SYMBIOSIS INSTITUTE OF TECHNOLOGY, NAGPUR

Generating descriptive image captions by using encoder-decoder architecture.

PBL Review Report – I

Submitted To:

Dr. Deepak Suresh Asudani

Assistant Professor

Submitted By:

Mukund Kuthe, 23070521082

4th Semester, Section B

Rushi Parhad, 23070521122

4th Semester, Section B

Sharvayu Zade, 23070521135

4th Semester, Section B

August 2025

Articulate problem statements and identify objectives

Problem statement:

The goal of image captioning is to produce a set of descriptions written in natural language in order to represent what an image is about. Conventional methods demand huge labelled data sets and long learning period which is time consuming and restrict generalization about new areas. Also object detection is a problem as we also need to know a background and connections to create credible and consistent text.

The approach to such limitations adopted by pretrained vision-language models such as BLIP-2 is the use of frozen image encoders with large language models that allow high quality captions. The work is dedicated to using such pretrained models to produce descriptive and context-aware captions with decreased requirements of using large-scale labeling and better results under various conditions.

Objectives:

1. **Use Pretrained Models** - Exploit the BLIP-2 to create captioning of a high-quality image without lengthy re-education.

2. **Context-Aware in Descriptions**- create captions that not only describe objects, but their relationships and the context of an entire scene.

3. **Reducing Data Dependence**- Minimalize the use of highly annotated datasets by transfer learning using pretrained vision-language models.

4. **Assist Various Domains** -Giving correct captions on diverse kinds of images and different settings without training the domain during training.

5. **Multiple Outputs** – Experiment across generation parameters (temperature, top-k, top-p, typical_p) so as to generate various and descriptive captions.

Identify engineering systems variables, and parameters to solve the problems.

Engineering Systems / Tools:

1. Hugging Face Transformers (pipeline) - An API over the BLIP image captioning model.
2. BLIP Pretrained Model (Salesforce/blip-image-captioning-base) Vision Vision language model that generates high-quality descriptive captions on large-scale, uncurated datasets.
3. Python Programming language is used to implement and integrate.

Variables:

1. image_path-Saves the file path name of input image (testing.jpg).
2. capt1, capt2, capt3, capt4, capt5- Store captions created by different parameters of caption.
3. image_to_text- The Hugging Face pipeline object to process image-to-text.

Parameters:

1. do_sample - Promotes the generation through the use of sampling in place of deterministic decoding so that the model can generate diverse captions.
2. temperature - controls the variability of the token choice; the higher the value, the more imaginative the captions.
3. top_k - Narrows down the scope of tokens to the k best probability ones.
4. top_p – Uses nucleus sampling, which picks the smallest subset of tokens ie the cumulative probability.
5. Typical_p – Typical p controls how much the model prefers tokens whose surprise is close to the average surprise of the distribution.

Identify existing processes/ solution methods for solving the problem

In the paper “Improving Image Captioning Descriptiveness by Ranking and LLM-based Fusion”[1], an entire pipeline is shown by initially retrieving multiple candidate captions using state-of-the-art captioners and then ranking the candidates using an image-text alignment metric; these ranked candidates are then fused into a single, more detailed description by a Large Language Model.

Expanding the premise of beginning with flawed text, in the article "From Alt-text to Real Context: Revolutionizing image captioning using the potential of LLM"[2] the raw web alt-text is treated as a poor quality, noisy source of caption and the LLM prompting is used to expand and situate those tersely described images. Rather than generating captions using only pixels, it takes an existing textual metadata and a sparse amount of visual cues to generate context-aware captions which add inferred background and likely actions, as well as cultural or situational context.

Paper “FuseCap: Leveraging Large Language Models for Enriched Fused Image Captions”[3] refines that next step and combines visual (detectors, OCR, attributes) and textual cues in a multimodal fusion step driven by an LLM. FuseCap gathers structured visual evidence and assembles these signals with the candidate captions or prompts, and uses an LLM to create a coherently semantically consistent caption that ties the mixed inputs together.

Generalizing the concept of LLM-based enrichment to style and form, the paper “Image Caption Extending Using LLM and Style Transfer”[4] takes a base caption (obtained during captioning by a captioner, or using fused inputs as an intermediate step in a captioner) and then applies LLM-based extension to the element of that base caption, followed by a style-transfer step to give a caption in a desired tone (e.g. poetic, formal, humorous).

The Paper “Image Captioning Using Multimodal LLMs”[5] discusses end-to-end multimodal LLM applications that take image embeddings and text requests directly and reason over them together to generate captions in a single real-time pass. This is unlike in the past staged pipelines where visual understanding was executed within multiple models; it is advantageous that this reasoning capability of the LLM is accessible just like the inference pipeline minimalization process.

Compare and contrast alternative solution processes to select the best process

First-generation models of image captioning are usually based on encoderdecoder models with benchmark training on datasets like MS-COCO. As much as COCO contains high quantity of image to caption pairs, captions tend to be short, object-specific and devoid of more elaborate contextual information. Models trained only with such data thus demonstrate a generic, template-quotient output and do not transfer to serving the over-all domains that COCO does not cover in vision and language. Moreover, the predefined vocabulary and narrow variety of sentences shapes the model to be limited in processing infrequent simply known as rare concepts or delicate interconnection between entities in terms of relationships. In spite of the attention mechanism or transformer-based encoders being added, these models still cannot escape the coverage and biases of the training data and thus they perform poorly in an open-domain setting.

However, the BLIP-2 model can avoid most of these limitations since it combines a vision encoder with a big pre-trained language model in a very small query transformer. In contrast to relying only on supervised training on domain-bounded datasets such as the COCO, BLIP-2 relies on additional linguistic diversity and world knowledge of LLMs, which produce more semantically rich and contextual captions, without needing of fine-tuning on a specific task. The loss of language by BLIP-2 allows decreasing the dependency on pricey dataset-specific retraining and, at the same time, leads to substantially lowering generalization and quality of the captions.[6]

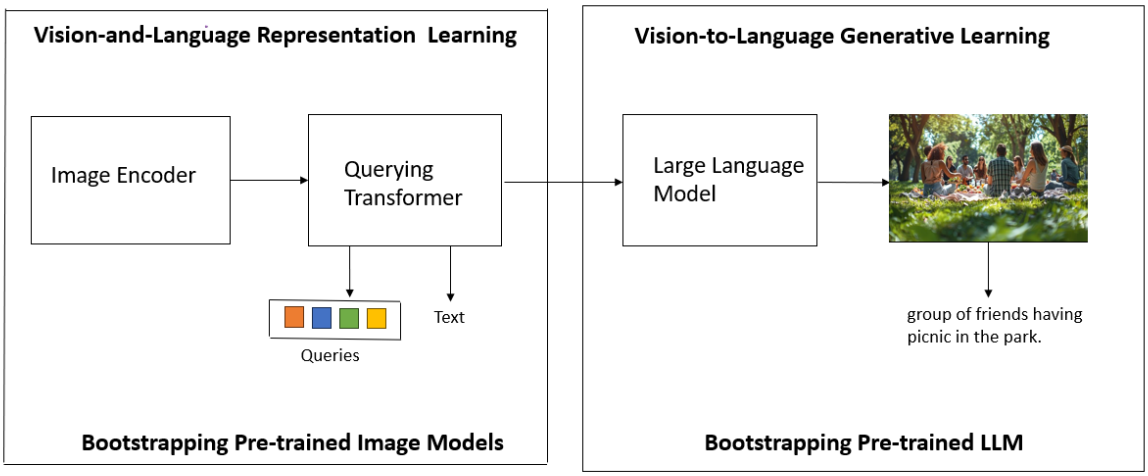

Fig 1: BLIP-2's framework

Read, understand, and interpret technical and non-technical information

BLIP-2 Pipeline Steps:

1. Image Encoder

Step 1 : Input image and cutting into patches

You have a picture (say, a 224 × 224 image of a cat wearing sunglasses).

The encoder cuts the image into small squares of size 16 × 16 pixels.

Across the width: 224/16 =14 patches Across the height: 224/16 =14 patches

Total patches = 14×14 =196 small image pieces.

Step 2 : Turning patches into numbers

Each small patch has 16 × 16 × 3 = 768 numbers (pixel values).

The model changes these 768 numbers into a vector of length 768. Now we have 196 vectors, each with 768 number. The image has 3 channels (R, G, B).

Step 3 : Adding position code:

The model adds a small "positional code" to each patch vector, so it knows where the patch came from in the image (top-left, middle, bottom-right, etc.).

Each patch vector size: 768 and Positional vector size: 768

For patch 1:
$z1=[0.12,0.88,-0.32,\dots]$
$p1=[0.05,-0.11,0.09,\dots]$

Add: $z1+p1 = z1(0) = [0.17,0.77,-0.23,\dots]$

Step 4: Understanding the Image and Selecting Key Features

The 196 patch vectors (each size 768) go through Transformer layers.

Each layer: Self-Attention → patches share info about the whole image, Feed-Forward → strengthen features.

BLIP-2 then sends these 196 vectors to the Q-Former.

2. Querying Transformer

A Q-Former architecture that links an image encoder with other different multimodal tasks. It begins when an Input Image is fed to Image Encoder, which is subject to treatment first. This encoder picks out the image features. Thereafter, they are passed through a stack of transformer blocks in Q-Former applied to all other blocks in the network. One individual layer comprises three major components: Self Attention where the self-attentive queries are enabled to attend to one another; Cross Attention that links the self-attentive queries to the encoder image features; and Feed Forward net that processes the attention outputs. The Learned Queries are learnable embeddings that serve as tokens to distill useful information about the task (in the image features).

The right side of the diagram identifies the manner in which Q-Former accommodates various tasks through dissimilar output heads. The learned query results are run through Image-Text Matching to perform semantic similarity compared to the texts embeddings. In Image-Grounded Text Generation, a Self Attention and Feed Forward layers network adds the set of learned queries and text tokens to each other in accordance with Attention Masking strategy. This kind of masking leads to various types of attention behaviors to occur such as directional dual modality in encoding stages and multimodal causal modality in generation and one in specific tasks. Lastly, the model is able to generate text conditional on image as well as text given an Input Text.[7]

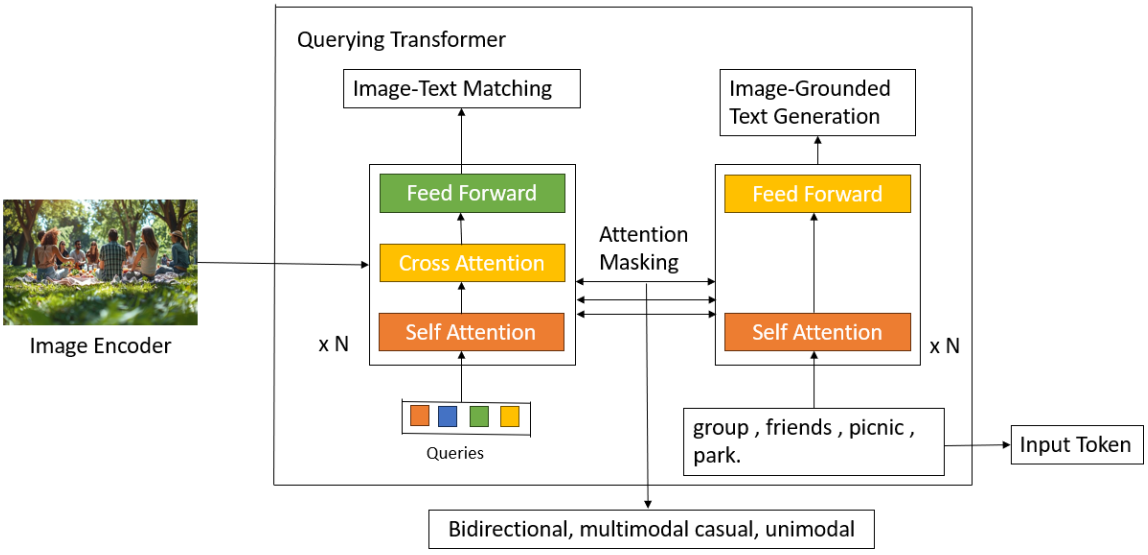

Fig 2: Querying Transformer

Step 1: Input get from Image Encoder and Learned Queries

We get the vectors values from the encoder and map it as Image Matrix(I)

Image Matrix(I)

$$I = \begin{bmatrix} 0.2 & 0.1 & 0.4 \\ 0.5 & 0.3 & 0.7 \\ 0.9 & 0.6 & 0.2 \\ 0.4 & 0.8 & 0.5 \end{bmatrix}$$

Learned Queries

The model uses its own internal, trainable query vectors to interrogate the image. Query Matrix (Q)

$$Q = \begin{bmatrix} 0.6 & 0.2 & 0.1 \\ 0.1 & 0.7 & 0.3 \end{bmatrix}$$

Step 2: Self Attention

Formula : $Q^{sa} = (QQ^{T)}\, Q$

Where: Q is Query Matrix and Q^T is the Transpose of it.

$$Q^{sa} = \begin{bmatrix} 0.3725 & 0.4275 & 0.1910 \\ 0.3055 & 0.4945 & 0.2178 \end{bmatrix}$$

Step 3: Cross Attention

Formula : $Q^{ca} = (Q^{sa}I^T)I$

$$Q^{ca} = \begin{bmatrix} 0.5321 & 0.4892 & 0.4423 \\ 0.5288 & 0.4913 & 0.4445 \end{bmatrix}$$

Step 4: Feed-Fordward Network(FFN)

Formula: $Q^{final} = Q^{ca} + FFN(Q^{ca})$

Where: $FFN(Q^{ca}) = (GELU((Q^{ca} \cdot W1)+b1) \cdot W2)+b2$

W1: The initial weight matrix that increases the vector to a higher, latent dimension.

b1: The initial biasing vector; this is added to the final product of the first multiplications.

GELU: It consists of the activation function (Gaussian Error Linear Unit) that adds non-linearity to the model and enables its ability to learn more complex patterns.

W2: The second weight matrix, this is what brings the vector in its initial dimension.

b2: The second bias vector to which addition comes as the last resort.

$$Q^{final} = \begin{bmatrix} 1.3303 & 1.2230 & 1.1058 \\ 1.3220 & 1.2283 & 1.1113 \end{bmatrix}$$

Step 5: Setup for the token generation

In Q^{final} we combine token [START] to create the first token . In combining we denote it as Z.

$$Z = \begin{bmatrix} 1.3303 & 1.2230 & 1.1058 \\ 1.3220 & 1.2283 & 1.1113 \\ 0.1000 & 0.1000 & 0.1000 \end{bmatrix}$$

Step 6: Generation of the first token

1. Calculate the Mask Score

Formula: Masked Score = (ZZ^T) + Mask

Where:

$$\text{Mask} = \begin{bmatrix} 0 & -\infty & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 \end{bmatrix}$$

Therefore Score Row = [0.37, 0.37, 0.03]

2. Calculate Attention Weights

Formula: Weights = softmax(Score Row)

Where:

$$\text{Softmax}(x_i) = e^{x_i} / \sum_j e^{x_i}$$

Weights = [0.369, 0.369, 0.262]

3. Output Vector

Formula: O = Weights * Z

O = [0.978, 0.903, 0.817]

4. Prediction of Word

Formula: Logit = O * W_{vocab}

Where: W_{vocab} is a weight matrix for the vocabulary

Logit = {“cat”:0.247, “garden”: 2.945 }

The model picks the word with the highest score so it takes the word **‘garden’.**

3. Large Language Model

In BLIP-2, the Large Language Model (LLM) is the component that turns the visual information extracted by the image encoder and refined by the Q-Former into natural language output. After the Q-Former produces a compact set of “visual tokens” (for example, 32 vectors summarizing the important parts of the image), these are mapped into the LLM’s embedding space so that the LLM can treat them just like word embeddings. This lets the LLM “see” the image content in the same representational format it uses for text, enabling it to reason about both language and vision together.

With the visual tokens installed, the LLM integrates them with the text data it receives (such as a question or prompt), then runs a series of inputs through its so-called Transformer layers, a strategy that allows it to reason in multiple steps. It uses its existing linguistic knowledge to understand the visual scene and adhere to instructions to give a coherent response either by captioning it through text or providing an answer to a visual question or by describing the scene which would be used in retrieving tasks.

Parameters used in BLIP-2

When the Q Transformer and LLM finishes processing the context, it gives raw logit scores (real numbers, can be positive or negative).

Token – Logit: the 2.3, a 1.8, an 1.2, this 0.9, some 0.3, those -0.2, all -0.5

Converting logits → probabilities, We use the softmax function.

Softmax: $P(token_i) = e^{logit_i} / \sum_j e^{logit_i}$

Step 1: Exponentiate each logit = exp(1.2) = 3.320, exp(0.8) = 2.225, exp(0.1) = 1.105, exp(-0.5) = 0.607, exp(-1.0) = 0.368.

Step 2: Sum them up = 3.320 + 2.225 + 1.105 + 0.607 + 0.368 = 7.625

Step 3: Divide each exponential value by the sum.

Token	Logit	exp(Logit)	Probability
the	1.2	3.320	0.435
a	0.8	2.225	0.292
an	0.1	1.105	0.145
this	-0.5	0.607	0.080
some	-1.0	0.368	0.048

Table 1 : Probability Distribution

1. Top-k

Sort by probability: the (0.435), a (0.292), an (0.145), this (0.080), some (0.048). Keep the top 3 tokens: the, a, an. (top-k= 3)

Sum of the probabilities = 0.435 + 0.292 + 0.145 = 0.872.

Renormalize each probability by dividing by 0.872:

P(the) = 0.435 / 0.872 = 0.499

P(a) = 0.292 / 0.872 = 0.335

P(an) = 0.145 / 0.872 = 0.166

Now the LLM will select the word randomly.

2. Top -P

Accumulate probabilities from the top until cumulative ≥ 0.8 (top-p= 0.8)

after the: cumulative = 0.435

after a: cumulative = 0.435 + 0.292 = 0.727

after an: cumulative = 0.727 + 0.145 = 0.872 ≥ 0.8

Renormalize exactly as above (divide by 0.872):

P(the) = 0.499

P(a) = 0.335

P(an) = 0.166

Now the LLM will select the word randomly.

3. Temperature

Softmax: $P(\text{token}_i) = e^{\text{logit}_i/T} / \sum_j e^{\text{logit}_i/T}$

If the temperature is equals to 1, the logits remain unchanged before calculating the softmax, so the new probability distribution is exactly the same as the original one. In this case, it select the word with the highest probability from the distribution.

4. Typical_p

Step 1: Calculate Shannon Information Content

IC(x) = -log p(x)

Token	Probability	Information content
the	0.435	0.833
a	0.292	1.231
an	0.145	1.931
this	0.080	2.526
some	0.048	3.037

Table 2: Shannon Information Content

Step 2: Calculate Entropy

Entropy $H = \sum p_i [-\log(p_i)]$

H=(0.435)(0.833)+(0.292)(1.231)+(0.145)(1.931)+(0.080)(2.526)+(0.048)(3.037)

H=0.362+0.359+0.280+0.202+0.146=1.349

Step 3: Calculate "typicality" for token

Typicality = $|-\log(p_i) - H|$

Token	Info content	|Info − H|
the	0.833	0.516
a	1.231	0.118
an	1.931	0.582
this	2.526	1.177
some	3.037	1.688

Table 3: typicality content

Step 5: Sort tokens by typicality (Lower deviation = more typical)

Order: a (0.118), the (0.516), an (0.582), this (1.177), some (1.688)

Typical_p = 0.5

Now sum their original probabilities in the sorted order until ≥ 0.5.

a → 0.292 (cumulative = 0.292)

the → +0.435 (cumulative = 0.727)

Therefore the selected tokens for the typical_p are a, the.

References :

[1] S. Bianco, L. Celona, M. Donzella, and P. Napoletano, “Improving image captioning descriptiveness by ranking and LLM-based fusion,” *arXiv.org*, Jun. 20, 2023. Available: https://arxiv.org/abs/2306.11593

[2] N. V. Patel, N. A. Modi, N. H. Mistry, N. A. Mishra, N. Dr. R. Upadhyay, and N. A. Shah, “From Alt-text to Real Context: Revolutionizing image captioning using the potential of LLM,” *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, vol. 11, no. 1, pp. 379–387, Jan. 2025, doi: 10.32628/cseit25111238. Available: https://doi.org/10.32628/cseit25111238

[3] N. Rotstein, D. Bensaïd, S. Brody, R. Ganz, and R. Kimmel, “FuseCap: Leveraging large language models for enriched fused image captions,” *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 5677–5688, Jan. 2024, doi: 10.1109/wacv57701.2024.00559. Available: https://doi.org/10.1109/wacv57701.2024.00559

[4] N. Andriyanov and V. Dementiev, “Image caption Extending using LLM and style transfer,” in *Lecture notes in computer science*, 2025, pp. 108–118. doi: 10.1007/978-3-031-87663-9_9. Available: https://doi.org/10.1007/978-3-031-87663-9_9

[5] N. Tiet, “Image Captioning Using Multimodal LLMs,” M.S. thesis, Dept. of Computer Science, Lund University, Lund, Sweden, LU-CS-EX 2025-02, 2025. Available: http://lup.lub.lu.se/student-papers/record/9185108

[6] J. Li, D. Li, S. Savarese, and S. Hoi, “BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models,” *arXiv (Cornell University)*, Jan. 2023, doi: 10.48550/arxiv.2301.12597. Available: https://arxiv.org/abs/2301.12597

[7] Y. Chebotar *et al.*, “Q-Transformer: Scalable offline reinforcement learning via autoregressive Q-Functions,” *arXiv (Cornell University)*, Jan. 2023, doi: 10.48550/arxiv.2309.10150. Available: https://arxiv.org/abs/2309.10150