Hierarchical Attention Captioner (HAC): Transformer with Hierarchical Attention Based Image Captioner

Raghvendra Kumar (raghk23@iitk.ac.in)
Prof. Koteswar Rao J (kotesrj@iitk.ac.in)

April 19, 2025

Abstract

In this paper, we present Transformer CNN based architecture for image captioning, that integrates a CNN backbone with novel transformer enhancements to capture both fine-grained and global context via a learned adaptive fusion mechanism.Integrated with a Pretained-ResNet50-based CNN backbone with transformer-based encoder-decoder blocks. Our model leverages non-local channel mixing, graph residuals, and custom transformer modules to capture both local and global image context, producing coherent and contextually rich captions. We have trained model for Hindi captioning and English captioning.

1 Introduction

Vision-Language Models (VLMs) have emerged as a response to the growing need for systems that can process and integrate visual and textual data effectively,producing descriptive sentence autonomously from the images. The text based summarization of the video either recorded or online is one of the potential impact of effective image captioning. There is vast application of accurate and efficient image captioning like healthcare (medical images paired with diagnostic reports),pivotal in developing assistive technologies that cater to the needs of visually impaired individuals, autonomous systems (sensor feeds integrated with navigation commands), and social media (images combined with captions) and many more. Image captioning is a key task that necessitates a semantic comprehension of images as well as the capacity to generate accurate and precise description sentences. In our work, we started by utilizing a Pretrained CNN models for feature extraction. These models are well-known for their ability to extract rich and meaningful features from images. Within Encoder Layer after the extracted visual features by Resnet 50(or other), Non-Local Layer 1x1 convolution for channel-wise feature mixing is used to the Enhances spatial relationships between image regions.Then Positional Embeddings leyer are used for Learnable position codes for spatial awareness of the tokens(patches of the iamge feature, 7*7=49).Then Transformer Encoder Layers is employed, for Stacked self-attention blocks to model relationships between image regions.At the end of Encoder process of spatial feature used Graph Residual, Lightweight linear layer refining encoder output.

Decoder module which generates is completely transformer based, which has Token Embeddings,Maps vocabulary IDs to dense vectors then Positional Embeddings,Learnable sequence position codes (up to maximum length of tokens decided by us) and finally Transformer Decoder,which Masked Self-Attention, Autoregressive token relationships and Cross-Attention a standard multi head attemtion module (between the visual token to text tokens) Fuses visual features from encoder memory. Then a Feedforward Networks, for Position-wise transformations. At the ende of Decoder, Prediction Head a fully connected layer, Projects final embeddings to vocabulary probabilities.

For text tokenizing we have used **Bart Tokenizer)** from **'facebook'/ bart-base)**

2 Related Work

Most of the model are encoder decoder based concept for the captioning task where encoder learn visual spatial feature and decoder predict the text. Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge, present a generative model based on a deep recurrent architecture that combines recent advances in computer vision and machine translation and that can be used to generate natural sentences describing an image. The model is trained to maximize the likelihood of the target description sentence given the training image. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, describe how we can train this model in a deterministic manner using standard backpropagation techniques and stochastically by maximizing a variational lower bound. an approaches to caption generation that attempt to incorporate a form of attention with two variants: a "hard" attention mechanism and a "soft" attention mechanism. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering, propose a combined bottom-up and top-down attention mechanism that enables attention to be calculated at the level of objects and other salient image regions. This is the natural basis for attention to be considered. Within this approach, the bottom-up mechanism (based on Faster R-CNN) proposes image regions, each with an associated feature vector, while the top-down mechanism determines feature weightings

3 Proposed Method

3.1 Architecture Overview

Our model is a hybrid encoder-decoder architecture combining a CNN-Transformer visual encoder with an autoregressive Transformer text decoder; given an input image $I \in \mathbb{R}^{3\times H\times W}$, the encoder generates visual tokens $V \in \mathbb{R}^{N\times d}$ (where $N = 49$ for a 7×7 grid and d is the embedding dimension) due to resnet ouput layer dimension, and the decoder generates a caption $y = (y_1, \ldots, y_T)$ conditioned on V via cross-attention, optimizing the objective $\mathcal{L} = \sum_{t=1}^{T} \log P(y_t \mid y_{<t}, V)$.

3.2 Hybrid Visual Feature Extractor

Our hybrid visual feature extractor consists of a frozen ResNet-50 backbone that produces spatial features $F \in \mathbb{R}^{2048\times 7\times 7}$, which effectively capturing rich hierarchical information; these features are passed through a 1×1 convolutional layer $W_c \in \mathbb{R}^{2048\times 2048\times 1\times 1}$ to enhance channel-wise context, which yielding $F' = \text{GELU}(W_c * F + b_c)$, which is then flattened into $N = 49$ spatial tokens $X \in \mathbb{R}^{49\times 2048}$, as nneded for the cross attention later with 49 text tokens, linearly projected to d-dimensional embeddings $X' = XW_p + b_p$, and augmented with learnable positional embeddings $E_{\text{pos}} \in \mathbb{R}^{49\times d}$ to form the input $V^{(0)} = X' + E_{\text{pos}}$; the token sequence $V^{(0)}$ is then passed through a stack of Transformer encoder blocks (each with LayerNorm, multi-head self-attention, and MLP) to model long-range dependencies and contextual interactions, followed by a lightweight residual graph projection, enabling spatially-aware and globally contextualized visual token representations.

3.2.1 Transformer Encoder

While the ResNet-50 backbone efficiently extracts local spatial features $F \in \mathbb{R}^{2048\times 7\times 7}$, it lacks the capacity to model long-range dependencies and global context; therefore, a Transformer encoder is applied to the flattened and embedded tokens $V^{(0)} \in \mathbb{R}^{49\times d}$, enabling each token to attend to all others via multi-head self-attention, thereby enriching the representation with global semantic context, spatial interactions, and scene-level consistency that CNNs alone cannot provide. Graph Residual Link: A skip connection with a linear transform refines the final output. Uniqueness for this provision, the graph residual adaptively fuses global visual relationships, complementing the transformer's self-attention.

3.3 Transformer-based Text Decoder

The fully Transformer-based text decoder in our model is designed using a stack (4 layers) of enhanced Transformer decoder blocks, which transform encoded visual features into coherent natural language sequences (e.g., image captions). The decoder is responsible for generating the caption token-by-token in an autoregressive manner.

3.3.1 Token Embedding and Positional Encoding

The input to the decoder consists of token IDs representing either the ground truth caption (during training) or the generated sequence (during inference). Each token is passed through an embedding layer:

$$\mathbf{x} = \text{Embedding}(\texttt{tgt\_ids}) + \mathbf{E}_{\text{pos}},$$

where $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{1\times L\times d}$ are learnable positional encodings that capture sequential information essential for language modeling.

3.3.2 Masked Self-Attention

The decoder uses *causal (masked) self-attention* to ensure that each position in the sequence can only attend to earlier positions. This is critical for autoregressive generation:

$$\text{SelfAttn}(\mathbf{x}) = \text{MultiHead}(\mathbf{x}, \mathbf{x}, \mathbf{x}) \quad \text{with causal mask},$$

which enables the decoder to learn the context of previously generated tokens without peeking into the future.

3.3.3 Cross-Attention with Visual Memory

After self-attention, the decoder incorporates visual information via *cross-attention* over the memory produced by the encoder:

$$\text{CrossAttn}(\mathbf{x}) = \text{MultiHead}(\mathbf{x}, \mathbf{M}, \mathbf{M}),$$

where $\mathbf{M}$ denotes the memory tokens output by the CNN-transformer hybrid encoder. This enables the decoder to align words with relevant image regions during generation.

3.3.4 Feedforward Network and Layer Normalization

Each decoder block ends with a position-wise feedforward network (FFN) defined as:

$$\text{FFN}(\mathbf{x}) = \text{GELU}(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2.$$

Layer normalization and residual connections are applied after each major component (self-attention, cross-attention, FFN) to enhance training stability and performance.

3.3.5 Output Projection

The final decoder output is passed through a linear projection to produce vocabulary logits:

$$P(w_t) = \text{Softmax}(\mathbf{W}_{\text{out}} \cdot \mathbf{x} + \mathbf{b}),$$

which represents the probability distribution over the vocabulary at each time step.

3.3.6 Advantages

- **Causal masking** ensures autoregressive generation, critical for sequence modeling.
- **Self-attention** captures long-range dependencies within the generated sequence.
- **Cross-attention** grounds the textual generation in visual semantics.
- **Feedforward layers and layer normalization** ensure non-linearity and stable optimization.
  Uniqueness: Our decoder uses separate normalization for self cross-attention, stabilizing gradient flow during training.
- **Stacked layers** allow for deeper reasoning over both language and vision domains.
- Early Stopping control : Generation terminates when all sequences in the batch predict (EOS) or reach a maximum length that is controlled by us. so model can be custumized for 1 line captioner or detailed captioner.

4 Novelty and Contributions

The proposed model introduces a novel hybrid framework for image captioning by tightly integrating convolutional neural networks (CNNs) with transformer-based architectures. The uniqueness of this work stems from multiple carefully designed components that together improve semantic understanding and language generation. The key contributions and novelties of the proposed model are as follows:

1. **Hybrid Visual Feature Extractor:** Unlike standard CNN-only or transformer-only vision models, we propose a hybrid extractor where a ResNet-50 CNN backbone is combined with transformer encoder layers. This design leverages:
    - Local pattern recognition and spatial hierarchy from CNNs.
    - Global context modeling and long-range dependency capturing via transformers.
    - Lightweight self-attention over tokenized CNN features with learnable positional encodings.
2. **Transformer-Enhanced Visual Encoding:** The transformer encoder is applied post-CNN to refine spatial tokens. This step enhances semantic richness in visual features and allows for multi-scale reasoning over image regions. Additionally, a residual graph-like transformation is introduced over transformer outputs to further improve inter-region relationships.
3. **Attention-Driven Decoder:** The decoder is composed of stacked transformer blocks with both:
    - *Masked Self-Attention* for autoregressive text generation.
    - *Cross-Attention* over visual memory tokens, enabling dynamic alignment between generated words and image regions.

    This dual-attention structure enhances the decoder's ability to ground language in visual semantics.
4. **Efficient Training and Generalization:**
    - The CNN backbone is optionally frozen to speed up training and reduce overfitting.
    - Lightweight layers and efficient architecture choices maintain competitive performance while reducing computational complexity.
5. **Unified End-to-End Captioning Pipeline:** The model provides an end-to-end pipeline from raw images to natural language captions without relying on external object detectors, region proposals, or pre-extracted features.

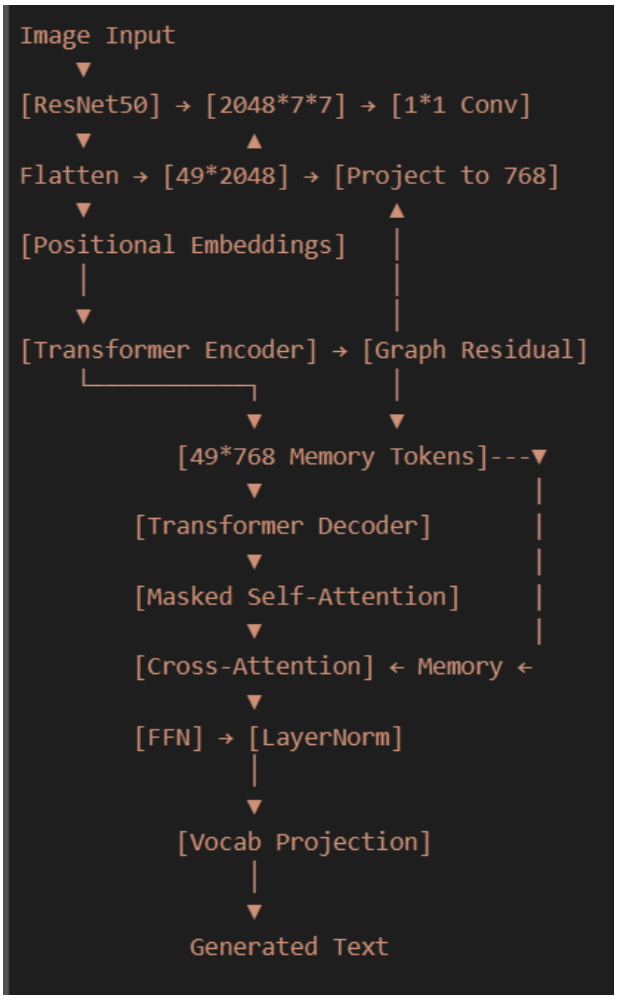

Figure 1: Model Architecture overview

Comparative Edge

Model is getting trained with very limited GPU memeory Tesla4 15Gb, with batch size of 32. Our work is trained for getting minimum wors that is most representative for the image. Also it takes lesser time than GP2, Clip etc. other large language based models.

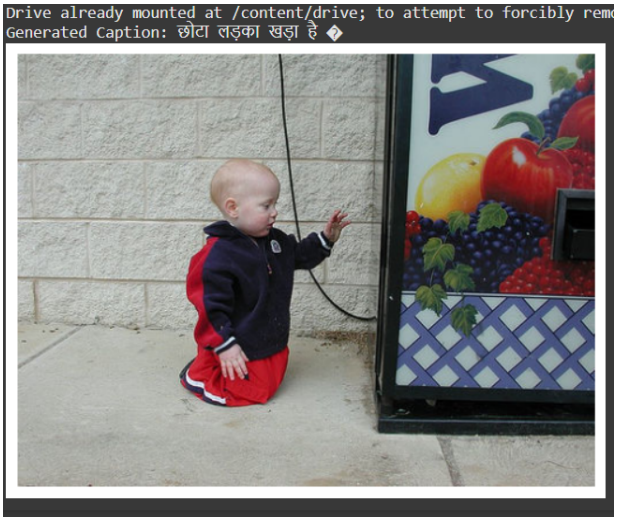

Figure 2: Example output

Figure 3: Example output

5 Experiments

5.1 Dataset and Training

We have used Flickr8k dataset, with hindi caption and english caption. The Flickr8k is a new benchmark collection for sentence-based image description and search, consisting of 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events. ... The images were chosen from six different Flickr groups, and tend not to contain any well-known people or locations, but were manually selected to depict a variety of scenes and situations. To train our model, we have use Clean-5Sentences-withComma.txt captions file for hindi and captions.txt for english caption.

5.2 Implementation Details

The Github Link below containing the all other details
https://github.com/RAGHAVAN000/Ticap-Captioner-model

5.3 Results and Comparisons

Multiple ouput example of image captioning is uploaded in the github link

6 Future Work

We can replace resnet-50 with ConvNext like transformer architecture for efficient yet little higher per-

formance. Also can build own tokenizer specific for the Hindilanguage. Can improve model architecture for Multi Lingual support.

References

[1] https://www.kaggle.com/datasets/adityajn105/flickr8k

[2] Gaurav Shinde et al, A Survey on Efficient Vision-Language Models. https://arxiv.org/pdf/2504.09724v1

[3] https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00571-w

[4] Ridoy et al, Compressed Image Captioning using CNN-based Encoder-Decoder Framework. https://arxiv.org/html/2404.18062v1

[5] Vinyals et al, Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge. https://arxiv.org/pdf/1609.06647v1

[6] Kelvin Xu et al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. https://arxiv.org/pdf/1502.03044v3

[7] Anderson et al, Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. https://openaccess.thecvf.com/contentcvpr2018/papers/AndersonBottom-UpandTop-DownCVPR2018paper.pdf

[8] Attention Is All You Need, Ashish Vaswani et al

7 Appendix

Model code is

model = AdvancedCaptionModel criterion=nn.CrossEntropyLoss optimizer = torch.optim.AdamW(model.parameters(), lr=1e-4)

Epoch 1, Loss: 1.8528 Epoch 2, Loss: 0.9171 Epoch 3, Loss: 0.7711 Epoch 4, Loss: 0.6918 Epoch 5, Loss: 0.6304 Epoch 6, Loss: 0.5747 Epoch 7, Loss: 0.5207 Epoch 8, Loss: 0.4646 Epoch 9, Loss: 0.4047 Epoch 10, Loss: 0.3466 Epoch 11, Loss: 0.2887 Epoch 12, Loss: 0.2349 Epoch 13, Loss: 0.1936 Epoch 14, Loss: 0.1555 Epoch 15, Loss: 0.1278 Epoch 16, Loss: 0.1054 Epoch 17, Loss: 0.0891 Epoch 18, Loss: 0.0774 Epoch 19, Loss: 0.0702 Epoch 20, Loss: 0.0655 Epoch 21, Loss: 0.0625 Epoch 22, Loss: 0.0576 Epoch 23, Loss: 0.0535 Epoch 24, Loss: 0.0508 Epoch 25, Loss: 0.0502 Epoch 26, Loss: 0.0483 Epoch 27, Loss: 0.0458 Epoch 28, Loss: 0.0444 Epoch 29, Loss: 0.0445 Epoch 30, Loss: 0.0417 Epoch 31, Loss: 0.0421 Epoch 32, Loss: 0.0406 Epoch 33, Loss: 0.0384 Epoch 34, Loss: 0.0389 Epoch 35, Loss: 0.0379 Epoch 36, Loss: 0.0380 Epoch 37, Loss: 0.0372 Epoch 38, Loss: 0.0360 Epoch 39, Loss: 0.0343 Epoch 40, Loss: 0.0350 Epoch 41, Loss: 0.0354 Epoch 42, Loss: 0.0332 Epoch 43, Loss: 0.0303 Epoch 44, Loss: 0.0312 Epoch 45, Loss: 0.0304 Epoch 46, Loss: 0.0301 Epoch 47, Loss: 0.0330 Epoch 48, Loss: 0.0316 Epoch 49, Loss: 0.0291 Epoch 50, Loss: 0.0300