CPSC 8430 – DEEP LEARNING
HOMEWORK 2
Vishnu Sai Nara

Video Caption Generation:

Overview

This assignment focuses on developing a model that generates meaningful and descriptive captions for short videos. The model needs to account for varying video characteristics, such as the presence of different objects and actions, and should handle the dynamic lengths of both video inputs and generated captions.

GitHub Link and Pretrained Model:

GitHubLink:https://github.com/vishnusai-nara/DeepLearning/tree/main/hw2/hw2_1

Pretrained Model:
https://drive.google.com/file/d/1BbkCZaZVoaZdqxsUUOCKJ-Rm_N20JG5d/view?usp=sharing

Testing and training were performed on a CUDA-supported environment (Palmetto Cluster), so CUDA-enabled hardware is recommended for testing.

Dataset:

The MSVD dataset was used for this project. MSVD consists of 1,970 YouTube short video clips, each one accompanied by multiple human-generated English captions. A subset of this dataset was utilized, consisting of:

- **1,450 videos** for training
- **100 videos** for testing

This dataset covers a diverse range of activities and scenarios, including animals, sports, and everyday actions, providing a broad spectrum for training the captioning model.

Data Preprocessing:

Several functions and classes were implemented to preprocess and handle the data efficiently for the caption generation task:

preprocess_data(): This function loads the captions from a JSON file and builds a dictionary by counting word frequencies. It filters out words which appears less than 3 times and creates mappings between words and their respective indices. Special tokens are included for handling the beginning (<BOS>), end (<EOS>), unknown (<UNK>), and padded words (<PAD>).

Special Tokens used:

<PAD>: Adds padding to sentences to ensure a fixed length.

<BOS>:Indicates beginning of the sentence.

<EOS>: Denotes end of the sentence.

<UNK>: Substitutes words which are not present in the vocabulary.

annotate_captions(): Transforms captions into sequences of word indices based on the filtered vocabulary.

create_minibatch(): Organizes captions into mini batches by sorting and padding them to uniform lengths.

DatasetWithFeatures class: Manages the training data by pairing video features with annotated captions.

TestDataset class: Loads the features for the test videos.

Model Architecture

A **sequence-to-sequence (Seq2Seq) model** with attention was implemented, consisting of the following components:

1. **Encoder**:
    - Handles the input video features by applying a linear projection to reduce their dimensionality, followed by dropout for regularization. The condensed features are then passed through a GRU layer to generate hidden states, which are leveraged by the attention mechanism.
2. **Attention Module**:
    - At each time step in the decoder, a context representation is produced. The decoder's active state is merged with the encoder outputs through multiple linear mappings.
    - A softmax function calculates attention weights, generating a weighted sum of the encoder outputs.

3. **Decoder**:
    - Generates output captions by transforming input word indices into dense vectors via an embedding layer.
    - The embeddings are merged with the attention-generated context representation and passed through a GRU to update the decoder's state. A linear layer then outputs the next word in the sequence.
    - During training, the model utilizes teacher forcing, feeding actual target words as input with a probability controlled by a sigmoid-based ratio.
4. **Seq2SeqModel**:
    - Integrates the encoder and decoder to form a complete model.
    - The model can operate in two modes:
        - **Training mode**: Generates captions using ground truth inputs.
        - **Inference mode**: Generates captions without ground truth inputs.

Training

Prior to training, the data undergoes preprocessing to build the vocabulary and remove rare words. The prepared dataset, containing video features and captions, is subsequently fed into a DataLoader to enable smooth and efficient batch processing.The training configuration consists of 200 epochs with a batch size of 128. An Adam optimizer is used with a learning rate of 0.001, and the model's performance is optimized using Cross-Entropy Loss as the loss function.

During the training, the models will learn by:

The training process begins by passing the video features through the encoder to produce hidden states. These hidden states are used by the decoder to generate captions.The loss is computed by evaluating the difference between the predicted captions and the actual ground truth captions. Lastly, the model parameters are adjusted using backpropagation to reduce the loss.

At the end of the training, the model is saved as **"model_vishnu.h5"**, and the training losses will be logged.

Testing

During the testing phase, the trained Seq2Seq model with attention is evaluated by generating captions for the test videos. The quality of these generated captions is measured using the BLEU score, providing an assessment of how well the model performs.

Execution Process

Training Command: To train the model from scratch use:

python training.py MLDS_hw2_1_data/training_data/feat MLDS_hw2_1_data/training_label.json

After training, the model will be saved as "model_vishnu.h5".

Testing Command: To test the pretrained model, run:

sh hw2_seq2seq.sh MLDS_hw2_1_data/testing_data video_captions.txt

Pretrained Model: Download the pretrained model, index2word.pickle, blue_eval.py, and testing_label.json. Run the test command.

Results:The results are saved in the specified output file, `video_captions.txt`, containing the generated captions for the test videos. The BLEU score is evaluated by matching the generated captions against the reference captions found in the testing_label.json file, offering a quantitative measure of the model's performance.

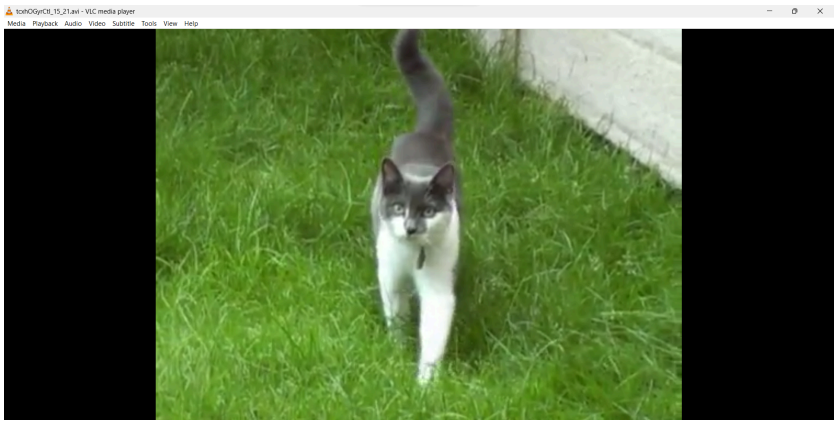
Results: The model achieved an average BLEU score of **0.6406**, reflecting the quality of the generated captions.

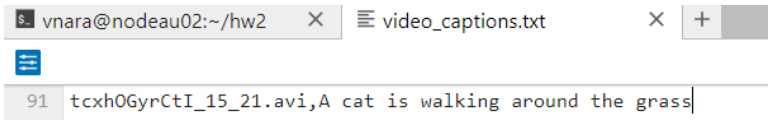
```
[vnara@nodeau02 hw2]$ sh hw2_seq2seq.sh MLDS_hw2_1_data/testing_data video_captions.txt
Ensure the model, index-to-word pickle, and 'testing_label.json' are available.
/home/vnara/hw2/test.py:22: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the default pickle module implicitly. It is pos
sible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more de
tails). In a future release, the default value for `weights_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will
 no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add_safe_globals`. We recommend you start setting `weights
_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  state_dict = torch.load(model_path, map_location=device)
Average BLEU score: 0.6406011692544538
```

Sample Captions:

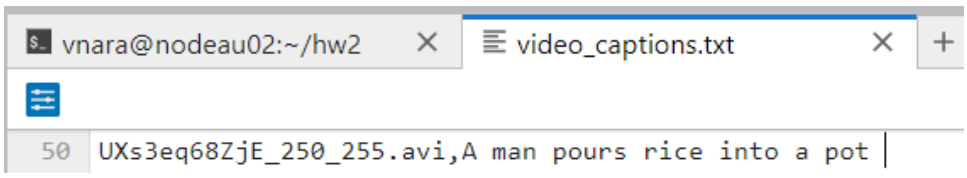
Here are a few examples of captions generated by the model:

Video 1: "A cat is walking around the grass ."

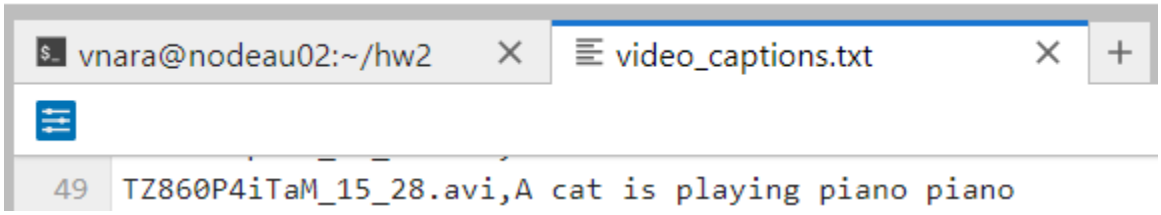


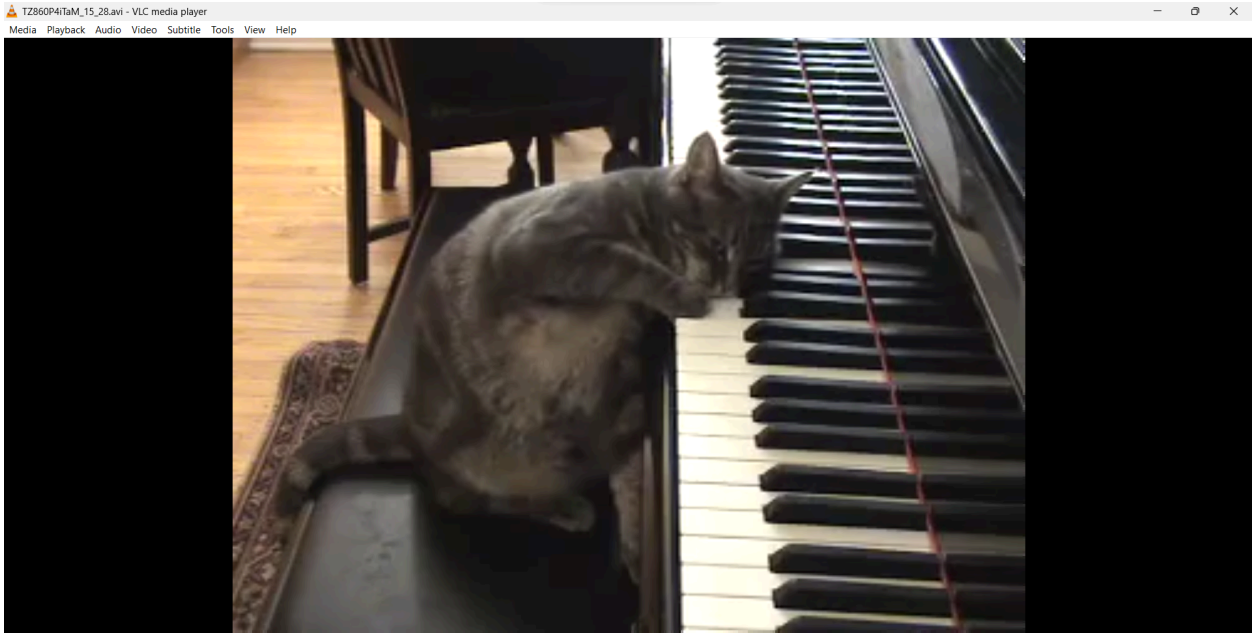


Video 2: "A man pours rice into a pot"



Video 3: "A cat is playing piano piano."



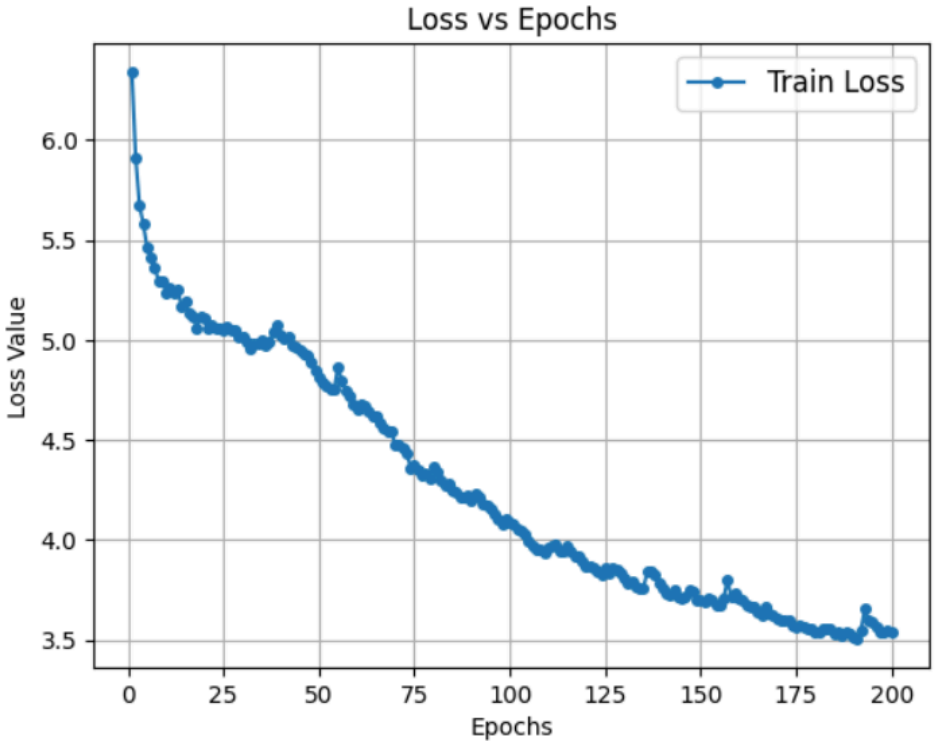


Some Parameters from the assignment:

```
[vnara@node0405 HW2]$ python training.py MLDS_hw2_1_data/training_data/feat 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_label.json' file as the secon
d 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: 7769.90 seconds
```

Graph for Loss over Epoch:



Conclusion

This assignment demonstrates the effectiveness of a sequence to sequence model with attention for generating captions for video clips. The model performs well on a subset of the MSVD dataset, achieving a BLEU score of 0.6406. This performance shows that the model can generate meaningful captions by understanding the content of short videos. Further improvements could involve experimenting with different architectures or increasing the dataset size to enhance caption quality.