Deep Learning Homework 2- Report

Prashanth Reddy Kadire
Dr. Feng Luo
CPSC 8430-Deep Learning
25 March 2022

Abstract:

Generating video captions automatically with natural language has been a challenge for both the field of natural language processing and computer vision. The Real-time videos will have complex dynamics, and the methods we use should be sensitive to temporal structure, and it should allow input and output of variable length. We will be using the Encoder-Decoder framework, and the MSVD dataset is used for model assessment. We use two Long Short Term Memory (LSTMs) units, one for encoding and the other for decoding. An encoder equipped with deep neural networks is used to learn video representation. On the other hand, the decoder will generate a sentence for the learned representation. For generating efficient captions, we use the Beam search algorithm.

INTRODUCTION:

In visual interpretation, Re-current Neural Networks (RNNs), which model sequence dynamics, proved to be efficient. Image description will try to handle a variable length output sequence of words. In contrast, a video description must handle a variable length input sequence of frames and variable length output sequence of words.

Model should be dynamic and able to learn new images. So let us build a sequence-to-sequence model that will take the input as frames and generate variable length text output. For better function of the sequence-to-sequence model, we will be using Long Short Term Memory (LSTM), a type of RNN.

LSTM's have been proved to be efficient in sequence-to-sequence tasks like Speech recognition and Machine translation. Encoding is done by a stacked LSTM which encodes the frames one by one. The output of a Convolutional Neural Network applied to each input frame's intensity values is used as input for the LSTM. Once all the frames are read, the model generates a sentence word by word. LSTMs are used to generate video captions by pooling the representation of all frames. They use mean-pools on CNN output features to get a single feature vector. The major drawback of this approach is that it completely ignores the organization of the video frames and fails to manipulate any temporal information.

A hierarchical recurrent neural encoder is proposed. In the encoding stage, a supplemental temporal filter layer composing LSTM cells is introduced to better denude the temporal dependency of the input frames. To our erudition, this work has reported the highest METEOR score on the MSVD data so far, which can be attributed to the more nonlinearity integrated by the temporal filter layer.

Attention mechanisms have been incorporated into various neural networks, since they sanction salient features to dynamically come to the forefront as needed. Several attention models have been reported to be subsidiary in sundry tasks including machine translation, image captioning and video captioning.

Dataset and Features:

The dataset we used in this experiment is Microsoft Video Description dataset. The Microsoft Research Video Description Corpus dataset consists of about 120K sentences. Workers on Mechanical Turk were paid to optically canvass a short video snippet and then summarize the action in a single sentence. The Dataset contains 1550 video snippets each ranging from 10 to 25 seconds. We have taken the split to 1450 and 100 videos for training and testing respectively. We have stored the features of each video in the format of 80×4096 after pre- processing using pre-trained CNN VGG19. During, training we don't reduce any number of frames from video to take full advantage.

1. A Woman is cutting a vegetable
2. A woman peels a potato
3. A man is cutting a vegetable

Fig: Frames from training video

Approach:

Our Model is a seq-to-seq model which takes sequence of video frames as input and generatesequence of words as output. The video frames are fed to the pre-trained CNN VGG19, which generates the spatial feature map of all the videos in the format of 80×4096. We have the feature maps generated for every video in the dataset.

Long Short-Term Memory Networks for Encoding and Decoding

We are required to handle the variable-sized input and variable-sized outputs, and we will be utilizing LSTM Recurrent Neural Network architecture. The entire input sequence (x1, x2, ..., xn) is first encoded, summarizing the video into one hidden state vector, which is utilized to decode a natural language description of the features shown.

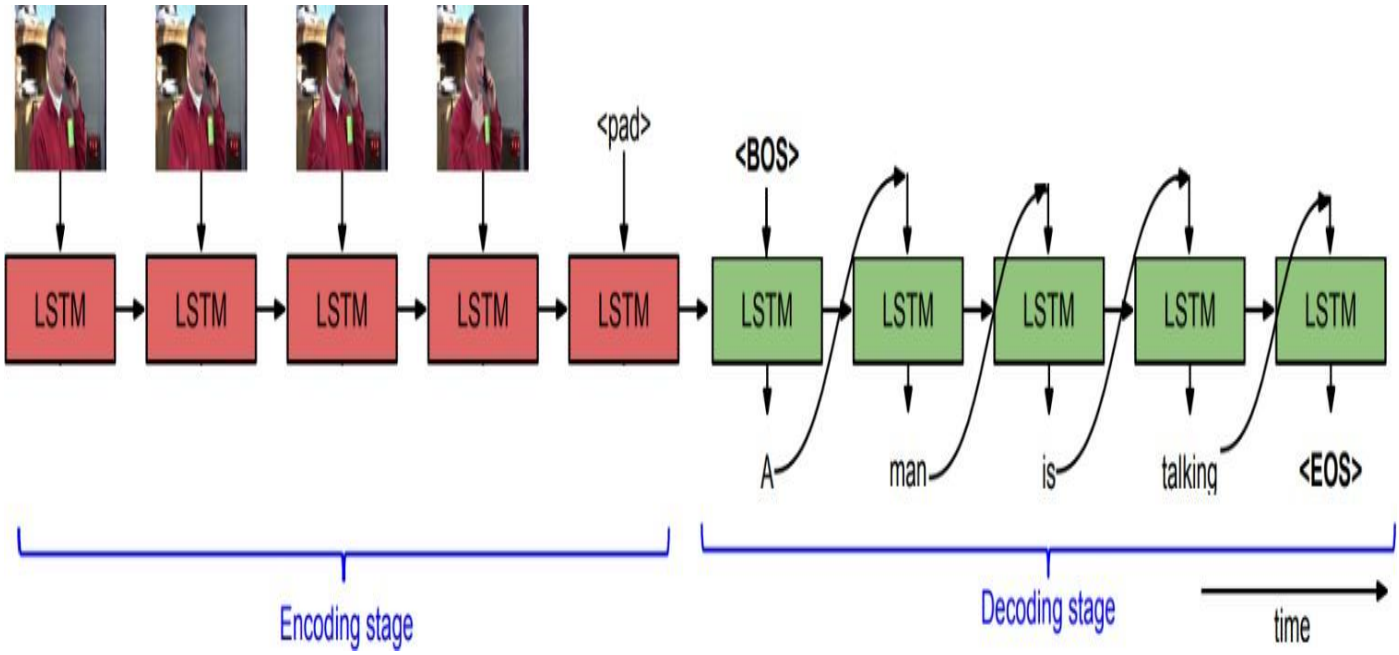

Fig: Architecture of LSTMs

Attention is an interface between the encoder and decoder that provides the decoder withinformation from every encoder hidden state.

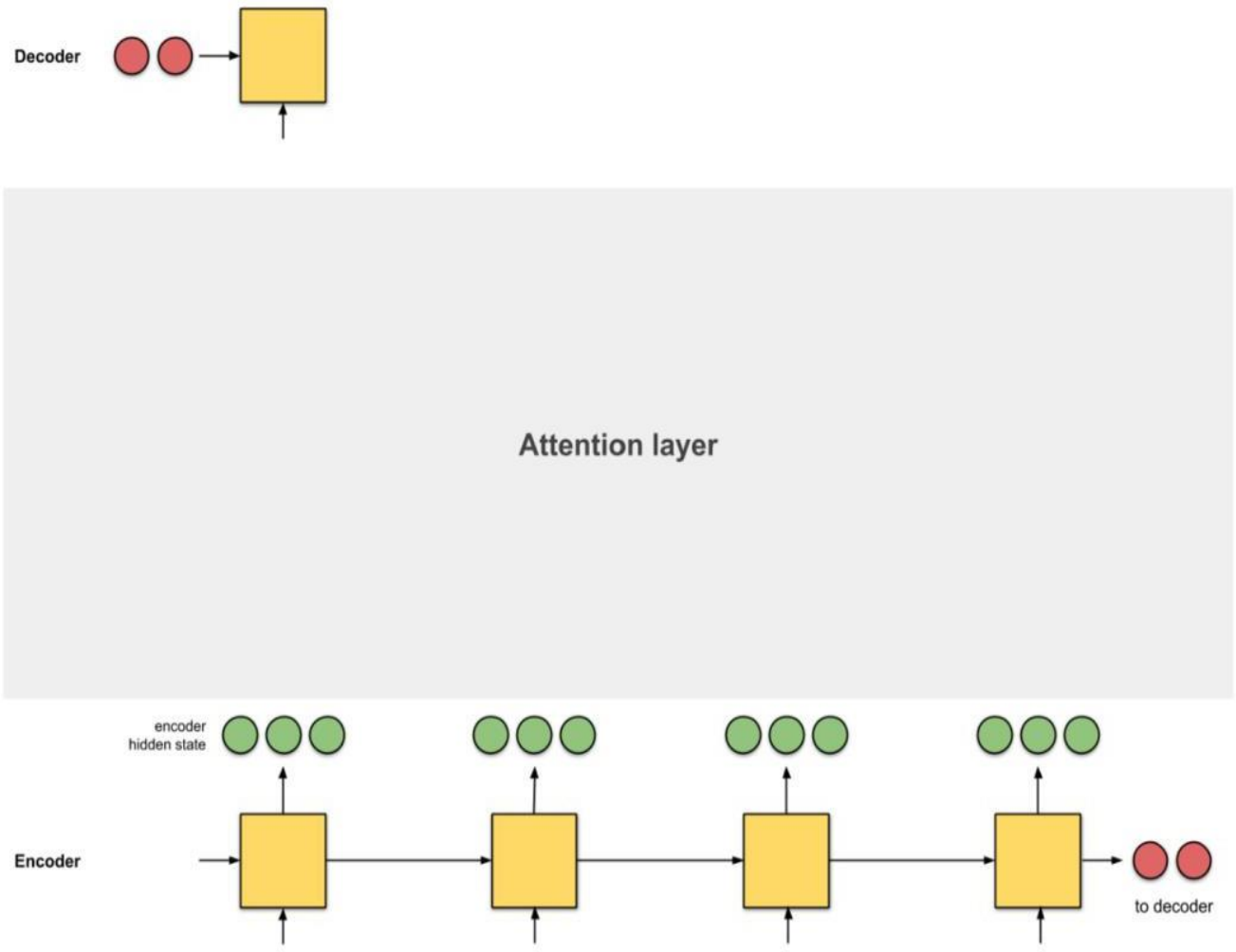

Fig: Attention Layer

BLEU Score

Bilingual Evaluation Understudy is an algorithm for evaluating the quality of text machine-translated from one natural language to another. Quality is considered the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU. BLEU was one of the first metrics to claim a high correlation with human judgments of quality and remains one of the most popular automated and inexpensive metrics.

Challenges

Sequence to sequence modeling scope is broader than just the machine translation task. However, there are very few source documents available on the same topic.

There are multiple challenges we can face during implementation:

i) Different attributes of video like action and objects.

ii) Variable length of I/O.

Requirement:

- tensorflow-gpu==1.15.0
- cuda==9.0
- python 3.9.7
- numpy== 1.14
- pandas==0.20

Execution Steps:

1) Padded sequences and makings are added in the preprocess step to the data. Packed padded arrangements allow us to only process the non-padded elements of our input sentence with our RNN. Masking forces the model to ignore certain aspects we do not want it to look at, such as attention over padded elements. This step is executed for performance boosting. The training and testing data were downloaded from the PowerPoint presentation provided and kept local under the 'MLDS_hw2_1_data' folder. Vocab size should minimum be 3.

2) Tokenizing:

a. <pad>: Pad the sentence to the same length

b. <bos>: Begin of the sentence, a sign to generate the output sentence.

c. <eos>: End of sentence, a sign of the end of the output sentence.

d. <unk>: Use this token when the word isn't in the dictionary or just ignore the unknown word.

Two object files are built to hold the dictionary of id and caption of respective videos. Object files 'vid_id.obj',' dict_caption.obj',' dict_feat.obj' are being written to build dictionary.

For the Execution of sequence.py, I used the following command in the Palmetto cluster at my active node:

python sequence.py prashanthreddykadire/Desktop/MLDS_hw2_data/training_data/feat/ prashanthreddykadire/Desktop/Prashanth/MLDS_hw2_data/training_label.json

```
(tf_gpu_env) [prashanth@login Prashanth] python sequence.py
prashanthreddykadire/Desktop/MLDS_hw2_data/training_data/feat/
prashanthreddykadire/Desktop/Prashanth/MLDS_hw2_data/training_la
bel.json From 6098 words filtered 2881 words to dictionary with minimum
count [3]

{Desktop/Prashanth/ conda/envs/tf_gpu_env/lib/python3.9.7/site-
packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an
ndarray from ragged nested sequences (which is @ 1ir-tuples-or ndarrays
with different lengths or shapes) is deprecated. If you meant to do this,
you must specify 'dtypesobject' 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: 7.711684516342827

Unique tokens: 6443

ID of Sith videos iTA0rWPE4nY_17_23.avi

Shape of features of 11th videos (88, 4096)

Caption of 11th video: A man places chicken into a container

(tf_gpu_env) [prashanth@login Prahanth]$
```

ii) Our next task is to train the model we created. We have two python files here. We will build the sequence-to-sequence model with sequence.py. We will train the model with the help of train.py. Please refer to the below table for Hyperparameters used in the experiment. A hyperparameter is a parameter whose value is used to control the learning process. An example of a model hyperparameter is the

topology and size of a neural network. Examples of algorithm hyperparameters are learning rate and batch size**.**

Hyperparameter	**Values**
Learning rate	0.001
UseAttention	True
MaxEncoderSteps	64
MaxDecoderSteps	15
EmbeddingSize	1024
BatchSize	50
BeamSize	5(if Beamsearch is True)

I have run the following command in the palmetto cluster on my active node using ssh.

pythontrain.py /Desktop/Prashanth/MLDS_hw2_data/testing_data/feat/
/desktop/Prashanth/MLDS_hw2_data/testing_label.json ./output_testset_prashanth.txt

The first argument is the path of the feature map, which is in the format of the .npy file, and the second argument I sth path of the testing_label.json file, which has captions for a particular video id.

```
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:0850/1450
Training done for batch:0700/1450
Training done for batch:0750/1450
Training done for batch:0800/1450
```

I have calculated Bleu score after every epoch. You can see the array of bleu scores in thedescending order of the scores.

```
[(tf_gpu_env) [prashanth@login Prashanth]$ python bleu.eval.py
test_prashanth.txt Originally, average bleu score, 0.268943791701406
By another method, average bleu score is 0.5740704024416032
(tf_gpu_env) [prashanth@login Prahanth]$
```

I have calculated the Bleu score using the following command.

python bleu_eval.py test_prashanth.txt

you can see the average score is reaching almost 0.57 i.e., ~0.6 after four epochs

After training the model for 200 Epochs, the bleu score is near to 0.610

References:

1) http://www.cs.utexas.edu/users/ml/papers/venugopalan.iccv15.pdf
2) https://gist.github.com/vsubhashini/38d087e140854fee4b14