**ABSTRACT**

Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. In this paper, we implements an image caption generator using the VGG16 architecture in TensorFlow and Keras. The project begins by extracting image features through VGG16 and storing them alongside corresponding captions. The text data undergoes preprocessing and tokenization, facilitating model training. A novel encoder-decoder architecture, combining VGG16 and LSTM, is defined for image caption generation. The model is trained using a data generator function, and the trained model is saved for future use. Evaluation metrics, including BLEU-1 and BLEU-2 scores, assess the model's performance. The code showcases generating captions for sample images and visualizing actual versus predicted captions. Additionally, a real image is tested with the trained model. The project highlights the integration of computer vision and natural language processing, emphasizing the importance of VGG16 for feature extraction in image captioning tasks.

# Table of Contents

[Table of Contents 2](#_Toc151477204)

[List of Figures 2](#_Toc151477205)

[1. CHAPTER 2](#_Toc151477206)

[INTRODUCTION 2](#_Toc151477207)

[1.1 What is image captioning 2](#_Toc151477208)

[1.2 CNN : Convolutional Neural Networks 2](#_Toc151477209)

[1.3 VGG16 2](#_Toc151477210)

[1.4 LSTM : Long Short-Term Memory 2](#_Toc151477211)

LITERATURE REVIEW 5

[2. CHAPTER 2](#_Toc151477212)

[METHODOLOGY 2](#_Toc151477213)

[2.1 Components 2](#_Toc151477214)

2.2 Dataset 7

[2.3 Image data preprocess 2](#_Toc151477215)

2.4 Define Model 8

[3. CHAPTER 2](#_Toc151477216)

[RESULTS & DISCUSSION 2](#_Toc151477217)

[CONCLUSION 2](#_Toc151477218)

FUTURE SCOPE 12

BIBLIOGRAPHY 13

# List of Figures

Figure 2.1 – Model workflow  [0](#_3j2qqm3)6

Figure 2.2 – Final Model Structure  [0](#_1y810tw)9

Figure 3.1 – Training image 10

Figure 3.2 – Resultant captions 10

Figure 3.3 – Final caption 10

Figure 3.4 – Testing image 11

Figure 3.5 –Final caption 11

Figure 3.6 –Bleu score 11

# CHAPTER

## INTRODUCTION

### What is image captioning

Image captioning aims at generating a natural language description of an image. Open domain captioning is a very challenging task, as it requires a fine-grained understanding of the global and the local entities in an image, as well as their attributes and relationships. Creating an image caption generator using Python is an exciting venture at the crossroads of computer vision and natural language processing. This project involves training a model to automatically generate descriptive captions for images, enhancing accessibility for visually impaired individuals and enriching user experiences. Leveraging pre-trained convolutional neural networks (CNNs) for feature extraction and recurrent neural networks (RNNs) for sequential processing, the model learns to associate images with contextual captions. Through this Python endeavor, you'll explore the synergy between deep learning, image processing, and language understanding, demonstrating the practical application of cutting-edge technologies in the realm of artificial intelligence.

### CNN : Convolutional Neural Networks

Convolutional Neural Networks (CNNs) play a pivotal role in image caption generation, serving as powerful tools for extracting meaningful features from images. In this context, CNNs are employed to analyze and comprehend the visual content before generating descriptive captions. Typically, a pre-trained CNN, such as VGG16, ResNet, or Inception, is used to extract hierarchical features from the input image. These features capture important visual elements, forming a semantic representation of the image. This representation is then fed into the subsequent layers of the image captioning model, often coupled with Recurrent Neural Networks (RNNs), to decode and generate coherent, contextually relevant textual descriptions. The integration of CNNs in image caption generation exemplifies their efficacy in image feature extraction and contributes to the overall success of the model in understanding and describing visual content.

### VGG16

VGG16, a prominent Convolutional Neural Network (CNN) architecture, plays a crucial role in image caption generation by extracting hierarchical features from input images. Comprising 16 weight layers, VGG16 is renowned for its simplicity and effectiveness. In the context of image captioning, a pre-trained VGG16 model serves as a feature extractor. The model processes the input image through multiple convolutional and pooling layers, progressively capturing intricate visual details. These learned features form a semantic representation of the image, capturing its essential characteristics. This rich feature set is then utilized in conjunction with Recurrent Neural Networks (RNNs) within the image captioning model. The combination of VGG16's capacity for hierarchical feature extraction and RNNs' sequential processing contributes to the generation of coherent and contextually relevant textual descriptions, showcasing VGG16's pivotal role in understanding and describing the visual content within the image captioning framework.

### LSTM : Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are integral components in image caption generators, enhancing the model's ability to generate sequential and contextually coherent descriptions. LSTMs serve as the decoding mechanism, receiving the hierarchical image features extracted by Convolutional Neural Networks (CNNs). Their distinctive architecture, featuring memory cells and gates, allows them to capture and retain relevant information over extended sequences. In the context of image captioning, LSTMs enable the model to understand and generate textual descriptions by learning the sequential dependencies present in the image features. The memory cells store essential context, ensuring that the model can maintain relevance and coherence throughout the caption generation process. Through their capacity for handling sequential data, LSTMs contribute significantly to the success of image caption generators, enabling them to produce detailed and contextually nuanced textual descriptions that align with the visual content extracted from the input images.

**LITRATURE REVIEW**

|  |  |  |
| --- | --- | --- |
| **Author(s)** | **Date** | **Conclusion** |
| Liu, Shuang & Bai, Liang & Hu, Yanli & Wang, Haoran (2018) | 2018 | Proposed CNN-RNN and CNN-CNN methods for image captioning. CNN-RNN uses both networks for encoding and decoding, while CNN-CNN uses only CNN, resulting in faster training with potential higher loss. |
| A. Hani, N. Tagougui and M. Kherallah (2019) | 2019 | Introduced an encoding-decoding model for image captioning, featuring retrieval-based and template-based captioning. Retrieval-based captioning correlates test images with captions, while prototype-based describing uses Inception V3 as the encoder with attention mechanisms. |
| Karpathy & Li (2014) | 2014 | Proposed a joint embedding space for ranking and generation, scoring sentence and image similarity based on R-CNN object detections with outputs of a bidirectional RNN. |
| Fang et al. (2014) | 2014 | Introduced a three-step pipeline, incorporating object detections. In contrast, their attention framework learns alignments from scratch, extending beyond objectness to abstract concepts. |

# CHAPTER

## METHODOLOGY

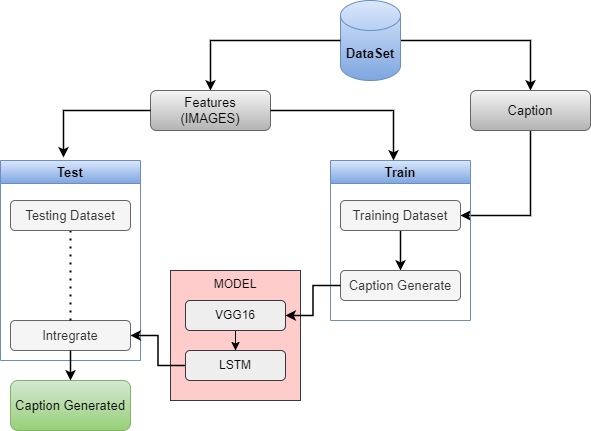


Fig2.1: Model workflow

### Components

**Python:** A versatile programming language commonly employed in machine learning and image captioning projects due to its simplicity and extensive libraries.

* **Pickle:** A Python module used for serializing and deserializing objects, crucial for saving and loading machine learning models.
* **NumPy:** A powerful numerical computing library in Python, essential for handling arrays and matrices in machine learning applications.

**TensorFlow:** An open-source machine learning framework, widely used for building and training deep learning models, including those for image caption generation.

* **Keras:** A high-level neural networks API running on top of TensorFlow, simplifying the process of building and training deep learning models.

**Tokenization:** The process of converting text into individual tokens or words, facilitating natural language processing tasks by breaking down text into manageable units.

**Mapping:** Refers to the creation of associations between elements, such as mapping words to numerical representations, a common step in natural language processing.

**BLEU** (Bilingual Evaluation Understudy): A metric used for evaluating the quality of generated text, commonly employed in image captioning to assess the similarity between generated and reference captions.

**2.2 Dataset**

Flickr 8k dataset <https://www.kaggle.com/datasets/adityajn105/flickr8k>

We have 8K images along with 45K captions in our used dataset. The Flickr 8k dataset is a publicly available image-to-sentence description benchmark. There are 8091 photos in this collection, each with five captions. These photos were taken from a variety of groups on the Flickr website. Each caption gives a detailed description of the objects and events seen in the photograph. The collection represents a wide range of events and settings and excludes photographs of well-known persons and locations, making it more generic. The dataset has the following characteristics that make it ideal for this project are:

* When many captions are mapped to a single image, the model becomes more general and avoids over fitting.
* Using a variety of training images allows the image captioning model to cope with a variety of image types, making the model more robust.

### Image data preprocess

1. IMAGE DATA PREPARATION
2. The image should be transformed into appropriate features that can be used to train a deep
3. learning model. In order to train any image in a deep learning model, feature extraction is
4. required.
5. I have divided the dataset into three parts namely train\_image , validation\_image and test\_image
6. containing 4855 ,1618 and 1618 images respectively.
7. 14

The image should be transformed into appropriate features that can be used to train a deep learning model. In order to train any image in a deep learning model, feature extraction is required. I have divided the dataset into three parts namely train\_image and test\_image containing 7281 and 810 images respectively.

The VGG-16 network has 16 weight layers, and the higher the number of layers, the better the feature extraction from images. The VGG-16 network has a simple architecture with 3\*3 convolutional layers and a max pooling layer in between to lower the image volume size. The image's last layer, which predicts classification, is deleted, and the image's internal representation just before classification is returned as a feature. The input image should have a dimension of 256\*256\*3, and this model collects image features and provides a 1-dimensional 4096 element vector.

**2.4 Define Model**

We've previously extracted the features from all of the images; now we shall use the Keras Model to determine the structure of the final model. It will be divided into two parts:

* Sequence Processor – The textual input will be handled by an embedding layer, which will be followed by the LSTM layer.
* Decoder - To make the final prediction, we will combine the output from the above two layers and process it via the dense layer. The number of nodes in the final layer will be equal to the size of our vocabulary.

Training the model To train the model, we will be using the 90% 0f 8091 training images by generating the input and output sequences in batches and fitting them to the model using model.fit\_generator() method. We also save the model to our models folder. This will take some time depending on your system capability.

Visual representation of the final model is given in the figure below:

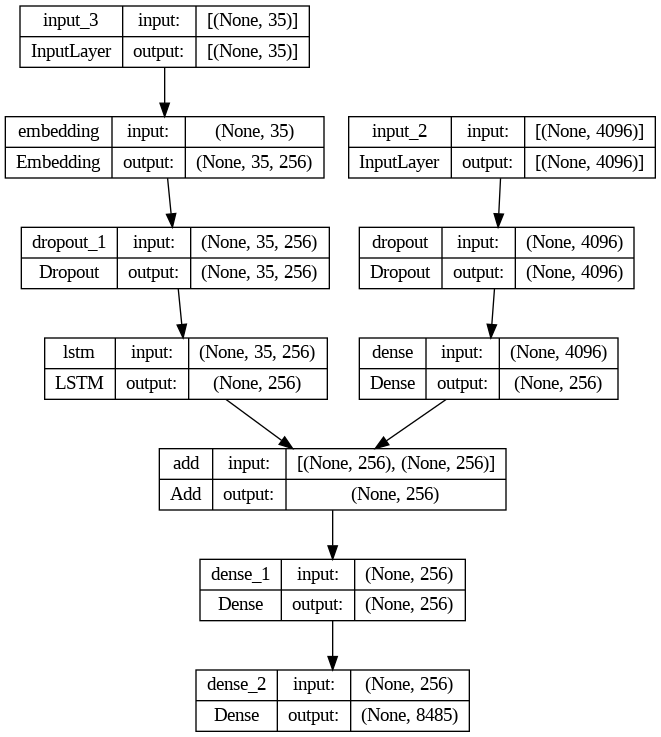


Fig2.2: Final Model Structure

# 3. CHAPTER

# RESULTS & DISCUSSION

We give this image to generate caption.



Fig3.1: Training image

After the tokenizer and passed through VGG16 Convolutional Neural Networks model this caption are selected for further modification with test data.

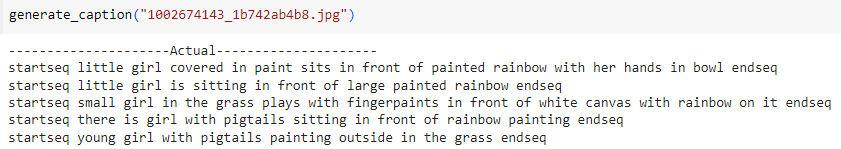


Fig3.2: Resultant captions

After model to train with testing data we finally generate the caption.



Fig3.3: Final caption

Let’s see the model with real test data.

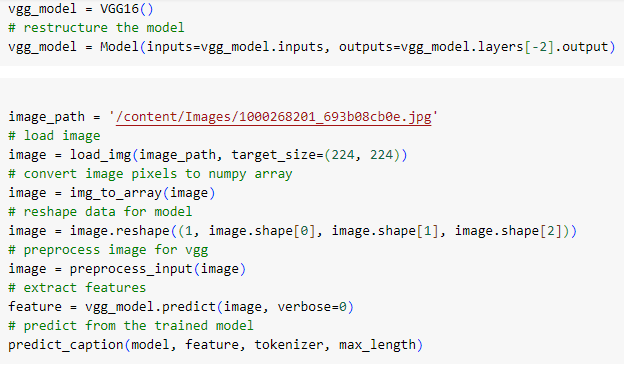


Fig3.4: Testing image



Fig3.5: Final caption

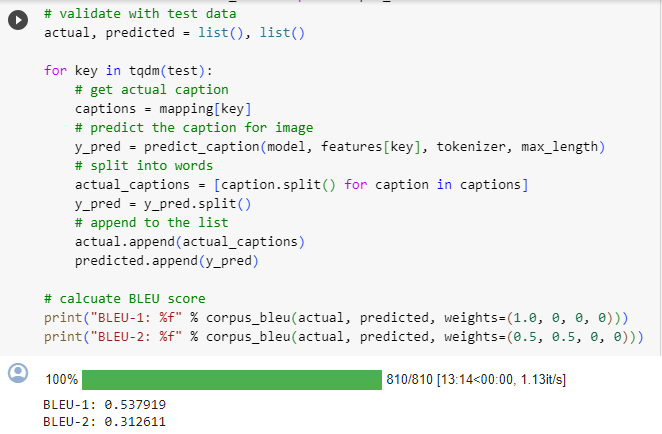


Fig3.6: Bleu Score

# CONCLUSION

The image caption generator demonstrates promise in bridging computer vision and natural language processing. While successful in automating visual content description, challenges remain in handling ambiguity, improving common-sense reasoning, and addressing biases. Further refinement is needed to enhance handling diverse visual scenarios and ensure consistent, high-quality outputs.

**FUTURE SCOPE**

* Real-Time Processing
* Explainability
* Multi-Modal Approaches
* User Feedback Integration
* Enhanced Semantic Understanding

# BIBLIOGRAPHY

* A. Aker and R. Gaizauskas. Generating image descriptions using dependency relational patterns. In ACL, 2010.
* D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, 2014.
* K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In EMNLP, 2014.
* J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In ICML, 2014.
* Show and Tell: A Neural Image Caption Generator <https://arxiv.org/pdf/1411.4555v2.pdf>
* Dataset <https://www.kaggle.com/datasets/adityajn105/flickr8k>