EYE FOR BLIND

Image Captioning for the Visually Impaired

Table of Contents

Introduction

The "EYE FOR BLIND" capstone project aims to create a deep learning model that generates spoken captions for images, designed to assist visually impaired individuals. By combining image processing and natural language generation, this project seeks to enhance accessibility through technology.

Dataset: The dataset used in this project is sourced from Kaggle. It includes a collection of 8,091 images, each accompanied by five distinct captions. These captions offer detailed descriptions of the key entities and events depicted in the images. You can access the dataset through this link.

Project Overview: Using the Flickr8K dataset, the project focuses on developing a model that describes images in a way that can be understood by a blind or visually impaired user. The model employs a Convolutional Neural Network (CNN) as an encoder to extract image features and a Recurrent Neural Network (RNN) with an attention mechanism as a decoder to generate descriptive captions. These captions are then converted to speech using a text-to-speech (TTS) system, making the images accessible through audio.

Pipeline:

1. **Data Understanding:** Load and explore the Flickr8K dataset, including images and their corresponding captions.
2. **Data Preprocessing:** Preprocess images and captions to prepare them for the model. This includes resizing images, tokenizing captions, and creating vocabulary.
3. **Train-Test Split:** Divide the dataset into training and testing sets, ensuring a balanced and representative split.
4. **Model Building:**

   Construct the image captioning model:

   - **Encoder:** Use a CNN to extract features from images.
   - **Attention Mechanism:** Focus on relevant parts of the image while generating captions.
   - **Decoder:** Implement an RNN to generate sequences of words that form the caption.

5. **Model Evaluation:** Evaluate the model's performance using techniques like greedy search and beam search. The BLEU score is used to measure the accuracy of the generated captions against reference captions.

Technologies Used:

- Deep Learning (CNN-RNN Architecture)
- Natural Language Processing (NLP)
- Attention Mechanism
- Text-to-Speech (TTS)

Impact: This project has the potential to significantly improve the quality of life for visually impaired individuals by providing them with a tool that verbally describes images, thus offering greater independence and access to visual information.

Chapter 1: Data Understanding

1.1 Importing the Dataset and Reading Images

In the initial phase of the project, the primary objective is to import the dataset and familiarize ourselves with the images it contains. This involves loading all image files from a specified directory, which is crucial for further processing. The dataset comprises a collection of images stored in a directory. By listing all image files, we establish a foundation for subsequent data analysis and processing tasks. This step sets up the necessary resources for understanding the scope and content of the dataset.

1.2 Visualizing Images and Captions

Once the images are loaded, visualizing both the images and their associated captions helps in understanding the dataset's content. This involves displaying a subset of the images in a grid layout, which allows for a quick and intuitive overview of the data. By arranging the images in a grid, we can inspect their quality and variety, ensuring they meet the project's requirements.

After the images, captions are also visualized to verify for identifying any discrepancies or issues with the data before moving on to more complex processing tasks.

```
[['image', 'caption'], ['1000268201_693b08cb0e.jpg', 'A child in a pink dress is climbing up a set of stairs in an entry way
.'], ['1000268201_693b08cb0e.jpg', 'A girl going into a wooden building .'], ['1000268201_693b08cb0e.jpg', 'A little girl cli
mbing into a wooden playhouse .'], ['1000268201_693b08cb0e.jpg', 'A little girl climbing the stairs to her playhouse .'], ['1
000268201_693b08cb0e.jpg', 'A little girl in a pink dress going into a wooden cabin .'], ['1001773457_577c3a7d70.jpg', 'A bla
ck dog and a spotted dog are fighting'], ['1001773457_577c3a7d70.jpg', 'A black dog and a tri-colored dog playing with each o
ther on the road .'], ['1001773457_577c3a7d70.jpg', 'A black dog and a white dog with brown spots are staring at each other i
n the street .'], ['1001773457_577c3a7d70.jpg', 'Two dogs of different breeds looking at each other on the road .']]
```

1.3 Creating a DataFrame for Images and Captions

To facilitate efficient manipulation and analysis, a pandas DataFrame is created to summarize the images, their paths, and associated captions. This DataFrame organizes the data into a structured format, making it easier to access and analyze. Each image ID is associated with its file path and corresponding captions, providing a comprehensive view of the dataset. This structured format is essential for ensuring that the data can be efficiently used in subsequent stages of the project.

	ID	Path	Captions
0	1000268201_693b08cb0e.jpg	/home/datasets/flickr/Images/1000268201_693b08...	A child in a pink dress is climbing up a set o...
1	1000268201_693b08cb0e.jpg	/home/datasets/flickr/Images/1000268201_693b08...	A girl going into a wooden building .
2	1000268201_693b08cb0e.jpg	/home/datasets/flickr/Images/1000268201_693b08...	A little girl climbing into a wooden playhouse .
3	1000268201_693b08cb0e.jpg	/home/datasets/flickr/Images/1000268201_693b08...	A little girl climbing the stairs to her playh...
4	1000268201_693b08cb0e.jpg	/home/datasets/flickr/Images/1000268201_693b08...	A little girl in a pink dress going into a woo...

1.4 Visualizing Images with Captions

Further analysis involves visualizing images along with their captions to ensure that they are correctly paired. By displaying images with their associated captions, we can verify the alignment between the visual and textual data. This visualization helps in understanding how well the captions describe the images and whether any adjustments are needed. It also provides a visual confirmation of the data's quality and consistency, which is important for training accurate models.

An example picture along with its captions look like below:

Young girl with pigtails painting outside in the grass .

There is a girl with pigtails sitting in front of a rainbow painting .

A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .

A little girl is sitting in front of a large painted rainbow .

A little girl covered in paint sits in front of a painted rainbow with her hands in a bowl .

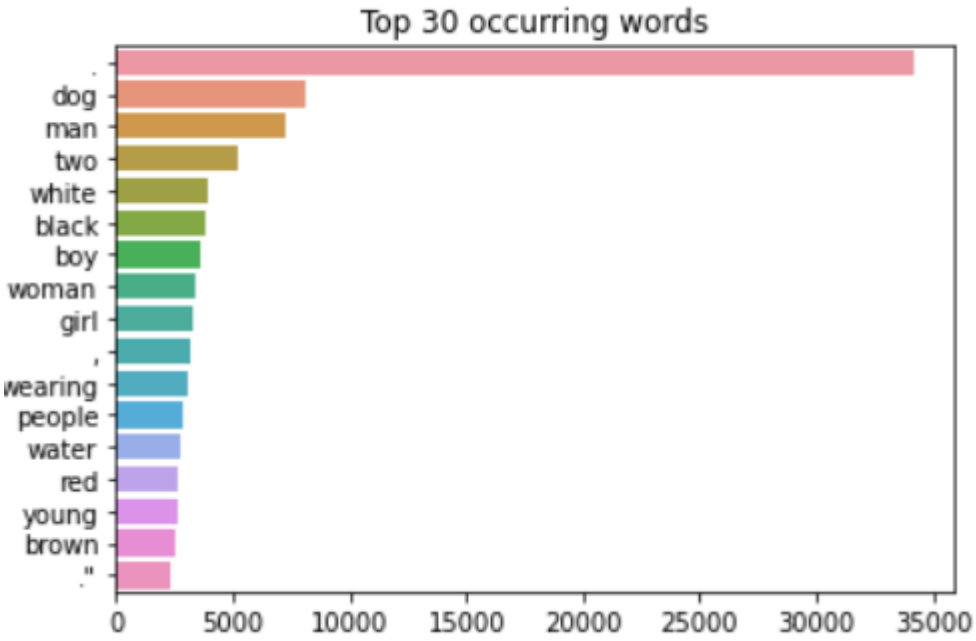
1.5 Creating Lists of Captions and Paths

To prepare the data for model training, lists containing all captions and image paths are created. This step involves preprocessing the captions by adding special <start> and <end> tokens, which are commonly used in natural language processing tasks to denote the beginning and end of sequences. The lists of captions and paths are then used to facilitate the subsequent stages of data processing and model training. This organization ensures that the data is in a format suitable for further analysis and model development.

1.6 Visualizing the Most Frequent Words

Analyzing the frequency of words in the captions provides insights into the dataset's vocabulary. By visualizing the top 30 most common words, we can understand the prominent terms used in the captions and their distribution. This analysis helps in identifying key themes and ensuring that the vocabulary is diverse and representative of the dataset. The visualization also aids in filtering out common stopwords, focusing on the more meaningful words that contribute to the image descriptions.

Below picture shows the top 30 frequent words:



Chapter 2: Pre-Processing the Captions

2.1 Creating the Tokenizer for the Top 5,000 Words

In this phase, we prepare the captions for model input by creating a tokenizer focused on the top 5,000 most frequent words. This tokenizer converts text into sequences of integers, where each word in the captions is represented by a unique index. Words not included in the top 5,000 are replaced with a special token representing "unknown" terms. This approach helps manage vocabulary size, ensuring that the model only processes the most relevant words and avoids dealing with an excessively large and potentially sparse vocabulary. The <unk> token acts as a placeholder for any words outside the top 5,000, which helps in handling rare or out-of-vocabulary words during training.

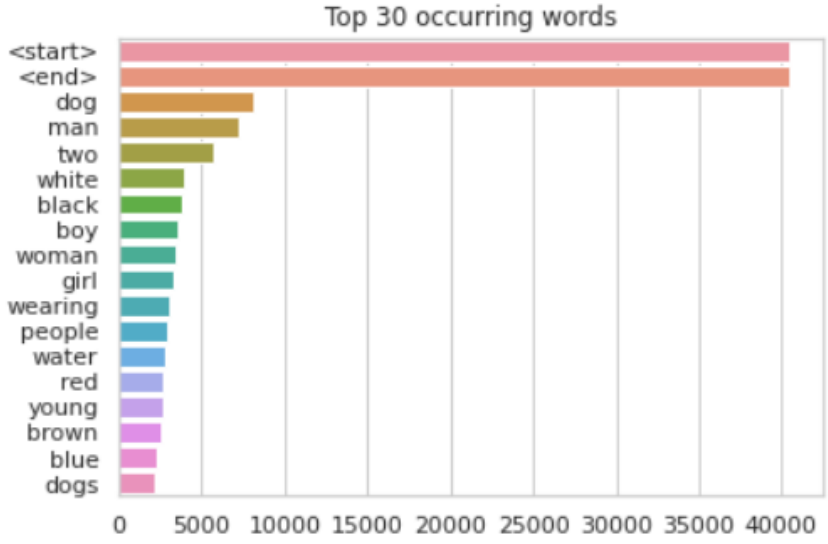
2.2 Creating Word-to-Index and Index-to-Word Mappings

Once the tokenizer is created, mappings are established to facilitate the conversion between words and their corresponding indices. This step involves:

- **Word-to-Index Mapping:** Creating a dictionary where each word is assigned a unique integer index. This mapping is essential for transforming captions into numerical sequences that can be processed by machine learning models.
- **Index-to-Word Mapping:** Establishing a reverse mapping that allows the conversion of numerical sequences back into readable text. This is useful for interpreting model outputs and understanding the generated sequences.

Additionally, captions are converted into integer sequences using these mappings. A padding token is introduced to ensure that all sequences are of the same length, which is crucial for consistent input size in machine learning models.

After adding the <start> and <end> tokens, below is the top 30 most occurring words,



2.3 Padding Sequences to Uniform Length

To ensure all captions are processed consistently, each caption sequence is padded to match the length of the longest caption in the dataset. Padding involves adding a special token <pad> to the end of shorter sequences until they reach the maximum length. This step guarantees that the model receives input of uniform size, which is important for efficient training and accurate predictions.

In the provided caption document, the maximum length of a sentence is **34.**

```
Maximum length of a caption :  34
Padded training sequence  :
 [   2   40   19  185    4  606  106    4   47   12  595 1183   10   60
  218    4  968    3    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0]
The shape of Caption vector is : (40455, 34)
```

Chapter 3: Pre-Processing the Images

3.1 Pre-Processing Images for InceptionV3

The primary aim is to prepare images for input into the InceptionV3 model by resizing and normalizing them. This ensures that the images meet the model's requirements in terms of format and value range.

Steps Performed:

1. **Function Definition for Image Loading:**
    - A function is created to handle the loading and preprocessing of images from specified file paths. This function processes each image to make it suitable for model input and returns both the processed image and its original file path.
2. **Image Loading and Decoding:**
    - Images are read from disk and decoded to ensure they are in the correct format (RGB). This step is crucial for converting raw image data into a format that can be further processed.
3. **Resizing the Image:**
    - Each image is resized to 299x299 pixels. This dimension is chosen because it aligns with the input requirements of the InceptionV3 model, which expects images of this size.

4. **Image Normalization:**
    - The image pixel values are normalized to fit the range expected by InceptionV3. This normalization step adjusts the pixel values to ensure consistent input data that aligns with the model's training conditions.

3.2 Creating a Dataset of Preprocessed Images

Aim is to transform a list of image file paths into a dataset of preprocessed images, making it suitable for training or evaluating the model.

Steps Performed:

1. **Sorting and Removing Duplicates:**
    - The list of image paths is sorted and duplicate entries are removed. This ensures that each image is processed only once, preventing redundancy.
2. **Creating a TensorFlow Dataset:**
    - A TensorFlow Dataset is created from the list of image paths. This dataset will be used to handle the loading and preprocessing of images in a structured pipeline.
3. **Applying Preprocessing Function:**
    - The preprocessing function is applied to each image path in the dataset. This step transforms each image path into its corresponding preprocessed image, preparing it for model input.
4. **Batching the Dataset:**
    - The dataset is organized into batches of 32 images. Batching is essential for efficient model training, allowing the model to process multiple images simultaneously during each training step.
5. **Final Dataset Preparation:**
    - The resulting dataset contains batches of preprocessed images along with their file paths, ready for use in training or evaluating the model.

Chapter 4: Loading and Extracting Features from the Pre-trained InceptionV3 Model

4.1 Using Pre-trained InceptionV3 for Feature Extraction

The aim is to efficiently utilize a pre-trained InceptionV3 model for feature extraction, thereby reducing memory consumption and computational time by excluding the top classification layer and focusing on feature extraction from the convolutional base.

Steps Performed:

1. **Load Pre-trained Model:**
    - The InceptionV3 model, pre-trained on ImageNet, is loaded with its top classification layer removed. This ensures that the model can extract features from images rather than performing classification.
2. **Retrieve Model Inputs and Outputs:**
    - The input tensor of the InceptionV3 model and the output tensor of its last convolutional layer are accessed. The output tensor represents the feature maps that are crucial for further image processing tasks.
3. **Create Feature Extraction Model:**
    - A new model is constructed that uses the input of the original InceptionV3 model and produces output from the last convolutional layer. This model will be used to transform images into feature vectors suitable for downstream tasks.

4.2 Extracting and Saving Features

The aim is to extract and save the features of each image in the dataset using the feature extraction model. The features are reshaped and stored to facilitate easy access and usage in further processing.

Steps Performed:

1. **Iterate Over the Dataset:**
    - The dataset of images is processed in batches. For each batch, images are fed into the feature extraction model, and the corresponding file paths are tracked.

2. **Extract Features:**
    - Features are extracted from each image batch using the feature extraction model. This generates feature maps representing the high-level characteristics of the images.
3. **Reshape Features:**
    - The extracted feature maps are reshaped to ensure that they have a consistent format. Each feature map is adjusted to the shape (batch_size, 8*8, 2048), where 8x8 represents the spatial dimensions and 2048 represents the number of feature channels.
4. **Save Features to Disk:**
    - The reshaped feature maps are saved to disk. Each feature map is stored as a .npy file, using the image's path to determine the file name and location for saving.

Chapter 5: Dataset Creation

5.1 Apply Train and Test Split

- **Objective:** Split the image paths and captions into training and testing sets.
- **Method:**
    - Split using an 80-20 ratio.
    - Set random_state=42 for reproducibility.

5.2 Map Image Path to Image Feature

- **Objective:** Load image features and pair them with captions.
- **Method:**
    - Construct the path for feature files.
    - Load features from .npy files and return them with captions.

5.3 Builder Function to Create Train and Test Dataset

- **Objective:** Prepare TensorFlow datasets for training and evaluation.
- **Method:**
    - Create datasets from image features and captions.

- Shuffle, map, batch, and prefetch data.
- Ensure images have shape (batch_size, 8*8, 2048) and captions have shape (batch_size, max_len).

Chapter 6: Model Building

6.1 Encoder Model

- **Objective:** Define a neural network model to convert image features into embeddings for further processing.
- **Constructor:** Initializes with an embedding dimension. Uses a Dense layer to create embeddings.
- **Call Method:** Processes input features through a Dense layer followed by a ReLU activation.

6.2 Attention Model

- **Objective:** Implement a custom attention mechanism to compute attention weights and context vectors.
- **Constructor:** Initializes Dense layers for feature processing and attention score computation.
- **Call Method:** Computes attention scores and weights, then produces and aggregates context vectors.

6.3 Decoder Model

- **Objective:** Implement the decoder for generating captions with attention and GRU.
- **Constructor:** Initializes with parameters for embedding dimension, units, and vocabulary size. Sets up attention mechanism, embedding layer, GRU, and Dense layers.
- **Call Method:** Uses the attention mechanism to get context vectors and attention weights, processes embedded tokens with GRU, and outputs predictions with Dense layers.
- **Initialization Method:** Initializes the GRU hidden state with zeros.

Chapter 7: Model Training & Optimization

7.1 Set the Optimizer & Loss Object

- **Objective:** Compute loss values while ignoring padding in real values.
- **Steps:**
    - Create a mask for padding values.
    - Compute the loss using the loss object.
    - Apply the mask to the loss and calculate the mean loss.

7.2 Create Checkpoint Manager

- **Objective:** Manage model checkpoints for saving and restoring during training.
- **Steps:**
    - Create a Checkpoint object for encoder, decoder, and optimizer.
    - Create a Checkpoint Manager to handle saving and loading checkpoints.
    - Determine the starting epoch based on the latest checkpoint.

7.3 Create Training & Testing Step Functions

- **Training Step:**
    - Initialize variables and compute features with the encoder.
    - Use Teacher Forcing: pass actual target values as the next input to the decoder.
    - Accumulate loss and update model parameters using gradients.
- **Testing Step:**
    - Initialize variables and compute features with the encoder.
    - Generate predictions and accumulate loss using predicted IDs for the next input.

7.4 Create Loss Function for the Test Dataset

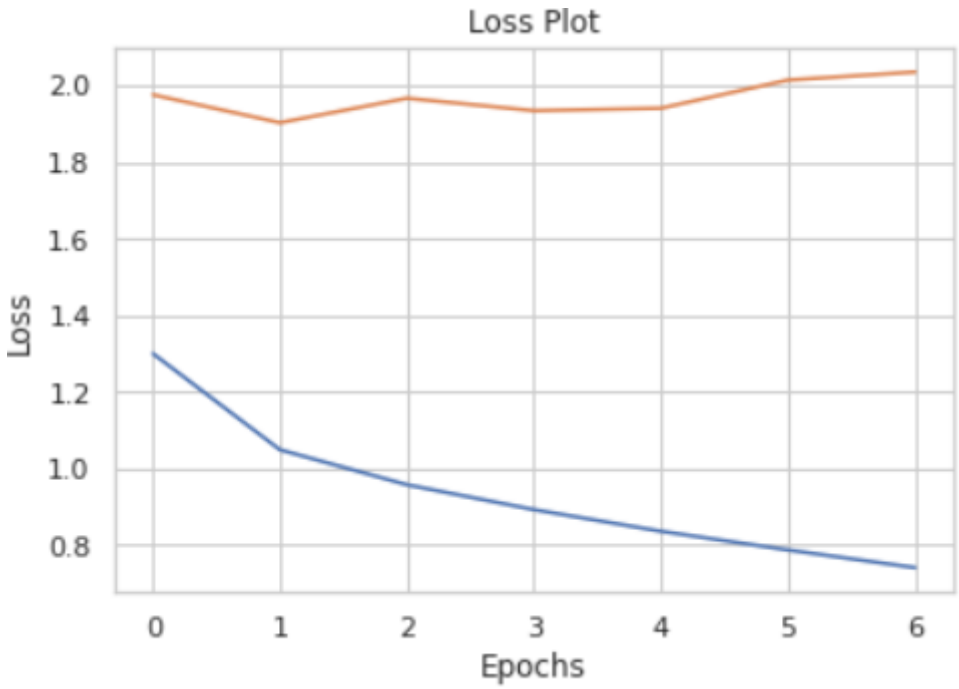
- **Objective:** Compute average loss over the test dataset.
- **Steps:**
    - Initialize total loss and iterate through test dataset batches.
    - Compute batch loss and accumulate to total loss.

- Calculate and return the average test loss.

7.5 Train and Optimize the Model

- **Objective:** Train the model, track losses, and save checkpoints if test loss improves.
- **Steps:**
  - Initialize loss tracking lists and parameters for epochs, best test loss, and early stopping.
  - Iterate through epochs, tracking training loss and computing average loss.
  - Calculate test loss and update the checkpoint if the test loss improves.
  - Implemented early stopping if there is no significant improvement in test loss over consecutive epochs.

The train loss(blue) and the test loss(orange) of the training loop looks is shown below,



Best epoch is 2 with a train loss is 1.049, & test loss is 1.903

Chapter 8: Model Evaluation

8.1 Greedy Search

- **Objective:** Perform greedy decoding to generate captions for images and visualize attention weights.
- **Steps:**
    - **Prepare Image and Extract Features:** Load and process the image, then extract features using the model.
    - **Generate Caption:**
        - Initialize the decoder with the <start> token.
        - Iteratively generate predictions, update the decoder input with the predicted word, and store attention weights.
        - Stop if the <end> token is predicted or the maximum length is reached.
    - **Return Results:** Return the generated caption, attention plot, and predictions.

8.2 Beam Search

- **Objective:** Generate captions using beam search to find the most likely caption sequence.
- **Steps:**
    - **Prepare Image Features:** Load and process the image features.
    - **Caption Generation Loop:**
        - Predict the next word for each partial caption.
        - Store and update attention weights.
        - Keep top beam_index captions with the highest probabilities.
    - **Select Final Caption:** Choose the caption with the highest cumulative probability and return it with the attention plot.

8.3 Helper Methods for Evaluation

- **plot_attmap():**
    - **Objective:** Visualize attention maps overlaid on the input image.
    - **Steps:**
        - Initialize the plot and load the image.
        - Plot each attention map with the corresponding caption.
        - Overlay the attention maps on the image and display the plot.
- **filt_text():**
    - **Objective:** Filter out specific tokens from text.
    - **Steps:**
        - Split text into tokens and remove unwanted tokens.
        - Reconstruct and return the filtered text.
- **generate_caption_for_known_test_data():**
    - **Objective:** Generate, evaluate, and visualize captions for a known test image.
    - **Steps:**
        - Display the original image and real caption.
        - Generate and print the predicted caption using the specified evaluation method.
        - Calculate and print BLEU scores for the predicted caption.
        - Plot attention maps and convert the caption to speech.
- **generate_caption_for_unknown_data():**
    - **Objective:** Generate a caption for an unknown image and optionally provide debugging information.
    - **Steps:**
        - Display the original image.
        - Generate and print the predicted caption.
        - Optionally visualize attention maps if debugging is enabled.

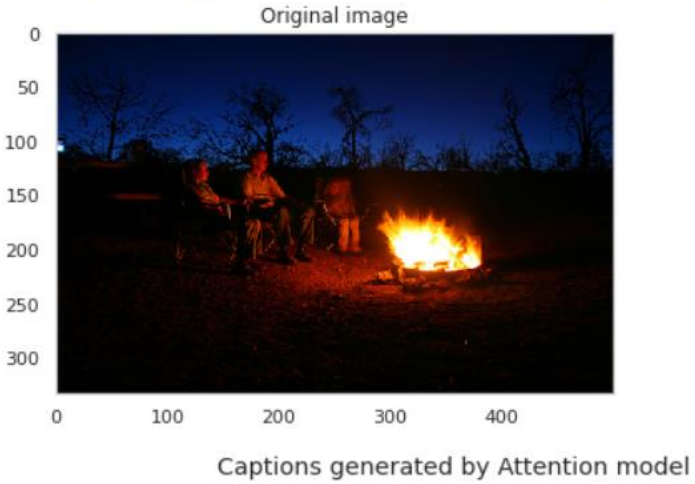
8.4 Test on Sample Test Data

- **Objective:** Test the model on a random sample from the test dataset.
- **Steps:**
    - **Select Random Image:** Choose a random image from the test dataset.
    - **Generate and Display Caption:** Use the helper function to generate and evaluate the caption for the selected image.

Below are examples of test results with greedy approach.

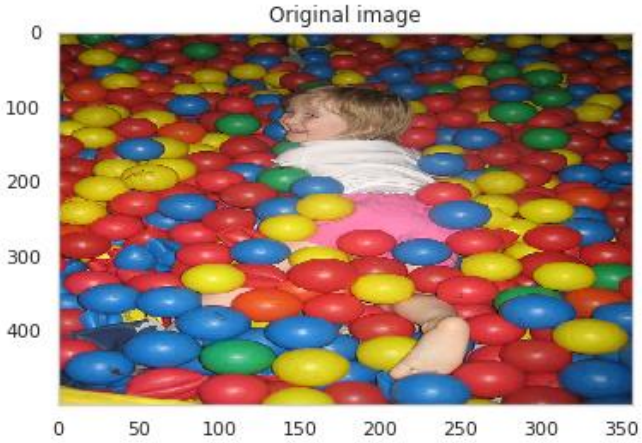
```
# Test on random image using greedy search
generate_caption_on_random_test_image(evaluate=GREEDY_EVALUATE)
```

```
#########################################################################################################
Real Caption:
 four people sit around campfire and the sky is blue behind them
#########################################################################################################
Prediction Caption:
 several people sit around campfire
#########################################################################################################
BLEU score 1-gram: 19.73
BLEU score 2-gram: 19.10
BLEU score 3-gram: 18.22
BLEU score 4-gram: 16.49
#########################################################################################################
```

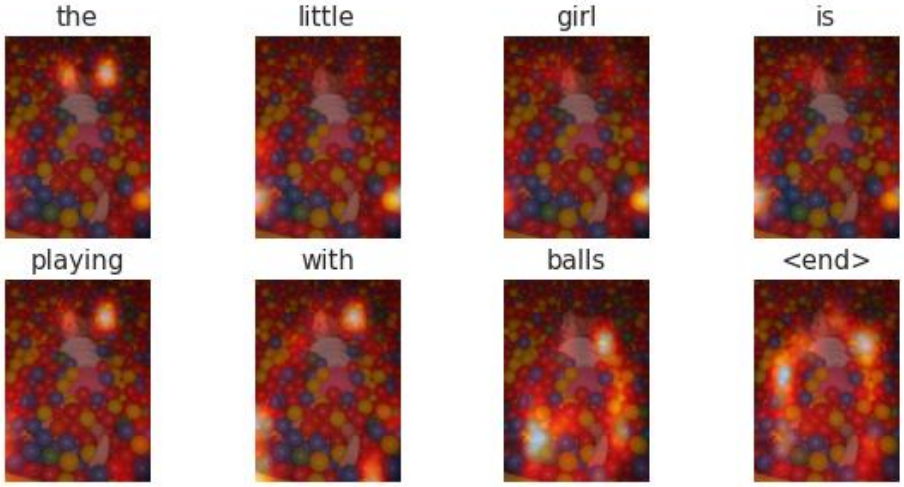


Captions generated by Attention model

```
#################################################################################################################
Real Caption:
 little girl plays in ball pit
#################################################################################################################
Prediction Caption:
 the little girl is playing with balls
#################################################################################################################
BLEU score 1-gram: 28.57
BLEU score 2-gram: 21.82
BLEU score 3-gram: 0.00
BLEU score 4-gram: 0.00
#################################################################################################################
```



Captions generated by Attention model

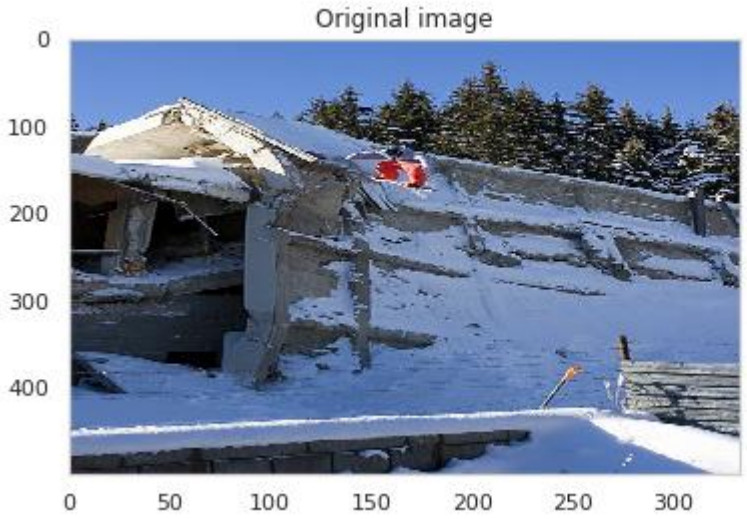


Below are examples of test results with beam approach

```
# Test on random image using beam search
generate_caption_on_random_test_image(evaluate=BEAM_EVALUATE)
```

```
################################################################################
Real Caption:
 snowboarder jumps off an old snowcovered building
################################################################################
Prediction Caption:
 snowboarder on jumping over snow covered wall
################################################################################
BLEU score 1-gram: 14.29
BLEU score 2-gram: 0.00
BLEU score 3-gram: 0.00
BLEU score 4-gram: 0.00
################################################################################

Audio for the predicted caption:
```



```
##################################################################
Real Caption:
 the boy is jumping into the swimming pool
##################################################################
Prediction Caption:
 boy jumping into pool
##################################################################
BLEU score 1-gram: 36.79
BLEU score 2-gram: 21.24
BLEU score 3-gram: 0.00
BLEU score 4-gram: 0.00
##################################################################

Audio for the predicted caption:
```

9. Experiment on unseen data

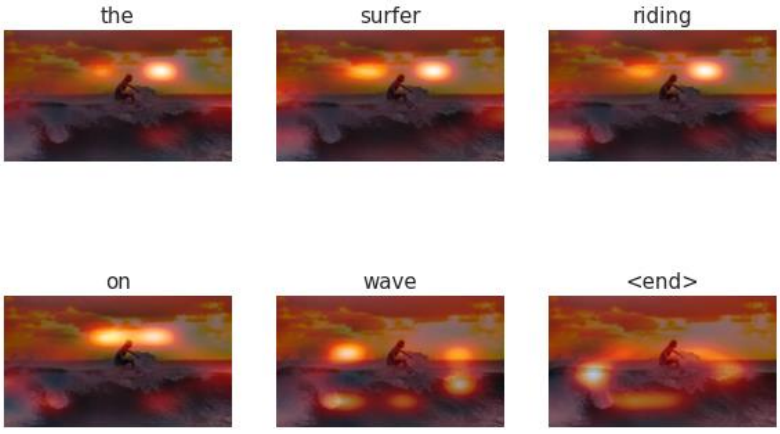
The model is tested with random images sourced from online. Below are few results.

A man surfing

```
generate_caption_for_unknown_data('/home/EyeForBlind/unknown_data/surfing.jpeg', evaluate=GREEDY_EVALUATE, debug=True)
```

```
#########################################################################################################
Predicted Caption:
 the surfer riding on wave
#########################################################################################################
```

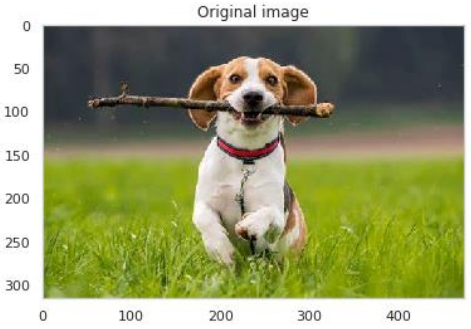
Captions generated by Attention model



In [154…

```
generate_caption_for_unknown_data('/home/EyeForBlind/unknown_data/dog_stick.jpeg', evaluate=GREEDY_EVALUATE, debug=True)
```

```
############################################################################################################
Predicted Caption:
 dog is running through the grass
############################################################################################################
```



Captions generated by Attention model

Conclusion

In this project, we have successfully developed, trained, optimized, and evaluated a neural network model for image captioning. The comprehensive approach ensures that the model not only generates meaningful captions for images but also handles various aspects of training, evaluation, and optimization efficiently.

Key Achievements:

- **Implemented Encoder-Decoder architecture** with attention mechanisms for robust image captioning.
- **Developed a comprehensive training pipeline**, including optimizers, loss functions, and checkpoint management.
- **Applied Greedy Search and Beam Search** for effective caption generation strategies.
- **Utilized BLEU scores for quantitative evaluation** of caption quality.
- **Visualized attention maps** and provided text-to-speech functionality for enhanced model interpretability and usability.
- **Demonstrated successful model testing** with meaningful captions and detailed performance analysis.

Lessons Learned:

- **Balancing Model Complexity**: Maintaining a balance in model complexity is essential for optimal performance and training efficiency.
- **Enhancing Captions with Attention**: Attention mechanisms improve caption relevance but require careful management and visualization.
- **Training Strategy Effectiveness**: Combining greedy and beam search strategies results in more accurate and nuanced captions.
- **Efficient Checkpoint Management**: Effective checkpointing prevents data loss and allows training continuation without starting over.
- **Accurate Loss Calculation**: Properly masking padding values in loss calculations ensures reliable training and evaluation metrics.
- **Comprehensive Evaluation Metrics**: Using BLEU scores along with qualitative assessments provides a complete picture of caption quality.
- **Visualizing and Interacting**: Attention maps and text-to-speech features enhance model interpretability and user experience.
- **Robust Testing and Debugging**: Testing with diverse data and using debugging tools like attention maps help refine model performance and robustness.