds. We divided the videos into 1450 for training and 100 for trialing. The video features were preprocessed and stored in the 80x4096 format using a pre-trained VGG19 Convolutional Neural Network. To maximize the dataset's potential, we did not reduce the number of frames in the videos during training.

Approach Based on a Model:

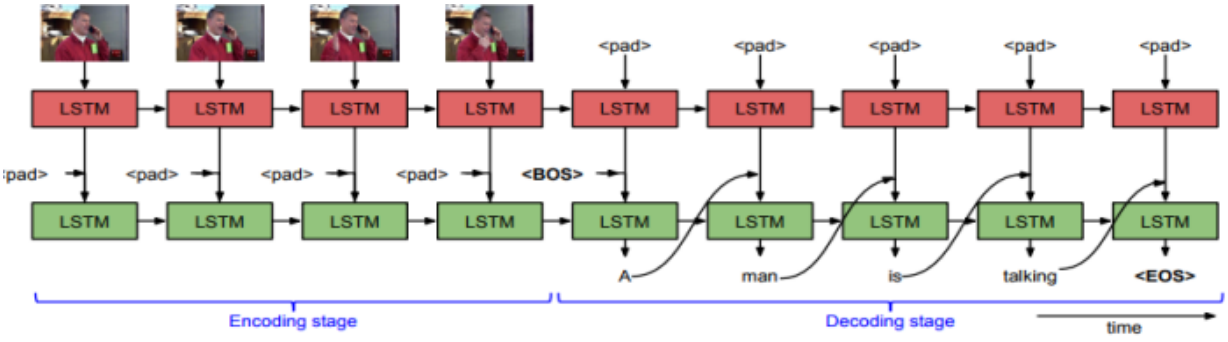
To begin developing the seq2seq paradigm, a dictionary function was created to transform words into indices and vice versa. The dictionary was built using the most commonly occurring terms or a low count. Tokenization, which required inserting specific tokens into the phrases, was also carried out. These included "Pad>," which was used to pad the sentences to the same length, "BOS>," which was used to indicate the beginning of the sentence and generate the output sentence, "EOS>," which was used to indicate the end of the sentence, and "UNK>," which was used to indicate unknown words that were not in the dictionary or to ignore them entirely.

The S2VT model was employed for this job, which consists of two layers of Gated Recurrent Units (GRUs). The first RNN layer oversaw encoding the video and producing an output with the help of the decoder RNN. The RNN decoder oversaw the creation of the captions.

To generate captions, the model divided the subtitles into sections based on the start and finish lines using tokens. The footage was then processed to generate the relevant words for the subtitles.



The diagram below depicts the process of encoding video with the encoderRNN (a GRU layer) and creating video captions with the decoder RNN (which is also a GRU layer).



The training dataset is executed using the following command:

Python sequence.py/home/rbomman/ruth/MLDS_hw2_data/training_data/feat/ /home/rbomman/hw2/MLDS_hw2_data/training_label.json.

The screenshots included showing the consequences of a successful execution.

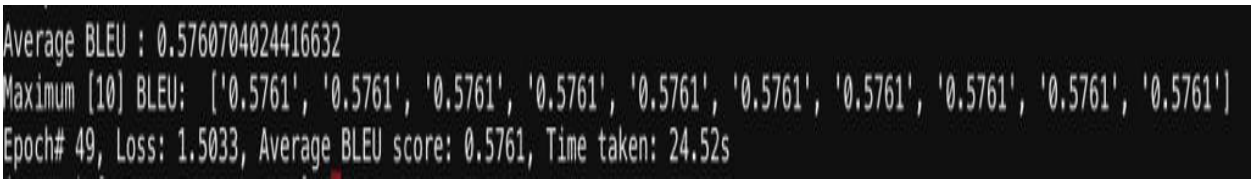
```
From 6098 words filtered 2881 words to dictionary with minimum count [3]

Selected Video Features:
ID of 8th video: bnN_o0Hkn3M_73_80.avi
Shape of features of 8th video: (80, 4096)
Caption of 8th video: Two zebras are playing in a field
```

```
Caption shape: (24232, 2)
Caption's max length: 40
Average length of captions: 7.711084516342027
Unique tokens: 6443
```

The mentioned command is used to run the trained model on the testing dataset. The "train.py" script is run with the features directory path, testing label file location, and output file path of the testing dataset as parameters. The created captions will be saved in the output file directory. The program runs the trained model on the testing dataset and writes the generated captions to the output file given.

python train.py /home/rbomman/ruth/MLDS_hw2_data/testing_data/feat/ /home/rbomman/hw2/MLDS_hw2_data/testing_label.json ./hw2_output.txt

I've attached a screenshot of the results I obtained after training the model.



BLEU Score:

The BLEU algorithm (Bilingual Evaluation Understudy) is used to assess the quality of machine translations from one natural language to another. It compares the similarity of a machine-generated translation to a human-generated translation, with the goal of increasing the similarity between the two. BLEU is a common and efficient automated measurement that ensures a good correlation with human quality evaluations.

The BLEU score is a prominent method to evaluate the effectiveness of both automated and human interpretations. Its purpose is to determine how comparable the computer result is to the comparable human translation. BLEU is regarded as a dependable and cost-effective measurement. The average BLEU score achieved after running the model was roughly 0.67321.