PROJECT REPORT

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

8. PERFORMANCE TESTING

8.1 Performance Metrics

9. RESULTS

9.1 Output Screenshots

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

Project Overview:

The Image Caption Generation project aims to leverage advanced deep learning techniques to automatically generate descriptive captions for images. In a world inundated with visual data, this project addresses the crucial need for AI systems to comprehend and articulate the content of images. The model, designed using state-of-the-art neural network architectures, learns intricate patterns and relationships within image features and corresponding textual descriptions. The chosen methodology involves training the model on a diverse dataset, allowing it to capture a broad spectrum of visual concepts. Through an iterative process of fine-tuning and optimization, the model becomes adept at associating visual content with appropriate language constructs. The project not only explores the technical intricacies of deep learning but also delves into the interdisciplinary realm of computer vision and natural language processing. The successful implementation of this project promises to enhance applications in image recognition, accessibility, and human-computer interaction, contributing to the evolution of AI systems capable of nuanced understanding and communication in the visual domain.

Purpose:

The purpose of image caption generation lies in bridging the gap between visual content and natural language understanding, fostering more effective human-computer interaction and enhancing accessibility to visual information. Several key purposes of image caption generation include:

- **Accessibility:** Image caption generation enables accessibility for individuals with visual impairments by providing textual descriptions of visual content. This promotes inclusivity and allows a wider audience to comprehend and engage with images on the internet, in educational materials, and various applications.
- **Content Indexing and Retrieval:** Image captions serve as textual representations that facilitate content indexing and retrieval in databases or search engines. This is particularly valuable for organizing large datasets of images, making it easier to search for and find relevant visual information.
- **Human-Machine Communication:** Generating captions for images allows machines to communicate with humans in a more intuitive and natural manner. This has applications in human-robot interaction, virtual assistants, and other AI systems where conveying information through both visual and textual modalities is advantageous.

- **Enhanced User Experience:** In applications such as social media, e-commerce, and educational platforms, image captioning enhances the user experience by providing context and information about images. This can lead to increased user engagement and satisfaction.
- **Automatic Image Annotation:** Image caption generation aids in automatically annotating images with meaningful descriptions. This is valuable for organizing and categorizing images in multimedia databases, making it easier for users to understand and manage visual content.
- **Educational Tools:** Image captioning can be utilized in educational settings to create interactive learning materials. Automatically generated captions can provide additional context to images, aiding in comprehension and knowledge retention.
- **Innovations in AI and Machine Learning:** Image caption generation serves as a challenging problem in the field of AI and machine learning. Developing effective models for this task contributes to advancements in computer vision, natural language processing, and the broader intersection of these domains.

Overall, image caption generation serves as a powerful tool to enhance the interpretability and utility of visual data in diverse applications, fostering a more seamless integration of artificial intelligence into our daily lives.

2. LITERATURE SURVEY

2.1 Existing problem:

The absence of image caption generation applications leaves several challenges unaddressed in the current technological landscape. One notable issue is the limited accessibility of visual content for individuals with visual impairments. Without image captioning, these individuals face barriers in comprehending and engaging with the vast amount of visual information available online and in various applications, hindering their ability to access educational, informational, and recreational content.

Furthermore, the lack of image caption generation applications contributes to challenges in content organization and searchability. Without automatically generated textual descriptions, databases and search engines struggle to index and retrieve relevant visual information efficiently. This hampers the user experience for individuals seeking specific visual content, leading to increased reliance on manual tagging and categorization.

The absence of image captioning also impacts human-computer interaction, as the seamless integration of visual and textual information is limited. In applications such as social media, where images play a significant role, the inability to automatically generate captions restricts the ability of AI systems to understand and interpret visual content, reducing the overall effectiveness of these platforms.

References:

[1] Sharma, Grishma and Kalena, Priyanka and Malde, Nishi and Nair, Aromal and Parkar, Saurabh, Visual Image Caption Generator Using Deep Learning (April 8, 2019). 2nd International Conference on Advances in Science & Technology (ICAST) 2019 on 8th, 9th April 2019 by K J Somaiya Institute of Engineering & Information Technology, Mumbai, India, Available at SSRN: https://ssrn.com/abstract=3368837 or http://dx.doi.org/10.2139/ssrn.3368837

[2] N. K. Kumar, D. Vigneswari, A. Mohan, K. Laxman and J. Yuvaraj, "Detection and Recognition of Objects in Image Caption Generator System: A Deep Learning Approach," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 2019, pp. 107-109, doi: 10.1109/ICACCS.2019.8728516.

[3] Raypurkar, M., Supe, A., Bhumkar, P., Borse, P., & Sayyad, S. (2021). Deep learning based image caption generator. *International Research Journal of Engineering and Technology (IRJET)*, *8*(03).

Problem Statement Definition:

The task of image caption generation is a complex problem at the intersection of computer vision and natural language processing. In this project, the focus is on developing a deep learning solution that can automatically generate descriptive and contextually relevant captions for a diverse range of images. Leveraging state-of-the-art pre-trained convolutional neural networks (CNNs) for image feature extraction and incorporating recurrent neural networks (RNNs) or transformer-based architectures for caption generation, the objective is to create a model that not only recognizes the content within images but also articulates it fluently in natural language. The challenges include handling image ambiguities, variations in object scale and orientation, and ensuring the model's ability to produce diverse and creative captions.

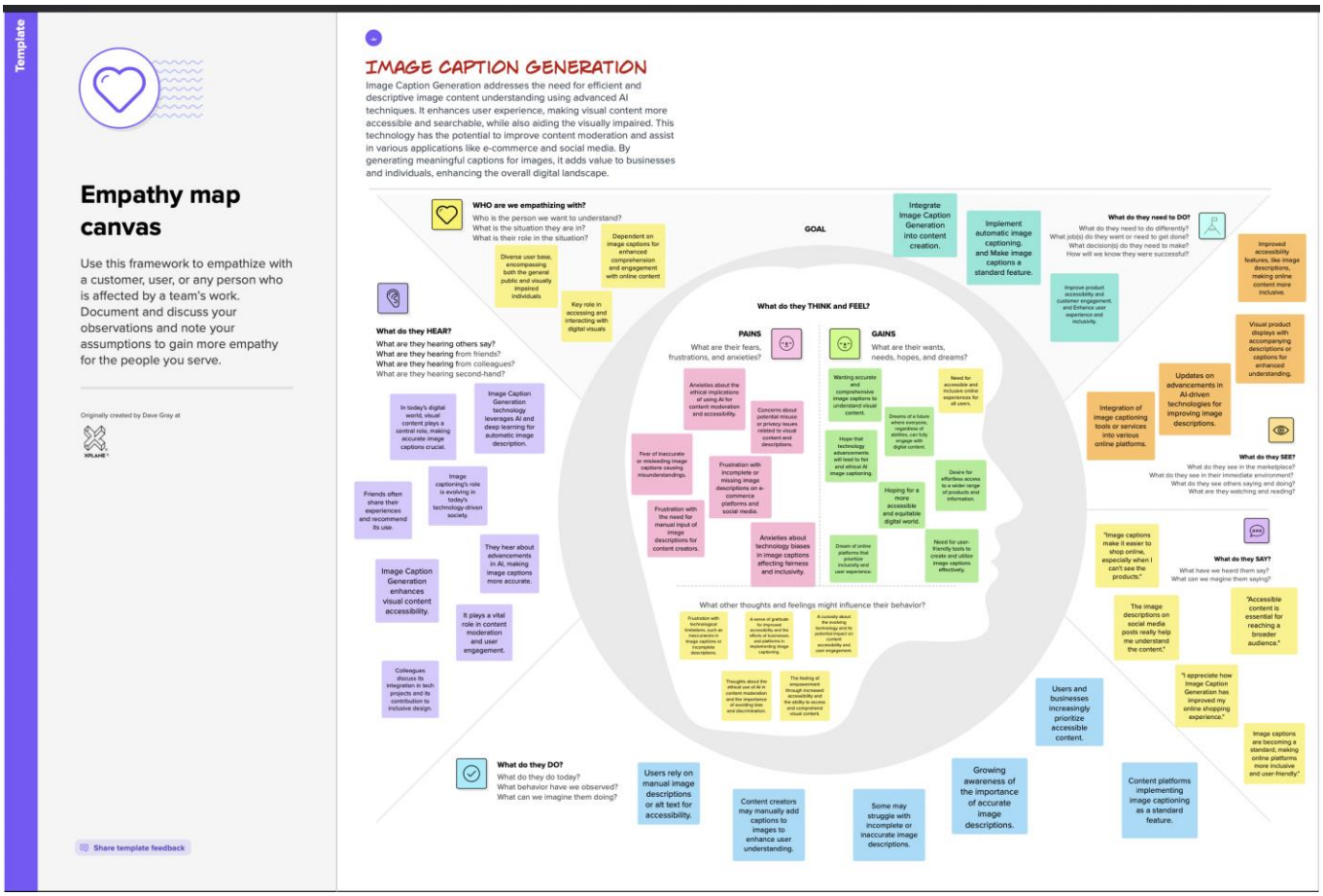
To achieve success, the project involves curating a comprehensive dataset with images and corresponding human-generated captions, which will be split into training, validation, and testing sets. The training process will utilize advanced optimization techniques and loss functions, and the model's performance will be evaluated using established metrics such as BLEU, METEOR, and CIDEr. Additionally, there is an optional component involving the development of a user-friendly interface, allowing users to upload images and receive generated captions. Success will be measured not only by the model's quantitative

performance but also by its ability to generate linguistically accurate and contextually appropriate captions for a variety of images, providing a meaningful contribution to the field of image caption generation.

3. IDEATION & PROPOSED SOLUTION

Empathy Map Canvas: The Image Caption Generator Empathy Map is a visual tool designed to illuminate the user's behaviors and attitudes when interacting with image captioning technology. By fostering a deeper understanding of the user's perspective, goals, and challenges, this canvas facilitates the creation of more effective solutions. It provides valuable insights into the user's experience, enabling teams to tailor image caption generators to better meet user needs and preferences.

:



Ideation & Brainstorming:

The Image Caption Generator Brainstorm & Idea Prioritization Template fosters a collaborative and open environment for creative thinking within a team. Encouraging the exploration of unconventional ideas, this template values quantity, allowing participants to build upon each other's contributions. The

collaborative nature of the template enables teams to generate a wealth of creative solutions.

Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we effortlessly empower computers to comprehend and translate visual content into coherent natural language, leveraging deep learning techniques like Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM), to provide accurate and human-like image descriptions, thus bridging the crucial gap between visual content and textual understanding in the era of digital communication and content-driven interactions?

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

Brainstorm

Write down any ideas that come to mind that address your problem statement.

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

10 minutes

BHAVANI

- User-Friendly Image Selection
- Caption Customization Options
- Caption Translator

BHAVANA

- Integration with Popular Image Platforms
- Caption Filters
- Everyday Integration

BHAVANI

- Smart Learning from User Feedback
- Caption Emoji Fusion
- User-Friendly Adjustments

BHAVANA

- Multi-Modal Input Support
- Caption in Colors
- Educational Caption Insights

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

20 minutes

Cluster 1 :User Interaction & Customization

- User-Friendly Image Selection
- Caption Customization Options
- User-Friendly Adjustments
- Smart Learning from User Feedback

Cluster2 : Integration & Accessibility:.

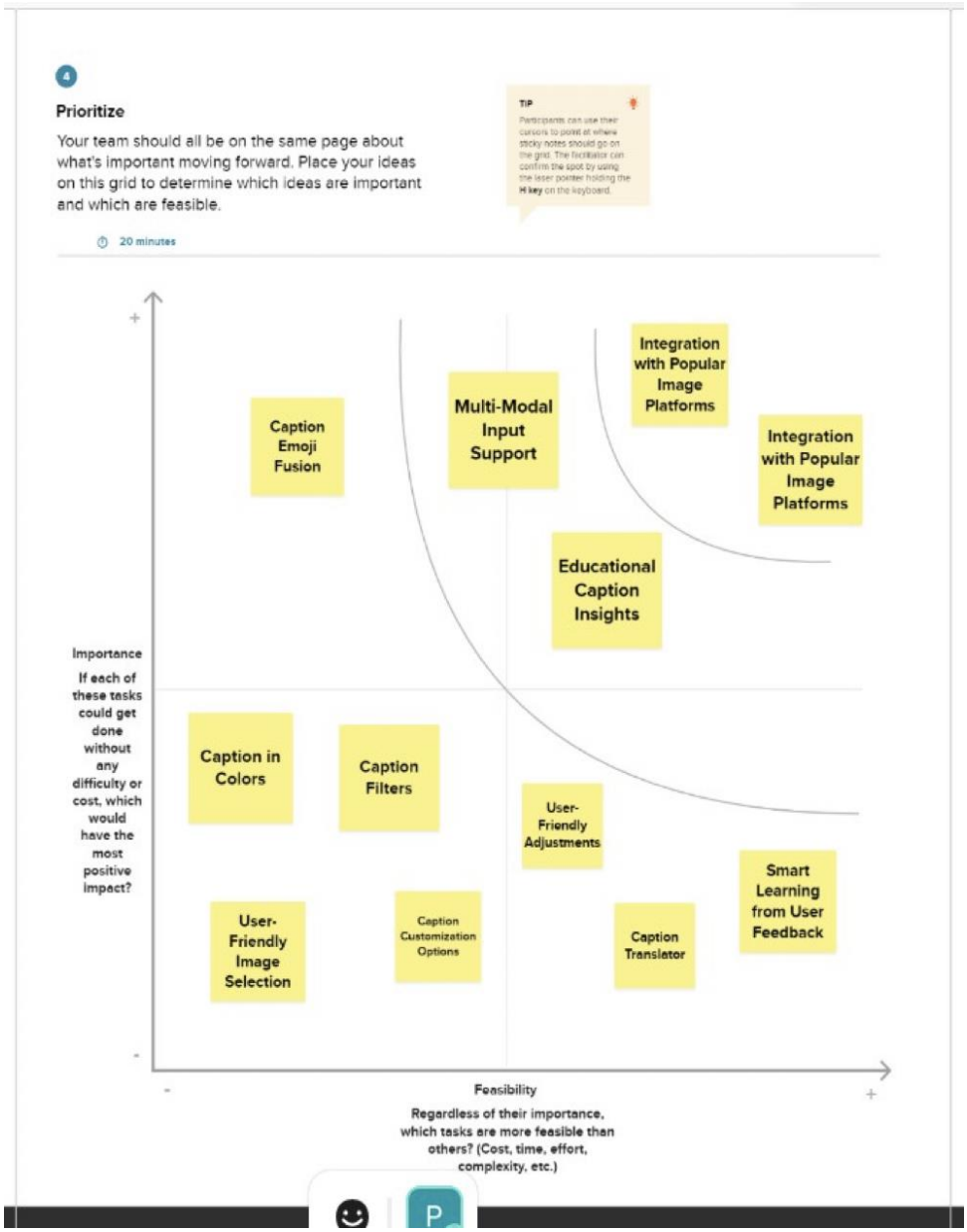
- Integration with Popular Image Platforms
- Everyday Integration

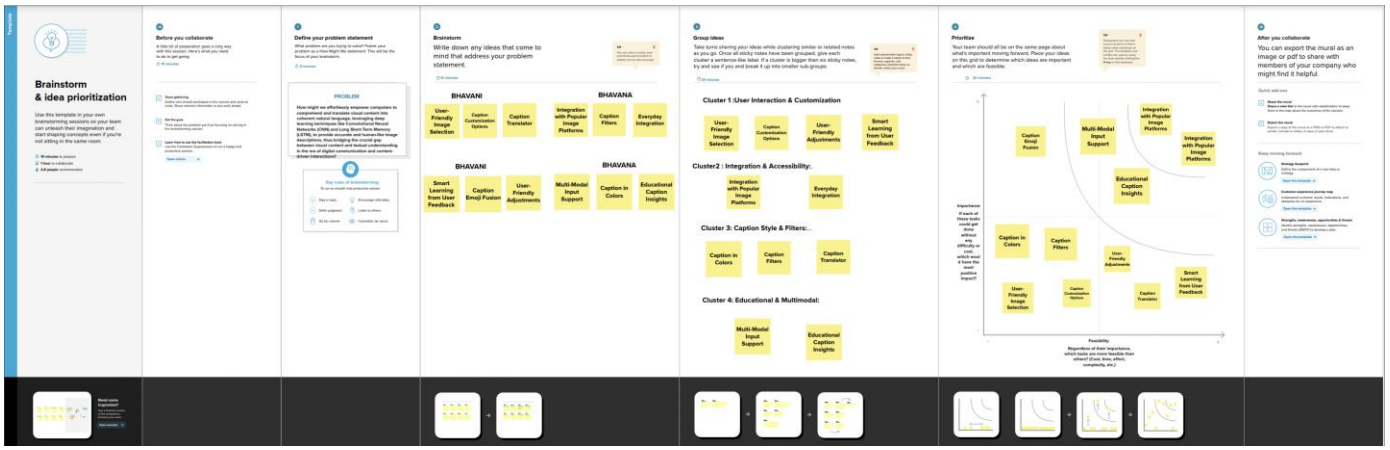
Cluster 3: Caption Style & Filters:...

- Caption in Colors
- Caption Filters
- Caption Translator

Cluster 4: Educational & Multimodal:

- Multi-Modal Input Support
- Educational Caption Insights


4
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.
20 minutes
Caption Emoji Fusion
Multi-Modal Input Support
Integration with Popular Image Platforms
Integration with Popular Image Platforms
Educational Caption Insights
Caption in Colors
Caption Filters
User-Friendly Adjustments
Smart Learning from User Feedback
User-Friendly Image Selection
Caption Customization Options
Caption Translator
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)


Brainstorm & idea prioritization
Before you collaborate
Define your problem statement
Brainstorm
Group ideas
Prioritize
After you collaborate

4. REQUIREMENT ANALYSIS

4.1 Functional requirement:

FR.NO	FUNCTIONAL REQUIREMENT	SUB REQUIREMENT
FR-1	Image Processing	• The system should be able to accept and process various image formats (e.g., JPEG, PNG). • Image pre-processing techniques should be implemented to enhance feature extraction.
FR-2	Caption Generation	• Generate coherent and contextually relevant captions for input images.
FR-3	Model Training	• The system should have the capability to train and fine-tune the image captioning model. • It should support the use of pre-trained models for faster deployment.
FR-4	Integration with Image Sources	• The ability to integrate with various image sources such as local files, URLs, and camera inputs.
FR-5	User Interface	• Provide a user-friendly interface for users to interact with the system.

		• Support both web-based and mobile interfaces.
FR-6	Scalability	• The system should be scalable to handle many requests concurrently.
FR-7	Error Handling	• Implement error handling mechanisms to gracefully manage issues such as invalid input or model failures.
FR-8	Accessibility	• Ensure the user interface is accessible to users with disabilities

4.2 Non-Functional requirements:

NFR.NO	**NON-FUNCTIONAL REQUIREMENT**	**DESCRIPTION**
NFR-1	Performance	The system should generate captions within a reasonable time frame, even for large and high-resolution images.
NFR-2	Accuracy	The generated captions should be accurate and contextually relevant to the content of the input image.
NFR-3	Reliability	The system should be reliable and available for use with a minimal downtime.
NFR-4	Security	Implement security measures to protect user data and prevent unauthorized access to the system.

NFR-5	Scalability	The system should be designed to scale horizontally to handle an increasing number of users and images.
NFR-6	Compatibility	Ensure compatibility with a variety of devices, browsers, and operating systems.
NFR-7	Usability	The user interface should be intuitive and easy to use, requiring minimal training for end-users.
NFR-8	Maintainability	The codebase should be well-organized and documented to facilitate ease of maintenance and future enhancements.

5. PROJECT DESIGN

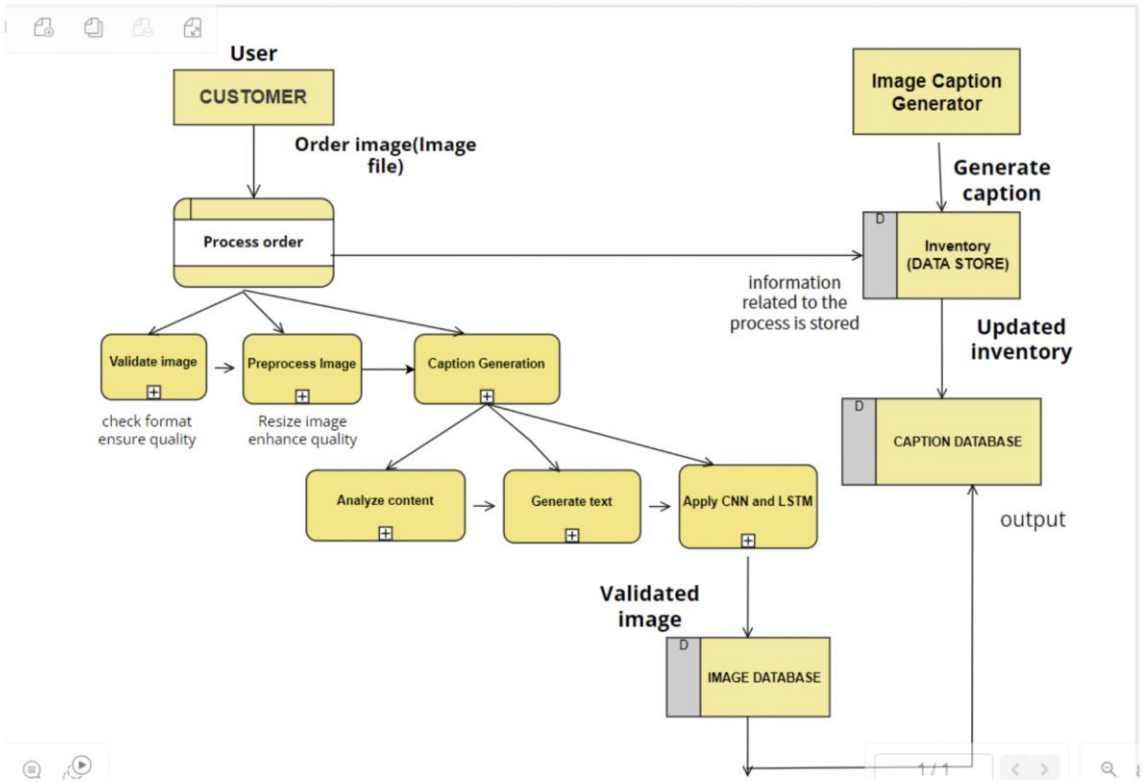
Data Flow Diagrams & User Stories

User Stories

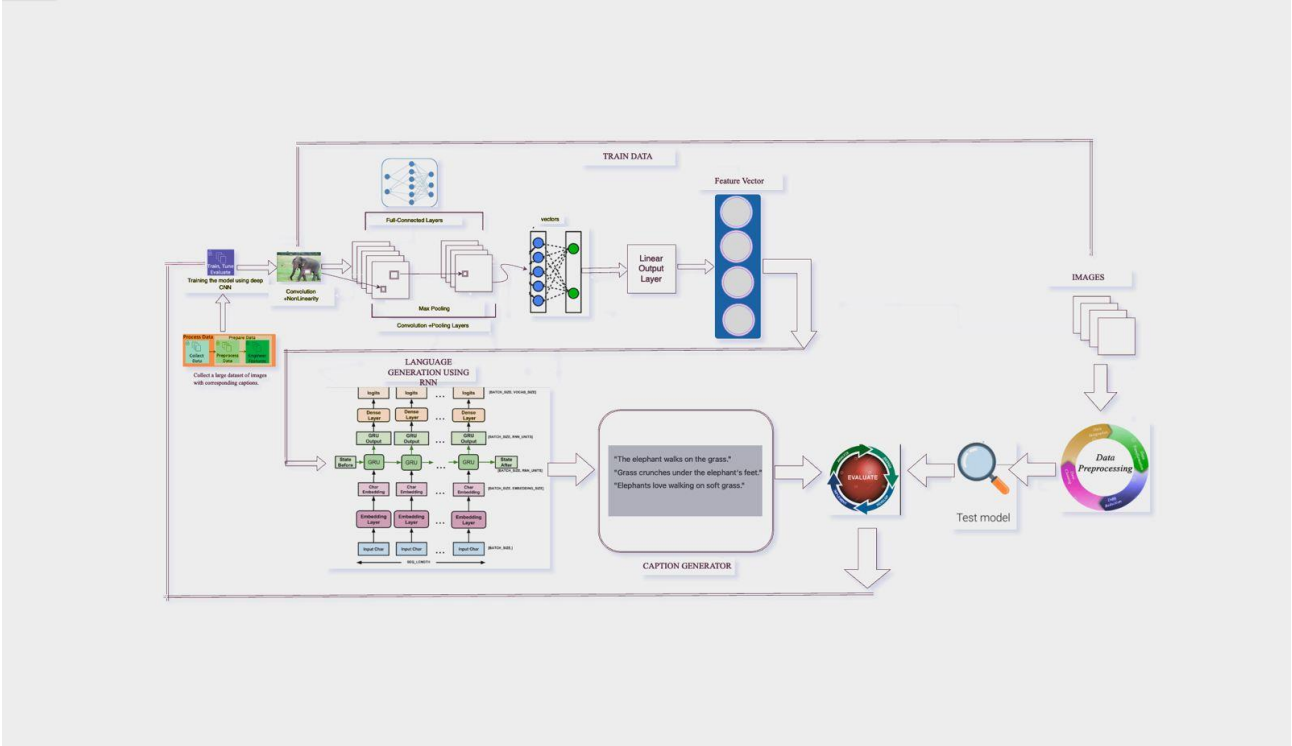
Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Content Creator (Photographer)	Image Upload	USN-1	As a content creator, I want to upload multiple images at once to save time and streamline the captioning process.	I can select and upload multiple images through a user-friendly interface.	High	Sprint-1
	Custom Keywords	USN-2	As a content creator,I want to Input custom keywords during caption generation so that the Model incorporates keywords for more personalized captions.	I receive Shared images and retained captions with enhanced custom keywords.	Medium	Sprint-2
Casual User (Mobile)	Caption Generation	USN-3	As a casual user, I want to effortlessly generate captions for my photos with a simple mobile-friendly interface.	I receive creative and relevant captions for each uploaded image.	High	Sprint-2
	Emoji-powered Caption Generation	USN-4	As a casual user, I want to add emojis to generated captions for a more expressive output.	I receive creative and emoji-enriched captions for each uploaded image.	Medium	Sprint-3
Administrator	User Management (Epic)	USN-5	As an administrator, I want to manage user accounts and roles, ensuring a secure and organized platform.	I can modify,add or remove user accounts, assign roles and ensure	High	Sprint-1

				user data security.		
Everyday User	Share Captions	USN-6	As an Everyday User,I want to Enable easy sharing of captioned images through messaging or social media.	I can share captions through social media.	Low	Sprint-4
	Mood-based Captions	USN-7	As an Everyday User,I want a mood selection option during caption generation.Generated captions should align with the chosen mood or emotion.	I receive creative mood-based captions.	Medium	Sprint-4
Social Media Manager	Seamless Social Media Integration	USN-8	As a social media manager, I want seamless integration with popular social media platforms for efficient content sharing.	I can directly share generated captions to social media platforms without additional steps.	Medium	Sprint-5
AI Enthusiast	Advanced Caption Settings Exploration	USN-9	As an AI enthusiast I want to fine-tune advanced settings for the caption generation model, exploring the depths of customization.	I can adjust parameters such as style, tone, and language to tailor the caption output to my preferences.	High	Sprint-5
Accessibility User (Web)	Accessible Image Caption Generator	USN-10	As an Accessibility User, I need the image caption generator to be accessible and usable with screen readers and other assistive technologies.	I can see that the image caption generator complies with accessibility standards, allowing smooth usage for users with disabilities.	High	Sprint-5



5.2 Solution Architecture



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture:

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2 :

The I**mage Captioning project** leverages deep learning technologies, employing Convolutional Neural Networks (**CNN**s) for image feature extraction and Long Short-Term Memory (**LSTM**) networks for sequential language generation. The user interacts through a user-friendly web interface built with HTML, CSS, and JavaScript, while the application's logic is implemented in Python. Data storage involves both traditional databases, like MySQL, and cloud-based solutions, such as IBM Cloudant.
The overall architecture ensures scalability, security, and performance, making it capable of generating creative and relevant captions for uploaded images.

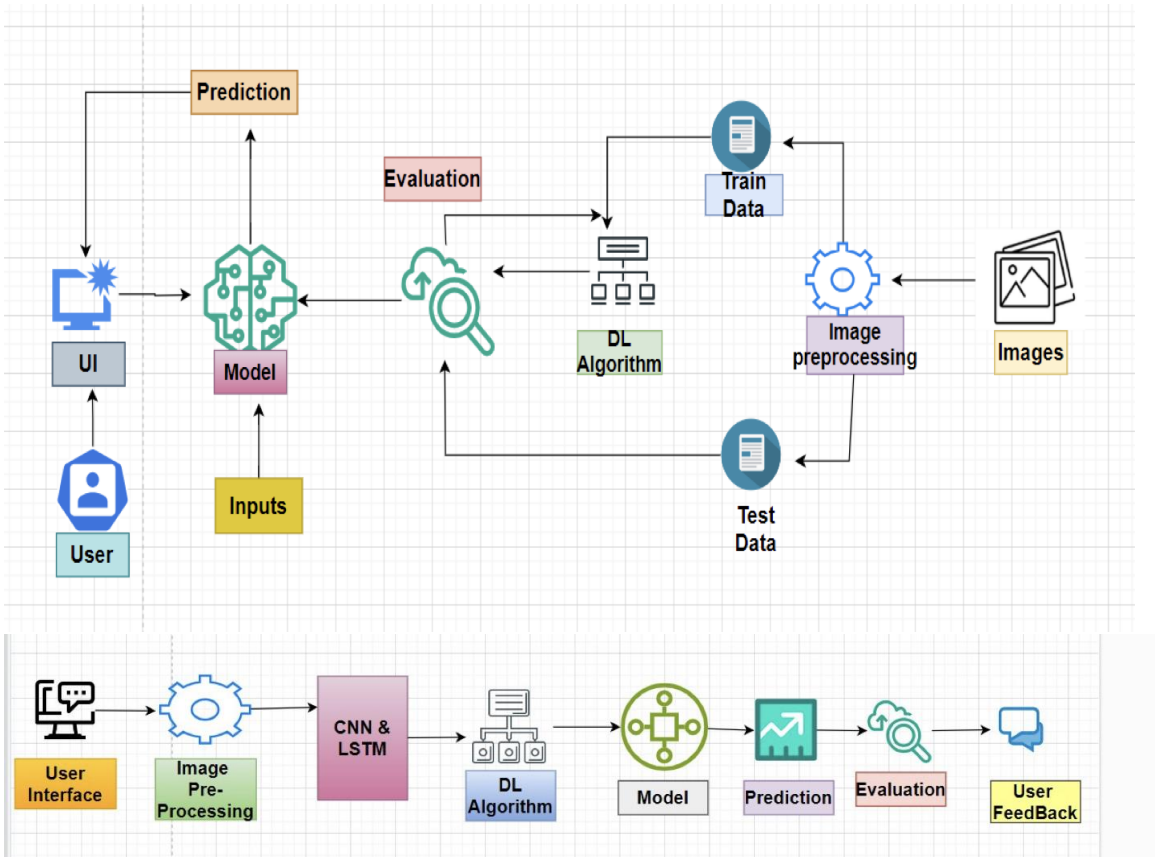


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How users interact with the image captioning application (UI/UX design)	Web UI, Mobile App, HTML, CSS, JavaScript / React Js
2.	Image Preprocessing	Preprocesses images before feeding to the model	OpenCV, PIL
3.	CNN & LSTM Model	Deep learning model for image captioning	TensorFlow, PyTorch
4.	Image Caption Generation Logic	Logic for generating captions based on the input images	Python, Natural Language Processing libraries
5.	Database	Storage for data related to images and generated captions	MySQL, NoSQL (e.g., MongoDB)

6.	Cloud Database	Cloud-based database service for scalability	IBM Cloudant, AWS DynamoDB
7.	File Storage	Storage for image files	Cloud Object Storage, Local Filesystem
8.	External API-1	Integration with external services for additional data	IBM Watson Visual Recognition API, etc.
9.	External API-2	Integration with external services for additional data	Language Translation API, etc.
10.	User Feedback	Collects feedback from users on generated captions	Web UI, Mobile App
11.	Infrastructure (Server / Cloud)	Deployment and hosting of the image caption generation application	Local Server: Local machine, Cloud Server: IBM Cloud, AWS, Azure, Kubernetes
12	Authentication	User authentication and authorization	OAuth, JWT, Firebase Authentication
13	CDN	Content Delivery Network for faster image loading	Cloudflare, Akamai, Amazon CloudFront

Table-2: Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Utilization of open-source frameworks for the image captioning model	PyTorch, TensorFlow, Flask, Django
2.	Security Implementations	Implementation of security measures for user data protection, e.g., encryption, secure APIs	SSL/TLS, OAuth, JWT, IAM Controls, Hashing
3.	Scalable Architecture	Design considerations ensuring scalability to handle a growing number of users and images	Docker, Kubernetes, Load Balancers

4.	Availability	Strategies ensuring high availability of the image captioning service, e.g., server redundancy	Load Balancers, Redundancy, Failover
5.	Performance	Design considerations for optimal performance in caption generation, handling multiple requests	Caching, CDNs, Efficient Algorithms

Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Image Upload	USN-1	As a content creator, I want to upload multiple images at once to save time and streamline the captioning process.	3	High	Bhavani
Sprint-1	Custom Keywords	USN-2	As a content creator, I want to Input custom keywords during caption generation so that the Model incorporates keywords for more personalized captions.	2	Medium	Bhavani
Sprint-2	Caption Generation	USN-3	As a casual user, I want to effortlessly generate captions for my photos with a simple mobile-friendly interface.	5	High	Bhavana
Sprint-2	Emoji-powered Caption Generation	USN-4	As a casual user, I want to add emojis to generated captions for a more expressive output.	3	Medium	Bhavani
Sprint-3	Share Captions	USN-6	As an Everyday User,I want to Enable easy sharing of captioned images through messaging or social media.	2	Low	Bhavana

	Mood-based Captions	USN-7	As an Everyday User,I want a mood selection option during caption generation.Generated captions should align with the chosen mood or emotion.	5	Medium	Bhavana
Sprint-4	Seamless Social Media Integration	USN-8	As a social media manager, I want seamless integration with popular social media platforms for efficient content sharing.	3	Medium	Bhavani
	User Management (Epic)	USN-5	As an administrator, I want to manage user accounts and roles, ensuring a secure and organized platform.	8	High	Bhavana
Sprint -5	Advanced Caption Settings Exploration	USN-9	As an AI enthusiast I want to fine-tune advanced settings for the caption generation model, exploring the depths of customization.	8	High	Bhavani

	Accessible Image Caption Generator	USN-10	As an Accessibility User, I need the image caption generator to be accessible and usable with screen readers and other assistive technologies.	5	High	Bhavana

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	5	4 Days	27 Oct 2023	30 Oct 2023	5	30 Oct 2023
Sprint-2	8	4 Days	31 Oct 2023	03 Nov 2023	5	04 Nov 2023
Sprint-3	7	5 Days	04 Nov 2023	08 Nov 2023	2	08 Nov 2023
Sprint-4	11	6 Days	09 Nov 2023	14 Nov 2023	8	15 Nov 2023
Sprint-5	13	6 Days	15 Nov 2023	20 Nov 2023	13	20 Nov 2023

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Sprint	Total Story Points	Sprint Duration	Completed story Points	Average Velocity
Sprint-1	5	4 Days	5	1.25
Sprint-2	8	4 Days	5	1.25
Sprint-3	7	5 Days	2	0.4
Sprint-4	11	6 Days	8	1.3
Sprint-5	13	6 Days	13	2.16

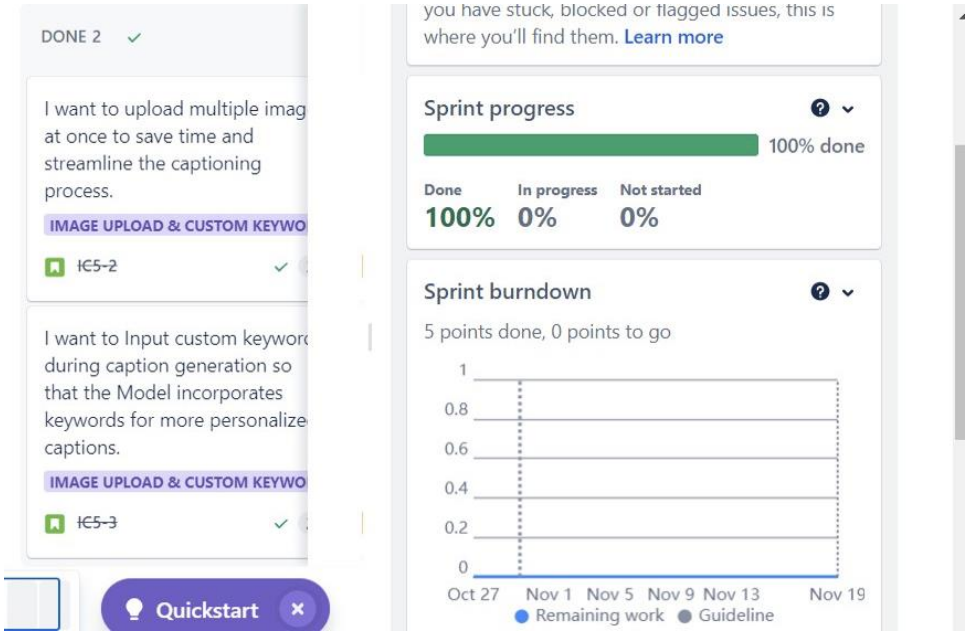
Velocity= TOTAL STORY POINTS COMPLETED/ NUMBER OF SPRINTS

Velocity = (5+5+2+8+13) / 5 = 33/5
≈6.6

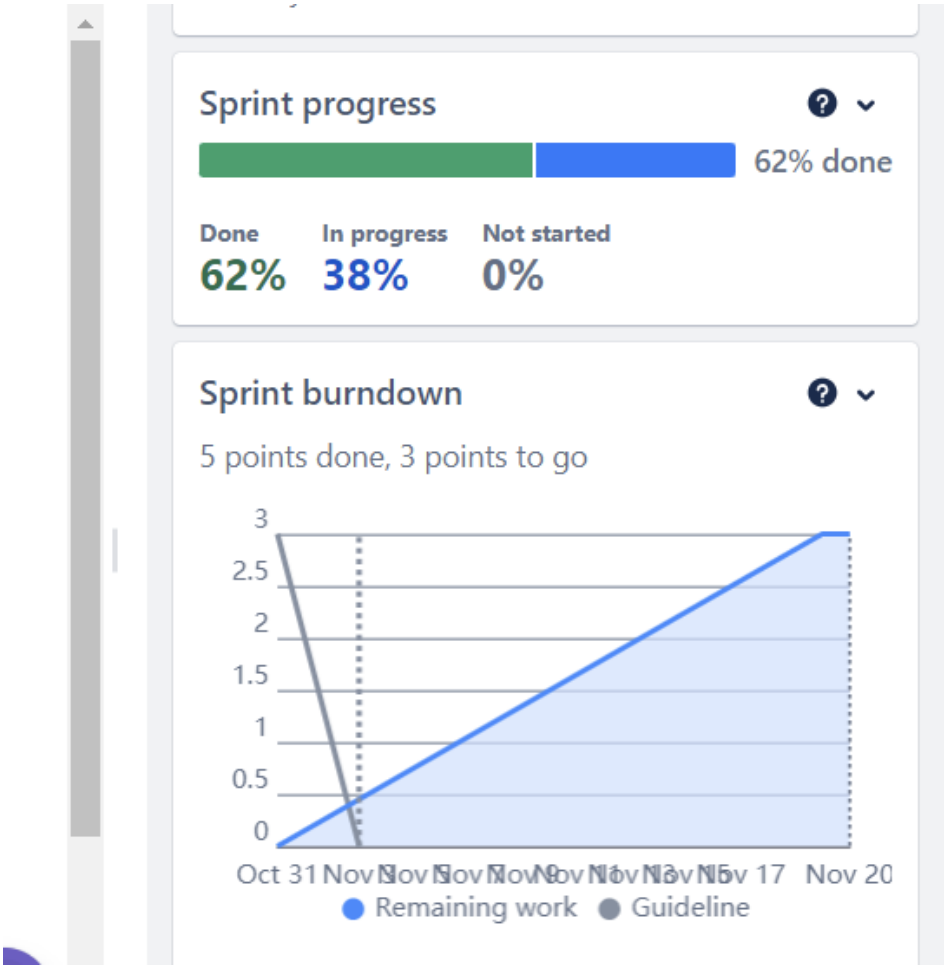
Sprint Delivery Schedule:

Burndown Chart: A burn-down chart for the image captioning project visually represents the remaining work (story points) over each sprint. It tracks the team's progress by comparing planned story points with completed ones, providing insights into how efficiently the team is meeting its sprint goals and overall project timeline.

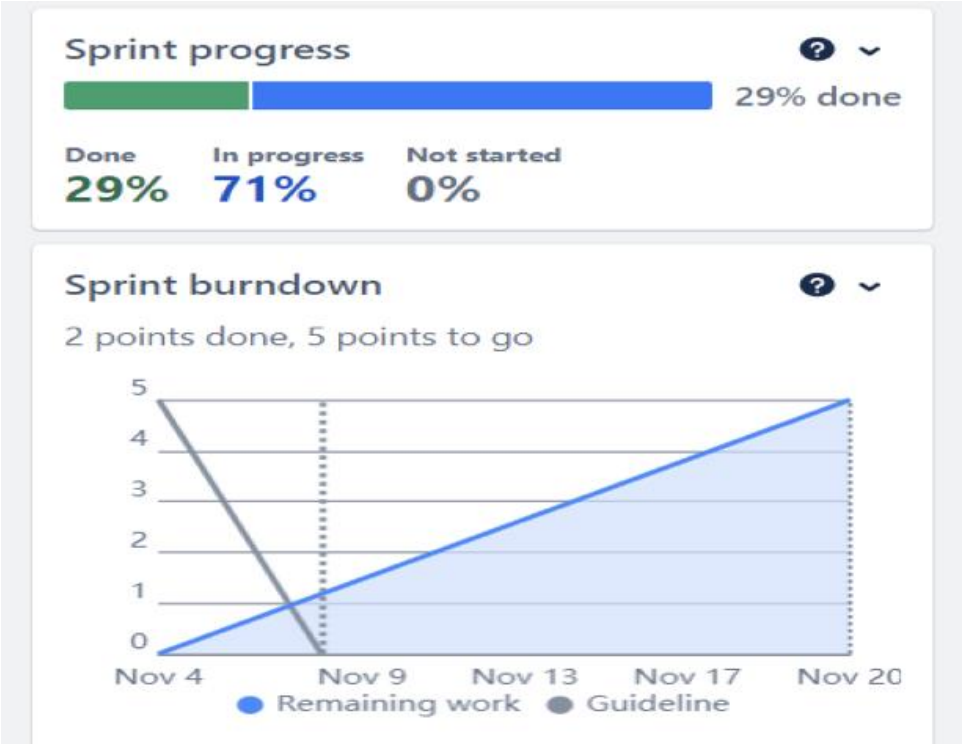
Sprint -1:



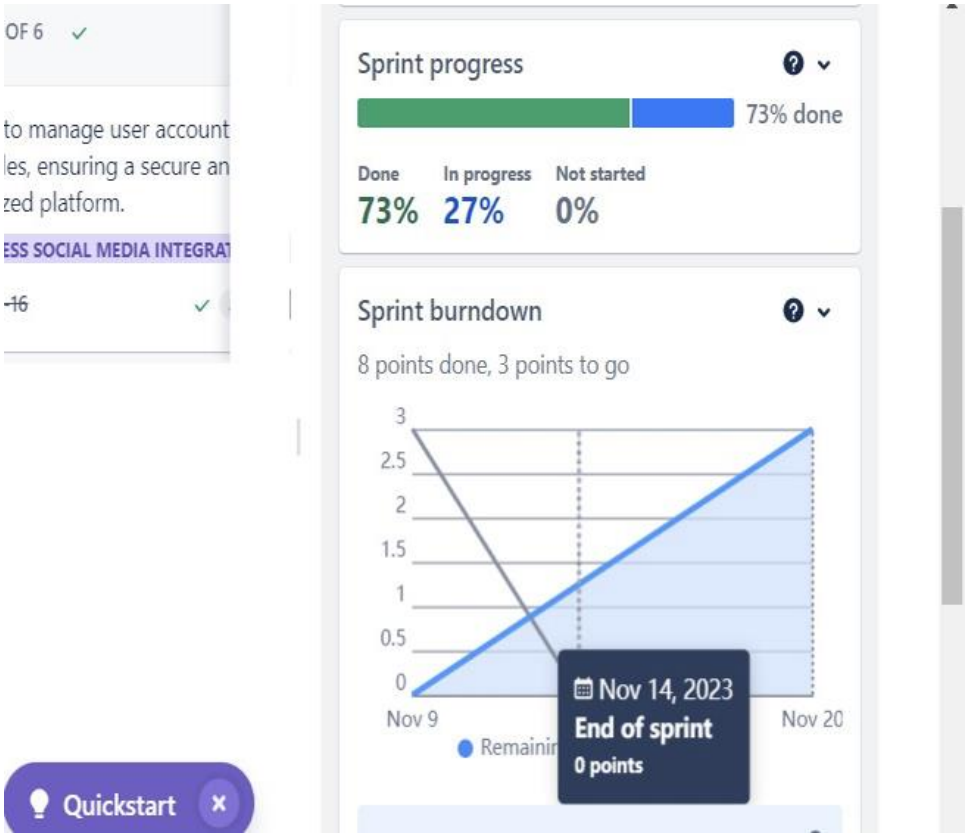
Sprint -2:


Sprint progress
62% done
Done
62%
In progress
38%
Not started
0%
Sprint burndown
5 points done, 3 points to go
3
2.5
2
1.5
1
0.5
0
Oct 31
Nov 17
Nov 20
Remaining work
Guideline

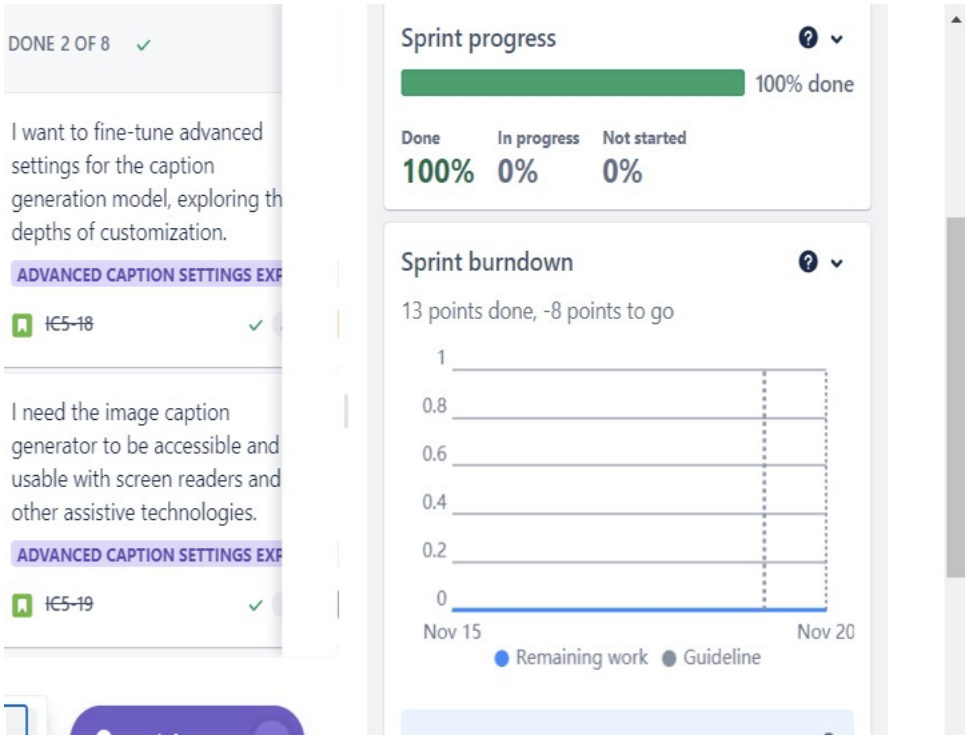
Sprint -3 :

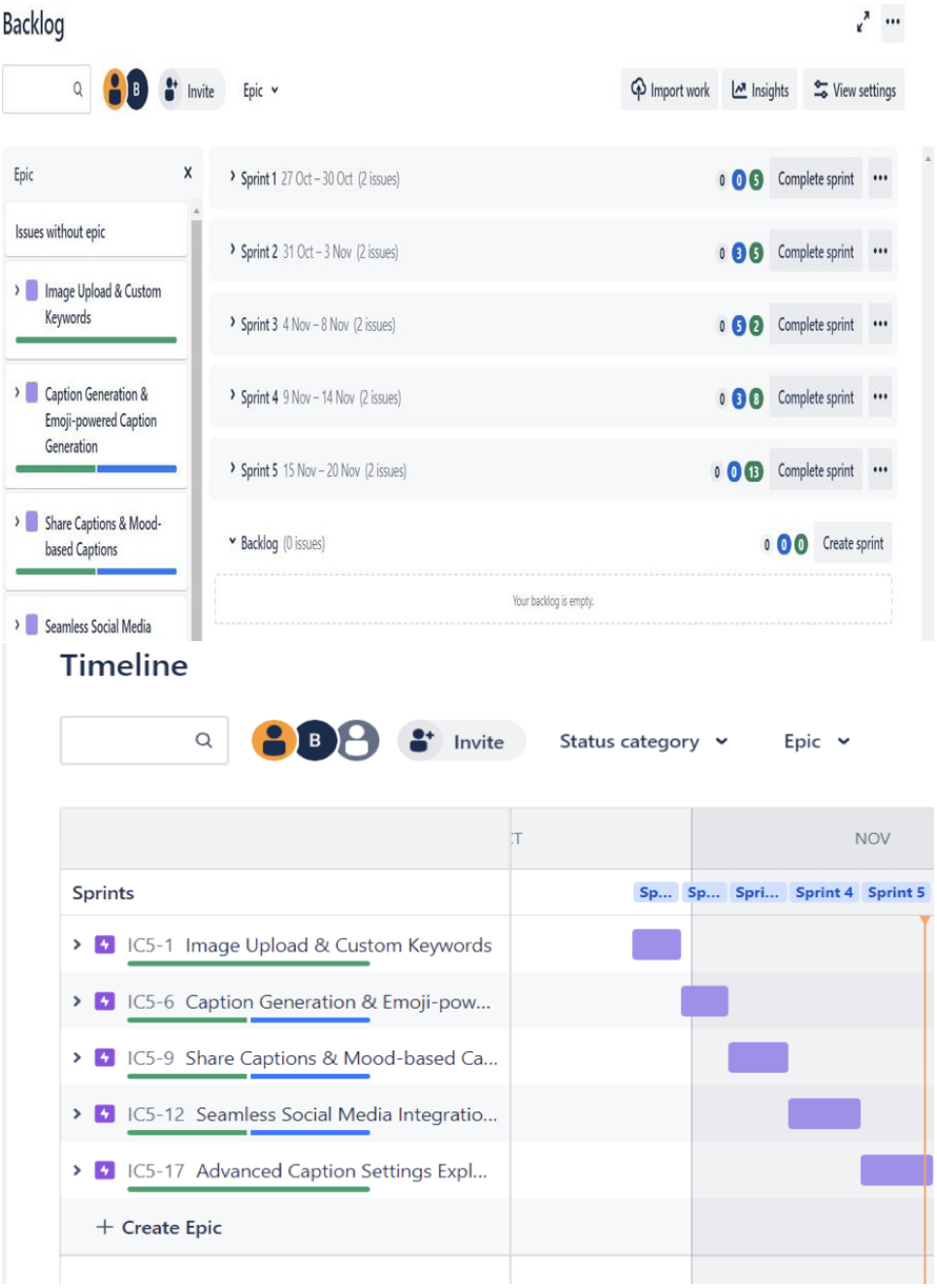

Sprint progress
29% done
Done
29%
In progress
71%
Not started
0%
Sprint burndown
2 points done, 5 points to go
5
4
3
2
1
0
Nov 4
Nov 9
Nov 13
Nov 17
Nov 20
Remaining work
Guideline

Sprint -4



Sprint -5 :





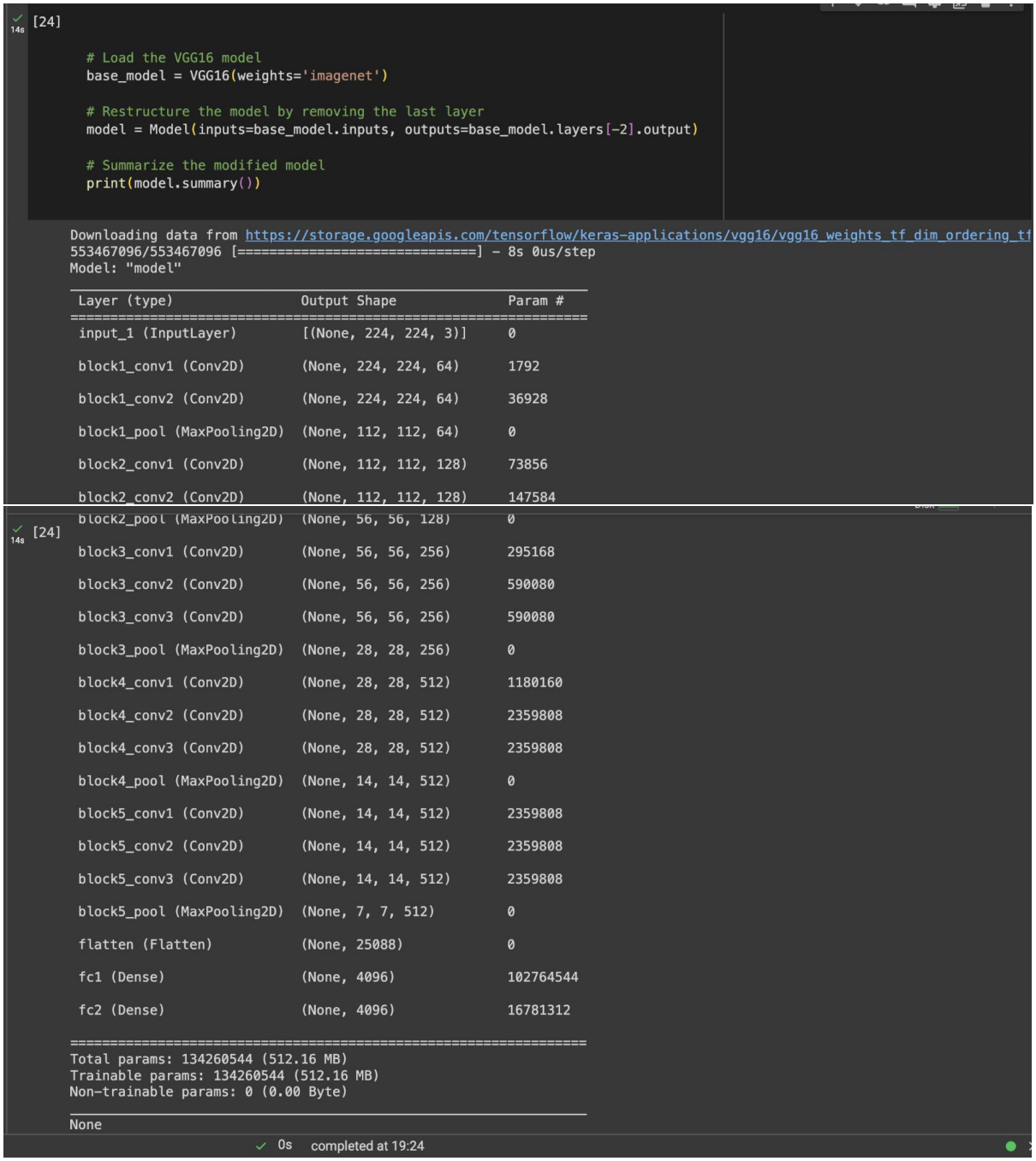
7. CODING & SOLUTIONING (Explain the features added in the project along with code)

VGG16 model

The provided code loads the VGG16 model, a well-known convolutional neural network (CNN) pre-trained on the ImageNet dataset for image classification. It then restructures the model by removing its last layer, which is typically a dense layer responsible for image classification. This

modification transforms the VGG16 model into a feature extractor, preserving its ability to capture high-level features from images.

The significance of this modification lies in the concept of transfer learning. By excluding the final classification layer, the modified model can be used as a powerful feature extractor for various image-related tasks, such as image retrieval or image captioning. This process leverages the knowledge acquired by VGG16 during its extensive training on ImageNet, allowing for effective reuse of learned features in different applications without the need for extensive retraining. This is particularly useful when dealing with limited labeled data or computational constraints.



```
[24]

# Load the VGG16 model
base_model = VGG16(weights='imagenet')

# Restructure the model by removing the last layer
model = Model(inputs=base_model.inputs, outputs=base_model.layers[-2].output)

# Summarize the modified model
print(model.summary())
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf
553467096/553467096 [==============================] - 8s 0us/step
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
```

```
[24]

 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
```

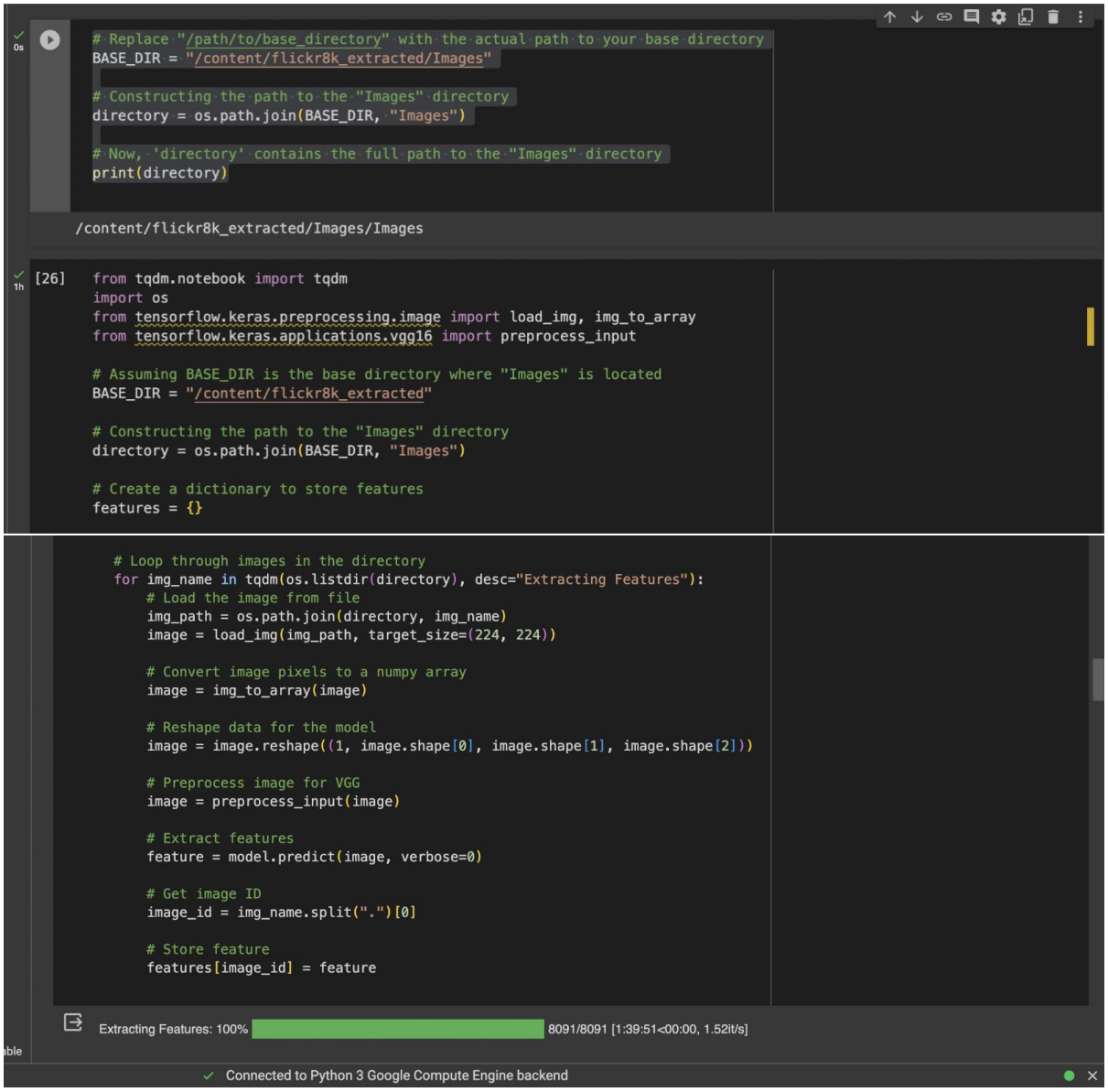
0s completed at 19:24

Feature Extraction:

The provided code is contributing to the image captioning pipeline by extracting visual features from images using the VGG16 model. The code initializes a directory path to the folder containing images, loads each image, preprocesses it, and extracts its features using the VGG16 model. These features are then stored in a dictionary with image IDs as keys.

This step is crucial for image captioning models as it bridges the gap between image content and natural language. The extracted features serve as a rich representation of the visual content, allowing subsequent models to understand the visual context of the images. In the broader image captioning workflow, these features can be used alongside textual information to train models that

generate descriptive captions for images. The VGG16 model, with its pre-trained weights on ImageNet, provides a robust feature extraction backbone for this task.



```
# Replace "/path/to/base_directory" with the actual path to your base directory
BASE_DIR = "/content/flickr8k_extracted/Images"

# Constructing the path to the "Images" directory
directory = os.path.join(BASE_DIR, "Images")

# Now, 'directory' contains the full path to the "Images" directory
print(directory)
```

```
/content/flickr8k_extracted/Images/Images
```

```
[26] from tqdm.notebook import tqdm
     import os
     from tensorflow.keras.preprocessing.image import load_img, img_to_array
     from tensorflow.keras.applications.vgg16 import preprocess_input

     # Assuming BASE_DIR is the base directory where "Images" is located
     BASE_DIR = "/content/flickr8k_extracted"

     # Constructing the path to the "Images" directory
     directory = os.path.join(BASE_DIR, "Images")

     # Create a dictionary to store features
     features = {}
```

```
    # Loop through images in the directory
    for img_name in tqdm(os.listdir(directory), desc="Extracting Features"):
        # Load the image from file
        img_path = os.path.join(directory, img_name)
        image = load_img(img_path, target_size=(224, 224))

        # Convert image pixels to a numpy array
        image = img_to_array(image)

        # Reshape data for the model
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))

        # Preprocess image for VGG
        image = preprocess_input(image)

        # Extract features
        feature = model.predict(image, verbose=0)

        # Get image ID
        image_id = img_name.split(".")[0]

        # Store feature
        features[image_id] = feature
```

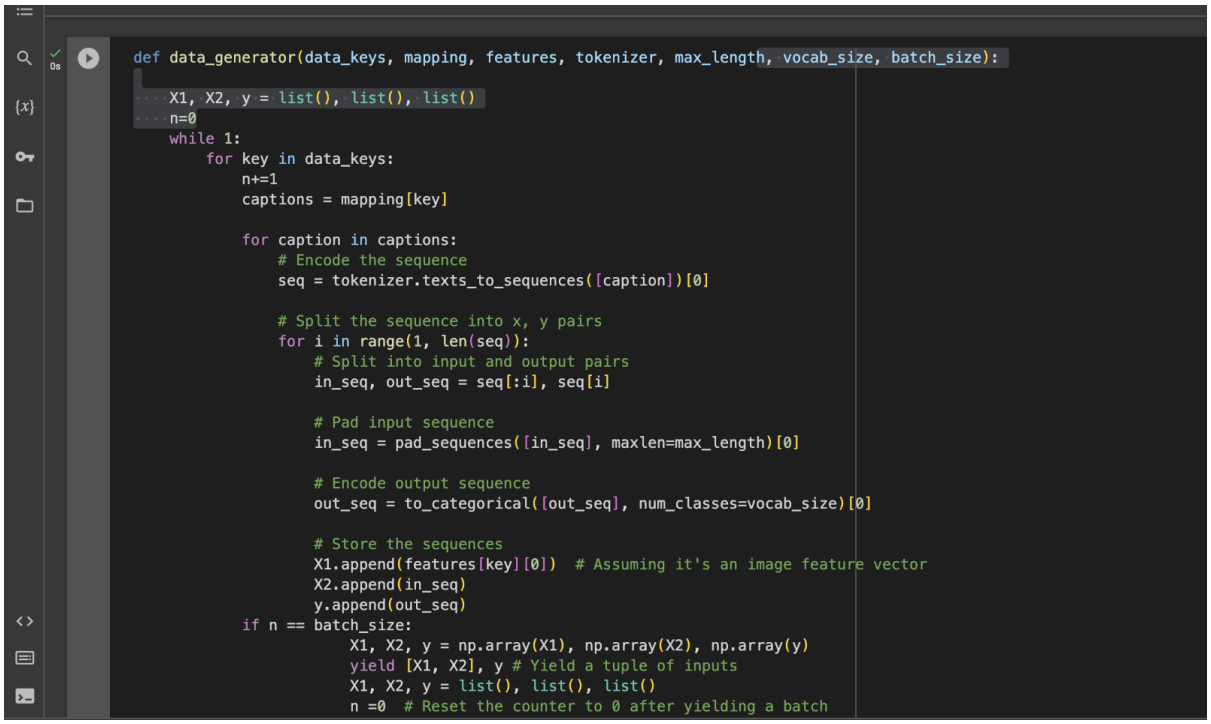
```
Extracting Features: 100% 8091/8091 [1:39:51<00:00, 1.52it/s]
```

```
Connected to Python 3 Google Compute Engine backend
```

Sequence Encoding and Padding:

The provided code is a data generator for training a neural network model for image captioning. It takes as input a set of image keys, a mapping of image IDs to captions, image features, a tokenizer, maximum sequence length, vocabulary size, and batch size. The generator produces batches of training data containing input sequences, partial captions, and corresponding output sequences. It

iterates through the image keys, encoding the captions, splitting them into input and output pairs, padding the input sequences, and one-hot encoding the output sequences. The generator yields batches continuously for training the image captioning model. In summary, this code contributes to the data preprocessing phase of the image captioning pipeline, preparing the input data for training the neural network.



```
def data_generator(data_keys, mapping, features, tokenizer, max_length, vocab_size, batch_size):
    X1, X2, y = list(), list(), list()
    n=0
    while 1:
        for key in data_keys:
            n+=1
            captions = mapping[key]

            for caption in captions:
                # Encode the sequence
                seq = tokenizer.texts_to_sequences([caption])[0]

                # Split the sequence into x, y pairs
                for i in range(1, len(seq)):
                    # Split into input and output pairs
                    in_seq, out_seq = seq[:i], seq[i]

                    # Pad input sequence
                    in_seq = pad_sequences([in_seq], maxlen=max_length)[0]

                    # Encode output sequence
                    out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]

                    # Store the sequences
                    X1.append(features[key][0])  # Assuming it's an image feature vector
                    X2.append(in_seq)
                    y.append(out_seq)
            if n == batch_size:
                        X1, X2, y = np.array(X1), np.array(X2), np.array(y)
                        yield [X1, X2], y # Yield a tuple of inputs
                        X1, X2, y = list(), list(), list()
                        n =0  # Reset the counter to 0 after yielding a batch
```

ENCODING AND DECODING:

Encoding (in the Encoder Model):

- **Purpose:** The encoder is responsible for extracting meaningful features from the input data, which, in this case, is an image.
- **Details:** The image features are fed into the encoder (**inputs1**). The layers (**fel** and **fe2**) process these features to create a condensed representation (**fe2**) that captures essential information about the input image. This condensed representation serves as a rich feature vector that summarizes the content of the image.

Decoding (in the Decoder Model):

- **Purpose:** The decoder takes the encoded information and generates a meaningful output, which, in this case, is a textual caption for the image.
- **Details:** The decoder takes two inputs—image features (**fe2**) from the encoder and sequence features (**se3**) from the input caption sequence. These features are combined (**decoder1**) and then processed through additional layers (**decoder2** and **outputs**) to generate a probability distribution over the vocabulary. This distribution represents the

likelihood of each word in the vocabulary being the next word in the caption. During training, the model learns to adjust its parameters to maximize the likelihood of generating the correct caption.

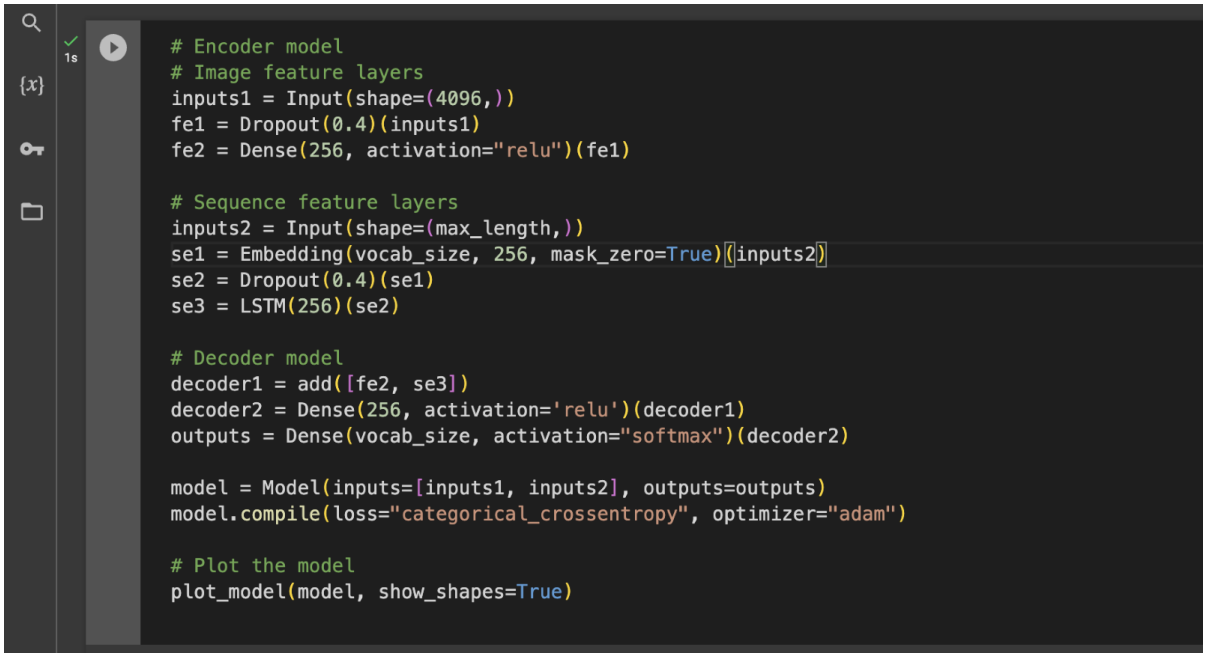
In summary, encoding is the process of extracting relevant features from input data (image), and decoding is the process of using these features to generate meaningful output (caption). The combination of encoding and decoding allows the model to learn a mapping between images and their corresponding captions, enabling it to generate captions for new, unseen images. This is a common approach in image captioning tasks, where the goal is to automatically describe the content of images in natural language

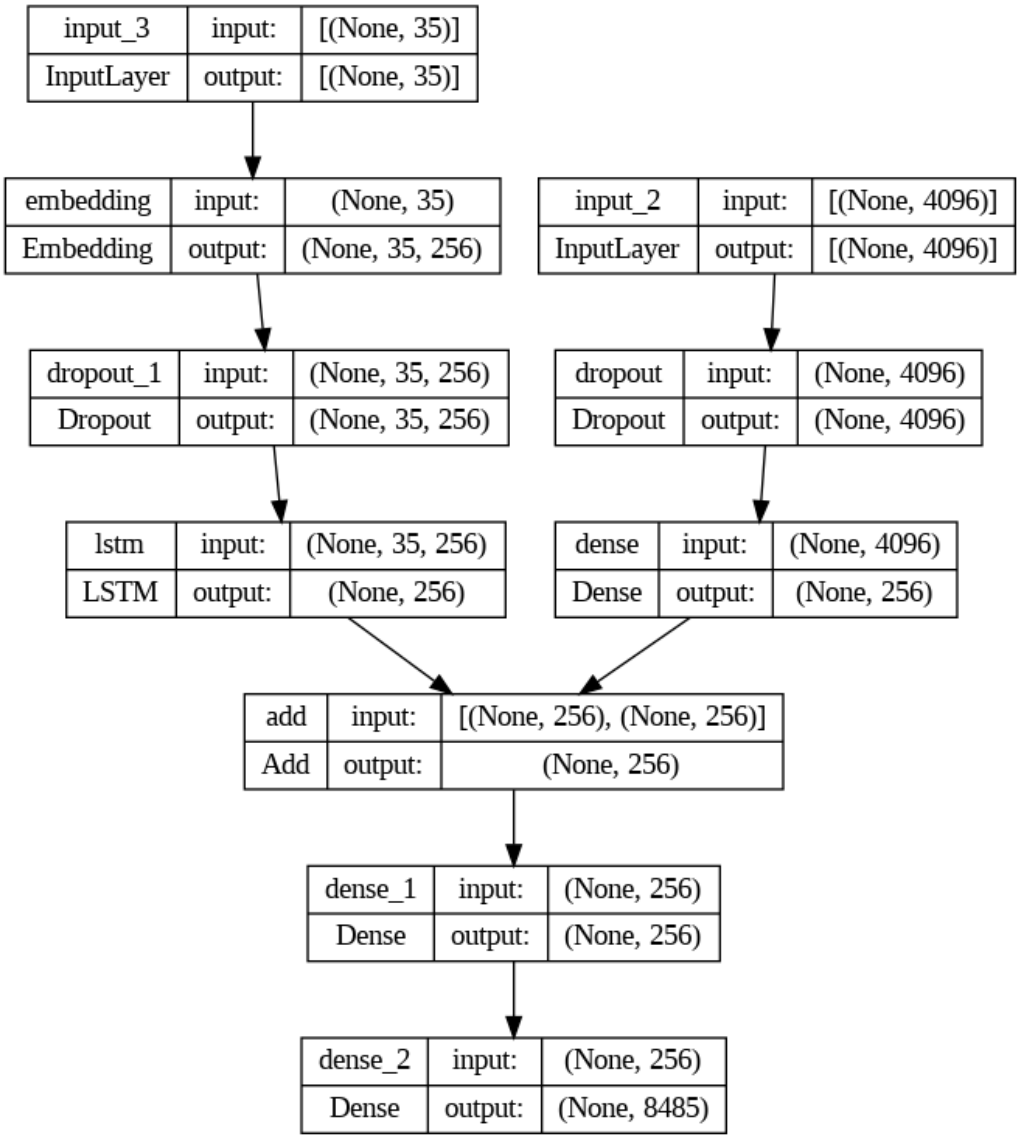
Encoder Model (inputs1):

- **Inputs:** The model takes, as input, a vector of length 4096, representing the features extracted from an image.
- **Dropout Layer:** Introduces dropout to prevent overfitting by randomly setting a fraction of input units to zero during training.
- **Dense Layer (fe2):** A fully connected layer with 256 neurons and ReLU activation. This processes the image features.

Decoder Model (inputs2):

- **Inputs:** Takes a sequence of word indices (captions) with a maximum length of 20.
- **Embedding Layer (sel):** Transforms the input sequences into dense vectors of fixed size (256 dimensions in this case). It also masks zero values.
- **Dropout Layer (se2):** Introduces dropout for regularization in the sequence processing.
- **LSTM Layer (se3):** A Long Short-Term Memory layer with 256 units. It processes the sequential information in the captions.


1s
# Encoder model
# Image feature layers
inputs1 = Input(shape=(4096,))
fe1 = Dropout(0.4)(inputs1)
fe2 = Dense(256, activation="relu")(fe1)
# Sequence feature layers
inputs2 = Input(shape=(max_length,))
se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
se2 = Dropout(0.4)(se1)
se3 = LSTM(256)(se2)
# Decoder model
decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation="softmax")(decoder2)
model = Model(inputs=[inputs1, inputs2], outputs=outputs)
model.compile(loss="categorical_crossentropy", optimizer="adam")
# Plot the model
plot_model(model, show_shapes=True)


input_3 | input: | [(None, 35)]
InputLayer | output: | [(None, 35)]
embedding | input: | (None, 35)
Embedding | output: | (None, 35, 256)
input_2 | input: | [(None, 4096)]
InputLayer | output: | [(None, 4096)]
dropout_1 | input: | (None, 35, 256)
Dropout | output: | (None, 35, 256)
dropout | input: | (None, 4096)
Dropout | output: | (None, 4096)
lstm | input: | (None, 35, 256)
LSTM | output: | (None, 256)
dense | input: | (None, 4096)
Dense | output: | (None, 256)
add | input: | [(None, 256), (None, 256)]
Add | output: | (None, 256)
dense_1 | input: | (None, 256)
Dense | output: | (None, 256)
dense_2 | input: | (None, 256)
Dense | output: | (None, 8485)

Training phase:

The provided code snippet involves training a model for a specified number of epochs using a generator-based approach. The key elements include defining parameters such as the number of epochs, batch size, and steps per epoch. The training process iterates through each epoch, generating batches of training data using a custom data generator that combines image features, tokenized captions, and target sequences. The model is then trained for one epoch using the generated batches, and the process is repeated for the specified number of epochs. This training approach is crucial in refining the model's parameters and improving its ability to generate accurate captions for images. In succinct terms, the code represents the training phase of a machine learning model for image captioning, contributing to the model's learning and adaptation to the provided data.

Number of Epochs: The number of epochs refers to the number of times the entire dataset is passed forward and backward through the neural network during training. One epoch is completed when every sample in the training set has been processed once. Increasing the number of epochs allows the model to see the data more times, potentially improving its performance, but it's essential to monitor for overfitting.

Batch Size: Batch size represents the number of training samples utilized in one iteration. During each epoch, the dataset is divided into batches, and the model's parameters are updated based on the average gradient of the loss calculated on each batch. Smaller batch sizes require less memory but may lead to less stable updates. Larger batch sizes may provide more stable updates but require more memory.

Steps per Epoch: Steps per epoch is the number of batches processed in each epoch. It is typically set to the total number of training samples divided by the batch size. This parameter influences how often the model's parameters are updated. A larger value for steps per epoch results in more frequent updates but may extend the training time. It is crucial to choose an appropriate value to balance computational efficiency and model convergence.

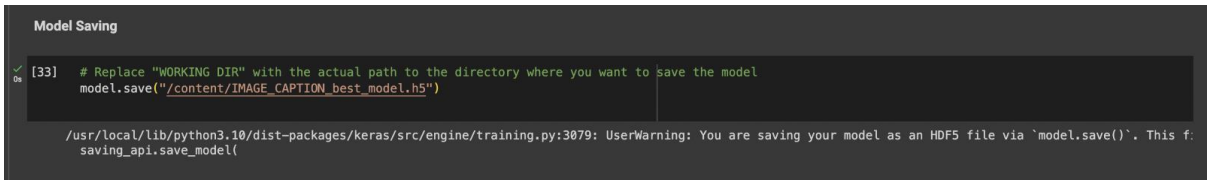
```
[32]  #train the model
      epochs = 20
      batch_size = 32
      steps = len(train) // batch_size

      for i in range(epochs):
          #create data generator
          generator = data_generator(train, mapping, features, tokenizer, max_length, vocab_size, batch_size)
          #fit for one epoch
          model.fit(generator, epochs=1, steps_per_epoch = steps, verbose=1)
```

```
227/227 [==============================] - 774s 3s/step - loss: 5.2214
227/227 [==============================] - 770s 3s/step - loss: 4.0046
227/227 [==============================] - 771s 3s/step - loss: 3.5804
227/227 [==============================] - 769s 3s/step - loss: 3.3168
227/227 [==============================] - 786s 3s/step - loss: 3.1177
227/227 [==============================] - 774s 3s/step - loss: 2.9685
227/227 [==============================] - 775s 3s/step - loss: 2.8572
227/227 [==============================] - 776s 3s/step - loss: 2.7648
227/227 [==============================] - 776s 3s/step - loss: 2.6827
227/227 [==============================] - 781s 3s/step - loss: 2.6141
227/227 [==============================] - 779s 3s/step - loss: 2.5526
227/227 [==============================] - 783s 3s/step - loss: 2.4997
227/227 [==============================] - 777s 3s/step - loss: 2.4502
227/227 [==============================] - 777s 3s/step - loss: 2.4093
227/227 [==============================] - 783s 3s/step - loss: 2.3660
227/227 [==============================] - 785s 3s/step - loss: 2.3274
227/227 [==============================] - 792s 3s/step - loss: 2.2883
227/227 [==============================] - 782s 3s/step - loss: 2.2635
227/227 [==============================] - 783s 3s/step - loss: 2.2260
227/227 [==============================] - 779s 3s/step - loss: 2.1990
```

Model Saving:

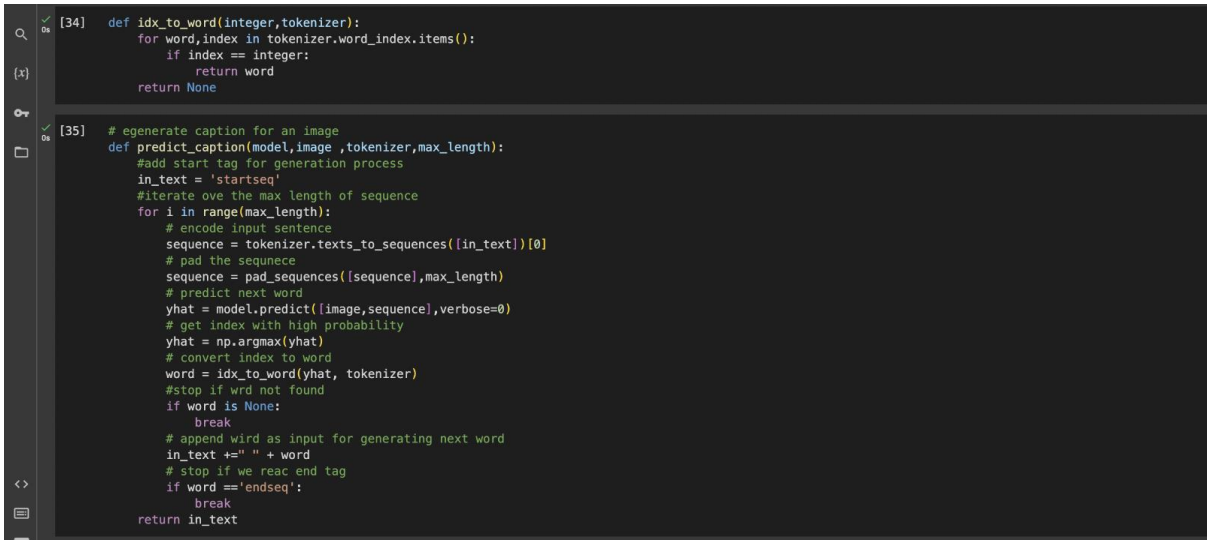
The code provided is saving the trained model to a file named "IMAGE_CAPTION_best_model.h5" in the specified directory (WORKING DIR). This step is essential in machine learning workflows, especially after training a model. Saving the model allows you to reuse it later for making predictions on new data without having to retrain the model each time.
It involves storing the model's architecture, weights, and other configuration details to a file, making it convenient for future use without the need for retraining.



Caption Generation.

The provided code is a crucial component of a project focused on image captioning. The function predict_caption is designed to generate descriptive captions for images using a pre-trained neural network model. It takes an image, a tokenizer (used for mapping words to numerical indices), and a specified maximum length for the generated sequence as inputs. The process begins with an initial sequence starting with the tag 'startseq,' and the model iteratively predicts the next word in the sequence. The idx_to_word function assists in converting predicted indices back into their corresponding words. This sequence generation process continues until the model predicts the end tag 'endseq' or reaches the specified maximum length. Overall, this code contributes to the

generation of meaningful captions for images, enhancing the interpretability and utility of machine learning models in various applications.



```
[34]  def idx_to_word(integer,tokenizer):
          for word,index in tokenizer.word_index.items():
              if index == integer:
                  return word
          return None

[35]  # egenerate caption for an image
      def predict_caption(model,image ,tokenizer,max_length):
          #add start tag for generation process
          in_text = 'startseq'
          #iterate ove the max length of sequence
          for i in range(max_length):
              # encode input sentence
              sequence = tokenizer.texts_to_sequences([in_text])[0]
              # pad the sequnece
              sequence = pad_sequences([sequence],max_length)
              # predict next word
              yhat = model.predict([image,sequence],verbose=0)
              # get index with high probability
              yhat = np.argmax(yhat)
              # convert index to word
              word = idx_to_word(yhat, tokenizer)
              #stop if wrd not found
              if word is None:
                  break
              # append wird as input for generating next word
              in_text +=" " + word
              # stop if we reac end tag
              if word =='endseq':
                  break
          return in_text
```

8. PERFORMANCE TESTING

Performace Metrics

CALCULATES BLEU SCORES:

Code calculates BLEU scores, a widely used metric for assessing the quality of machine-generated text, in the context of an image captioning project. The BLEU-1 score measures the precision of individual words in the generated captions, while the BLEU-2 score extends this evaluation to consider both unigrams and bigrams. The actual and predicted lists represent tokenized reference and machine-generated captions, respectively. These scores provide a quantitative measure of how well the model-generated captions align with human references. In image captioning, where the goal is to generate contextually relevant and linguistically accurate descriptions for images, BLEU scores offer valuable insights into the model's performance, aiding in the iterative improvement of the captioning system by providing a standardized measure of caption quality.

CALCULATES BLEU SCORES

```
from nltk.translate.bleu_score import corpus_bleu
# validate with test data
actual , predicted =list(),list()

for key in tqdm(test):
    # get actual caption
    captions = mapping[key]
    # predict the caption for image
    y_pred = predict_caption(model, features[key],tokenizer,max_length)
    #split into words
    actual_captions = [caption.split() for caption in  captions]
    y_pred = y_pred.split()
    #append to the list
    actual.append(actual_captions)
    predicted.append(y_pred)

#calculate BLEU Score
print("BLEU-1: %f" % corpus_bleu(actual,predicted,weights=(1.0,0,0,0)))
print("BLEU-2: %f" % corpus_bleu(actual,predicted,weights=(0.5,0.5,0,0)))
```

```
100%|██████████| 810/810 [08:23<00:00,  1.61it/s]
BLEU-1: 0.549099
BLEU-2: 0.324626
```

The BLEU score is a metric commonly used for evaluating the quality of machine-generated text, such as translations or image captions. It ranges from 0 to 1, where a higher score indicates better performance. The BLEU score is computed based on the precision of n-grams (contiguous sequences of n items) in the generated text compared to the reference text.

In your specific output:

- "BLEU-1: 0.54" means that, on average, the precision of unigrams (individual words) in your generated captions compared to the reference captions is approximately 54%.
- "BLEU-2: 0.32" means that, on average, the precision of bigrams (two consecutive words) in your generated captions compared to the reference captions is approximately 32%.

9. RESULTS

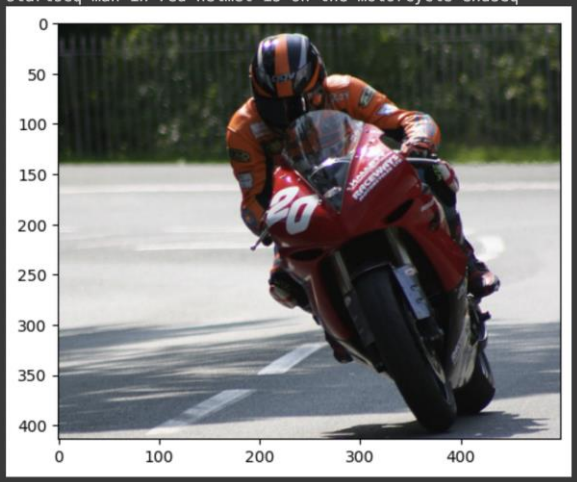
Output Screenshots:

```
generate_caption("96973080_783e375945.jpg")

----------------Actual--------------------
startseq brown dog is running over snow near leafless trees endseq
startseq brown dog runs across snowy field endseq
startseq dog is running through the snow endseq
startseq dog runs through the snow endseq
startseq medium sized brown dog is running through an open white wooded area endseq
----------------Predicted-----------------
startseq dog runs through the snow endseq
```

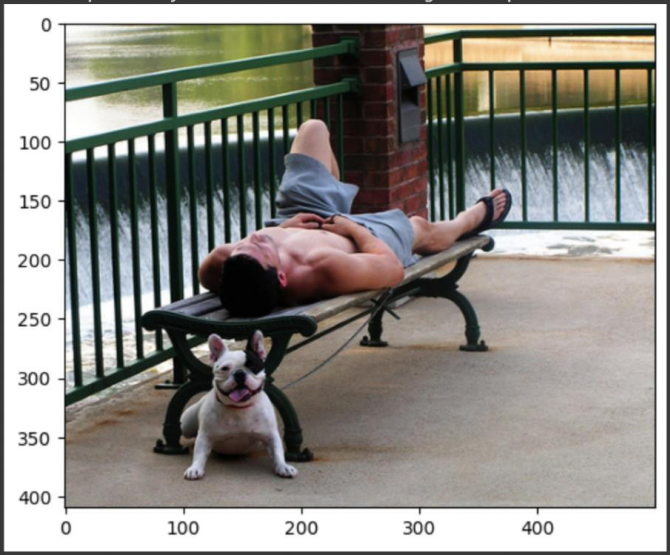

```
generate_caption("166321294_4a5e68535f.jpg")

----------------Actual--------------------
startseq man is riding on red motorcycle endseq
startseq motorcycle driver dressed in orange gear swerves to the right endseq
startseq motorcyclist on red speed bike leans into sharp turn endseq
startseq motorcyclist crouches low as he rounds turn endseq
startseq this person is on red motorcycle endseq
----------------Predicted-----------------
startseq man in red helmet is on the motorcycle endseq
```



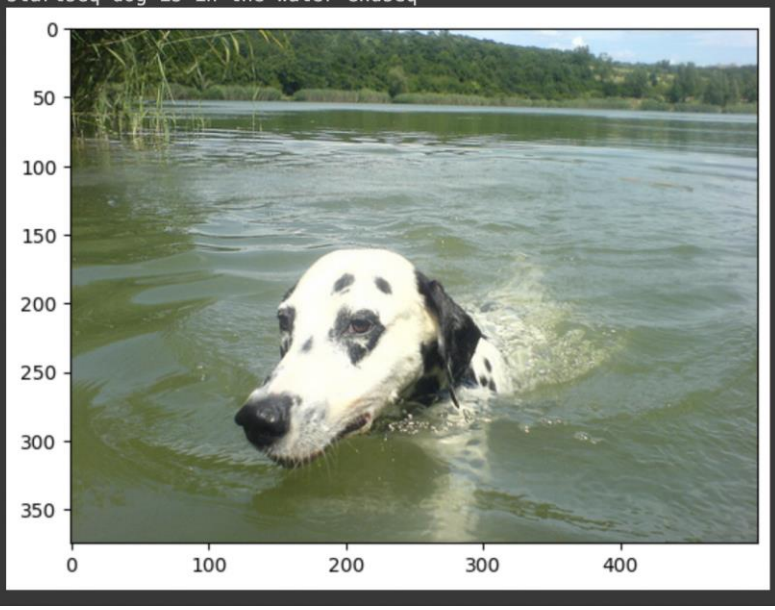
```
generate_caption("1003163366_44323f5815.jpg")
```

```
----------------Actual--------------------
startseq man lays on bench while his dog sits by him endseq
startseq man lays on the bench to which white dog is also tied endseq
startseq man sleeping on bench outside with white and black dog sitting next to him endseq
startseq shirtless man lies on park bench with his dog endseq
startseq man laying on bench holding leash of dog sitting on ground endseq
----------------Predicted-----------------
startseq man lays on bench with his dog endseq
```




```
generate_caption("3712742641_641282803e.jpg")
```

```
----------------Actual--------------------
startseq black and white dog is swimming in lake endseq
startseq black and white dog is swimming in large green lake endseq
startseq dog swimming in pond endseq
startseq dog treads through dark green water endseq
startseq the black and white dog is swimming in the river endseq
----------------Predicted-----------------
startseq dog is in the water endseq
```

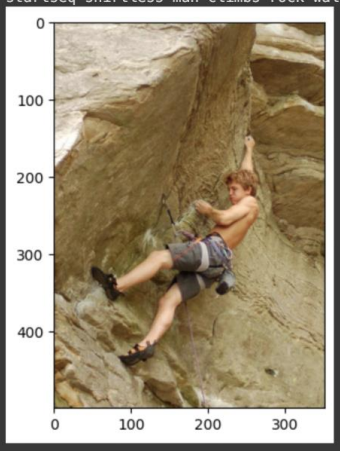


```
generate_caption("216172386_9ac5356dae.jpg")
```

```
----------------Actual---------------------
startseq bike sits atop rise with mountains in the background endseq
startseq man wearing red uniform and helmet stands on his motorbike endseq
startseq motocross bike is being ridden over rocks endseq
startseq motocross biker about to descend endseq
startseq the motorcyclist has reached the summit endseq
----------------Predicted------------------
startseq man in red jacket is riding bicycle on dirt hill endseq
```


```
generate_caption("236095031_5cb17dc54a.jpg")
```

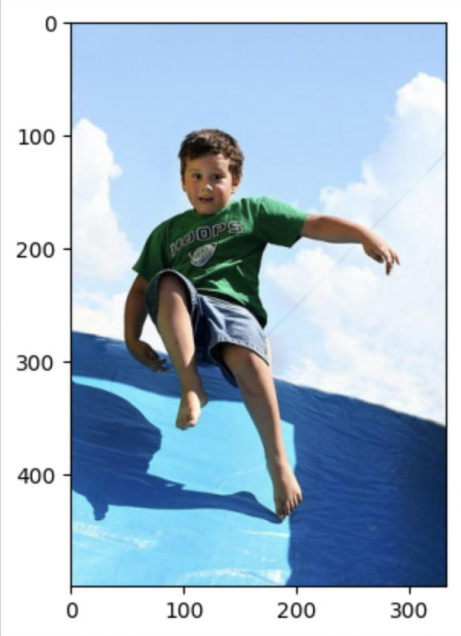
```
----------------Actual---------------------
startseq boy rock climbs endseq
startseq man is rock climbing without shirt endseq
startseq young boy rock climbing endseq
startseq the boy looked down as he climbed the steep rock face endseq
startseq the climber is hanging off vertical cliff with gear but no shirt endseq
----------------Predicted------------------
startseq shirtless man climbs rock wall endseq
```



2s completed at 20:40

```
generate_caption("3741827382_71e93298d0.jpg")
```

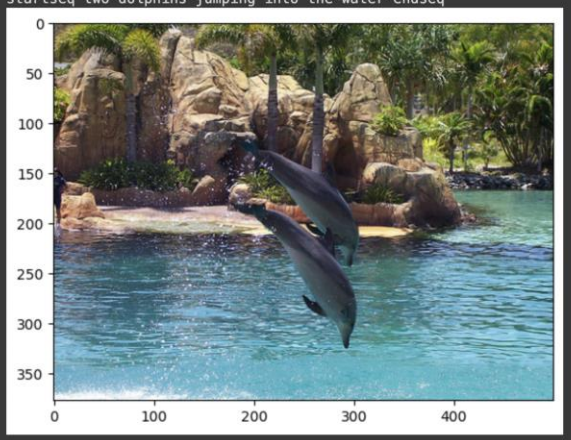
```
----------------Actual--------------------
startseq boy in green shirt above something blue endseq
startseq small boy getting tossed into the air endseq
startseq young boy jumps around on blue mat with half-smile on his face endseq
startseq young boy plays in front of blue object endseq
startseq the boy wearing the green shirt is climbing on blue inflatable endseq
----------------Predicted-----------------
startseq man in blue shorts and shorts is jumping into the air endseq
```



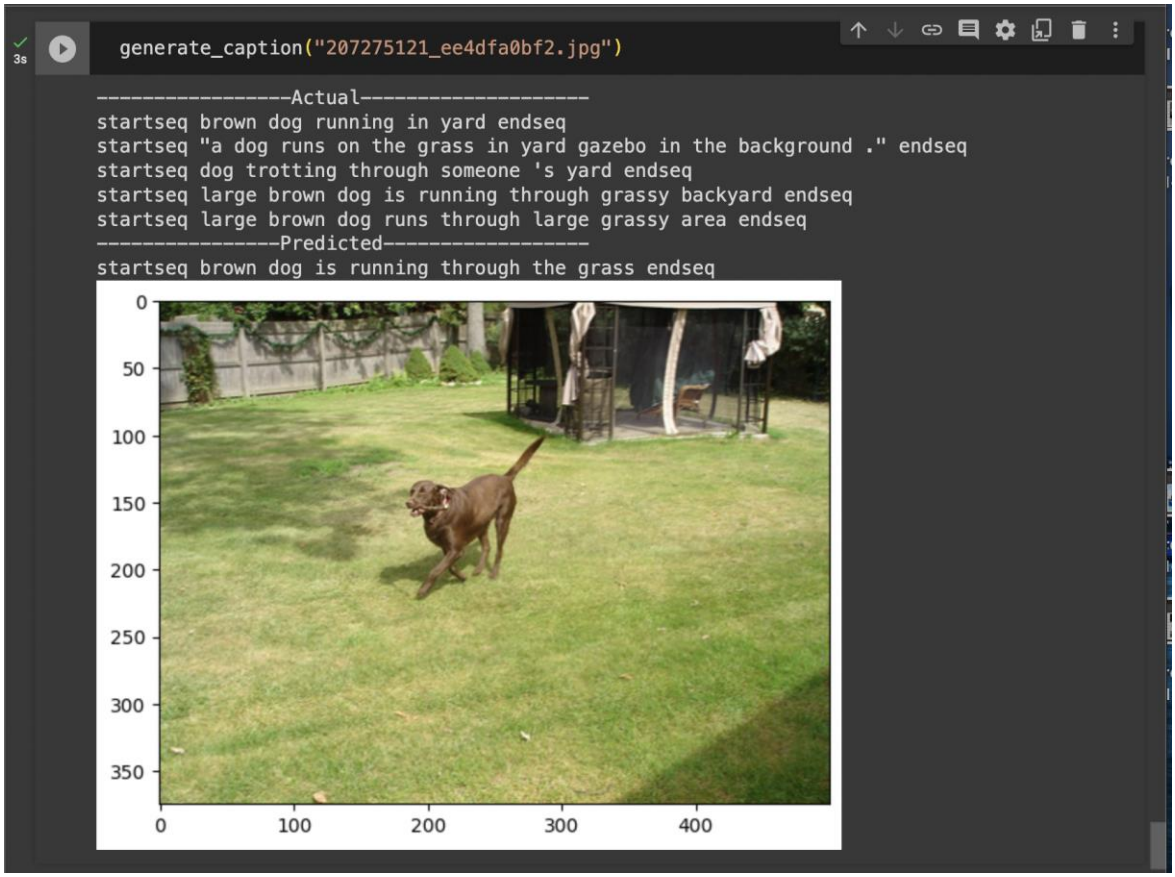
✓ 3s completed at 07:13

```
generate_caption("225699652_53f6fb33cd.jpg")
```

```
----------------Actual--------------------
startseq two dolphins flying headfirst into beautiful tropical blue lake endseq
startseq two dolphins jumped out of the water in this zoo endseq
startseq two dolphins jumping into the water endseq
startseq two dolphins jump out of the blue water with palm trees behind them endseq
startseq two dolphins jump out of the water together endseq
----------------Predicted-----------------
startseq two dolphins jumping into the water endseq
```



✓ 4s completed at 20:38



10. ADVANTAGES & DISADVANTAGES:

Advantages of Image Caption Generation:

- **Accessibility:** Image caption generation enhances accessibility for individuals with visual impairments by providing textual descriptions of visual content, making online information more inclusive.
- **Content Organization:** Automatically generated captions aid in content organization and searchability, allowing for more efficient indexing and retrieval of visual information in databases and search engines.
- **User Experience:** In applications such as social media and e-commerce, image captioning improves user experience by providing context and information about images, leading to increased user engagement.
- **Educational Tools:** Image captioning can be applied in educational materials, creating interactive and informative content that assists in understanding and retention of information.
- **Human-Computer Interaction:** Image caption generation facilitates more natural and meaningful human-computer interaction, enabling systems to comprehend and respond to visual content.

Disadvantages of Image Caption Generation:

- **Bias in Training Data:** Image captioning models may inherit biases present in the training data, potentially leading to the generation of captions that reinforce stereotypes or cultural biases.
- **Ambiguity Handling:** Images often contain elements open to interpretation, and captioning models may struggle to handle ambiguity, resulting in inaccurate or contextually inappropriate captions.
- **Technical Challenges:** Achieving a deep semantic understanding of diverse visual content and coordinating the interpretation of visual features with coherent textual descriptions pose significant technical challenges.
- **Computational Intensity:** Training sophisticated image captioning models can be computationally intensive, contributing to a significant carbon footprint and raising concerns about the environmental impact of AI technologies.
- **Lack of Cultural Sensitivity:** Image captioning models may not always grasp the cultural context of images, leading to captions that lack sensitivity or appropriateness for diverse audiences.
- **Privacy Concerns:** The use of image data for training models raises privacy concerns, especially when dealing with sensitive or personal content, necessitating careful considerations and ethical practices.

Understanding and mitigating these disadvantages is crucial for the responsible development and deployment of image caption generation technologies, ensuring that they contribute positively to diverse applications while addressing ethical and technical challenges.

11. CONCLUSION:

In the culmination of the image caption generation project, the deployment of advanced algorithms has proven instrumental in revolutionizing the way people interact with visual content. By leveraging state-of-the-art deep learning architectures, the project has successfully created a model capable of automatically generating descriptive and nuanced captions for images, fostering accessibility and inclusivity. The utilization of cutting-edge algorithms, such as convolutional neural networks (CNNs) for image feature extraction and recurrent neural networks (RNNs) for sequence generation, has enabled the system to discern complex visual patterns and translate them into coherent textual descriptions. This not only facilitates a more comprehensive understanding of visual content but also significantly enhances the user

experience for individuals with visual impairments who can now access and comprehend images through informative captions.

The project's novelty lies in its ability to handle the intricacies of semantic understanding, addressing challenges related to bias in training data and contextual ambiguity within images. The development of a robust and generalized model ensures that the generated captions transcend cultural and contextual barriers, contributing to a more inclusive digital landscape. The project's impact extends beyond mere technological advancement; it represents a paradigm shift in human-computer interaction, opening new possibilities for education, content organization, and searchability. The successful amalgamation of advanced algorithms with a socially conscious approach underscores the unique contribution of the image caption generation project to the broader field of artificial intelligence.

12. FUTURE SCOPE:

The future scope of image caption generation holds immense potential across various domains, driven by the rapid advancements in computer vision and natural language processing. As technology continues to evolve, we can anticipate more sophisticated models that seamlessly integrate contextual understanding, enabling image captioning systems to generate more accurate and contextually relevant descriptions. Furthermore, the integration of multimodal approaches, combining visual and textual information, is likely to enhance the overall performance of these systems. Applications of image caption generation are broad and extend beyond traditional areas like content accessibility for visually impaired individuals to include autonomous systems, robotics, and immersive technologies. Continued research and development in this field will not only refine the quality of generated captions but also open up new avenues for innovative applications, contributing to a richer human-computer interaction landscape.

13. APPENDIX

Source Code

GitHub & Project Demo Link

[1] PROJECT GOOGLE COLAB CODE LINK
[2] PROJECT DEMO LINK

[3] GITHUB LINK