Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Unified Approach to Text Summarization and Image Captioning Using NLP and VGG16

[1]Dr.S.Velmurugan, [2]Dr.K.Subramanian, [3]Dr. B.Abirami

[1] Assistant Professor, Department of Computer Science with Data Analytics
Kongunadu Arts and Science College, Coimbatore, Tamilnadu, India, E-mail: ssvelmurugan@yahoo.com
[2] Sr.Assistant Professor, Department of IT and Analytics,
Xavier Institute of Management and Entrepreneurship (XIME),Bangalore, India,E-mail:subramanian@xime.org
[3] Assistant Professor, Department of Computer Science and Engineering
SRM Institute of Science and Technology, Ramapuram Campus, Chennai, Tamilnadu, India
E-mail: abivsb.sanrak@gmail.com

Abstract— Text summarization is essential for transforming extensive documents into brief, well-structured summaries, simplifying the process of accessing and managing large amounts of information. The proposed model offers a robust solution for summarizing large volumes of multimodal data by combining advanced text summarization and image captioning techniques. Leveraging Natural Language Processing (NLP), the text summarization module employs syntactic and semantic analysis, entity recognition, and sentiment analysis to extract vital information. By calculating word and sentence weights, it identifies and prioritizes the most critical content, ensuring concise and coherent summaries. Additionally, the model incorporates an automated image captioning system using the VGG16 architecture for feature extraction and a caption generator to produce descriptive and contextually relevant captions for images. This comprehensive approach enhances the accessibility and manageability of complex datasets, making it a valuable tool for information distillation in diverse applications.

Keywords—Image Captioning, Text Summarization, NLP, VGG16, LSTM.

I. INTRODUCTION

Interpreting and articulating the content of images is a sophisticated challenge that computers have historically struggled to master. Advances in computer vision and deep learning, however, have enabled the development of systems capable of analyzing images and producing contextually relevant captions. Despite the inherent difficulty in crafting precise image descriptions, modern techniques and algorithms facilitate the creation of captioning models that yield meaningful outputs. This paper presents a methodology for image captioning through the integration of deep neural networks, specifically utilizing Convolutional Neural Networks (CNN) for feature extraction and Long Short-Term Memory (LSTM) networks for sequential language generation. The objective is to transform an image input into a syntactically accurate and semantically rich sentence that effectively conveys its content.

Creating natural language descriptions for images, known as image captioning, has advanced remarkably in recent years due to innovations in computer vision and deep learning. Despite the intricate nature of the task, researchers have made significant progress in mimicking the human brain's ability to swiftly interpret and describe visual content. This progress has paved the way for machine learning models adept at analyzing and annotating images with precise and contextually appropriate captions. Nevertheless, constructing grammatically accurate and semantically meaningful sentences to describe images remains a challenging endeavor. The task of image captioning remains a significant challenge, requiring systems to recognize objects in an image and articulate appropriate descriptions, combining computer vision with natural language processing techniques [1]. Approaches to image captioning include template-based frameworks, retrieval-driven methods, and advanced deep learning techniques for novel caption generation. These methodologies aim to incorporate semantics into captions, enabling more detailed and contextually relevant descriptions. The complexity is heightened by the need to handle diverse semantic modes, which has led to innovative approaches like top-down and bottom-up generation techniques [2]. Recent advancements, such as the Context Sequence Memory Network (CSMN), have addressed issues like the vanishing gradient problem in recurrent neural networks, improving the ability

of models to retain long-term dependencies for better caption accuracy [3]. Image captioning also proves essential in managing the vast amounts of image data available today, facilitating content-based retrieval through generated captions [6]. Emerging architectures, such as those leveraging visual relationship graphs, further enhance caption quality, highlighting the continuous evolution of this domain [12].

I leverage the VGG16 model as a feature extractor to process images and extract meaningful features. VGG16 is a deep convolutional neural network (CNN) [3] known for its effectiveness in image classification tasks. By using the pre-trained VGG16 model, I can extract high-level features from images, which are then used as [3] input to an LSTM (Long Short-Term Memory) network. The LSTM network is responsible for generating captions for the images based on the extracted features. LSTMs are a type of recurrent [4] neural network (RNN) known for their ability to model sequential data and capture long-term dependencies. In the context of image captioning, the LSTM takes the extracted image features as input and [4] generates a sequence of words that form the caption describing the image. By combining the feature extraction capabilities of VGG16 with the sequential modeling power of LSTM, my project aims to generate accurate and contextually relevant captions for images, contributing to the advancement of image captioning research. In today's world, where we are inundated with vast amounts of information, extracting key insights efficiently can be challenging. Text summarization, an essential task in natural language processing (NLP), seeks to address this issue by condensing lengthy texts into concise summaries that retain the most critical information. This paper introduces a web-based text summarization application specifically designed for extractive summarization. It serves as a practical tool to help users more effectively extract and manage information.

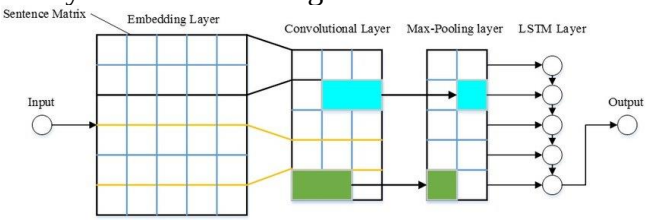


Fig 1: Architecture of CNN and LSTM

There are two primary approaches to automatic text summarization: 1) Abstractive and 2) Extractive. Extractive summarization involves selecting key sentences or information directly from the original text. This method utilizes linguistic or statistical features to identify the most important sentences. In contrast, abstractive summarization aims to understand the input text and rephrase it concisely while maintaining the core meaning. Extractive summarization is often referred to as sentence ranking and is typically divided into two phases: 1) Pre-processing and 2) Processing. This paper will focus on explaining the extractive text summarization method.

The motivation behind the development of this application arises from the growing need to address information overload. As highlighted by recent surveys on automatic text summarization techniques [15] [17], there is an increasing demand for automated solutions that can accurately and efficiently summarize textual content. Traditional manual summarization methods are labor-intensive and time-consuming, making them impractical for processing large volumes of text. As a result, automated summarization tools are becoming increasingly essential for navigating the vast amounts of information available online.

The approach taken in our application draws upon advanced NLP techniques, utilizing tools such as the SpaCy library [18] for text processing and feature extraction. Specifically, our application implements extractive summarization, where sentences are selected from the original text based on their importance scores, which are derived from word frequency and sentence weighting. While abstractive summarization techniques involve generating summaries through paraphrasing and synthesizing information, our focus is on providing users with concise summaries that accurately represent the original content.

II. BACKGROUND

Image captioning has been a challenging task in computer vision and natural language processing. Various methods have been proposed to address this challenge, leveraging deep learning techniques and neural networks. Convolutional Neural Networks (CNNs) are commonly used for feature extraction from images, with models like VGG16 being popular choices [1, 2, 3]. These extracted features are then fed into recurrent neural networks (RNNs), often Long Short-Term Memory (LSTM) units, to generate captions [1, 3]. Training these networks typically involves maximizing the likelihood of the generated

captions [4]. Yao et al. [5] employed a network trained on MSCOCO attributes to enhance performance on standard image captioning metrics. Anderson et al. [6] introduced an attention-based network model, while Aneja et al. [7] updated the language decoder in LSTM networks to increase diversity in generated captions. Wang et al. [8] also explored the use of CNNs in decoding captions. Vinyals et al. [9] fed the output of the fully connected layer of a CNN into an LSTM unit for caption generation. Xu et al. [10] used visual features from lower layers of a CNN and applied attention mechanisms to select spatial feature maps, which were then fed into an LSTM. Jia et al. [11] reworked the LSTM unit to incorporate linguistic options, providing direction to the LSTM for sentence generation. You et al. [12] and Wu et al. [13] advanced evaluation metrics such as CIDEr and BLEU for image captioning. Pan et al. [14] applied transferred semantic features and characteristics extracted from natural scenes and videos, enhancing these features with a shift unit before inputting them into an LSTM for captioning videos. These works collectively contribute to the ongoing development of image captioning models, improving the quality, diversity, and accuracy of generated captions.

Automatic text summarization is a key area of Natural Language Processing (NLP) that has been extensively explored, with numerous techniques and methodologies being developed over time. Researchers have conducted surveys to assess the strengths and limitations of various summarization methods. Recent evaluations have focused on the effectiveness of different summarization algorithms and their performance on benchmark datasets.

El-Kassas et al. [15] have provided a comprehensive overview of the field, covering key concepts, methodologies, and evaluation metrics used in text summarization research. They highlight recent advancements in both extractive and abstractive summarization techniques, discussing the challenges and opportunities in the field. This survey serves as an important resource for researchers and practitioners working in NLP.

Nenkova and McKeown [16] have also surveyed text summarization techniques, focusing on various approaches for generating summaries from textual content. Their survey addresses extractive techniques, which involve selecting sentences or passages from the original text, as well as abstractive techniques, which generate summaries by paraphrasing and synthesizing information. The survey offers valuable insights into the trade-offs associated with choosing the most appropriate summarization approach..

Ferreira et al. [19] evaluate various sentence scoring techniques for extractive text summarization, offering insights into the effectiveness of different scoring mechanisms in identifying the most salient sentences. Additionally, Forman presents empirical studies on feature selection metrics for text classification, which form the basis of many extractive summarization methods. These studies contribute to a deeper understanding of how specific features and scoring techniques can enhance the quality and relevance of automatically generated summaries.

III. METHODOLOGY

A. VGG16

The VGG-16 model, a significant breakthrough in convolutional neural networks (CNNs), has made a lasting impact on the field of computer vision. Proposed by Simonyan and Zisserman [4], the model consists of 13 convolutional layers, each followed by a Rectified Linear Unit (ReLU) activation function, which enhances the model's ability to capture complex, non-linear features from images [1], [2], [3]. These convolutional layers are vital for extracting hierarchical features, making VGG-16 particularly effective for image classification tasks. To reduce the spatial dimensions of feature maps while preserving critical information, VGG-16 employs max-pooling layers after every two convolutional layers [1]. This technique helps to summarize the most significant features within each region of the feature map, improving computational efficiency and mitigating overfitting.

Following the convolutional and max-pooling layers, VGG-16 contains three fully connected layers, each with 4096 neurons [1]. These layers refine the extracted features further, using ReLU activations to introduce non-linearity, which enables the model to learn complex patterns in the data. The final output layer of VGG-16 uses softmax activation to produce probabilities for each of the 1000 classes in the ImageNet dataset [1], allowing the model to make predictions about the image's content based on the

features it has extracted.

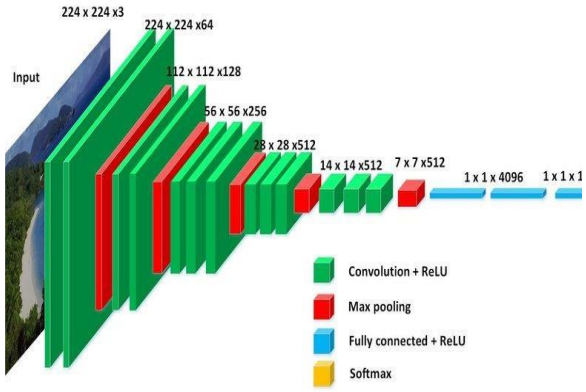


Fig 2: VGG-16 Architecture Overview

The success of VGG-16 has led to its widespread adoption in numerous computer vision applications, demonstrating the effectiveness of deep convolutional neural networks in extracting meaningful insights from images [1, 2, 3]. Its architecture has become a foundation for subsequent advancements in CNNs, marking the ongoing evolution of deep learning for image analysis and recognition tasks. The figure 2 would typically label these layers and show how the image progressively passes through different stages of feature extraction and classification.

B. LSTM

Long Short-Term Memory (LSTM) networks play a crucial role in image caption generators by enhancing the model's ability to generate sequential and contextually coherent descriptions. These networks serve as the decoding mechanism, receiving hierarchical image features extracted by [1] Convolutional Neural Networks (CNNs). The distinctive architecture of LSTMs, featuring memory cells and gates, allows them to capture and retain relevant information over extended sequences. In the context of image [2] captioning, LSTMs excel at understanding and generating textual descriptions by learning the sequential dependencies present in the image features. The memory cells within LSTMs store essential context, ensuring that the model can maintain relevance and coherence throughout [3] the caption generation process. By handling sequential data effectively, LSTMs contribute significantly to the success of image caption generators, enabling them to produce detailed and contextually nuanced textual descriptions aligned with the [4] visual content extracted from input images.

The functionality of LSTMs is governed by several key components, including the forget gate, [5] input gate, new information update, output gate, and SoftMax activation. The forget gate evaluates the relevance of the previous hidden state in the current context, deciding what information to retain or discard. The input gate determines the importance of new information at the current time step and regulates how much of it should be added to the current cell state. After evaluating the forget and input gates, the network [6] computes new information based on the current input and previous hidden state, updating the cell state accordingly. The output gate plays a crucial role in determining the information to output as the current hidden state, considering both the current input and updated cell state. Finally, the SoftMax activation function is applied to the hidden state to generate a probability distribution over the vocabulary, [7] representing the likelihood of each word in the vocabulary being the next word in the sequence. This distribution helps select the most probable word for generating the next part of the sequence, such as in image captioning tasks. As seen in Figure 1, captions are generated from the visual content extracted by the CNN and processed by the LSTM network.

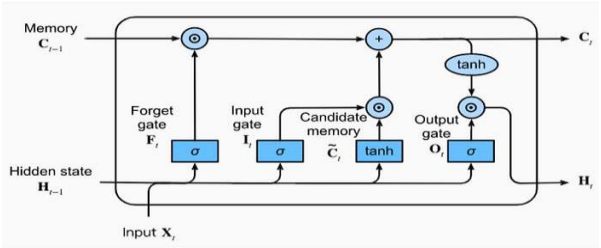


Fig: 3LSTM Architecture

A. PREREQUISITES

TensorFlow [11]: TensorFlow is a versatile open-source platform that provides comprehensive support for machine learning and artificial intelligence. It is compatible with various hardware platforms, has a user-friendly API, and enjoys strong community support, making it a top choice for developers.

Keras: Keras is an advanced neural network API written in Python that is user-friendly and simplifies the process of building deep learning models. It can run on top of frameworks like CNTK or Theano. Known for its simplicity and ease of use, Keras provides a range of pre-trained models that facilitate transfer learning and improvement of existing models.

Related Python Libraries:

NumPy: Useful for performing arithmetic operations and handling numerical data efficiently.
Pickle: Allows for serializing and deserializing Python objects, which is essential for saving model states and data.
tqdm: A library that provides a progress bar widget, useful for tracking long-term tasks or for importing large files.
Tokenizer: Tokenization is a crucial step in natural language processing (NLP) tasks such as text classification and sentiment analysis. It involves splitting text into tokens (words, phrases, or symbols), enabling the text to be processed in a format that can be understood by machine learning models.

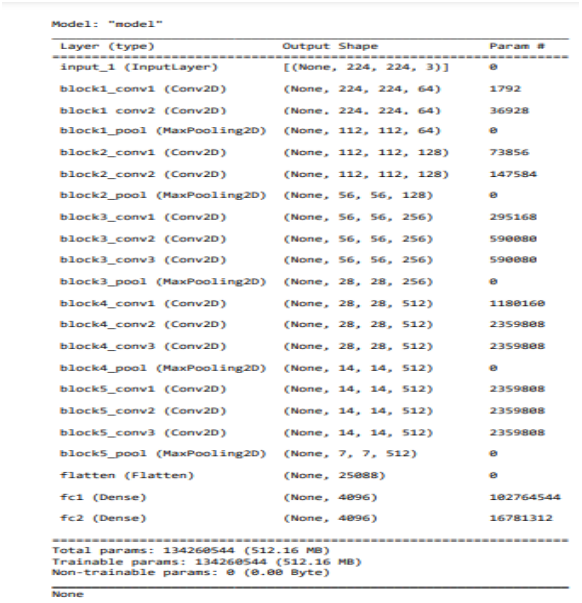
B. TEXT SUMMARY PROCESS:

Tokenization: Break the input text into individual terms or words. This step allows word-level analysis.
Stop Words Removal: Eliminate common words like "the", "is", "and", which don't contribute significantly to meaning, focusing on more meaningful keywords.
Part-of-Speech Tagging: Assign part-of-speech labels to each remaining word, providing context (e.g., noun, verb, adjective).
Frequency Calculation: Calculate the frequency of each keyword in the text. This measures the importance or relevance of the keyword.
Normalization: Normalize the frequency of each keyword by dividing it by the maximum frequency to bring all keyword weights to a similar scale.
Summation: Sum the weighted frequencies of all keywords to aggregate their importance in the text.
Sentence Extraction: Extract sentences with higher weighted frequencies, as they contain the most significant keywords and are considered more relevant for the summary.
This process helps select the most relevant sentences to construct a concise and meaningful summary from the original text.

II. PROPOSEDMODEL

The feature extraction phase for the image captioning project using the VGG-16 model begins with loading the pre-trained VGG-16 model through TensorFlow's Keras API. This model, renowned for its effectiveness in image classification tasks, has been trained on the vast ImageNet dataset. Once loaded, the model is restructured to capture the activations from the penultimate fully connected layer, just before the SoftMax layer. This restructuring allows the extraction of rich, high-level features from the input images. After restructuring, the images in the Flicker8k dataset are carefully processed to ensure they meet the required dimensions and formatting standards for seamless integration with the VGG-16 model. These preprocessing steps ensure that the images are optimally prepared for feature extraction. Subsequently, the preprocessed images are passed through the restructured VGG-16 model, progressing through its convolutional layers. As the images pass through each layer, they undergo transformations that capture increasingly complex features, helping to identify essential characteristics like objects, textures, and spatial relationships. During this process, the VGG-16 model extracts features from each image, storing them in a convenient dictionary format. This dictionary maps each image's unique identifier (ID) to its corresponding feature vector, allowing for easy retrieval during the caption

generation phase. This feature extraction process is fundamental to the overall image captioning task, as it provides the essential information needed to generate accurate and contextually relevant captions for the Flicker8k dataset. Ultimately, these extracted features serve as the backbone for the captioning model, ensuring that the generated captions are aligned with the content of the images.



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
 flatten (Flatten)           (None, 25088)             0
 fc1 (Dense)                 (None, 4096)              102764544
 fc2 (Dense)                 (None, 4096)              16781312
=================================================================
Total params: 134260544 (512.16 MB)
Trainable params: 134260544 (512.16 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Fig 4: Restructured VGG16 Model

A. Text Preprocessing and Tokenizing

The captions are converted to lowercase to ensure uniformity and consistency in the text. This step helps standardize the text and avoids discrepancies arising from different capitalizations. Next, any digits, special characters, and extra spaces are removed from the text. This cleaning process eliminates noise, ensuring that only relevant information remains. Additionally, start and end tags are added to each caption. These tags are crucial for the model to understand the beginning and end of a caption, providing context and structure to the generated descriptions.

After preprocessing, a Tokenizer is employed to tokenize the captions. This Tokenizer converts the preprocessed captions into sequences of integers, where each integer represents a unique word in the vocabulary. This tokenization step is essential because it prepares the captions for input into the model, facilitating the training and evaluation process. By converting text into a numerical format, the model can process and learn patterns from the captions effectively.

B. Encoder-Decoder

The encoder-decoder model for image captioning in the provided code is structured into two primary branches: one for processing image features and the other for processing textual sequences. In the image feature branch, the model accepts an input shape of (4096,), which corresponds to the feature vector extracted from a pre-trained VGG-16 model. This input is first passed through a dropout layer with a rate of 0.4 to mitigate overfitting, followed by a dense layer with 256 units and ReLU activation to further process the image features. The sequence feature branch, on the other hand, takes an input shape of (max_length,), where each element represents a tokenized word in the caption. This input is transformed into a dense representation using an embedding layer that converts each integer-encoded word into a 256-dimensional vector. A dropout layer with a rate of 0.4 is applied to prevent overfitting, followed by an LSTM layer with 256 units that captures sequential dependencies in the text. The decoder then combines the features from both the image and text branches via element-wise addition. This combined information is processed by a dense layer with 256 units and ReLU activation to refine the representation before passing through a final softmax output layer that generates a probability distribution over the vocabulary, determining the most likely next word in the caption. The model is compiled with categorical cross-entropy loss and the Adam optimizer. The overall architecture, including input/output shapes and layers, is illustrated using plot_model, providing a comprehensive view of how the image captioning process is structured. This working model is detailed in Figure 1.

C. Train The Model

Training the model involves using a data generator that processes the dataset in batches, which helps in preventing session crashes by efficiently managing memory usage. The generator encodes input sequences, splits them into input-output (X, y) pairs, and pads the sequences to ensure uniform length across the dataset. This step is essential to ensure that the model receives input data in a consistent format, especially for sequence-based tasks like image captioning

The model is compiled using categorical cross-entropy loss, which is suitable for multi-class classification problems, and the Adam optimizer, which helps in adjusting the learning rate dynamically during training. The model is trained over multiple epochs, where each epoch involves iterating over the entire training data to update the model's weights. Validation sets are used during the training process to determine the optimal epoch, ensuring the model's performance generalizes well to unseen data. After each epoch, the model's performance is evaluated, and the checkpoint with the highest validation accuracy is saved as the best model. This approach ensures that the model's parameters are optimized, avoiding overfitting and underfitting issues. After the model is trained, it can be used for generating captions for new images. The process involves extracting features from the input image and then predicting the next word in the sequence based on the context provided by previously predicted words. By leveraging the learned features and sequential dependencies, the model generates contextually relevant and accurate captions for a variety of images.

D. Model Evaluation

To evaluate the performance of the image captioning model, we use the BLEU score, specifically BLEU-1 and BLEU-2 scores, which are calculated using the corpus_bleu function from the NLTK library. The BLEU score is a metric commonly used in machine translation and captioning tasks to assess the quality of generated text. The BLEU-1 score measures the precision of the generated captions by comparing unigrams (single words) between the predicted captions and the ground truth captions. A higher BLEU-1 score indicates that the predicted words are closer to the actual words in the ground truth. The BLEU-2 score, on the other hand, evaluates the precision and recall of bigrams (pairs of adjacent words) in the predicted and actual captions. This score is more comprehensive as it considers the word order and structure within the captions. By calculating both BLEU-1 and BLEU-2 scores, we can quantitatively assess the model's performance. Higher BLEU scores suggest that the model is generating captions that are both accurate and contextually relevant, matching the ground truth more closely.

E. Deployment

The image captioning model has been successfully deployed using the Flask framework, providing an intuitive web interface for users to interact with the system. Through this web application, users can upload images, which are then processed by the deployed model to generate descriptive captions. Flask handles the backend processing, including image preprocessing and caption generation, ensuring a seamless and responsive user experience. The deployment of the model makes it accessible to a broader audience without the need for specialized software or technical expertise, offering a straightforward way for users to benefit from the image captioning functionality.

A. Generation of Caption

1. Add a Start Tag for the Generation Process: The generation process begins with the special start tag <start> to signify the beginning of the caption generation.
2. Encode Input Sequence: The image features are encoded together with the start tag <start>.
3. Pad the Sequence: The encoded sequence is padded to a fixed length, typically 20 tokens, to ensure uniformity for the model input.
4. Predict the Next Word: The model is used to predict the next word based on the encoded and padded sequence. For example, the model might predict the word "a."
5. Get Index with High Probability: The index of the predicted word "a" with the highest probability is selected.
6. Convert Index to Word: The index is then converted to its corresponding word using the vocabulary. In this case, "a" is chosen.
7. Stop if Not Found: The process continues until either an end tag <end> is generated, or a maximum

sequence length is reached.

8. Append Word as Input to Generate Next Word: The word "a" is appended to the input sequence to generate the next word in the caption.
9. Repeat Steps 4-8: Steps 4 to 8 are repeated, and the model continues generating the caption. For

instance, the predicted caption could be "sunny beach with people playing" (as shown in Figure 5).

10. Stop Once We Reach an End Tag: The process stops when the model generates the end tag <end>, or when the maximum sequence length is reached.

B. Text Summarization

The following steps outline the process for summarizing a given text using a natural language processing (NLP) model:

Step 1: Initialization

The Summarizer class is initialized with an NLP model that performs various text-processing tasks. This model helps to process the input text and supports the summarization workflow.

Step 2: Text Cleaning and Sentence Splitting

The _clean_text method processes the input text by replacing newline characters with spaces to ensure proper sentence splitting. This ensures the text is formatted correctly for further processing.

Once the text is cleaned, the NLP model is applied to the cleaned text to create a doc object. This doc object is then processed to extract individual sentences, which are stored in a list called sents.

Step 3: Calculating Word Weights

The _get_word_weights method processes the doc to extract lemmatized and lowercase words, while excluding stopwords and punctuation.

Word Counts: The Counter class is used to calculate the frequency of each word in the text.

Max Count: The maximum word count (max_count) is calculated to normalize word frequencies, ensuring that word frequencies are on a comparable scale.

Word Weights: The weight of each word is calculated as the ratio of its count to the maximum word count.

The formula to calculate word weights can be written as:

word_weights[word] = word_counts[word] ÷ max_count

Where:

word_counts: A dictionary containing word counts for each word in the document.

max_count: The maximum count among all word frequencies.

word_weights: A dictionary containing word weights, where each weight is the ratio of a word's count to the maximum count.

Step 4: Calculating Sentence Weights

The _get_sentence_weights method iterates through each sentence in the document and processes it in a manner similar to the word weight calculation. It uses the word weights calculated earlier to assign weights to individual sentences. The sentences with higher total weights are considered more important

and are likely to be included in the final summary.

Step 4: Calculating Sentence Weights (continued)
For each sentence, the lemmatized and lowercased words are extracted, and a score is assigned to the sentence based on the sum of word weights in the sentence divided by its length.
Sentence Weight Calculation:

The score for each word wiw_iwi in the sentence is determined using the pre-calculated word_weights.

The weight for the sentence is calculated as the sum of the individual word scores divided by the cleaned sentence length.

The formula for the sentence weight can be written as:

$$\text{Sentence weight} = \frac{\sum \text{score}(w_i)}{\text{cleaned\_length}}$$

Where:

- score(w_i) is the weight of each word in the sentence.
- cleaned_length is the length of the cleaned sentence (i.e., the number of words after cleaning).
- nnn is the number of words in the cleaned sentence.

This formula ensures that longer sentences, which may have more words, are normalized by their length, and sentences with higher-weighted words will have higher scores.

Step 5: Selecting Top Sentences
The _get_top_n_sentences method sorts the sentence weights in descending order, identifying the top-ranked sentences based on their weights.

- **Top Sentences**: The method selects the sentences with the highest weights.
- **Indices of Top Sentences**: The indices of the top sentences are stored in the top_sentences_idx list.
- **Retrieving Sentences**: The actual sentences corresponding to these indices are retrieved from the sents list and stored in the summary list.

Finally, the summary list, which contains the selected sentences, is joined together to create the final summarized text. This final output represents a condensed version of the original text, containing the most important information extracted from the top-ranked sentences.

IV. DATASET

The Flickr8k dataset consists of 8092 images, each accompanied by five unique English captions, divided into three subsets: 6000 images for training, 1000 for testing, and 1000 for validation. This dataset is specifically designed for image captioning tasks, offering diverse descriptions for each image to help in training models to generate accurate and contextually relevant captions. Preprocessing of the dataset involved several important steps, including the removal of punctuation, converting all text to lowercase for consistency, and eliminating numerical values to reduce noise. The dataset is organized into two directories—one containing the images and the other storing the corresponding text descriptions. The caption.txt file acts as the central reference, linking each image with its five corresponding captions, which makes it easy to pair images with their descriptions during training. This structured and well-preprocessed dataset serves as an ideal foundation for training models capable of generating natural language captions based on visual content.

V. RESULTS

Some of our results are shown in Fig. 6 and Fig. 7. We examined the performance of our improved image captioning model using a variety of image examples. The generated sentences effectively described the main elements of the images, capturing both significant and minor details. For instance, the model accurately described an image with two dogs playing. However, there were some inaccuracies, such as misidentifying a road and a rainbow in another image. These errors are understandable, as similar objects can be challenging to differentiate, even for humans. In terms of grammar, the generated sentences demonstrated good adherence to grammatical rules. However, our model's attention mechanism had limitations. It could only recognize the most significant parts of the images, with the attention weights remaining the same at each step. This is because the features are input at the first step of the LSTM, providing enough information to generate a reasonable sentence. Our future models aim to address this limitation by refining the attention mechanism to improve its ability to focus on different parts of the image at each step of the caption generation process.

Fig 6: Generated Caption 1

Fig 6 figure illustrates the generated caption for an image, showcasing the model's ability to describe the key elements present in the visual content. For instance, in this image, the caption generated by the model could be something like, "Two dogs are playing in a park," effectively conveying the main subjects and their actions within the image.

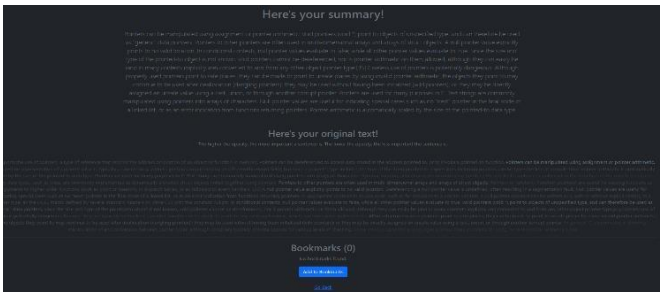


Fig 7: Generated Caption 2

Fig 7 presents another example of a generated caption for a different image. The caption effectively captures the main elements of the image, though some inaccuracies may still arise due to the limitations in distinguishing between similar objects or details. For example, if the image contains a rainbow and a road, the model might mistakenly identify one for the other. Despite this, the generated caption provides useful

context, such as describing "A road with a rainbow in the sky," although some fine details may not be captured perfectly.

The system allows users to specify the desired number of sentences for the summary using the "Number of Sentences" field. This provides flexibility in tailoring the summary length according to user preferences. Additionally, users can bookmark the generated summary by providing a label for easy reference, ensuring that important summaries are stored for later access. Once the summarization process is complete, the summarized text is displayed for review. To enhance user experience, the system also identifies and highlights the top significant words from the original text. This feature helps users quickly grasp the key points of the content, providing visual cues that emphasize the most relevant information. The overall design not only simplifies the summarization process but also enables a more interactive and user-friendly approach to content management and analysis.

Fig 8: User input

Fig 9: User

V. CONCLUSION

The proposed image captioning model leverages a deep learning architecture that combines an encoder-decoder framework with attention mechanisms. The encoder processes image features using a pre-trained Convolutional Neural Network (CNN), while the decoder generates captions using a Long Short-Term Memory (LSTM) network. The model integrates both image and text data, ensuring that the generated captions are contextually accurate and relevant to the visual content. Additionally, an attention mechanism is incorporated to allow the model to focus on specific parts of the image while generating each word in the caption, enhancing the quality and specificity of the output. This architecture is designed to handle a wide variety of images, capturing both prominent and subtle details for effective caption generation. The model is trained on the Flickr8k dataset and is capable of producing captions that describe the visual content in a way that is helpful for applications such as accessibility tools for the visually impaired.

REFERENCE

[1] Hossain, M. Z., Sohel, F., Shiratuddin, M. F., & Laga, H. (2018). A comprehensive survey of deep learning for image captioning. *arXiv:1810.04020v2 [cs.CV]*. https://doi.org/10.48550/arXiv.1810.04020

[2] You, Q., Jin, H., Wang, Z., Fang, C., & Luo, J. (2016). Image captioning with semantic attention. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4651–4659. https://doi.org/10.1109/CVPR.2016.503

[3] Park, C., Kim, B., & Kim, G. (2017). Attend to you: Personalized image captioning with context sequence memory networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 895–903. https://doi.org/10.1109/CVPR.2017.97

[4] Johnson, J., Karpathy, K., & Fei-Fei, L. (2016). DenseCap: Fully convolutional localization networks for dense captioning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4565–4574. https://doi.org/10.1109/CVPR.2016.493

[5] Aneja, J., Deshpande, A., & Schwing, A. G. (2018). Convolutional image captioning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5561–5570. https://doi.org/10.1109/CVPR.2018.00582

[6] Srivastava, G., & Srivastava, R. (2018). A survey on automatic image captioning. *Mathematics and Computing: ICMC 2018. Communications in Computer and Information Science*, 834. Springer. https://doi.org/10.1007/978-3-319-96678-9_62

[7] Wang, H., Zhang, Y., & Yu, X. (2020). An overview of image caption generation methods. *Computational Intelligence and Neuroscience*, 2020, Article ID 3062706, 13 pages. https://doi.org/10.1155/2020/3062706

[8] Sharma, P., Ding, N., Goodman, S., & Soricut, R. (2018). Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2556–2565. https://doi.org/10.18653/v1/P18-1246

[9] Sairam, G., Mandha, M., Prashanth, P., & Swetha, P. (2021). Image captioning using CNN and LSTM. *4th Smart Cities Symposium (SCS 2021)*, Online Conference, Bahrain, 274–277.

[10] Agrawal, V., Dhekane, S., Tuniya, N., & Vyas, V. (2021). Image caption generator using attention mechanism. *12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1–6. https://doi.org/10.1109/ICCCNT51525.2021.9579758

[11] Shi, Z., Zhou, X., Qiu, X., & Zhu, X. (2020). Improving image captioning with better use of captions. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7454–7464. https://doi.org/10.18653/v1/2020.acl-main.670

[12] Geetha, G., Kirthigadevi, T., Godwin Ponsam, G., Karthik, T., & Safa, M. (2020). Image captioning using deep convolutional neural networks (CNNs). *Journal of Physics: Conference Series*, 1712, 012015. https://doi.org/10.1088/1742-6596/1712/1/012015

[13] Wang, C., Yang, H., Bartz, C., & Meinel, C. (2016). Image captioning with deep bidirectional LSTMs. *Proceedings of the 24th ACM International Conference on Multimedia (MM '16)*. https://doi.org/10.1145/2988257.2988611

[14] Poddar, A. K., & Rani, R. (2023). Hybrid architecture using CNN and LSTM for image captioning in Hindi language. *International Conference on Machine Learning and Data Engineering*, Procedia Computer Science, 218, 686–693. https://doi.org/10.1016/j.procs.2023.06.092

[15] El-Kassas, W. S., Salama, C. R., Rafea, A. A., & Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165, 113990. https://doi.org/10.1016/j.eswa.2020.113990.

[16] Nenkova, A., & McKeown, K. (2012). A survey of text summarization techniques. In *Mining Text Data* (pp. 43-76). Springer. https://doi.org/10.1007/978-1-4614-3223-4_3

[17] Gambhir, M., & Gupta, V. (2017). Recent automatic text summarization techniques: A survey. *Artificial Intelligence Review*, 47(1), 1–66. https://doi.org/10.1007/s10462-016-9485-6

[18] Explosion AI. (n.d.). *SpaCy: Industrial-strength natural language processing in Python*. Retrieved from https://spacy.io/

[19] Ferreira, R., De Souza Cabral, L., Lins, R. D., Silva, G. P. E., Freitas, F., Cavalcanti, G. D. C., Lima, R., Simske, S. J., & Favaro, L. (2013). Assessing sentence scoring techniques for extractive text summarization. *Expert Systems with Applications*, 40(14), 5755–5764. https://doi.org/10.1016/j.eswa.2013.04.035

[20] Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3(Mar), 1289–1305. https://www.jmlr.org/papers/volume3/forbes03a/forbes03a.pdf

[21] Erkan, G., & Radev, D. R. (2004). LexRank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22, 457–479. https://doi.org/10.1613/jair.1522

[22] Cao, Z., Wei, F., Dong, L., Li, S., & Zhou, M. (2015). Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI* (pp. 2153-2159). https://doi.org/10.1609/aaai.v29i1.9405.