Image captioning

Amirmahdi Aramideh
October 2024

1 problem statement

Automatically generating descriptive captions for images is a challenging task with applications in accessibility, image retrieval, and content generation. This project aims to build a deep learning model using the Flicker8k dataset to generate accurate captions for images by combining convolutional neural networks (CNNs) for image features and recurrent neural networks (RNNs) for language generation.

1.1 challenges

1. Handling Diversity: Dealing with the wide variety of image types, scenes, and objects present in the Flickr 8k dataset.

2. Model Complexity: Balancing model complexity and performance to achieve accurate results while maintaining computational efficiency.

3. Limited Dataset Size: Working with the constraints of the Flickr 8k dataset, which, while substantial, may limit the model's ability to generalize to a wider range of images.

4. Evaluation Metrics: Choosing appropriate metrics to evaluate the quality of generated captions, as subjective human judgment often plays a role in assessing caption quality.

5. Cross-Modal Learning: Effectively combining and aligning information from both visual and textual modalities in the deep learning model.

2 Review of available methods

1. **Convolutional Neural Networks (CNN) + Recurrent Neural Networks (RNN)**: This method combines CNNs, known for their ability to effectively extract visual features from images, with RNNs like LSTM networks, which excel in generating sequences. In this setup, the CNN processes the image to capture key visual elements, and the RNN uses these features to sequentially generate a caption, one word at a time. This approach leverages the strengths of both models for image analysis and language generation.

2. **Attention Mechanism**: The attention mechanism is integrated into the CNN-RNN architecture to improve the captioning process. By allowing the model to focus on different regions of the image when generating each word, the attention mechanism helps the system create more accurate and contextually relevant descriptions. This makes it easier for the model to emphasize important parts of the image at each step in the caption generation.

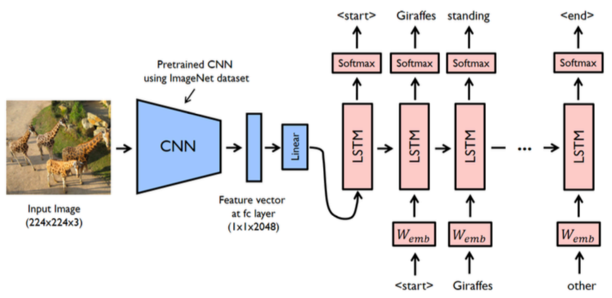
3. **Transformer Networks**: Transformer-based models, originally developed for natural language processing, have been successfully applied to image captioning. These models use self-attention mechanisms to capture relationships between different regions of an image and the words in the generated caption. Compared to RNNs, Transformers can process entire sequences in parallel, which allows for faster and more efficient training while producing more coherent and context-aware captions.

4. **Reinforcement Learning**: Some approaches integrate reinforcement learning to optimize caption generation. This method uses a feedback loop where the model is rewarded for generating more accurate and meaningful captions. Over time, the model adjusts its output to maximize rewards, leading to captions that are not only precise but also more fluent and human-like.

5. **Multimodal Approaches**: In this method, visual data from images is combined with additional textual information, such as pre-trained word embeddings or other linguistic features. This fusion of visual and language models enables a richer

understanding of the image content, resulting in higher-quality captions that capture more detail and nuance.

6. **2.6 Transfer Learning**: Leveraging pre-trained models from large datasets like ImageNet can significantly improve image captioning performance. By transferring knowledge from these models, which have learned to recognize a wide variety of features, the captioning model can more effectively analyze images and generate descriptive captions, even with limited data.



3. Model Architecture

In this project, the image captioning model consists of two main components: the **Encoder** (a CNN-based image feature extractor) and the **Decoder** (an RNN-based sequence generator). These components work together to generate meaningful captions for a given image.

1. **Encoder (EncoderCNN)**:

    1. The encoder uses a pre-trained **ResNet-50** model to extract features from input images. ResNet-50 is a deep convolutional neural network trained on the ImageNet dataset, which is capable of identifying a wide variety of visual features.
    2. The classification layer of ResNet-50 is removed, and the model is set to non-trainable mode to prevent further updates to the pre-trained weights.
    3. The output features from ResNet are then passed through a **linear layer** to transform them into a lower-dimensional embedding space. A **batch normalization** layer is also applied to stabilize the learning process.

2. **Decoder (DecoderRNN)**:

    1. The decoder is a recurrent neural network (RNN) that takes the image features extracted by the encoder and generates captions sequentially.
    2. It starts with an **embedding layer** that converts word indices into dense vector representations.
    3. A **Long Short-Term Memory (LSTM)** network is used to model the sequential nature of language, generating one word at a time based on both the image features and the previous word.
    4. The decoder includes dropout layers for regularization to prevent overfitting, and a fully connected **linear layer** to predict the next word in the sequence.
    5. During training, the decoder processes input captions and learns to predict the following words in the sequence. During inference (caption generation), it can predict words one by one until the caption is complete.

3. **ImageCaptioning (Combined Model)**:

    1. The **ImageCaptioning** class integrates both the encoder and decoder into one unified model.
    2. The encoder extracts the image features, and the decoder generates the corresponding captions.
    3. The model can operate in two modes: training mode, where it learns to predict captions based on input data, and generation mode, where it produces captions based on unseen images.

4 Dataset

The **Flickr8K dataset** is a widely recognized benchmark used for image captioning tasks. It was curated from images on the Flickr platform and paired with captions that were manually written by human annotators to describe the content.

Key aspects of the Flickr8K dataset include:

1. **Dataset Size and Composition**: The dataset contains 8,000 images, which is where it gets its name, "Flickr8K." Each image is accompanied by five unique captions, resulting in a total of 40,000 captions overall. The images cover a broad spectrum of topics, ranging from various objects and scenes to diverse activities, making it well-suited for training and testing image captioning models.

2. **Caption Accuracy**: The captions are written by human annotators, ensuring that the descriptions are accurate and contextually relevant to the images. These high-quality annotations help models better understand the visual content and improve their captioning performance.

3. **Language Style**: The captions in Flickr8K are generally formal and descriptive, closely resembling the style typically found in curated photo collections or databases. This formal tone can be beneficial for applications where structured, detailed descriptions are required.

4. **Dataset Splits**: Flickr8K is split into training, validation, and test sets. Specifically, there are 6,000 images in the training set, while the validation and test sets each consist of 1,000 images. This standardized division allows for effective training, validation, and performance benchmarking of captioning models.

5 Model Training Using CNN + LSTM

In this stage of the project, I started by organizing the dataset based on captions, leading to a total of 30,000 samples. The training set includes 6,000 images, each paired with five captions, resulting in 30,000 training samples. Similarly, the validation set contains 1,000 images, producing 5,000 samples, while the test set comprises roughly 5,000 samples (since some images in the test set have more than five captions). In total, the dataset has 40,460 captions, matching the total number of samples.

I trained the model in two distinct modes, taking into account both the total number of tokens in the dataset and the number of tokens specifically in the training data. For the training set, the token count stands at 7,698 (excluding special tokens), while the total token count across the dataset is 8,911. In the first phase, where I applied a basic convolutional neural network (CNN) alongside a recurrent neural network (RNN), I only used tokens from the training set.

For the CNN, I leveraged a pre-trained **ResNet50** model, keeping it frozen during training—meaning the network's parameters were not updated. However, the final linear layer of ResNet50 was modified and integrated into the training process. The output from the ResNet50 model (which contains image features) was merged with the embedding layer's output, ensuring that the dimensions matched the embedding size.

This combined output (image features acting as a token) was added to the beginning of the caption tokens. During this process, the last token of the caption is excluded, following a similar approach to language modeling where the model receives the image and caption and generates a sequence of the same size as the caption, without the final token.

Additionally, to handle varying caption lengths within a batch, I padded the captions to match the size of the longest sequence, ensuring that all captions could be properly concatenated with the image feature vector. After this concatenation, the combined sequence passed through several LSTM layers to generate the caption, followed by a final linear layer for classification to adjust the output size.

The optimal hyperparameters for the model were:

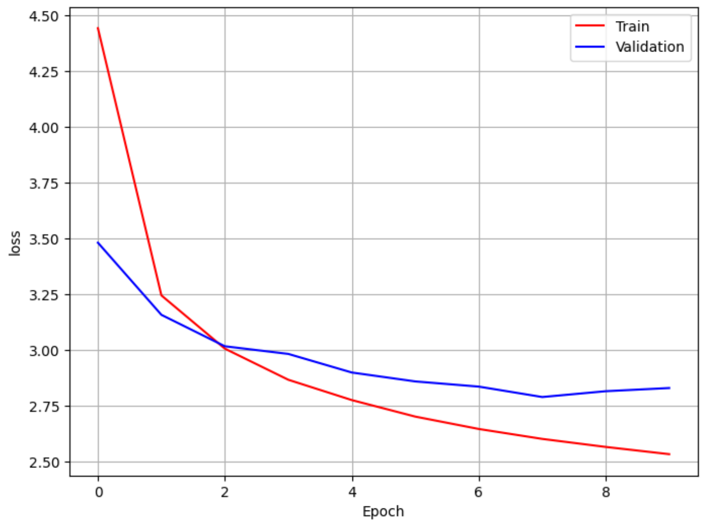
- **Embedding dimension**: 300
- **Hidden dimension**: 500
- **Number of layers**: 2
- **Embedding dropout**: 0.5
- **LSTM dropout**: 0.5
- **Batch size**: 128
- **Learning rate**: 1.25
- **Weight decay**: 1e-4
- **Optimizer**: SGD

6 Results

Training Process:

The graph provided illustrates the loss curves for both the training and validation sets across multiple epochs. Initially, both losses start high, with the training loss (in red) and validation loss (in blue) decreasing as training progresses. The training loss steadily declines throughout, indicating that the model is learning effectively from the dataset. However, the validation loss levels off around the 5th epoch and begins to slightly increase toward the end, suggesting potential overfitting.

This suggests that while the model continues to improve on the training data, it may not generalize as well to unseen data beyond a certain point in training.



Evaluation on Test Set:

The model was evaluated using BLEU scores, which measure how well the generated captions match human-generated captions in terms of overlapping n-grams. Here are the results of the BLEU scores calculated for the test set:

- **BLEU-1**: 0.6108
- **BLEU-2**: 0.3907
- **BLEU-3**: 0.2519
- **BLEU-4**: 0.1636

These scores indicate that the model performs relatively well on the BLEU-1 metric, capturing unigrams (single words) effectively, but the performance decreases as we move to higher-order n-grams (BLEU-2, BLEU-3, BLEU-4), which involve larger word sequences. This is a common result in image captioning, where models can capture individual words well but may struggle to generate longer, coherent sequences.

Conclusion:

The model exhibits strong performance in generating relevant words (as indicated by BLEU-1), but there is room for improvement in generating more fluent, multi-word phrases (as seen in the drop-off in BLEU-3 and BLEU-4 scores). Further refinements could be made to the model, such as enhancing the LSTM layers or fine-tuning the ResNet, to improve its ability to generate more natural and contextually accurate captions.