Image Caption Generation using External Knowledge

CHIKKERI CHINMAYA
Dept. of Information Technology
NITK, Surathkal
chikkerichinmaya.211it017@nitk.edu.in

ADISH KERKAR
Dept. of Information Technology
NITK, Surathkal
adishrajendrakerkar.211it004@nitk.edu.in

MANJUNATH HARALE
Dept. of Information Technology
NITK, Surathkal
manjunathharale.211it039@nitk.edu.in

***Abstract*—This work demonstrates an end-to-end pipeline that integrates state-of-the-art object detection using YOLOv5, deep feature extraction with ResNet and BERT, and multi-modal embedding techniques to enhance contextual understanding of visual data. The implementation supports real-time object detection, semantic interpretation of detected objects, and fusion of image and text embeddings for downstream tasks like caption generation and scene understanding.**

***Index Terms*—Multi-modal Learning,Feature Extraction,Deep Learning Applications, Object Detection, Image Captioning**

I. Introduction

Object detection and multi-modal learning are at the forefront of advancements in computer vision and natural language processing (NLP), enabling systems to interpret and integrate visual and textual data effectively. This project aims to harness the power of deep learning models to process images, accurately identify objects, and enhance their interpretation with semantic information from external sources. Using YOLOv5, the system performs real-time object detection, classifying and localizing objects within images efficiently. To provide deeper context, the project incorporates a Wikipedia API to fetch concise summaries of the detected objects, enriching the system's understanding beyond basic labels. Furthermore, by leveraging ResNet-50 for image feature extraction and BERT for textual embedding, the project achieves a seamless fusion of visual and textual data. This multi-modal embedding forms a unified representation that can be utilized for tasks like image captioning and scene analysis, showcasing the integration of these technologies in building intelligent, context-aware systems.

II. Related Work

Many early neural models for image captioning encoded visual information using a single feature vector representing the image as a whole, and hence did not utilize information about objects and their spatial relationships. Karpathy and Fei-Fei in, as a notable exception to this global representation approach, extracted features from multiple image regions based on an R-CNN object detector [7] and generated separate captions for the regions. As a separate caption was generated for each region, however, the spatial relationship between the detected objects was not modeled. This is also true of their follow-on dense captioning work [10], which presented an end-to-end approach for obtaining captions relating to different regions within an image. Fang et al. in [6] generated image descriptions by first detecting words associated with different regions within the image. The spatial association was made by applying a fully convolutional neural network to the image and generating spatial response maps for the target words. Here again, the authors did not explicitly model any relationships between the spatial regions.

A family of attention based approaches to image captioning have also been proposed that seek to ground the words in the predicted caption to regions in the image. As the visual attention is often derived from higher convolutional layers of a CNN, the spatial localization is limited and often not semantically meaningful. Most similar to our work, Anderson et al. in [2] addressed this limitation of typical attention models by combining a "bottom-up" attention model with a "top-down" LSTM. The bottom-up attention acts on mean-pooled convolutional features obtained from the proposed regions of interest of a Faster R-CNN object detector. The top-down LSTM is a two-layer LSTM in which the first layer acts as a visual attention model that attends to the relevant detections for the current token and the second layers is a language LSTM that generates the next token. The authors demonstrated state-of-the-art performance for both visual question answering and image captioning using this approach, indicating the benefits of combining features derived from object detection with visual attention. Again, spatial information—which we propose in this work via geometric attention—was not utilized. Geometric attention was first introduced by Hu et al. for object detection in [9]. There, the authors used bounding box coordinates and sizes to infer the importance of the relationship of pairs of objects, the assumption being that if two bounding boxes are closer and more similar in size to each other, then their relationship is stronger.

In recent developments, namely the Transformer architecture have led to significant performance improvements for various tasks such as translation, text generation [4], and language understanding . In , the Transformer was applied to the task of image captioning. The authors explored extracting a single global image feature from the image as well as uniformly sampling features by dividing the image into 8x8 partitions. In the latter case, the feature vectors were fed

Identify applicable funding agency here. If none, delete this.

in a sequence to the Transformer encoder. In this paper we propose to improve upon this uniform sampling by adopting the bottom-up approach of [2]. The Transformer architecture is particularly well suited as a bottom-up visual encoder for captioning since it does not have a notion of order for its inputs, unlike an RNN. It can, however, successfully model sequential data with the use of positional encoding, which we apply to the decoded tokens in the caption text. Rather than encode an order to objects, our Object Relation Transformer seeks to encode how two objects are spatially related to each other and weight them accordingly

III. Methodology

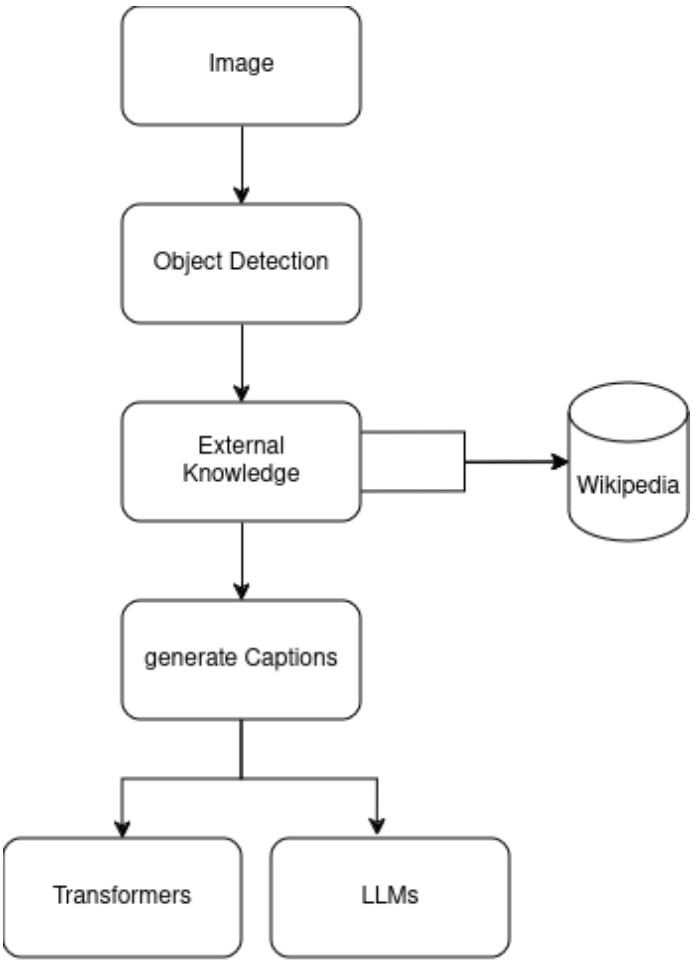

Fig. 1. Methodology Diagram

A. Feature Extraction Using ResNet-50 and BERT

To capture the visual content of images effectively, the ResNet-50 model, a pre-trained convolutional neural network, was employed for feature extraction. ResNet-50, trained on the ImageNet dataset, is renowned for its ability to learn rich and robust visual representations. The model processes images and generates high-dimensional feature vectors that encapsulate intricate details and patterns present in the input.

Before feeding the images into the model, a series of preprocessing steps were performed to ensure compatibility. The images were resized to meet the model's input dimensions, normalized to align pixel intensity values with the model's training conditions, and converted into tensor format for processing in PyTorch. These preprocessed images were then passed through ResNet-50, and the output was a feature vector representing the image's visual characteristics.

$$a + b = \gamma \tag{1}$$

B. Positional Encoding

Earlier, the models used to produce text were based on sequential models. Sequential models input the text in an orderly manner which allows the model to understand the ordering of words in the text. As in RNN based models, at any time step, the input includes previous hidden state (ht-1) and current word (xt), this allows the model to process words in sequential manner. But, in transformers, we do not have any sequential input of the words. So, if we directly input the words, it would not contain any information regarding its order. This would be same as considering it as a bag of words. And for generating captions, our model has to learn the structural and contextual meaning of the sentence. So, we use positional encoding for this purpose. Positional encoding provides an encoded vector for a sentence where each word is mapped to a embeddim dimensional vector in such a way that it contains the positional information of that word. Then, this positional encoding vector (shape : m, embeddim)(m is the number of tokens in the sentence) is added to embedding vector generated by Embedding layer (shape: m, embeddim). This vector is then passed to the transformer model to further operations.

C. Multi-Head Attention Mechanism

Multi-head attention is an extension of the standard attention mechanism that allows the model to focus on different parts of the input sequence simultaneously in parallel. Instead of learning a single set of attention weights, multi-head attention learns multiple sets (or "heads") of attention weights. Each head captures different relationships between the input tokens, enabling the model to focus on various aspects of the sequence

1) Capturing Diverse Relationships: Each attention head learns to focus on different parts of the sequence or different types of relationships between tokens. For example, one head may learn syntactic relationships (e.g., subject-verb-object), while another may learn semantic relationships (e.g., synonyms or thematic connections). This enables the model to capture more complex dependencies in the data.
2) Better Representation: By combining multiple attention heads, multi-head attention provides a richer representation of the input, improving the model's ability to understand and generate diverse patterns in the data.
3) Parallelization: Since the attention heads operate in parallel, multi-head attention can be computationally efficient and scale well with larger models.

4) Improved Generalization: By attending to different aspects of the input, multi-head attention helps the model generalize better to unseen data, making it more robust and effective in tasks like machine translation, image captioning, and more.

D. Object Detection using YOLOv5

YOLO (You Only Look Once) is a cutting-edge real-time object detection framework known for its speed and accuracy. Unlike traditional methods that process images in multiple stages, YOLO processes the entire image in a single evaluation. It predicts bounding box coordinates and class probabilities simultaneously, making it highly efficient for tasks requiring quick and precise object detection.

In this project, the YOLOv5 variant was employed due to its state-of-the-art performance and ease of integration. Using the PyTorch torch.hub library, the model was loaded seamlessly, leveraging the pre-trained yolov5s weights for object detection. The pre-trained model, trained on a large dataset, ensures high accuracy even for generic detection tasks. Given an input image, YOLOv5 outputs the coordinates of bounding boxes that enclose detected objects, along with their corresponding class labels (e.g., ”car,” ”person”) and confidence scores indicating the model's certainty in its predictions.

E. Semantic Enrichment Using External APIs

- To enhance the understanding of detected objects, this project integrates a semantic enrichment step by fetching contextual information using external APIs, specifically the Wikipedia API. The primary objective of this process is to provide meaningful summaries for the objects identified during detection, moving beyond simple labels to a more nuanced comprehension of their significance.
- A custom function was developed to query Wikipedia based on the names of detected objects. Upon receiving a query, the API retrieves a concise and relevant summary of the object, which is then integrated into the pipeline. In cases where no information is available, a placeholder message is returned, ensuring seamless handling of all inputs. This additional layer of semantic information enriches the output, making it particularly useful for context-aware applications such as educational tools, where detailed explanations are essential, or accessibility systems designed to aid visually impaired users.
- By coupling object detection with semantic interpretation, this step significantly enhances the system's capability to provide not just identification but also insightful descriptions of the scene, thereby making it more versatile and impactful.

This extracted feature vector serves as a compact and informative representation of the image, capturing both low-level details (such as edges and textures) and high-level semantics (such as object shapes and structures). These features are foundational for downstream tasks like multi-modal learning, image captioning, or scene understanding, as they provide a meaningful numerical summary of the visual data.

F. Caption Generation

- Caption generation is a crucial step aimed at producing natural language descriptions that effectively convey the details of detected objects and their surrounding scene. This task bridges the gap between visual perception and linguistic expression, enabling a richer understanding and communication of visual data.
- For this purpose, the project utilizes the T5 model, a transformer-based text-to-text generation framework. Renowned for its versatility in handling a variety of text generation tasks, the T5 model was fine-tuned on data derived from the fused embeddings of images and text. These embeddings, combining visual features extracted using ResNet-50 and textual features from BERT, encapsulate both the visual content and semantic context of the scene.
- By leveraging this multi-modal input, the T5 model generates captions that are not only descriptive but also contextually relevant. This ensures that the captions provide a comprehensive summary of the scene, including details about the objects, their relationships, and any inferred context. Such captions have wide-ranging applications, including accessibility tools for the visually impaired, content generation, and automated scene analysis in surveillance or multimedia indexing systems.

IV. Model Architecture

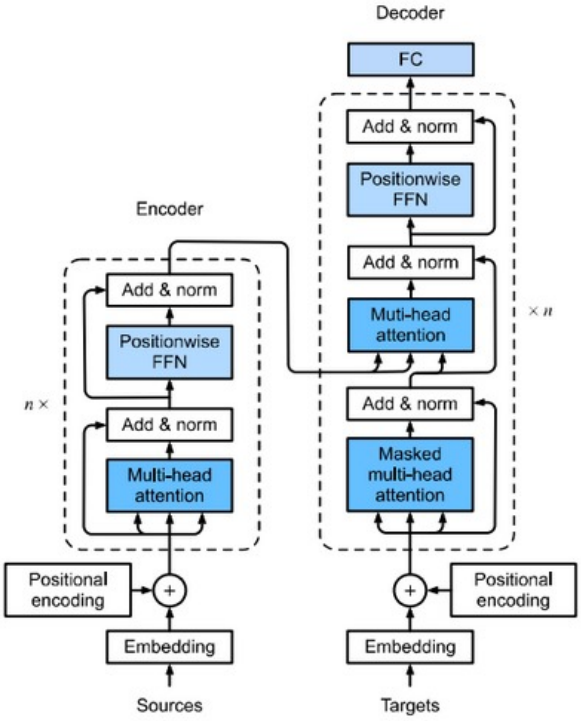

Fig. 2. Model Architecture

A. Decoder Layer

The DecoderLayer in your code is part of a Transformer architecture, specifically for the decoder part. This layer contains multiple sub-layers designed for self-attention, encoder-decoder attention, and feed-forward processing, which are all fundamental components of the Transformer model. Let's break down the layer and its components:

1) Multi-Head Attention (MHA)
   self.mha1 and self.mha2 are instances of the Multi-Head Attention (MHA) mechanism. MHA allows the model to focus on different parts of the input sequence simultaneously. There are two MHA layers here: self.mha1: This is the self-attention layer that attends to the target sequence (x). The input to the query, key, and value are all the same (i.e., x). This layer helps the decoder focus on different parts of its own input sequence while making predictions. self.mha2: This is the encoder-decoder attention layer. The key and value inputs are from the encoder's output (encoutput), and the query input is from the output of the first attention (out1). This allows the decoder to attend to the encoder's output while generating predictions.
2) Feed-Forward Neural Network (FFN)
   self.ffn is a point-wise feed-forward network that applies a transformation independently to each position in the sequence. It helps the model learn more complex representations. The dff parameter specifies the number of units in the hidden layer of the feed-forward network. The function pointwisefeedforwardnetwork is assumed to be defined elsewhere in the code, typically containing two layers with a ReLU activation in between.
3) Layer Normalization
   self.layernorm1, self.layernorm2, and self.layernorm3 are instances of Layer Normalization, which is used to stabilize the training process and speed up convergence. It normalizes the inputs to each layer across the batch, reducing internal covariate shift.
4) Dropout
   self.dropout1, self.dropout2, and self.dropout3 are dropout layers to prevent overfitting by randomly setting some of the activations to zero during training. The dropout rate is controlled by the rate parameter, which defaults to 0.1.
5) 5. Call Method
   This method defines the forward pass of the DecoderLayer. Here's what happens step by step:
   - Self-Attention (MHA1): The first attention block (mha1) computes self-attention for the target sequence (x). This helps the decoder focus on different positions in the current sequence while making predictions. The result (attn1) is passed through a dropout layer (self.dropout1), and then a residual connection is applied, followed by layer normalization (self.layernorm1).
   - Encoder-Decoder Attention (MHA2): The second attention block (mha2) performs encoder-decoder attention, where the query comes from the output of the first attention (out1), and the keys and values come from the encoder's output (encoutput). Again, dropout (self.dropout2) is applied, and a residual connection is added, followed by layer normalization (self.layernorm2).
   - Feed-Forward Network (FFN): The output from the second attention layer (out2) is passed through the feed-forward network (self.ffn). After passing through the FFN, dropout (self.dropout3) is applied, followed by another residual connection and layer normalization (self.layernorm3).

B. Encoder Layer

The EncoderLayer class you provided is the encoder part of the Transformer architecture. It includes two main sub-layers: self-attention and a feed-forward neural network (FFN). Just like the DecoderLayer, this encoder layer has its own attention mechanism, normalization, and dropout components.

Here's a detailed explanation of the EncoderLayer:

1) 1. Multi-Head Attention (MHA)
   self.mha: This is the Multi-Head Attention layer used for self-attention. The self-attention mechanism allows the encoder to focus on different parts of the input sequence while processing each token. In this case, all three inputs (queries, keys, and values) are the same (x), meaning the attention mechanism is applied to the input sequence itself.
2) Feed-Forward Neural Network (FFN)
   self.ffn: This is the point-wise feed-forward network that applies the transformation independently at each position in the sequence. It is used to help the model capture more complex relationships in the input data. The function (dmodel, dff) is assumed to be defined elsewhere in your code and typically includes two layers with a ReLU activation in between.
3) Layer Normalization
   self.layernorm1 and self.layernorm2: These are instances of Layer Normalization, which normalizes the input across the batch. This helps in stabilizing training and improving convergence speed by reducing internal covariate shift.
4) Dropout
   self.dropout1 and self.dropout2: These are Dropout layers used to prevent overfitting by randomly setting some of the activation values to zero during training. The dropout rate is controlled by the rate parameter (default is 0.1).
5) 5. Call Method
   - Self-Attention (MHA): The input x is passed through the self-attention mechanism (self.mha(x, x, x, mask)), where the query, key, and value are all the same (x). This allows each token to attend to all other tokens in the input sequence. The output of the self-attention mechanism (attnoutput)

is passed through dropout (self.dropout1) to prevent overfitting. A residual connection is added between the original input (x) and the attention output (attnoutput), followed by layer normalization (self.layernorm1).

- Feed-Forward Network (FFN): The output of the first sub-layer (out1) is passed through the feed-forward network (self.ffn(out1)), which is applied independently to each position in the sequence. The output from the FFN (ffnoutput) is then passed through dropout (self.dropout2). Again, a residual connection is added between the input to the FFN (out1) and the FFN output (ffnoutput), followed by layer normalization (self.layernorm2).
- Return: Finally, the method returns the final output (out2), which is the result of applying both self-attention and the feed-forward network, with the corresponding residual connections and layer normalization.

V. Results

A. Result For Caption Generation

1) IMG 1 Real Caption: ”dog leaps to catch ball” Predicted

Fig. 3. Output.

Caption: ”dog catches ball outside”

- BLEU-1: 38.94
- BLEU-2: 55.07
- BLEU-3: 63.26
- BLEU-4: 65.49

2) IMG 2 Real Caption: young girl is playing in fountain of water Predicted Caption: young boy plays in the water without shirt

- BLEU-1 score: 37.5
- BLEU-2 score: 61.237243569579455
- BLEU-3 score: 74.5091107671223
- BLEU-4 score: 78.25422900366436

B. Results For Object Detection

Based on the detected objects in the image, it is likely that the scene takes place indoors, possibly in a living room or home office. The presence of a potted plant suggests an indoor environment, as houseplants like aloe vera or snake plants are commonly found in such spaces. Additionally, the visible cell phone indicates a location where people frequently use phones, such as a living room or office. These elements together point to a domestic or office-related setting, with the plant and phone providing clues about the location’s functionality and atmosphere.

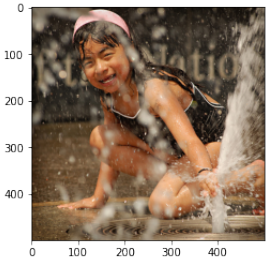

Fig. 4. Output.

Fig. 5. Ouput

VI. Conclusion adn Future Work

In conclusion, the use of combined masks in the Transformer decoder plays a crucial role in ensuring that the model attends to only valid information during training and inference. By incorporating both the look-ahead mask and the padding mask, the decoder can effectively prevent future token leakage and avoid attending to padding tokens, which are not part of the actual sequence. This combination of masks supports the autoregressive nature of the model, where each token is predicted based solely on the previously seen tokens. The careful application of these masks is key to maintaining the integrity of the sequence generation process, contributing to the success of the Transformer architecture in tasks such as machine translation, text generation, and other sequence-to-sequence tasks.

References

[1] P. Anderson, B. Fernando, M. Johnson, and S. Gould. SPICE: Semantic propositional image caption evaluation. In European Conference on Computer Vision, pages 382–398. Springer, 2016.

[2] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018

[3] M. Denkowski and A. Lavie. Meteor universal: Language specific translation evaluation for any target language. In Proceedings of the ninth workshop on statistical machine translation, pages 376–380, 2014

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[5] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2625–2634, 2015.

[6] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, et al. From captions to visual concepts and back. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1473–1482, 2015.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 580–587, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016. Vision and Pattern Recognition, pages 580–587, 2014.

[9] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3588–3597, 2018.

[10] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4565–4574, 2016.