# Parthiv Patel GitHub Repo: <https://github.com/Parthivp2205/CPSC-8430.git>

# CPSC - 8430 Deep Learning

Homework 1

How to run:  
hw2\_seq2\_training.ipynb: files that I used for training the model.

your\_seq2seq\_model: this folder contains the trained model (model\_weights.h5) and other json file required.  
  
ckpt.pt: which is inside your\_seq2seq\_model, contains the data from running each epoch.

My model: model\_weights.h5, which can be found under my folder your\_seq2seq\_model

hw2\_seq2seq.sh: is the bash script file to run the model\_seq2seq.py file.  
  
To run the bash file: first clone my repository and then run : ./hw2\_seq2seq.sh testing\_data testset\_output.txt   
  
**cause my training script stores the model in your\_seq2seq\_model folder.**

**Project Overview**The goal of this assignment is to design and implement a video caption generation system using a sequence-to-sequence (Seq2Seq) model. The model takes short video clips as input and generates natural language captions that describe the video content. This is a challenging task due to variable video lengths, diverse actions/objects, and the need for accurate alignment between frames and text. The baseline BLEU-1 score provided in the project guidelines was 0.6, and the objective was to meet or surpass this benchmark. The dataset used is the MSVD dataset, containing approximately 1,450 videos for training and 100 videos for testing. Experiments were conducted with a batch size of 32, 200 epochs, and a learning rate of 0.001. All code was implemented in Python using PyTorch, and training was performed with GPU acceleration when available. The workflow is:

1. Build a vocabulary from training captions
2. Load per-video feature tensors
3. Train Encoder - Decoder
4. Save a checkpoint & metadata
5. Run inference with greedy/beam search
6. Write video\_id, caption lines to a text file and generate the BLEU score.

**Vocabulary and Preprocessing**

To process captions, a vocabulary was built by scanning all training captions and retaining words with frequency above a threshold (minimum count = 4). Four special tokens were added:

* <PAD>: padding
* <BOS>: beginning of sentence
* <EOS>: end of sentence
* <UNK>: unknown words

Captions were tokenized, converted into indices, and bounded by maximum length constraints.

The Vocab class was implemented to maintain the word-to-index (stoi) and index-to-word (itos) mappings. Its decode() function reconstructs captions from predicted token IDs, ignoring special tokens.

**Data Handling**

Features were pre-extracted from video frames and stored in .npy format. The load\_feat() function reshapes and normalizes feature arrays, while list\_video\_ids() lists available video samples in a given directory.

Training and testing data were paired with corresponding captions from JSON label files, ensuring multiple ground truth captions per video. This allows the model to learn diverse captioning patterns for the same video.

**Dataset**

* Source: MSVD dataset provided in class
* Training set: 1450 videos
* Epoch: 200
* Batch size: 32
* Learning rate: 1e-3
* Min count: 4
* Captions: JSON files (training\_label.json and testing\_label.json) mapping each video ID to multiple ground truth captions

**Model Architecture**

The model follows a Seq2Seq structure with GRU units and optional attention. It is implemented as an S2VT wrapper class that integrates encoder, decoder, and decoding strategies:

* Encoder (Encoder class): GRU network that processes sequences of video features and outputs hidden states summarizing temporal information.
* Attention (Attention class): Scaled dot-product attention module. Computes weights over encoder outputs based on the current decoder hidden state.
* Decoder (Decoder class): GRU network with embeddings, optional attention integration, and a linear output projection over the vocabulary.

A diagram of a system

Description automatically generated

* S2VT Wrapper: Orchestrates encoder and decoder, and provides greedy\_decode() for inference.

This modular design ensures flexibility in training and decoding strategies.

**Functions and Their Role**

**Vocabulary (Vocab class)**

* build(): constructs word-to-index (stoi) and index-to-word (itos) mappings with frequency threshold
* tokenize(): splits caption into lowercase tokens
* encode(): maps a caption into a sequence of token IDs, with <BOS> and <EOS> markers

**Dataset (VideoCaptionDataset)**

* \_\_getitem\_\_: returns (video\_id, features, one\_random\_caption\_encoded)
* \_load\_feat(): handles .npy feature loading with reshaping
* Integrated with a **collate function** that pads variable-length sequences

**Model Functions**

* Encoder.forward(): encodes video features into hidden states
* Attention.forward(): computes attention weights + context vector
* Decoder.forward(): predicts token probabilities using embedding + GRU + linear layer
* S2VT.forward(): orchestrates encoder–decoder with teacher forcing
* S2VT.beam\_search(): performs beam search decoding (beam size = 3)
* Training **& Evaluation Functions**
* train\_model(): full training loop with teacher forcing, gradient clipping, Adam optimizer
* bleu1\_precision(): computes BLEU-1 score (precision-based) for generated captions
* run\_inference(): loads checkpoint, decodes test set captions, computes BLEU-1

**Training Strategy**

Training was implemented in the hw2\_seq2\_training.ipynb notebook. The model was trained for 200 epochs using the Adam optimizer with learning rate = 0.001.

* Loss Function: CrossEntropyLoss, ignoring <PAD> tokens.
* Teacher Forcing & Schedule Sampling: Teacher forcing was used early in training, gradually reduced via schedule sampling to reduce exposure bias.
* Gradient Clipping: Applied with a maximum norm of 1.0 to stabilize training.
* Checkpoints: Saved after each epoch (e.g., ckpt.pt) to resume or analyze progress.

This training strategy ensured gradual convergence and prevented instability.

**Training Process**

* **Epochs**: 200
* **Embedding Size**: 256
* **Hidden Size**: 256
* **Optimizer**: Adam (lr=0.0001)
* **Loss Function**: Cross-Entropy (ignores <PAD> tokens)
* **Regularization**: Gradient clipping (max norm = 1.0)
* **Schedule Sampling**: Gradually replaced teacher forcing with self-predicted tokens to reduce exposure bias
* **Attention**: Enabled, improving alignment with video frames

During training, the model gradually learned to map video features to natural captions, reducing loss steadily across epochs.  
  
first Epoch: loss=5.0095 - tf=1.00

A table of numbers and symbols

Description automatically generated with medium confidence

Last Epoch: loss=2.4042 - tf=0.00

A close-up of numbers

Description automatically generated

Evaluation and Result

* Evaluation Metric: BLEU-1 Score
* Decoding Strategy: Beam search with beam size = 3
* Result:
  + Average BLEU-1 = 0.669 on the test set
* This surpasses the baseline BLEU-1 target (~0.6) mentioned in the slides
* Example Outputs: Generated captions were coherent, capturing main video actions (e.g., “a man is playing guitar”, “a woman is cutting vegetables”)

A screenshot of a computer screen

Description automatically generated

**Conclusion**

In this project, I was able to successfully implement a Seq2Seq model with GRU, attention, and beam search for video captioning. Compared to greedy decoding, beam search significantly improved fluency and BLEU scores.

The final model was trained for **200 epochs** and achieved an **average BLEU-1 score of 0.669**, indicating effective caption generation capability.

