Image Caption Generator using CNN-LSTM and BLIP-2

Ali Zain
k214653@nu.edu.pk

Ali Shaz
k213260@nu.edu.pk

Bilal Ahmed
k214864@nu.edu.pk

April 16, 2025

Abstract

Image captioning, the task of automatically generating textual descriptions for images, lies at the intersection of computer vision and natural language processing. This project implements and compares two distinct approaches for image captioning on the Flickr8k dataset. The first approach utilizes a custom-built Convolutional Neural Network (CNN) based on MobileNetV2 features combined with a Long Short-Term Memory (LSTM) network decoder. The second approach leverages the pre-trained, large-scale Vision-Language model, BLIP-2, accessed via the Hugging Face Transformers library. We detail the data preprocessing steps, the architecture of the custom model, the training procedure including early stopping, and the methodology for using the BLIP-2 model. Qualitative comparisons between the captions generated by both models are presented, highlighting their respective strengths and weaknesses. The project demonstrates the effectiveness of both traditional encoder-decoder architectures and modern pre-trained models for image captioning.

Keywords: Image Captioning, Deep Learning, Computer Vision, Natural Language Processing, CNN, LSTM, MobileNetV2, BLIP-2, Flickr8k, PyTorch, Transformers.

1 Introduction

Generating descriptive captions for images is a fundamental challenge that bridges visual understanding and language generation. It has numerous applications, including aiding visually impaired individuals, enhancing image retrieval systems, and enabling human-robot interaction. This project aims to explore this task by implementing two different deep learning models.

The first model follows the popular encoder-decoder framework. We use a pre-trained Convolutional Neural Network (CNN), specifically MobileNetV2, as the image encoder to extract salient visual features. These features are then fed into a Long Short-Term Memory (LSTM) network, which acts as the decoder to generate the caption word by word.

The second model leverages the power of large pre-trained vision-language models. We utilize BLIP-2 (Bootstrapping Language-Image Pre-training 2), a state-of-the-art model known for its strong performance on various vision-language tasks, including image captioning. We access BLIP-2 through the Hugging Face Transformers library for zero-shot caption generation.

This report details the dataset used, the preprocessing steps performed, the architecture and training specifics of our custom CNN-LSTM model, the usage of the BLIP-2 model, and presents a qualitative comparison of the results obtained from both approaches on the Flickr8k dataset.

2 Methodology

2.1 Dataset

We utilized the Flickr8k dataset [1], a standard benchmark for image captioning.

- **Source:** Collected from Flickr.com.

- **Size:** Contains 8,000 images.
- **Annotations:** Each image is associated with five different human-written captions.
- **Splits:** We split the dataset into training (80%), validation (10%), and test (10%) sets based on image IDs to train our custom model and perform qualitative comparisons.

2.2 Data Preprocessing

The image captions required significant preprocessing before being used for training:

- **Loading:** Captions were loaded from the provided 'captions.txt' file, mapping image IDs to lists of captions.
- **Cleaning:** Captions were converted to lowercase, punctuation was removed, and words containing non-alphabetic characters or single letters were filtered out.
- **Vocabulary Creation:** A vocabulary was built from the cleaned training captions. Special tokens ('<PAD>', '<SOS>', '<EOS>', '<UNK>') were added. Words occurring less than a frequency threshold (e.g., 5 times) were mapped to the '<UNK>' token.
- **Tokenization:** Captions were converted into sequences of integer indices based on the vocabulary. '<SOS>' (start-of-sequence) and '<EOS>' (end-of-sequence) tokens were added to each numericalized caption.
- **Padding:** Caption sequences within each batch were padded to the maximum sequence length in that batch using the '<PAD>' token index for efficient processing.

Image preprocessing for the custom model involved resizing, center cropping, conversion to tensors, and normalization using ImageNet statistics, matching the requirements of the MobileNetV2 backbone. For BLIP-2, images were loaded as PIL objects, and preprocessing was handled internally by the BLIP-2 processor.

2.3 Model Architectures

2.3.1 Custom CNN-LSTM Model

This model follows an encoder-decoder structure:

- **Encoder (CNN):** We used a pre-trained MobileNetV2 model [2] provided by 'torchvision'. The final classification layer was removed. The output features from the convolutional layers were passed through an adaptive average pooling layer and a linear layer with Batch Normalization to project them into the embedding space (dimension: 256). The weights of the pre-trained MobileNetV2 base were frozen during training.
- **Decoder (LSTM):**
  - An embedding layer converts input word indices (from the caption) into dense vectors (dimension: 256).
  - An LSTM layer [3] (hidden size: 512, layers: 1) processes the sequence of word embeddings, conditioned on the image features. The image feature vector (from the encoder) is concatenated as the first element of the input sequence to the LSTM.
  - A final linear layer maps the LSTM hidden state output to the vocabulary size, producing logits for the next word prediction.

- **Training Detail:** The model was trained using sequence packing ('pack$_p$$added_s$$equence$')$tohandlevariable-lengthcaptionsefficientlyandignorepaddingduringLSTMcomputationandlosscalculation.$

2.3.2 BLIP-2 Model

We used a pre-trained BLIP-2 model [4] accessed via the Hugging Face ‘transformers‘ library [5].

- **Architecture Overview:** BLIP-2 typically consists of a frozen image encoder (e.g., ViT), a lightweight Querying Transformer (Q-Former), and a frozen large language model (LLM, e.g., OPT or Flan-T5). The Q-Former bridges the vision and language components.
- **Usage:** We used the ‘Blip2Processor‘ for preprocessing images and decoding output text, and ‘Blip2ForConditionalGene for inference. The specific checkpoint used was ‘Salesforce/blip2-opt-2.7b‘.
- **Inference:** Captions were generated in a zero-shot manner by providing the image to the processor and using the model’s ‘generate()‘ method.

2.4 Training (Custom Model)

- **Loss Function:** Cross-Entropy Loss (‘nn.CrossEntropyLoss‘), configured to ignore the padding token index.
- **Optimizer:** Adam optimizer [6] with a learning rate of 0.001.
- **Batch Size:** 64.
- **Epochs:** Trained for a maximum of 20 epochs.
- **Early Stopping:** Training was stopped early if the validation loss did not improve for 5 consecutive epochs (patience=5). The model weights corresponding to the best validation loss were saved.

2.5 Implementation Details

- **Framework:** PyTorch [7].
- **Libraries:** ‘transformers‘ (for BLIP-2), ‘torchvision‘ (for MobileNetV2), ‘Pillow‘ (for image loading), ‘numpy‘, ‘pandas‘, ‘matplotlib‘ (for plotting), ‘tqdm‘ (for progress bars).
- **Environment:** [Specify CPU/GPU, e.g., Google Colab GPU T4, or local machine specs].

3 Results and Discussion

3.1 Training Performance (Custom Model)

The custom CNN-LSTM model was trained on the Flickr8k training set and validated on the validation set after each epoch. Training loss consistently decreased, indicating learning. Validation loss also decreased initially and then plateaued, eventually triggering the early stopping mechanism after [Specify Epoch number if it triggered] epochs. The best model was saved based on the lowest validation loss achieved.

3.2 Qualitative Results and Comparison

We performed qualitative analysis by generating captions for sample images from the test set using both the best saved custom model and the pre-trained BLIP-2 model.
Analysis for Figure 1:

- **Reference Captions:**

nsert Reference Caption 1
nsert Reference Caption 2
...

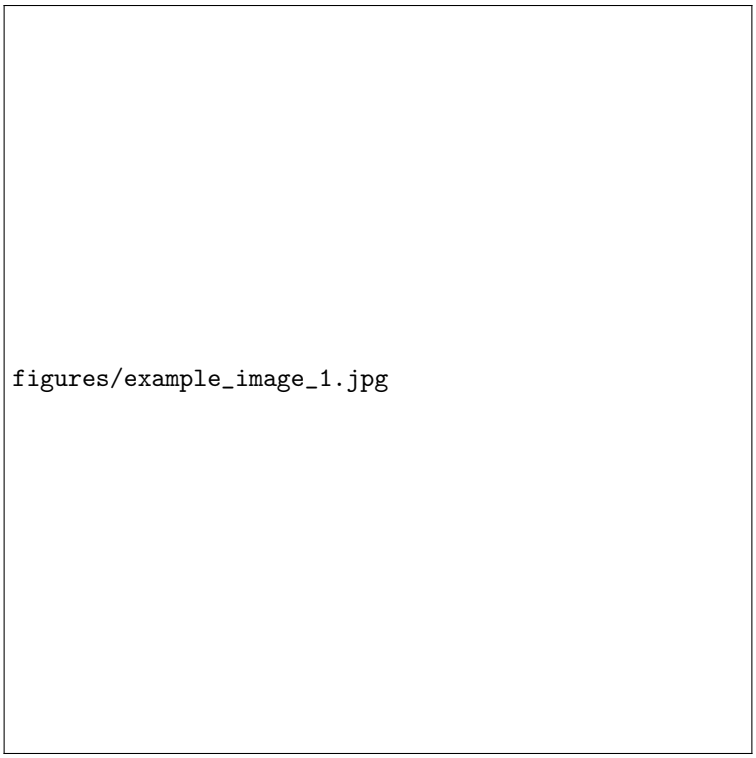

Figure 1: Example Image 1 from the Test Set.

- **Custom Model Caption:** [Insert Caption Generated by Custom Model]
- **BLIP-2 Caption:** [Insert Caption Generated by BLIP-2 Model]
- **Discussion:** [Compare the generated captions. Is one more detailed/accurate? Does one make mistakes? Does BLIP-2 use more complex language? Does the custom model capture the main elements?]

3.3 Discussion

Overall, the pre-trained BLIP-2 model generally produced more fluent, detailed, and contextually accurate captions compared to our custom CNN-LSTM model trained from scratch (with a frozen CNN base) on the relatively small Flickr8k dataset. This is expected, given BLIP-2's extensive pre-training on massive image-text datasets.

The custom model demonstrated its ability to learn the basic mapping between images and captions, often identifying the main subjects and actions. However, its captions were sometimes generic, grammatically simpler, or occasionally missed finer details compared to BLIP-2. [Add specific observations based on your examples].

Training the custom model required careful hyperparameter tuning and preprocessing, whereas BLIP-2 provided strong performance out-of-the-box (zero-shot). However, BLIP-2 also requires significantly more computational resources (memory, potentially inference time) compared to the lighter MobileNetV2-LSTM model.

4 Conclusion

This project successfully implemented two approaches for image captioning on the Flickr8k dataset. A custom CNN-LSTM model, using MobileNetV2 features and an LSTM decoder, was trained and demonstrated the ability to generate relevant, albeit simpler, captions. Early stopping based on validation loss was implemented

to prevent overfitting. A pre-trained BLIP-2 model was also employed, showcasing state-of-the-art zero-shot captioning capabilities, producing more detailed and fluent descriptions. The qualitative comparison highlighted the trade-offs between training custom models on smaller datasets versus leveraging large pre-trained models regarding performance, resource requirements, and development effort.

5 Future Work

Several avenues exist for future work:

- **Attention Mechanism:** Incorporate an attention mechanism into the custom decoder to allow the model to focus on specific image regions while generating words.
- **Beam Search:** Implement beam search decoding instead of greedy search during inference for the custom model, potentially improving caption quality.
- **Different CNN Backbones:** Experiment with other pre-trained CNNs (e.g., ResNet, EfficientNet) as the encoder for the custom model.
- **Fine-tuning:** Fine-tune the encoder CNN or even parts of the BLIP-2 model (if resources allow) on the Flickr8k dataset.
- **Larger Datasets:** Train the custom model on larger datasets like MS COCO or Flickr30k for potentially better performance.
- **Quantitative Evaluation:** Re-introduce quantitative metrics like BLEU, ROUGE, METEOR, or CIDEr scores for a more rigorous comparison (requires careful setup).

References

[1] Hodosh, M., Young, P. and Hockenmaier, J. (2013) *Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics*. Journal of Artificial Intelligence Research, 47, 853-899.

[2] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

[3] Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural computation, 9(8), 1735-1780.

[4] Li, J., Li, D., Savarese, S., & Hoi, S. (2023). *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. arXiv preprint arXiv:2301.12597.

[5] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Brew, J. (2020). *Transformers: State-of-the-Art Natural Language Processing*. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (pp. 38-45).

[6] Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980.

[7] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). *PyTorch: An imperative style, high-performance deep learning library*. Advances in neural information processing systems, 32.

Example code def $example_function() : pass$