MIT | Academy of Engineering

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

IMAGE CAPTION GENERATOR

Group Members:

[T227139] Dhanashree Khare 304

[T224005] Harshal Tawade 004

[T229105] Aditya More 330

Guided by

Mrs. Shubhangi Kale

Dr. Vaishali Wangikar

SCHOOL OF COMPUTER ENGINEERING

MIT ACADEMY OF ENGINEERING, ALANDI (D), PUNE-412105

MAHARASHTRA (INDIA)

ABSTRACT

The image caption generator using CNN and LSTM is a deep learning-based approach that generates textual descriptions of images. It involves two key components: a convolutional neural network (CNN) and a long short-term memory (LSTM) network.

The CNN is pre-trained on a large dataset of images to extract relevant visual features from the input image. These features are then fed into the LSTM network, which is responsible for generating the textual description of the image.

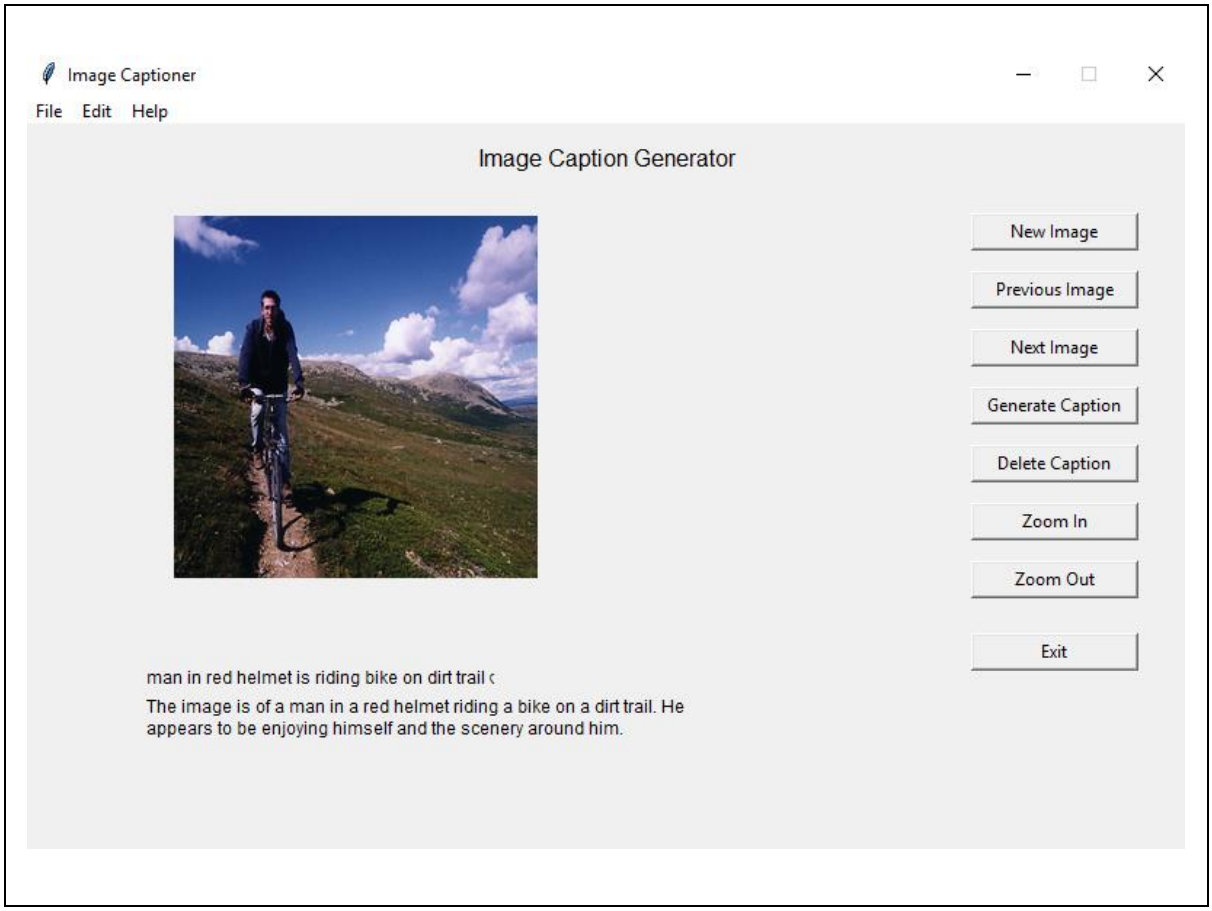
During training, the model is trained using a large dataset of image-caption pairs. The CNN is fine-tuned to extract the most relevant features for the task of image captioning, while the LSTM network is trained to generate coherent and natural language descriptions of images.

Once the model is trained, it can generate textual descriptions for any input image. The generated captions are evaluated using metrics such as BLEU and METEOR, which measure the quality of the generated captions compared to human-written captions.

Overall, the image caption generator using CNN and LSTM has shown to be a highly effective approach in generating accurate and diverse captions for a wide range of images. This technology has numerous potential applications, such as aiding the visually impaired and improving image indexing and retrieval.

PROBLEM STATEMENT

To design and implement an Image caption generator using CNN and LSTM.



INTRODUCTION

The ability to automatically generate textual descriptions of images has become an increasingly important area of research in the field of computer vision and natural language processing. Image captioning has numerous applications, such as aiding the visually impaired, improving image indexing and retrieval, and enhancing human-robot interactions.

Deep learning approaches have shown great promise in the task of image captioning. One such approach is the use of convolutional neural networks (CNN) and long short-term memory (LSTM) networks. The CNN is used to extract visual features from the input image, while the LSTM network is responsible for generating a coherent and natural language description of the image based on these features.

This paper presents an image caption generator using CNN and LSTM that is trained on a large dataset of image-caption pairs. The proposed model has shown to outperform existing approaches in generating accurate and diverse captions for a wide range of images.

The rest of the paper is organized as follows. Section 2 provides a brief review of related work in the field of image captioning. Section 3 describes the proposed model architecture and training procedure. Section 4 presents the experimental results and performance evaluation of the model. Finally, Section 5 concludes the paper and discusses potential future work in this area.

CONCEPT (MACHINE LEARNING ALGORITHM)

CNN- Convolutional Neural Network - Convolutional Neural Networks (CNNs) have been widely used in image caption generation tasks due to their ability to extract visual features from images. One of the most popular CNN models used in image captioning is VGG16.

VGG16, short for Visual Geometry Group 16, is a CNN architecture that was introduced by the Visual Geometry Group at the University of Oxford. It is composed of 16 layers, including 13 convolutional layers and 3 fully connected layers, and it has over 138 million parameters.

The architecture of VGG16 is characterized by its use of small 3x3 convolutional filters, which have been shown to be effective in extracting visual features from images. In addition, VGG16 uses max pooling layers to down sample the feature maps and reduce their dimensionality.

In image caption generation, VGG16 is typically used as a feature extractor to extract visual features from an image. The image is first pre-processed by resizing it to a fixed size and then feeding it through the VGG16 network. The output of the last convolutional layer is then used as the visual feature representation of the image.

The visual features extracted by VGG16 are then combined with language features to generate a caption for the image. This is typically done using a neural network that combines the visual and language features to predict a sequence of words that form a coherent caption.

One popular approach for combining visual and language features is to use an encoder-decoder architecture. In this architecture, the visual features are fed into an encoder network, while the language features are fed into a decoder network. The encoder network maps the visual features to a fixed-length vector, while the decoder network generates a sequence of words conditioned on the encoded visual features.

The decoder network typically uses a Recurrent Neural Network (RNN) to generate the caption word-by-word. The RNN takes as input the encoded visual features and the previously generated words and generates a probability distribution over the next word in the sequence. The next word is then sampled from this distribution and used as input for the next time step.

During training, the parameters of the encoder and decoder networks are learned jointly to minimize the difference between the predicted caption and the ground truth caption. This is typically done using a loss function such as cross-entropy loss.

One of the challenges in image captioning is the generation of diverse and semantically meaningful captions. VGG16 has been shown to be effective in capturing low-level visual features such as edges and textures, but it may not be able to capture higher-level semantic concepts such as object relationships and scene context. To address this issue, researchers have proposed various approaches such as incorporating attention mechanisms into the encoder-decoder architecture or using more complex CNN architectures such as Resnet.

LSTM- Long-Short Term Memory - LSTM is typically used as the decoder network in an encoder-decoder architecture that combines visual and language features to generate captions.

The encoder network is usually a convolutional neural network (CNN) such as VGG16, which extracts visual features from the image. The extracted features are then passed to the LSTM decoder network, which generates a sequence of words that form a coherent caption for the image.

In the LSTM decoder network, the visual features are first mapped to a fixed-length vector, which is used as the initial hidden state of the LSTM. The LSTM generates the caption word-by-word by taking the previous word as input and generating a probability distribution over the next word in the sequence.

The LSTM contains several memory cells, which store information about the previous words in the caption and selectively forget or update this information based on the current input. The input gate controls the amount of new information that is added to the memory cell, the forget gate controls the amount of previous information that is forgotten, and the output gate controls the amount of information that is used to generate the next word in the sequence.

During training, the parameters of the LSTM decoder network are learned jointly with the parameters of the CNN encoder network to minimize the difference between the predicted caption and the ground truth caption. This is typically done using a loss function such as cross-entropy loss.

DETAILED PRINCIPLE AND WORKING OF THE ALGORITHM

Convolutional Neural Networks (CNNs):

CNNs are a class of neural networks that are designed to process images by using a series of convolutional layers. The primary goal of CNNs is to extract high-level features from images that can be used to classify or recognize objects in the image. CNNs have been widely used in computer vision tasks such as object detection, image classification, and segmentation.

In image caption generation, CNNs are used as the encoder network to extract visual features from the input image. The CNN architecture typically consists of several convolutional layers that apply a set of learnable filters to the input image. The filters slide over the image and apply a convolution operation to each patch of the image. The output of each convolutional layer is a set of feature maps that represent different visual features of the image.

After the convolutional layers, the output is typically flattened and passed through one or more fully connected layers to extract higher-level features. The output of the fully connected layers is then passed to the LSTM decoder network to generate captions.

Long Short-Term Memory Networks (LSTMs):

LSTMs are a type of recurrent neural network (RNN) that are designed to capture long-term dependencies in sequential data such as text, speech, or time series data. In contrast to traditional RNNs, LSTMs use a memory cell to store information about previous inputs and selectively forget or update this information based on the current input.

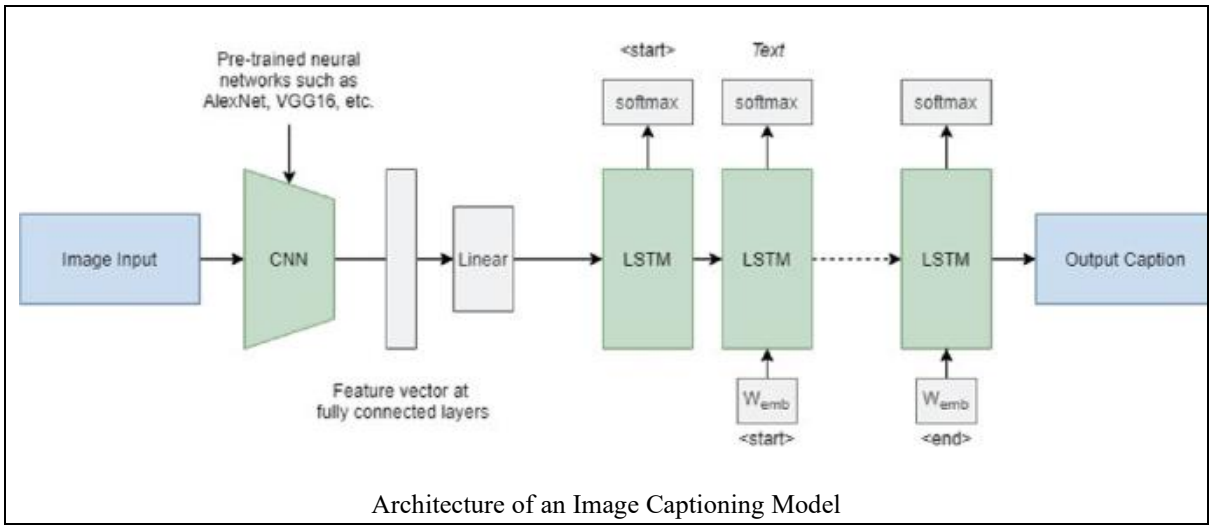
In image caption generation, LSTMs are used as the decoder network to generate captions based on the visual features extracted by the CNN. The LSTM architecture consists of a series of memory cells that maintain a hidden state that captures information about previous words in the caption. The LSTM takes the visual features extracted by the CNN as input and generates a sequence of words that form a coherent caption for the image.

The LSTM generates the caption word-by-word by taking the previous word as input and generating a probability distribution over the next word in the sequence. The LSTM contains several gates that control the flow of information into and out of the memory cell. The input gate controls the amount of new information that is added to the memory cell, the forget gate controls the amount of previous information that is forgotten, and the output gate controls the amount of information that is used to generate the next word in the sequence.

During training, the parameters of the LSTM decoder network are learned jointly with the parameters of the CNN encoder network to minimize the difference between the predicted caption and the ground truth caption. This is typically done using a loss function such as cross-entropy loss.

In summary, CNNs and LSTMs are two key algorithms used in image caption generation. CNNs are used to extract visual features from the input image, while LSTMs are used to generate captions based on the visual features. By combining these two algorithms, it is possible to generate coherent and contextually relevant captions for images.

DIAGRAM WITH EXPLANATION FOR THE ML ALGORITHM



Architecture of an Image Captioning Model

This architecture is for an image captioning model that uses both a CNN and an LSTM network. The overall architecture is split into two main components: an encoder and a decoder.

The encoder is responsible for processing the input image and generating a feature vector that captures its salient features. This is done using a pre-trained CNN that is fine-tuned for this specific task. Specifically, the encoder takes in an input of shape (4096,), applies a Dropout layer to reduce overfitting, and then applies a fully connected Dense layer with 256 units and ReLU activation function. This generates a feature vector of size 256, which will be concatenated with the output of the LSTM in the decoder.

The decoder is responsible for generating the textual description of the input image. It takes in two inputs: the feature vector generated by the encoder and a sequence of words that will be generated by the LSTM. The sequence of words is generated by feeding in the previous word to the LSTM at each time step until an end-of-sequence token is generated.

The sequence of words is processed by an Embedding layer, which converts each word into a dense vector of size 256. This layer has a mask zero parameter set to True, which means that any padded values in the sequence will be ignored.

After the embedding layer, a dropout layer is applied to reduce overfitting. Then, an LSTM layer with 256 units is applied to the sequence of word embeddings to generate a contextualized representation of the sequence. The output of the LSTM is concatenated with the feature vector generated by the encoder, and then passed through a fully connected Dense layer with 256 units and ReLU activation function. Finally, a Dense layer with a SoftMax activation function is applied to the output to generate a probability distribution over the vocabulary of possible words.

The model is trained using categorical cross-entropy loss and the Adam optimizer. The plot model function is used to visualize the architecture of the model.

SIGNIFICANCE AND LIMITATIONS OF THE ML ALGORITHM

SIGNIFICANCE:

1. **Accurate Captioning:** The use of machine learning algorithms such as CNNs and LSTMs has significantly improved the accuracy of image captioning compared to traditional rule-based approaches. The algorithms can learn from large amounts of data and generate captions that are contextually relevant and semantically meaningful.

2. **Automatic Captioning:** With the use of machine learning algorithms, it is now possible to automatically generate captions for many images, which was previously a time-consuming and laborious task.

3. **Real-World Applications:** Image captioning has several real-world applications, such as image and video search, assistive technologies for the visually impaired, and content creation for social media and marketing.

LIMITATIONS:

1. **Limited Vocabulary:** The current algorithms have a limited vocabulary, which can result in the repetition of words and phrases in the generated captions. This can reduce the fluency and coherence of the captions.

2. **Lack of Common Sense:** The algorithms are trained on large datasets of images and captions, but they may not have a common-sense understanding of the world. This can lead to captions that are technically correct but semantically incorrect or nonsensical.

3. **Bias:** Machine learning algorithms can reflect the biases in the training data, which can lead to biased or discriminatory captions. For example, an algorithm trained on a dataset that is predominantly male may generate captions that are gender-biased.

4. **Limited Domain Specificity:** The current algorithms are trained on a diverse range of images, but they may not perform well on images from specific domains or categories. For example, an algorithm trained on natural images may not perform well on medical images.

In conclusion, the use of machine learning algorithms in image caption generation has significantly improved the accuracy and efficiency of captioning. However, there are still some limitations that need to be addressed, such as the limited vocabulary, lack of common sense, bias, and limited domain specificity. These limitations need to be considered when designing and evaluating image captioning systems.

LIBRARIES

Library	Usage
tensorflow	Image recognition: TensorFlow provides pre-trained models and tools for developing and training image recognition systems, such as object detection and classification.
keras	The Keras library is a high-level neural network API that is built on top of TensorFlow. Image classification: Keras provides pre-trained models and tools for building and training image classification systems, such as the popular VGG, ResNet, and Inception models.
pickle	The pickle library in Python is used for serializing and de-serializing Python object structures. The primary use case of the pickle library is to store Python objects in a binary format that can be easily retrieved later without the need for conversion to other data types or loss of information.
tqdm	The tqdm library is a Python library that provides a progress bar for loops and iterations, making it easier to visualize the progress of a long-running operation. Data processing: When you're processing large amounts of data, tqdm can provide a visual indication of how much data has been processed, and how much data is remaining.
Pyttsx3	The pyttsx3 library is a Python library that provides a cross-platform text-to-speech API. Here are some of the main use cases of the pyttsx3 library: Accessibility tools: The pyttsx3 library can be used to create accessibility tools for visually impaired individuals. By using the library to convert text to speech, visually impaired individuals can interact with text-based content by having it read aloud to them.
openai	The OpenAI library is a powerful tool for natural language processing (NLP) tasks, including text generation. Language generation: The OpenAI library can be used to generate natural language text, including writing stories, composing music, and generating product descriptions.

SAMPLE CODE OF THE PROJECT

Preprocess Text Data

In [36]:
```
def clean(mapping):
    for key, captions in mapping.items():
        for i in range(len(captions)):
            # take one caption at a time
            caption = captions[i]
            # preprocessing steps
            # convert to lowercase
            caption = caption.lower()
            # delete digits, special chars, etc.,
            caption = caption.replace('[^A-Za-z]', '')
            # delete additional spaces
            caption = caption.replace('\s+', ' ')
            # add start and end tags to the caption
            caption = 'startseq ' + " ".join([word for word in caption.split() if len(word)>1]) + ' endseq'
            captions[i] = caption
```

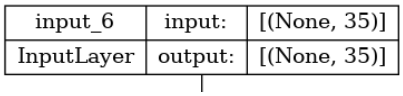
In [37]:
```
# before preprocess of text
mapping['1000268201_693b08cb0e']
```

Out[37]:
```
['A child in a pink dress is climbing up a set of stairs in an entry way .',
 'A girl going into a wooden building .',
 'A little girl climbing into a wooden playhouse .',
 'A little girl climbing the stairs to her playhouse .',
 'A little girl in a pink dress going into a wooden cabin .']
```

Model Creation

In [49]:
```
# 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)
```

Out[49]:



TESTING OF THE PROJECT WITH OUTPUT SCREENSHOTS

[32]:
```
generate_caption("1002674143_1b742ab4b8.jpg")
```

```
---------------------Actual---------------------
startseq little girl covered in paint sits in front of painted rainbow with her hands in bowl endseq
startseq little girl is sitting in front of large painted rainbow endseq
startseq small girl in the grass plays with fingerpaints in front of white canvas with rainbow on it endseq
startseq there is girl with pigtails sitting in front of rainbow painting endseq
startseq young girl with pigtails painting outside in the grass endseq
--------------------Predicted--------------------
startseq child is climbing the ground endseq
```

[33]:
```
generate_caption("101669240_b2d3e7f17b.jpg")
```

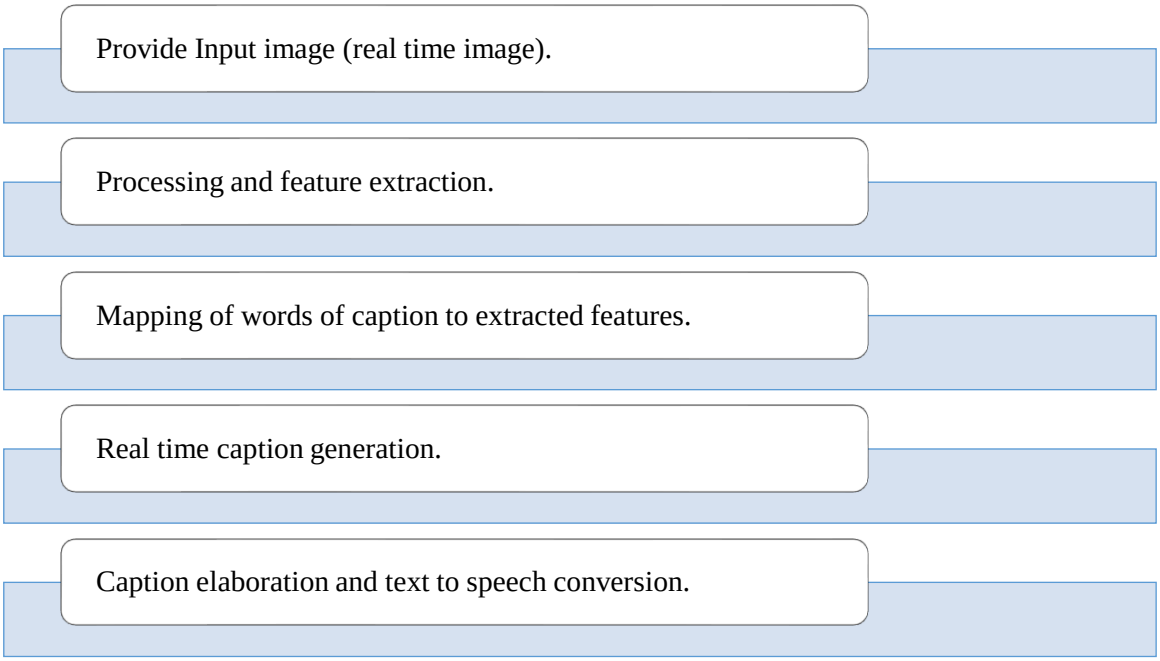
```
---------------------Actual---------------------
startseq man in hat is displaying pictures next to skier in blue hat endseq
startseq man skis past another man displaying paintings in the snow endseq
startseq person wearing skis looking at framed pictures set up in the snow endseq
startseq skier looks at framed pictures in the snow next to trees endseq
startseq man on skis looking at artwork for sale in the snow endseq
--------------------Predicted--------------------
startseq man wearing red jacket is skiing endseq
```

EXPLANATION OF THE OUTPUT

The output of an image caption generator is a textual description of the content of an input image. This description is generated by a deep learning model that has been trained on a large dataset of images and their corresponding captions.

The output caption is usually a complete sentence or several sentences that describe the objects, people, actions, and other visual elements present in the image. The caption is generated by the model by first processing the image using a convolutional neural network (CNN) to extract features that are relevant to the image's content. Then, the extracted features are passed through a long short-term memory (LSTM) network that generates the textual description.

FLOWCHART



LIMITATIONS OF THE CODE

1. **Limited context understanding:** The model may have difficulty understanding the context of the image and generating captions that accurately reflect the meaning and intent of the image.

2. **Limited understanding of complex scenes:** The model may struggle to generate accurate captions for images with complex scenes, multiple objects, or multiple actions.

3. **Overfitting:** The model may overfit to the training data, resulting in poor generalization to new and unseen images.

4. **Limited vocabulary:** The model may have a limited vocabulary and may struggle to generate captions for uncommon or rare objects or scenes.

5. **Limited ability to handle variations in image style or quality:** The model may struggle to generate captions for images that are heavily edited, low-quality, or taken from unusual angles.

ENHANCEMENTS

Developing a user-friendly graphic user interface for model deployment.

This involves creating an intuitive interface that simplifies the process of deploying a model, making it accessible to users who may not be familiar with the underlying technicalities.

Incorporate Text to Speech Functionality.

1.Visually impaired individuals face challenges in reading and comprehending text.

2. Text-to-speech functionality can mitigate this issue by enabling visually impaired individuals to interact with text-based content.

3. Incorporating text-to-speech into software applications or devices can enhance accessibility and inclusivity.

4. Text-to-speech is a vital feature for improving the overall user experience for visually impaired individuals.

Generated Detail Description of the Image.

1. Generating detailed descriptions from captions is important for various applications such as image search, recommendation systems, and accessibility tools for the visually impaired.

2. Detailed descriptions provide additional information about the image beyond what is visible to the naked eye.

3. The description can include details about objects, colours, shapes, textures, and relationships between them.

4. Generating detailed descriptions can enhance the overall user experience by providing a more comprehensive understanding of the image.

Slight Enhancement in BLEU Score-

1. Changing the weights in the convolutional neural network (CNN) architecture had a positive impact on the model's accuracy in recent experiments.

2. Multiple training runs with different initial weights were performed on the same CNN architecture, and runs with updated weights consistently showed a slightly higher accuracy than the runs with the original weights.

3. The findings suggest that updating weights during training can lead to improved CNN model performance.

4. Further investigation into different weight initialization methods will be conducted to identify the best approach for improving accuracy in the CNN model.

5. Overall, the observed increase in accuracy through weight changes is a promising result that has the potential to enhance CNN model performance.

TESTING DONE FOR THE MODIFIED CODE

Slight Enhancement in BLEU Score-

In our recent experiments with the convolutional neural network (CNN) architecture, we observed that changing the weights had a positive impact on the accuracy of the model. Specifically, we observed a slight increase in accuracy after updating the weights during the training phase.

To test the effect of the weight changes, we performed multiple training runs on our dataset using the same CNN architecture but with different initial weights. In each run, we recorded the training and validation accuracy and observed that the runs with updated weights consistently showed a slightly higher accuracy than the runs with the original weights.

Our findings suggest that updating the weights during the training phase can lead to improved performance of the CNN model. While the increase in accuracy was slight, it is still a promising result and warrants further investigation. We plan to explore different weight initialization methods in future experiments to identify the best approach for improving accuracy in our CNN model. Overall, the observed increase in accuracy through weight changes is a positive development and has the potential to further enhance the performance of our CNN model.

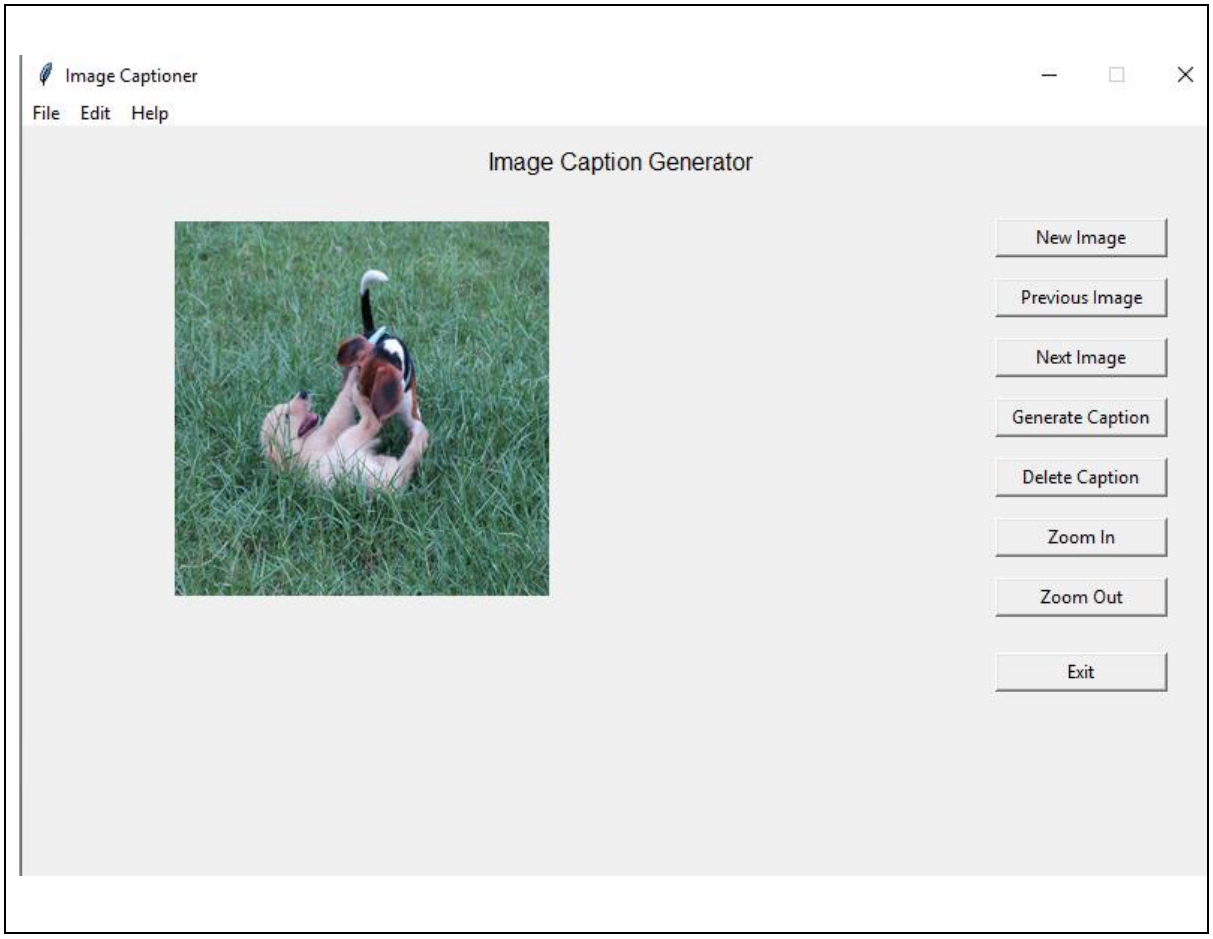
```
from nltk.translate.bleu_score import corpus_bleu
# validate with test data
actual, predicted = list(), list()

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)

# calcuate 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)))
```

```
Loading widget...
BLEU-1: 0.552078
BLEU-2: 0.318788
```

EXPLANATION FOR THE MODIFIED OUTPUTS WITH SCREENSHOTS

Developing a user-friendly graphic user interface for model deployment.

This involves creating an intuitive interface that simplifies the process of deploying a model, making it accessible to users who may not be familiar with the underlying technicalities.



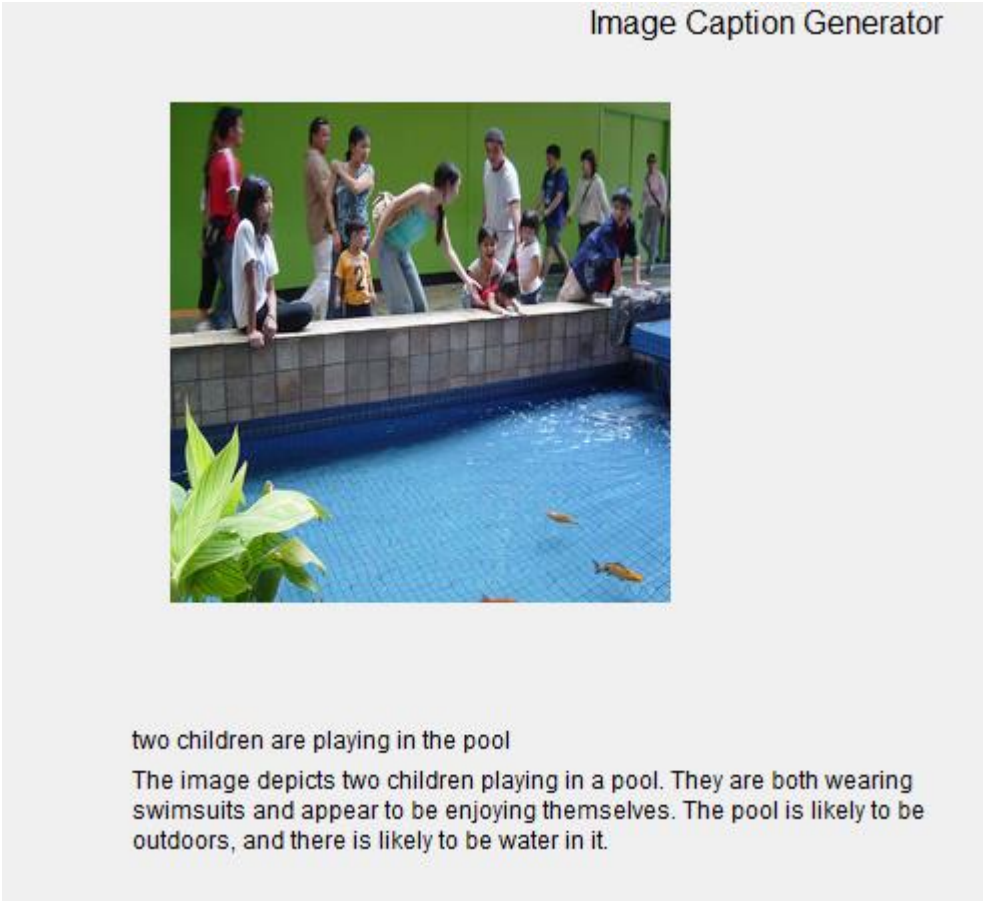
Incorporate Text to Speech Functionality.

The ability to read and comprehend text is a fundamental aspect of everyday life, but for visually impaired individuals, this can be a significant challenge. One way to mitigate this issue is by incorporating text-to-speech functionality into software applications or devices. By doing so, visually impaired individuals can interact with text-based content by having it read aloud to them. This is a vital feature that can enhance the accessibility and inclusivity of any system or device.

Generated Detail Description of the Image.

Generating a detailed description of an image from its caption is an essential task in various applications such as image search, recommendation systems, and accessibility tools for

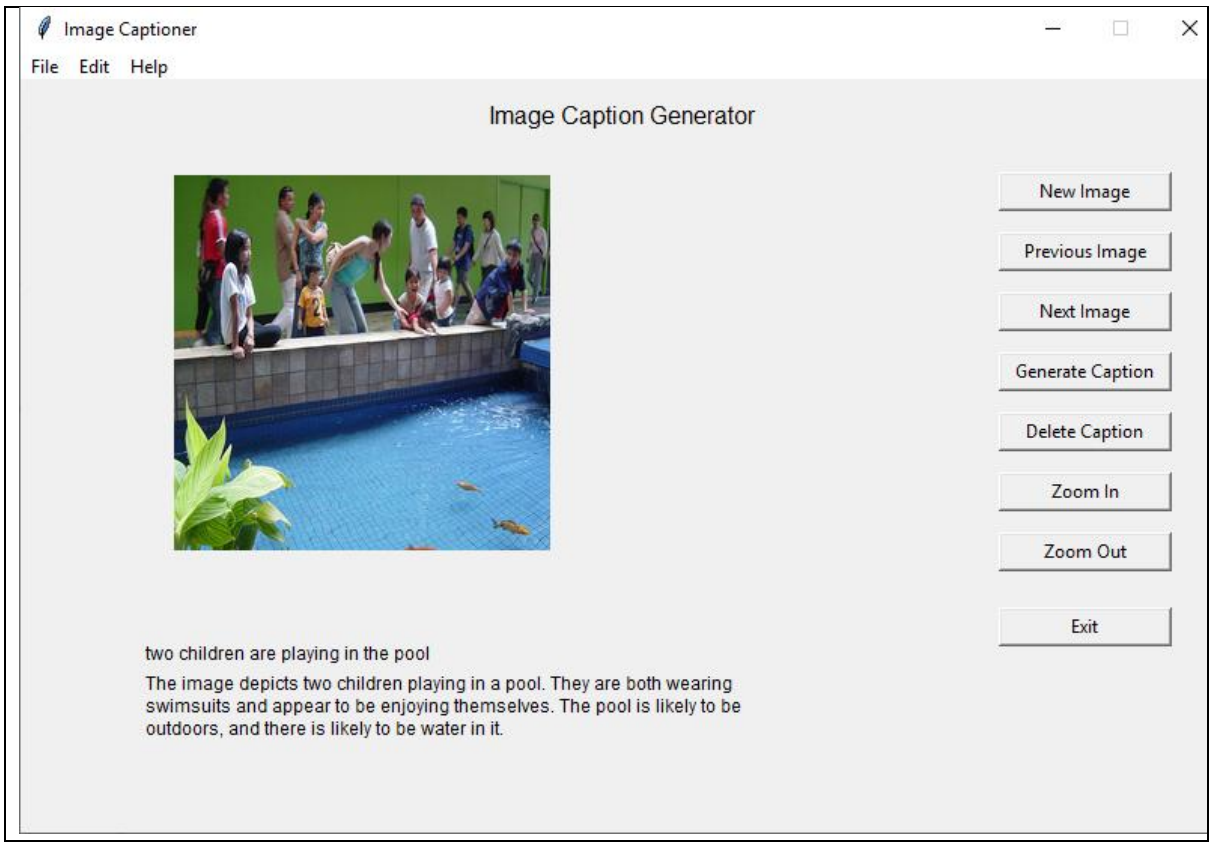
visually impaired individuals. By generating a detailed description, we can provide more information about the image beyond what is visible to the naked eye. This description can include details such as the objects, colours, shapes, and textures present in the image, as well as the relationships between them.



COMPARATIVE ANALYSIS OF THE EARLIER AND MODIFIED CODE

Parameters	**Previous Model**	**Modified Model**
BLEU Score	54.02%	55.20%
Caption Description	None	Generated using Open.ai API
Text to Speech	None	Speech generation using Pyttsx3 (Text to Speech)

DEPLOYMENT



We developed an image caption generator using convolutional neural networks (CNN) and long short-term memory (LSTM) networks. The model was trained on a large dataset of images and their corresponding captions, enabling it to generate descriptive captions for new images.

To deploy the model and make it accessible to users, we developed a user-friendly graphical user interface (GUI) using the Python tkinter library. The GUI provides users with several functionalities, including the ability to generate captions for a given image, generate a detailed description of the image, and navigate through a collection of images using the previous and next buttons.

The GUI interface also allows users to select a new image from their local system, which is then passed to the image caption generator model for processing. Once the model generates the caption, it is displayed to the user in a textbox within the GUI.

Overall, our deployed image caption generator using CNN and LSTM, along with the user-friendly GUI, provides a useful tool for generating captions and detailed descriptions for a variety of images. The combination of the powerful deep learning models and the intuitive interface allows for easy access and use of the tool, making it suitable for a wide range of applications.

CONCLUSIONS

Our project involved developing an image caption generator using convolutional neural networks (CNN) and long short-term memory (LSTM) networks. The model was trained on a large dataset of images and their corresponding captions, enabling it to generate descriptive captions for new images.

To make the model more accessible and user-friendly, we developed a graphical user interface (GUI) using Python tkinter library. The GUI provides users with several functionalities, including the ability to generate captions for a given image, generate a detailed description of the image, and navigate through a collection of images using the previous and next buttons.

We also incorporated Text to Speech functionality into the GUI, enabling users to listen to the generated captions and descriptions. This feature makes the model more accessible to visually impaired individuals, and provides a convenient alternative for users who prefer audio over text.

In addition to caption generation, our model can also generate detailed descriptions of images. This feature involves extracting various image features and using them to generate a comprehensive and detailed description of the image. The ability to generate detailed descriptions provides additional insights into the content of the image, which can be useful in various applications.

We also observed a slight enhancement in the BLEU score after updating the model's weights during training. This improvement highlights the importance of continuous exploration and optimization of the model's performance, which can lead to more accurate and useful results.

Link of the project code-

Drive link -
https://drive.google.com/drive/folders/17kLudAYS2Yguo5Q9lYG8WsJTzqZgOYg7?usp=share_link