**Sandya Rani Prasadam HW2**

**Github:** [**https://github.com/sandya-33/deep-learning\_hw2**](https://github.com/sandya-33/deep-learning_hw2)

I couldn’t upload my model because it is too big to upload. I could mail you if its necessary.

**Used tokens:**

Dictionary - most frequently word or min count  
<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

**Data preprocessing:** The script loads the training data, extracts the words from the captions, and creates a word dictionary.

**Model:**

The model has 2 layers namely, encoder and decoder. The LSTM-long short-term memory networks .  It is used in DL. It is a variety of RNN that is capable of learning long-term dependencies, especially in sequence prediction problems. LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data, apart from single data points such as images. But it takes so much memory and takes so much time to build model. The model uses the LSTM (Long Short-Term Memory) cell to encode the input sequence and generate the corresponding hidden states. The encoder processes the video and encodes it into the necessary format. The decoder is then used to segment the captions based on the beginning and ending tokens and perform video processing over the words to produce actual words.

**Attention Layer:**

The attention mechanism allows the decoder to focus on different parts of the encoder output at different time steps, allowing for better performance in tasks such as machine translation or image captioning.

The attention module takes in the hidden state of the decoder and the encoder outputs (which are the features extracted from the input image or sequence) and computes a weight for each encoder output that indicates how important it is to the current decoding step. These weights are then used to compute a weighted sum of the encoder outputs, which is passed to the decoder as an input.

**Parameters:**

Training Epoch = 200   
LSTM dimension = 256   
Learning rate = 0.001  
 vocab size = min count > 3

AdamOptimizer

**Dataset size:**

1450 videos for training  
100 videos for testing

Loss: 3.66263

Average Bleu score: 0.66638