EE 782 Advanced Topics in Machine Learning, Jul-Nov 2025, IIT Bombay

Assignment 1: End-to-End Image Captioning for Remote Sensing

Learning Objectives:

1. Build two working captioners for overhead imagery: CNN + LSTM and CNN + Transformer decoder.
2. Probe and repair typical LLM coding failures (dataloaders, padding/masks, shapes, device placement, loss/label types, decoding loops).
3. Decompose the problem into clean modules (data curation, tokenization, feature extraction, decoding, evaluation).
4. Deliver insight beyond BLEU: qualitative failure slices, Grad-CAM on vision features, attention/saliency on text, and a miscaption case study.
5. You may use LLMs for coding support, but you MUST REPORT PROMPT and ORIGINAL CODE OUTPUT in text cells of your ipynb, AND YOUR MODIFIED CODE in the code cells. Include copius comments.
6. Report within a single ipynb(Google Colab link with pubic view enabled preferred) with copius use of text cells to walk us through your plan, observations, learnings, and new ideas, with ablations, seeds, and limitations. Include Sections of a paper-style report in the ipynb text cells itself - Abstract, Introduction, Methods, Results, Discussion and Conclusions.
7. **Submit a link to a video of you walking us through the ipynb, explaining all major lines of code, your observations and learnings. A video that cannot be viewed or downloaded because of permission error will be given 0.**

1) Problem & Dataset (Remote-Sensing Captioning) [1]

Task:
Generate a single sentence describing a satellite/aerial image (e.g., land use, structures, scene layout).
Use RSICD data from Kaggle.Use approximately 10.9k images (RGB, various sizes), 5 captions/image.
Splits: train 8,000 / val 1,500 / test 1,421. Using pre-existing code will be penalized. Using LLM for coding is fine, but you must understand the code.

Preprocessing:
Resize to 224×224; start with ImageNet normalization and justify any change.
Tokenize captions with a word-level vocabulary (~10k; built on train only). Add <bos>, <eos>, <pad>. Limit max length (e.g., 22–24).
Save a train/val text stats table: vocab coverage, OOV %, length histogram.

2) Baselines to Implement (both required) [4]

2.1 CNN Encoder (shared)
Experiment with ResNet-18 and MobileNet (ImageNet weights).
Remove classifier; use global average pooling
Feature-cache mode (compute-light): Precompute and save per-image features as .pt (batched, torch.no_grad()).
End-to-end (last-layer fine-tune): Freeze all but last block; keep batch size small.

2.2 LSTM Decoder
Embedding dim 300–512; 1–2 LSTM layers, hidden 512.
Project image feature to initial hidden state or prepend as a learned <img> token—pick one and justify.
Train with teacher forcing + cross-entropy (ignore PAD with ignore_index).
Inference: greedy (beam-3 optional for extra learning).

2.3 Transformer Decoder
nn.TransformerDecoder with 2–4 layers, 4–8 heads, d_model=512.

Provide causal mask and key padding mask; project image feature to a short “memory” sequence (e.g., tile/proj to 1–4 tokens) with LayerNorm. (optional bonus)
Training/inference same loss/decoding policy as LSTM.
Optimizers: Adam (default betas). Try LR (e.g., 2e-4 LSTM/heads; 1e-4 CNN head; 2e-5 for any unfrozen Transformer layers). Try simple LR scheduling, such as decrease LR after a few epochs.

3) LLM-Aware Development & Debugging Diary (mandatory) [3]
You may scaffold code with an LLM, but you must document and fix at least 4 distinct, authentic issues:
Typical failures we expect:
Collate/padding: inconsistent lengths, missing batch_first=True, wrong ignore_index in loss.
Masks: absent causal mask; key-padding mask not propagated; masks on wrong device.
Shapes: [L,B,D] vs [B,L,D] confusion; CNN features [B, C, H, W] not pooled.
Device/precision: CPU tensors in CUDA graph; missing .eval() for validation; float64 drifts.
Loss/labels: CE fed one-hot; labels are float; PAD tokens included in loss.
Decoding: never-ending loops (no <eos> check), length collapse, or repeated tokens from missing temperature clamp.
Tokenizer drift: train/val built with different vocab or different max_len.
Report each LLM bug that you resolved:
Unit check (tiny code/assert or shape printout proving the fix)

4) Experiments & Extensions (pick ≥1 meaningful) [3]
Stay within the syllabus; keep lightweight and principled. Examples:
Backbone swap: ResNet18 ↔ MobileNet; report memory/speed vs BLEU/METEOR.
Input handling: try rotation-aware augmentation (0/90/180/270°) and report effect (overhead images are often rotation-invariant).
Vision-text interface:
LSTM: compare init-hidden vs <img> token injection.
Transformer: compare 1-token memory vs 4-token memory (projected).
Regularization: dropout placement study (embeddings vs decoder block).
Decoding detail: gentle length penalty during greedy.

5) Evaluation, Analysis & Explainability [2]
5.1 Metrics
BLEU-4 and METEOR (via nltk or torchmetrics).
Caption length stats (mean±std), and % of degenerate repetitions (≥3 identical tokens in a row).

5.2 Qualitative & Slice Analysis
10 success and 10 failure examples (images + GT captions + your caption).
At least 3 error slices (e.g., very bright vs very dark images; short (≤8) vs long (≥16) GT captions; scenes with “runway/harbor/farmland”). Plot per-slice BLEU-4 deltas.

5.3 Explainability
Grad-CAM (or Grad-CAM++) on the last conv block w.r.t. the EOS logit (or a salient content word’s logit). Show 6 overlays (3 good, 3 bad).
Text importance: LSTM: token occlusion (mask one token; observe Δlogit/Δloss), Transformer: average last-layer attention maps (note that attention≠explanation—state limitation).
Mis-caption case study (3 examples): hypothesize cause (domain shift, rotation, tiny objects, vocabulary gap) and propose a non-advanced fix you didn’t implement.