HOMEWORK-2

INTRODUCTION-

For the domains of natural language processing and computer vision, the automatic creation of natural language captions for movies has been a considerable issue. The complex dynamics of real-time movies, which need techniques sensitive to temporal structures and capable of handling inputs and outputs of different durations, make this challenge much more challenging. We have used an encoder-decoder framework and the MSVD dataset for model assessment to deal with this problem. Two (LSTMs) are used as our method, one for encoding and the other for decoding. Deep neural networks are used to train an encoder to learn how to represent the video, and a sentence is produced for this representation by the decoder. We use a Beam search technique to generate effective captions.

Sequence-to-sequence applications like voice recognition and machine translation have shown the effectiveness of LSTMs. A stacked LSTM that is used for encoding transmits the frames 1 by 1. The results of a CNN trained on the intensity values of each input frame serves as the LSTM's input.

They also use a two-step method where they first use CRFs to gather meaningful itemset of action, objects, tools, and regions and then use LSTM to convert those tuples into sentences. The small domain of cookery videos is where the method is used.

Features of Dataset-

After preprocessing with pre-trained CNN VGG19, we saved the characteristics of each clip in the format of 804096. To get the most out of the training, we don't cut any frames from the video

TOPIC	DESCRIPTION
Dataset	MSVD (Microsoft video Description) dataset
Contents	Consists of about 120k sentences
Data Collection	Workers on Mechanical Turk were paid to optically canvas a short video snippet and then summarize the action in a single sentence.
Video Snippets	1550 video snippets each ranging from 10 to 25 seconds
Data Split	1450 and 100 videos for training and testing respectively
Video Features	Stored in the format of 80*4096 after preprocessing using pre-trained CNN VGG19
Training	Do not reduce any number of frames from a video to take full advantage.

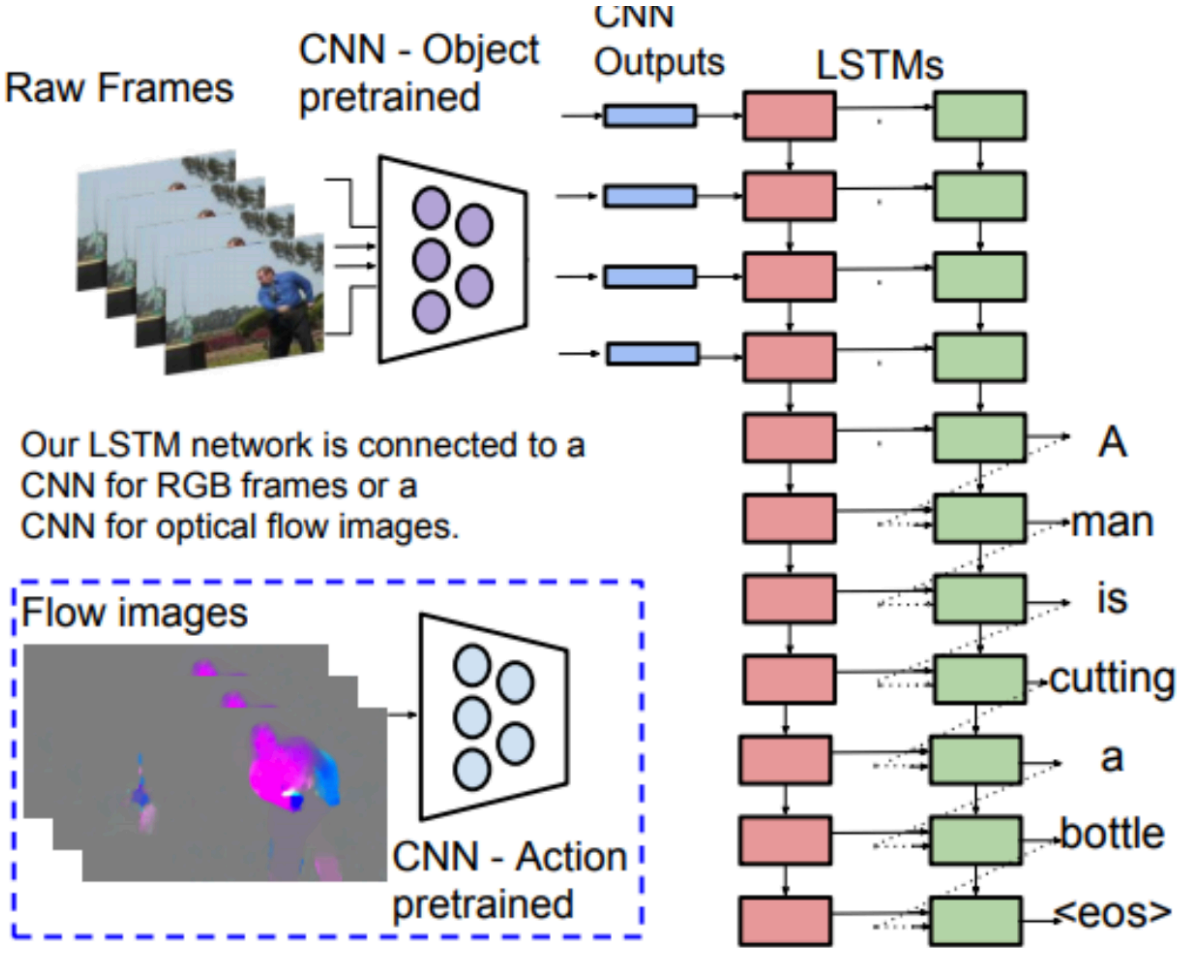
IMAGE DESCRIPTION-

1. A can of water is added to a cooking saucepan that is on a burner.
2. A can is being filled with water.
3. A man is filling the can with water.
4. The man filled the tomato sauce container with water.

A selection of images from a sample video in the training set is shown in the image above.

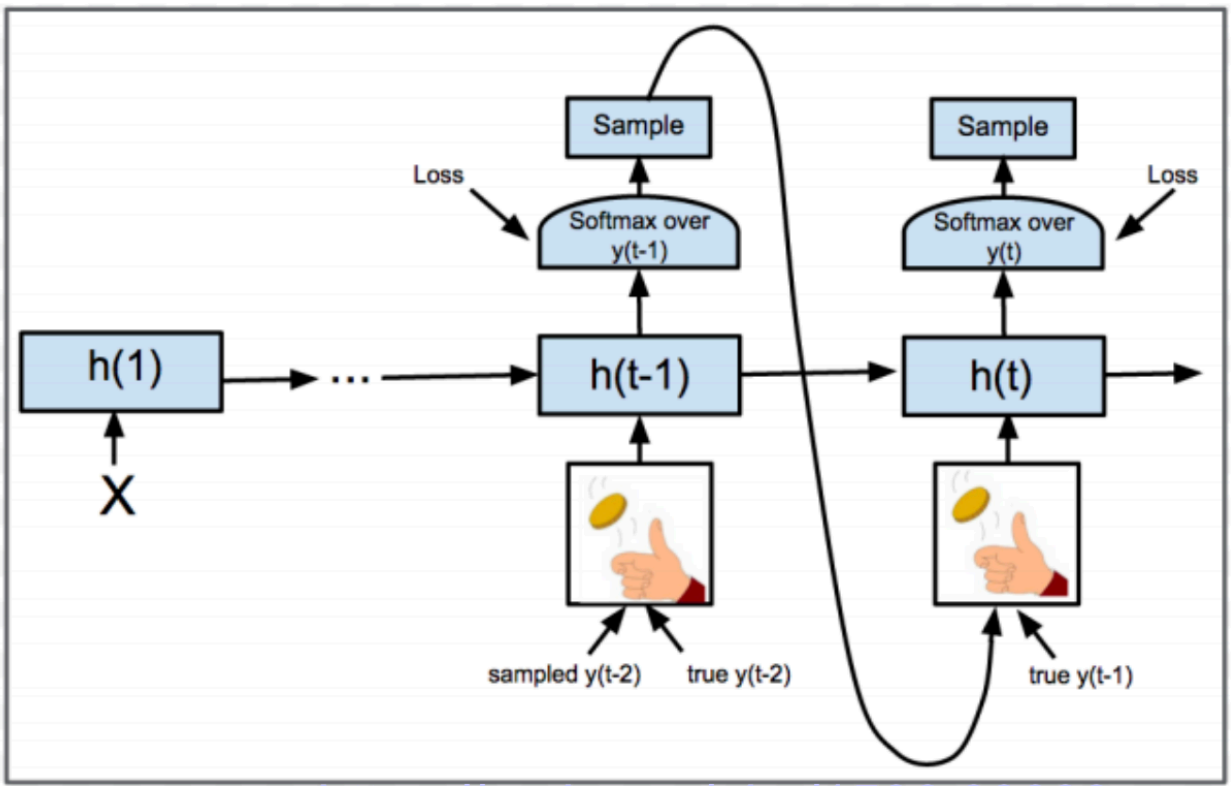
3. Accession-

This model is a seq-to-seq model that generates a series of words from a process of video frames.

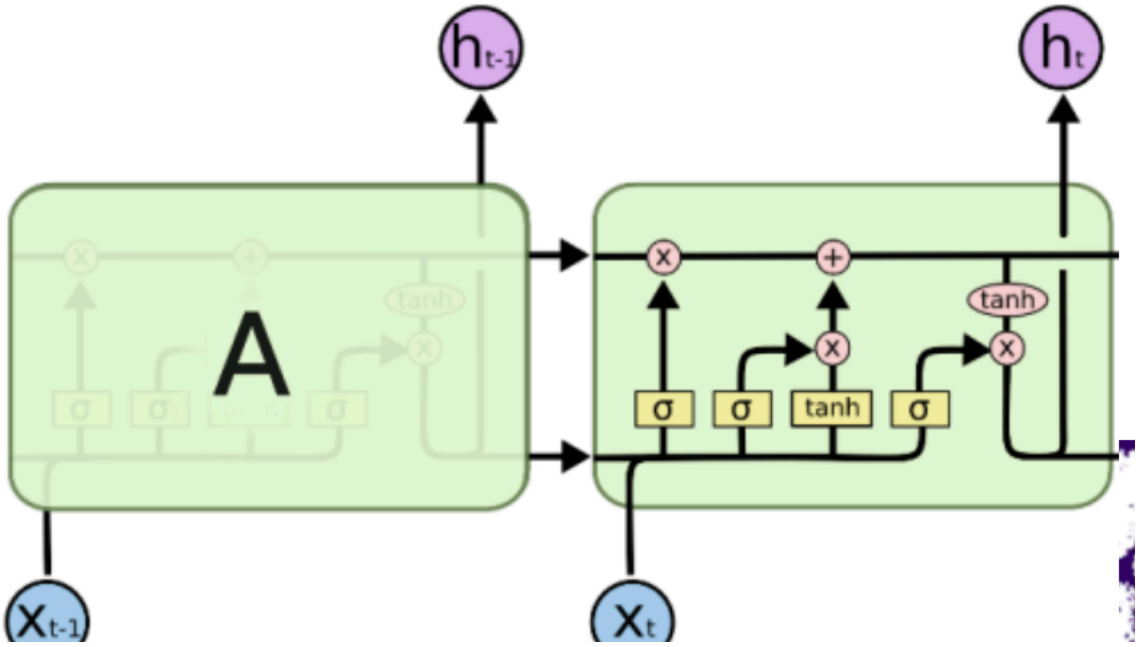


Schedule Sampling-

To solve the “exposure bias” problem, when training, we feed ground truth as input at the odds.

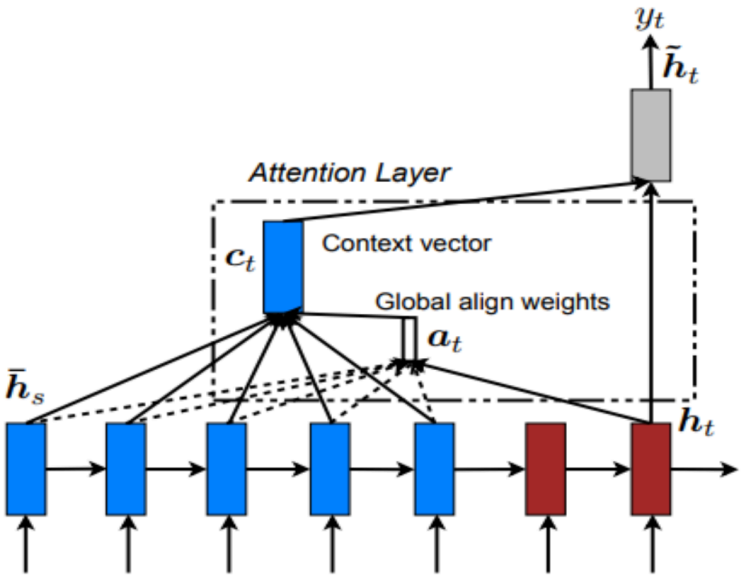


LSTM Unit-

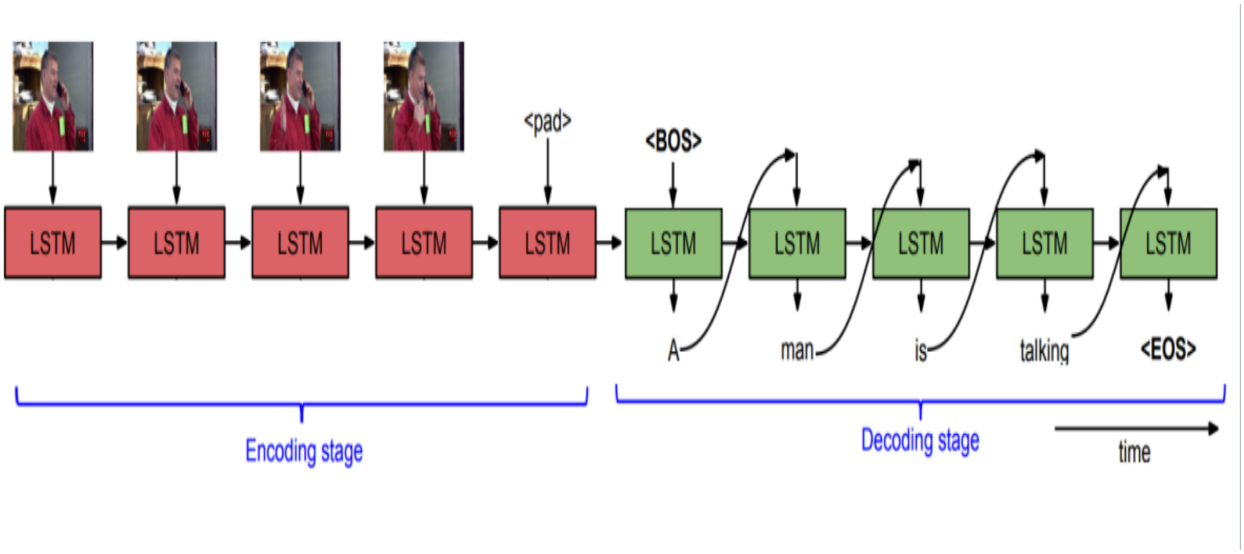


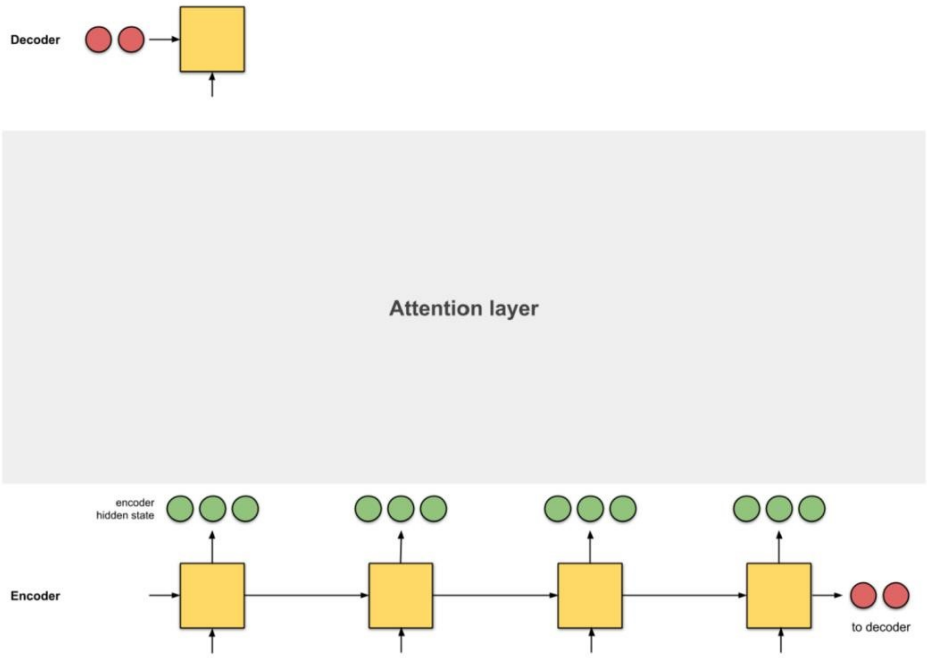
LSTM for Encoding and Decoding-

We will use the LSTM RNN structure



Model of LSTM'S-





Requirements-

The Model is operational through the utilization of the following technologies:

- TensorFlow-gpu version 1.15.0
- Cuda version 9.0
- python version 3.6.0
- NumPy version 1.14
- panda's version 0.20

Tokens-

- <PAD> : Pad the sentencen to the same length
- <BOS> : Begin of sentence, a sign to generate the output sentence.
- <EOS> : End of sentence, a sign of the end of the output sentence.
- <UNK> : Use this token when the word isn't in the dictionary or just ignore the unknown word.

```
python sequence.py /home/skomman/Sandeep/MLDS_hw2_data/training_data/feat/
```

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

```
/.conda/envs/tf_gpu_env/lib/python3.7/site-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a li
```

```
r-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
Caption dimenstion is: (24232, 2)
Caption's max length is: 40
Average length of the captions: 7.711084516342027
Unique tokens: 6443
ID of 11th video: iTA0rWPE4nY_17_23.avi
Shape of features of 11th video: (80, 4096)
Caption of 11th video: A man places chicken into a container
```

```
/home/skomman/skomman/MLDS_hw2_data/training_label.json
```

HyperParameter	Value
Learning rate	0.001
Use_attention	True
Max_encoder_steps	64
Max_decoder_steps	15
Embedding_size	1024
Batch_size	50
Beam_size	5(if beam search is True)

```
python train.py /home/skomman/Sandeep/MLDS_hw2_data/testing_data/feat/
/home/skomman/Sandeep/MLDS_hw2_data/testing_label.json ./output_testset_Sandeep.txt
```

```
Highest [10] BLEU scores:  ['0.5488', '0.5000', '0.4935', '0.4886', '0.4546']
Epoch# 4, Loss: 0.8858, Average BLEU score: 0.5000, Time taken: 28.79s
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:0650/1450
Training done for batch:0700/1450
Training done for batch:0750/1450
Training done for batch:0800/1450
```

I computed the Bleu score after each period.

Outputs-

```
Originally, average bleu score is 0.2689437917016406
By another method, average bleu score is 0.5760704024416632
```

The following command is computed for bleu score

```
python bleu_eval.py test_Sandeep.txt
```

After four epochs, the average score is about 0.57, i.e., 0.6.

After 200 epochs of training, the bleu score is about 0.610.

https://github.com/SandeepKOmmanaboyina/Deeplearning_HW2/tree/main/hw2/hw2_1