Look, Ask, Explain: The VQA Challenge

End-to-End Image Captioning

December 31, 2025

Learning Objective

1. Build a working captioner for overhead imagery: CNN + Transformer decoder.
2. Decompose the problem into clean modules (data curation, tokenisation, feature extraction, decoding, evaluation).
3. Deliver insight beyond BLEU: qualitative failure slices, Grad-CAM on vision features attention/saliency on text, and a miscaption case study.
4. Report within a single ipynb(Google Colab link with pubic view enabled preferred) with copious use of text cells to walk us through your plan, observations, learnings, and new ideas, with ablations, seeds, and limitations.

Problem & Dataset

Task

Generate a single sentence describing an image (e.g. structures, scene layout).
You can **use any one of the datasets**:

1. Use **RSICD (Satellite Imagery) data from Kaggle**. Contains approximately 10.9k images (RGB, various sizes), 5 captions/image. Splits into: train 8,000 / val 1,500 / test 1,421.
2. **Flickr8k dataset**: contains **8,092 images** collected from Flickr, mainly showing people, animals, and everyday activities. Each image is paired with **five descriptive captions**, making it suitable for introductory multimodal learning tasks.
    - Image type: RGB, natural images
    - Image sizes: Variable
    - Captions per image: 5
    - Primary task: Image captioning

Preprocessing

- **Resize to 224×224**; start with ImageNet normalization and justify any change.
- Tokenize captions with a word-level vocabulary ($\tilde{1}$0k; built on train only). Add special tokens `<bos>`, `<eos>`, and `<pad>`. Limit the maximum caption length (e.g., 22–24 tokens for the RSICD dataset).
- Save a train/val text stats table: vocab coverage, OOV(Out of Vocabulary) %, length histogram.

Baseline to implement

CNN Encoder

- Experiment with ResNet-18 and MobileNet (ImageNet weights).
- Remove classifier; use global average pooling.
- Use 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 the last block; keep batch size small.

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)
- Train with teacher forcing + cross-entropy (ignore PAD with ignore_index).
- Optimisers: 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 decreasing LR after a few epochs.

Experiments & Extensions (pick $\geq$ 1 meaningful)

- Backbone swap: ResNet18 $\leftrightarrow$ MobileNet; report memory/speed vs BLEU/METEOR.
- Input handling: try rotation-aware augmentation (0/90/180/270 deg) and report effect.
- Regularisation: dropout placement study (embeddings vs decoder block).

Evaluation, Analysis & Explainability

1. Metrics

- BLEU-4 and METEOR (via nltk or torchmetrics)
- Caption length stats (mean $\pm$ std), and % of degenerate repetitions ($\geq$ 3 identical tokens in a row).

2. Qualitative & Slice Analysis

- 10 success and 10 failure examples (images + GT captions + your caption).

3. Explainibility

- 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 in Transformer: average last layer attention maps (note that attention $\neq$ explanation—state limitation).
- Mis-caption case study (3 examples): hypothesise the cause (domain shift, rotation, tiny objects, vocabulary gap).

Happy Learning :)