Image Captioning Vision Transformer

Donggyu Kim

Introduction

This report outlines the development and assessment of an image captioning model utilizing the Flickr8k dataset. The Flickr8k dataset, comprising 8,000 images each annotated with five different captions, serves as a resource for training and testing image captioning systems.

The primary aim of this study is to investigate the effectiveness of transformer-based models combined with Vision Transformer (ViT) in generating descriptive captions for images. The transformer model, especially noted for its self-attention mechanism, enables parallel processing of sequences and has demonstrated substantial advancements over traditional models like recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) in various tasks.

This report commences by discussing the theoretical background of transformer models and the integration of ViT as the image feature extractor. A detailed description of the dataset and the preprocessing steps required for transforming images and captions for model training follows. We then elaborate on the architecture of our model, which includes a Vision Transformer (ViT) as the encoder and a transformer decoder for generating captions, emphasizing the unique attributes and the thought process behind the selected configurations.

The training procedure, encompassing parameter tuning and optimization strategies, is thoroughly explained. Finally, the model's performance is evaluated using the BLEU score, a metric for comparing machine-generated text to human-written references. We compare these results with those of baseline models to highlight the superior captioning abilities of our approach.

Experimental setup

I used the spyder IED to run the Pytorch and python libraries. The version of Pytorch was 2.1.2 with Python 3.10.8 and the version of Cuda was 11.8 for this programming assignment. To run the simulation of the learning, I used the laptop with CPU i7-8750 and GPU RTX 2070 max_Q(for the laptop).

Hyper Parameters and Model Description

There are several types of hyperparameters in this model.

1. Learning rate: For this model, Learning rate is set to 0.0001

2. Betas: These are the coefficients used for computing running averages of the gradient and its square, which are components of the Adam algorithm. beta1 (0.9 for this model) controls the decay rate of the moving average of past gradients (momentum).beta2 (0.98 for this model) controls the decay rate of the moving average of past squared gradients.

3. Epsilon: A very small number added to the denominator in the Adam update rule to prevent any division by zero. This term helps maintain numerical stability. It is set to `1e-9`

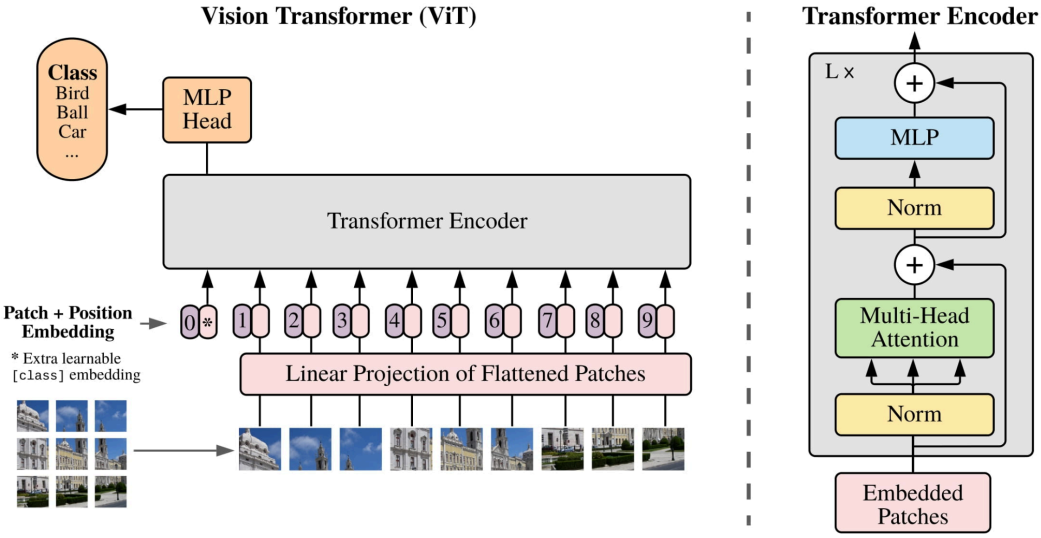
4. Number of epochs: I let the model train for 12 epochs. As the epoch value was adjusted from 20, it was confirmed that overfitting occurred as the difference between training and testing results increased when it exceeded 12.

5. Mini Batch Size:

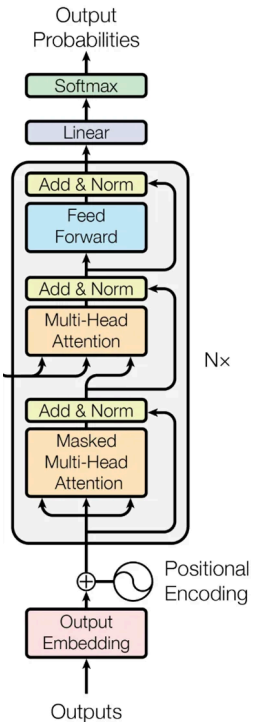
    a. The mini Batch size of this model is 50 which implies there are 120 iterations for each caption. I tried various sizes such as 50,100,200,20 but 50 showed the most accurate result

Model description

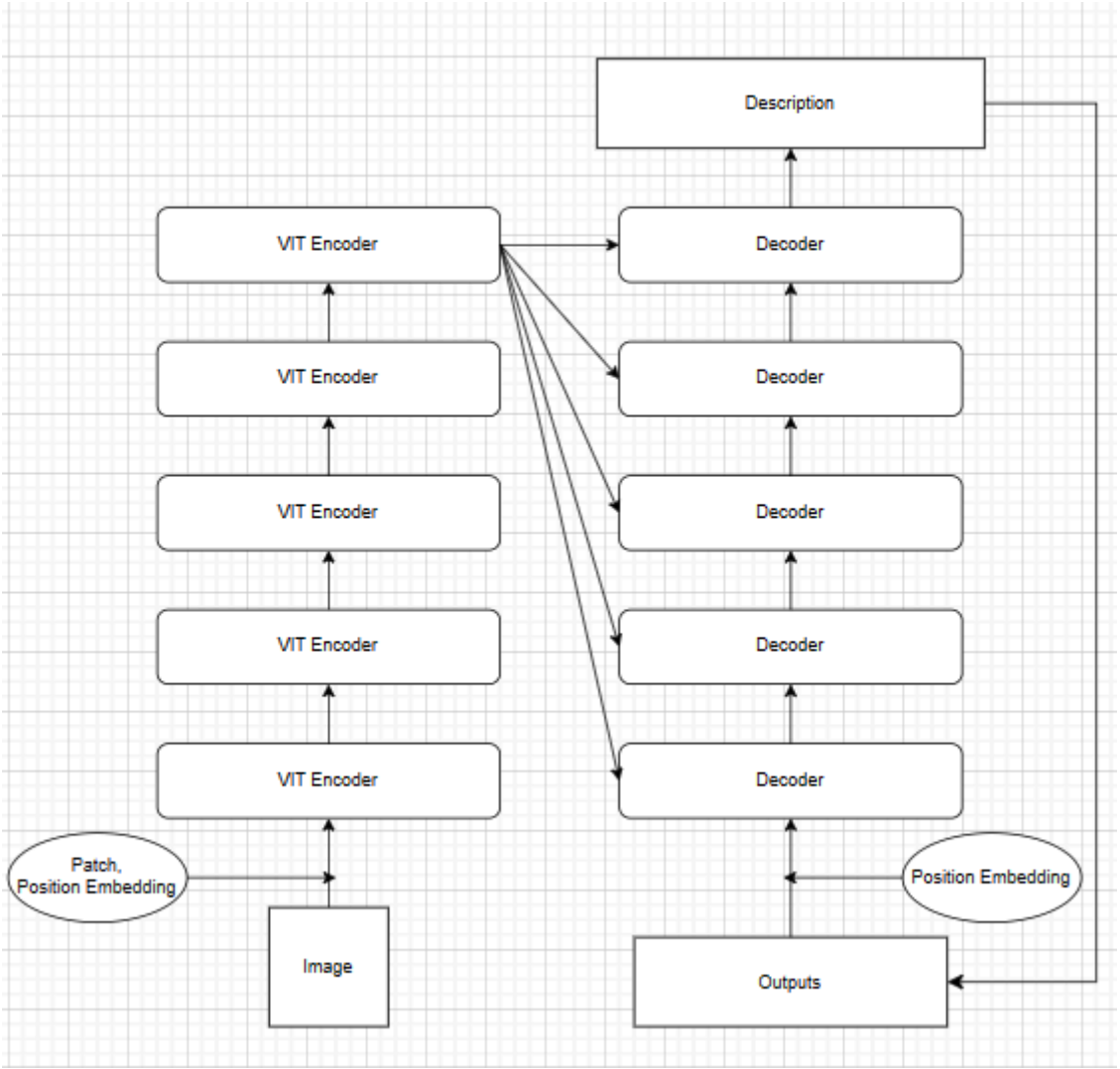
The Transformer model is designed to handle sequential data and is especially popular in tasks like machine translation. It differs from previous models because it dispenses with recurrence and convolutions entirely, relying on attention mechanisms to process data in parallel and capture sequential relationships at different scales**.** The model of this assignment consists of 5 VIT encoder layers and 5 decoder layers with positional encoding and token embedding with attention mechanisms.



Structure of VIT



Structure of Decoder



Specific details of each layer

VIsion Trasnformer(VIT):

- The ViT consists of a stack of 5 identical layers, similar in structure to the layers used in standard transformer encoders but adapted for image processing. Each layer has two sub-layers:

  Architecture: Vision Transformer (ViT) adapts the principles of transformers. It processes images by dividing them into fixed-size patches.

  Image Patching: Each image is split into patches, which are then flattened and linearly embedded. Positional encodings are added to these embeddings to preserve the positional context of each patch.

  Transformer Encoder: The core of ViT consists of a series of identical layers, each containing two sub-layers:

Multi-head Self-Attention Mechanism: This allows the model to focus on different parts of the image simultaneously, enhancing its ability to capture complex features across the spatial domain.

Position-wise Feed-Forward Networks: Each position (patch) is processed independently through two linear transformations with a ReLU activation in between, further refining the features.

Decoder:

- The decoder also consists of 5 identical layers. Each decoder layer has two sub-layers:

Multi-head self-attention mechanism: Similar to the encoder, but it is masked to prevent positions from attending to subsequent positions. This ensures that the predictions for position i can depend only on the known outputs at positions less than i.

Multi-head attention over the encoder's output: This layer helps the decoder focus on relevant parts of the input sentence, similar to how attention mechanisms in seq2seq models work.

Position-wise fully connected feed-forward network: Same as in the encoder.

Positional Encoding: Since the Transformer uses no recurrence or convolution, it uses positional encodings to consider the order of the sequence. Positional encodings are added to the input embeddings at the bottoms of the encoder and decoder stacks. These encodings have the same dimension as the embeddings, so the two can be summed. They use sine and cosine functions of different frequencies to encode the position of a token in the sequence.
.
Token Embedding: Token embeddings transform input tokens into vectors of a specified size (embedding dimension, 768). These embeddings are learned and represent a dense representation of the tokens in a high-dimensional space. In the Transformer model, these embeddings are input to both the encoder and decoder.

Embedding dimension: 768 - This parameter specifies the size of the embeddings used for the input tokens. All input data is converted into vectors of this dimension before being processed by the model.

Feedforward dimension: 768 - Within each encoder and decoder layer, there is a position-wise feedforward neural network. This network transforms the data from the embedding dimension to the feedforward dimension and back to the embedding dimension.

Number. of Attention head: 12- In the multi-head attention mechanism, the model divides the embedding dimension into smaller parts and performs parallel attention computations across these parts. Each "head" focuses on different parts of the input sequence, allowing the model to attend to different aspects of the input simultaneously.

Preprocessing Steps

Image processing:

Resizing: All images are resized to a consistent size (256x256 pixels) to ensure uniformity in input dimensions.

Normalization: All images are normalized to the tensor image with a mean of 0.5 and a standard deviation of 0.5 for each channel.

Data Augmentation

To increase the diversity of the training data and to prevent overfitting, data augmentation techniques are applied:

1. transforms.RandomHorizontalFlip()

This transformation randomly flips the image horizontally with a default probability of 50%. Horizontal flipping is a common augmentation technique for tasks where the horizontal orientation does not change the semantics of the image.

2. transforms.RandomRotation(degrees=(-30, 30))

This transformation rotates the image by a random angle within the specified range, which is between -30 and +30 degrees in this case. Random rotation helps the model to be invariant to orientation, increasing its ability to recognize objects or scenes regardless of their rotation.

4. transforms.RandomAutocontrast(p=0.5)

This transformation automatically adjusts the image contrast so that the distribution of its pixel values expands to cover the full range of intensities allowed in the image data type . The parameter p=0.5 means that there is a 50% chance this adjustment will be applied to any given image.

Caption preprocessing

Vocabulary Building: A vocabulary is created for target captions.

Numerical Encoding: Each token is converted into a numerical ID, using the vocabulary.

Padding: Captions are padded to ensure they have uniform length within each batch.

Training Procedure

During the training of an image captioning model using a Vision Transformer (ViT) as the encoder and a dedicated decoder, both the image and its corresponding caption are provided to the model. The ViT encoder processes the image to create a context-rich representation. This representation serves as the input to the decoder, which also receives the target caption. The caption

acts as a guide, helping the decoder learn the correct sequence of words to accurately describe the image:

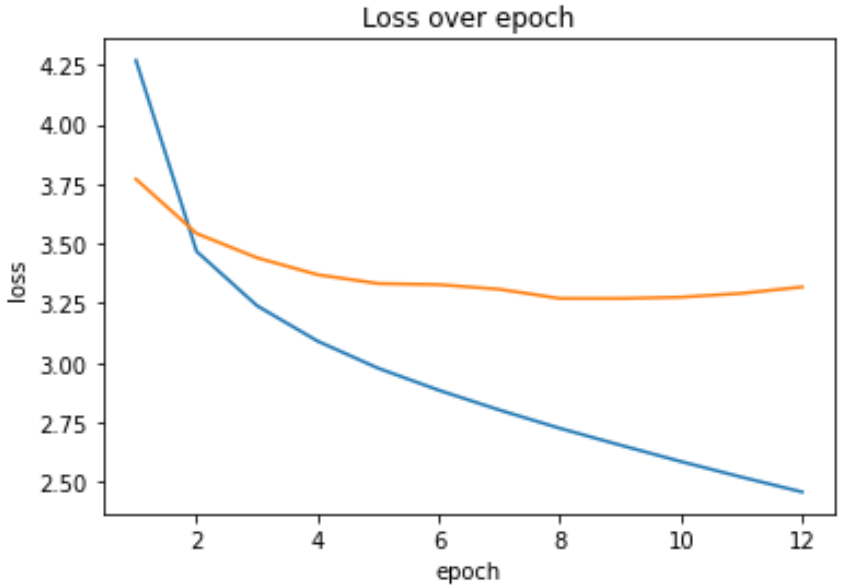
Validation Process

The validation process follows the same structure as the training phase but utilizes a separate dataset—the validation set—that the model has not encountered during training. This stage is essential for evaluating the model's performance on unseen data, ensuring it can generalize well across different images and captions not present in the training set.

Testing Model

During testing, the caption generation begins with only a start token, which is the initial state of the target sequence. At each step, the model predicts the output probabilities for all possible next words in the vocabulary. The decoder selects the most probable word (argmax of the output logits) as the next token in the sequence. This newly predicted word is added to the existing caption sequence, and the process is iterated. This continues until the model predicts an end token or reaches a predefined maximum length for the caption.

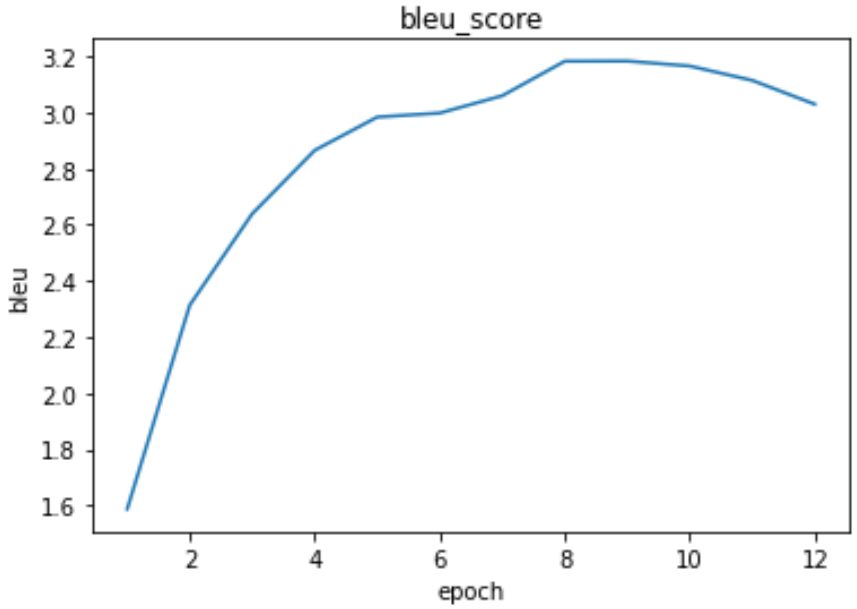
Training Loss and Validation Loss



Training Loss(Blue) and Validation Loss(Orange) versus Epoch

Both the training and validation loss curves exhibit a pronounced downward trend, indicative of the model's performance enhancement as training progresses. Initially, there is a steep reduction in loss across both datasets, reflecting significant learning gains. However, by around epoch 8, the validation loss begins to plateau, signaling a convergence point beyond which minimal improvements are observed. This plateau suggests that the model is close to optimal generalization, as further training does not yield significant reductions in validation loss, potentially leading to overfitting if continued. The training loss, on the other hand, continues to decline, which could indicate that the model is beginning to memorize the training data rather than learning to generalize from it.

.Bleu Score over epoch



BLEU score versus epoch

The BLEU score, represented on the y-axis, shows its relation to the training epochs on the x-axis in this graph. Initially, the BLEU score demonstrates a rapid ascent during the early epochs, indicating significant enhancements in the model's image description quality at the outset of the training. As the training progresses, the rate of improvement in the BLEU score starts to slow down, with the curve peaking and beginning to plateau around epoch 8. This indicates diminishing returns from continued training. The score appears to stabilize post-epoch 8, hovering around the 3.1 mark, suggesting that the model has nearly maximized its learning potential from the available training data. The observed plateau signifies that additional training is unlikely to yield substantial improvements in image description quality, implying that the model is nearing its optimal performance within the constraints of the current training setup and data.

Maximum bleu score was 3.183823.

Analysis of the final result

The experiments involved adjusting the architecture of Vision Transformer (ViT) components, specifically varying the numbers of encoders and decoders, the dimensions of the feedforward network and embeddings, and the number of attention heads.

The initial configuration included a single ViT encoder and decoder with a feedforward and embedding dimension of 518, and 8 attention heads. This base model achieved a BLEU score of 1.5 but showed difficulties in effectively learning from the training data, indicating potential under-specification or insufficient capacity to capture the complexity of the task.

To address this, the architecture was scaled up to 3 ViT encoders and decoders while maintaining the same dimensionality and number of heads, which improved the BLEU score to 2.5. However, this model still exhibited signs of underfitting, suggesting that further expansion might yield better results.

Further modifications involved increasing both the feedforward and embedding dimensions to 768 and raising the number of attention heads to 12. This configuration, with 3 encoders and decoders, pushed the BLEU score to 2.7, indicating a positive trend in performance enhancements with increased model complexity.

The most successful variant deployed 5 ViT encoders and 5 decoders with the larger dimensionality and attention heads, achieving a BLEU score of 3.18. This model not only outperformed its predecessors but also demonstrated the most balanced approach to learning from the dataset without significant overfitting.

Additional experiments with 6 and 12 ViT encoders and decoders were also conducted. The model with 6 encoders and decoders showed early signs of overfitting, triggering a stop in training, while the significantly larger model with 12 encoders and decoders struggled with training efficiency, likely due to its excessive complexity which could lead to difficulties in optimization and excessive computational demand.

These experiments underscore the importance of carefully balancing model complexity with training capability to optimize performance without incurring overfitting or underutilization of the model's capacity. The insights gained point to a model configuration with 5 ViT encoders and decoders as the most effective in this series of tests, providing a robust framework for further refinement and application.

In addition to architectural adjustments, the performance impact of varying input image sizes and patch sizes for the Vision Transformer (ViT) components was also explored. ViT architectures fundamentally transform image patches into a sequence of embeddings, which are then processed through the transformer model. The choice of image size and patch size is crucial as it determines the granularity of features extracted from the image and the sequence length of the transformer.

Image Sizes

Larger image sizes allow the model to capture more detailed features, potentially improving the model's understanding of complex visual inputs. However, larger images also increase the computational burden, requiring more memory and processing power. Smaller image sizes, while computationally less demanding, might overlook essential features, leading to poor model performance. In our experiments, the trade-off between image resolution and computational efficiency was balanced to optimize performance without excessively straining computational resources.

Patch Sizes

The size of the patches that the images are divided into affects the resolution of the input that the ViT processes. Smaller patch sizes result in higher resolution and more patches, which can help the model learn finer details but increases the sequence length that the transformers handle. Conversely, larger patches reduce the computational load by decreasing the sequence length but at the cost of potentially missing finer details in the image.

Feedforward and Embedding Dimensions

The experiments included variations in the feedforward and embedding dimensions, which are crucial for the model's capacity to process and represent information. Increasing these dimensions generally allows the model to capture more complex patterns and relationships in the data, which is reflected in the improved BLEU scores observed in our later experiments. Specifically, moving from dimensions of 518 to 768 provided the model with a larger representational space, enabling more effective learning and better generalization capabilities.

Conclusion

In conclusion, our series of experiments aimed at enhancing an image captioning model using the Flickr8k dataset has yielded valuable insights into the interplay between model architecture, image size, and patch size in Vision Transformer (ViT) applications. The series started with a baseline model featuring minimal complexity, which underperformed, indicating insufficient capacity to handle the complexity of the task.

Progressive scaling of the model's architecture showed a clear trend: increasing the number of encoders and decoders, along with expanding the dimensions of feedforward networks and embeddings, substantially improved performance, as evidenced by rising BLEU scores. The optimal model emerged with 5 ViT encoders and 5 decoders, which not only achieved the highest BLEU score of 3.18 but also maintained a balance between learning efficacy and computational efficiency.

Further explorations into image and patch sizes demonstrated that these parameters are crucial for optimizing input data processing. Smaller patches, although increasing the resolution and detail captured, imposed higher computational demands and longer sequence processing, which sometimes led to underperformance. Conversely, larger patches simplified processing but at the cost of potentially missing finer visual details crucial for accurate image captioning.

The interplay between architectural complexity and input data configuration underscores the necessity of a balanced approach to model design. This balance ensures that the model is sophisticated enough to learn effectively from the data without exceeding computational limits or losing critical information due to overly coarse input resolutions.

These findings emphasize that careful tuning of both architectural and input parameters is essential to developing efficient and effective image captioning systems. This approach not only optimizes performance on the current dataset but also provides a framework for adapting to other datasets and tasks in the field of deep learning and computer vision.