HOMEWORK 2 (CPSC 8430 – DEEP LEARNING)

SHRAVANI KONDA

Video caption generation

The goal for this assignment is to develop a model that can automatically generate descriptive captions for short videos, considering the diverse attributes of the videos, such as varying objects and actions, and handling the variable length of both the input videos and the output captions.

GitHub Link:

https://github.com/shravanik31/Deep-Learning/tree/main/hw2/hw2_1

Please download the model from https://drive.google.com/file/d/19dvvQgTKG4UelaMULE6lmvMWQ06yPlLJ/view?usp=sharing as downloading from the GitHub may have some issues (compression and extraction). I have performed this assignment on Palmetto cluster using CUDA. So please test the model with CUDA supporting device.

Dataset:

The dataset that is used for this project is the Microsoft Video Description Corpus (MSVD), which consists of 1970 short video clips from YouTube, each accompanied by multiple human-generated English captions. For our assignment, we utilize a subset of this dataset, comprising 1450 videos for training and 100 videos for testing. The videos cover a wide range of content, from sports and animals to everyday activities, providing a diverse set of visual and contextual scenarios for caption generation.

Data Preprocessing:

In the data preprocessing stage, different functions and classes are defined for preprocessing and handling data for a video captioning task.

- The **preprocess_data** function reads a JSON file containing video captions, builds a dictionary of words based on their frequency, and filters out words with a count less than three. It also creates mappings between words and indices, including special tokens for padding, beginning of sentence, end of sentence, and unknown words.
- The special tokens that are used are:
    1. <PAD> : This token is used to pad the sentence to the equal length.
    2. <BOS> : This is used as beginning of sentence, a sign to start generating the output sentence.
    3. <EOS> : This token is used for ending of sentence, a sign at the end of the output sentence.
    4. <UNK> : This token is used when the word is not present in the dictionary or to just ignore the unknown word.
- **annotate_captions** function converts captions into sequences of indices based on the filtered dictionary.

- The **create_minibatch** function sorts and pads the captions to create minibatches for training.
- The **DatasetWithFeatures** class loads video features and pairs them with the processed captions for training, while the **TestDataset** class loads features for testing videos. These components are used to prepare and manage the data for training a video captioning model.

Model Building:

In the model building stage, a sequence-to-sequence (Seq2Seq) model is defined with attention, composed of several key components:

- **EncoderRNN**: This module processes the input video features. It first applies a linear projection to reduce the feature dimension to the hidden size of the Gated Recurrent Units (GRU) layer. Then, it applies dropout for regularization and passes the processed features through a GRU layer to produce a sequence of hidden states, which are used by the attention mechanism in the decoder.

- **Attention**: This module calculates the context vector for each time step in the decoder. It takes the decoder's current hidden state and the encoder's outputs as inputs. The attention mechanism concatenates the decoder hidden state with each encoder output, projects this combined vector through several linear layers, and applies a softmax function to obtain attention weights. These weights are then used to compute a weighted sum of the encoder outputs, resulting in the context vector.

- **DecoderRNN**: This module generates the output caption. It uses an embedding layer to convert input word indices into dense vectors, which are then combined with the context vector from the attention module. The combined vector is fed into a GRU layer to update the decoder's hidden state. The output of the GRU layer is passed through a linear layer to predict the next word in the caption. The decoder supports teacher forcing during training, where the ground truth word is used as the input for the next time step with a certain probability, which is controlled by a sigmoid function (teacher forcing ratio).

The **Seq2SeqModel** takes video features as input, passes them through the encoder to get the encoder outputs and last hidden state, and then feeds these into the decoder to generate the output caption. The model can operate in two modes: 'train' for training and 'inference' for generating captions without ground truth inputs.

Training:

The training stage starts by preprocessing the data to create a dictionary of frequently used words and filtering out infrequent words. The training data, which includes video features and corresponding captions is loaded into a DataLoader for batch processing.

The training parameters used are:

- Number of epochs: 200
- Batch size: 128
- Learning rate: 0.001
- Optimizer: Adam
- Loss function: Cross-Entropy Loss

During each epoch, the model's parameters are updated to minimize the loss on the training data. The training process involves feeding video features to the encoder, generating captions with the decoder, and computing the loss based on the predicted and ground truth captions. After training, the model is saved to a file, and the training losses are recorded.

Testing:

The testing stage evaluates a trained sequence-to-sequence (Seq2Seq) model with attention for video captioning by generating captions for a test dataset and calculating the BLEU score to assess the quality of the captions. It loads the trained model and index-to-word mapping, creates a DataLoader for the test dataset, and writes the generated captions to an output file. The BLEU score is calculated by comparing the generated captions with the ground truth captions from a JSON file.

Execution Process:

Training:

Initially, training is performed on the model by running the below command:

python3 training.py /scratch/shravak/HW2/MLDS_hw2_1_data/training_data/feat /scratch/shravak/HW2/MLDS_hw2_1_data/training_label.json

I have specified my respective paths to the training data features folder and training_label.json file in the command. If you want to perform the training, please specify your paths to the respective folders in the same order as in the command.

A trained model with name “model_shravani.h5” is saved at the end of training process.

Testing:

Before starting the testing process, please download the pretrained model “model_shravani.h5” (https://drive.google.com/file/d/19dvvQgTKG4UelaMULE6lmvMWQ06yPlLJ/view?usp=sharing), testing_label.json, index_to_word.pickle and blue_eval.py files to your respective directory.

To test the model, the following shell script (**hw2_seq2seq.sh**) is run with the below command:

sh hw2_seq2seq.sh /scratch/shravak/HW2/MLDS_hw2_1_data/testing_data output_captions.txt

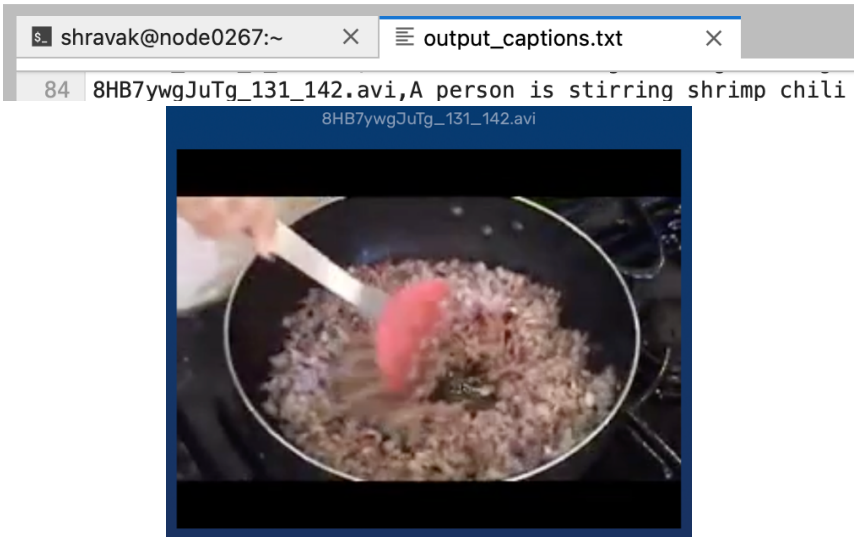
I have specified my respective paths to the testing data folder and output_captions.txt file in the command. If you want to perform the testing, please specify your paths to the respective folders in the same order as in the command.

After the testing process is completed, the resulting captions are stored into an output file.

Results:
The average BLEU score that I have obtained is **0.625.**

```
[shravak@node0267 ~]$ sh hw2_seq2seq.sh /scratch/shravak/HW2/MLDS_hw2_1_data/testing_data output_captions.txt
Please download the model, index-to-word pickle, and 'testing_label.json' to run the shell script.
Average BLEU score is 0.6255014286512114
```

Below are some of the generated captions with the videos:



shravak@node0267:~ | output_captions.txt

```
75 8MVo7fje_oE_125_130.avi,A man is draining pasta
```

shravak@node0267:~ | output_captions.txt

```
55 Je3V7U5Ctj4_569_576.avi,A man is mixing a in tortilla
```

Below are some insights of the assignment:

```
[shravak@node0267 ~]$ python3 training.py /scratch/shravak/HW2/MLDS_hw2_1_data/training_data/feat /scratch/shravak/HW2/MLDS_hw2_1_data/training
_label.json
To initiate the training process, please specify the path to your training data features as the first argument and the path to your 'training_l
abel.json' file as the second argument.
Number of filtered words: 2486
Number of unique videos: 1450
Average caption length: 9.71
Caption length distribution: min=3, max=42, median=9.0
Video ID: xBePrplM4OA_6_18.avi and it's caption: A woman goes under a behind and gets pooped on
```

```
Training finished
Total training time: 7993.48 seconds
[shravak@node0267 ~]$
```

Below is the plot for Loss over Epochs:


Loss vs Epochs Plot
Train Loss
Loss
Epochs
6.25
6.00
5.75
5.50
5.25
5.00
4.75
4.50
4.25
0
25
50
75
100
125
150
175
200