Deep Learning Term Project CS60010 Spring Semester 2024

Task: Build encoder decoder models for Automatic Image Captioning
Group ID: G-12
Group Name: Quad Squad

Team Members:

1) **Utsav Suthar (23CS60R49)**
2) **Krishna K. (23CS60R54)**
3) **Sandhya S. (23CS60R62)**
4) **Akash Pattani (23CS60R74)**

Part A:

Introduction: The objective of this project is to develop a deep learning model capable of generating captions for images. To achieve this, we leverage the VGG16 convolutional neural network (CNN) architecture for feature extraction and Long Short-Term Memory (LSTM) networks for caption generation.

Model Architecture:

- **VGG16 for Feature Extraction:** VGG16, a pre-trained CNN model, is employed to extract high-level features from input images. These features capture the image's content in a meaningful manner.
- **LSTM for Captioning:** The extracted features from VGG16 are then fed into an LSTM network. LSTM is chosen due to its ability to model sequences effectively, making it suitable for generating captions based on the extracted image features.

Data Preprocessing:

- **Image Data:** The image data is preprocessed to fit the input dimensions required by the VGG16 model. Images are resized and normalized to ensure compatibility.
- **Caption Data:** Caption data associated with the images is preprocessed, tokenized, and padded to facilitate training with the LSTM model.

Model Training:

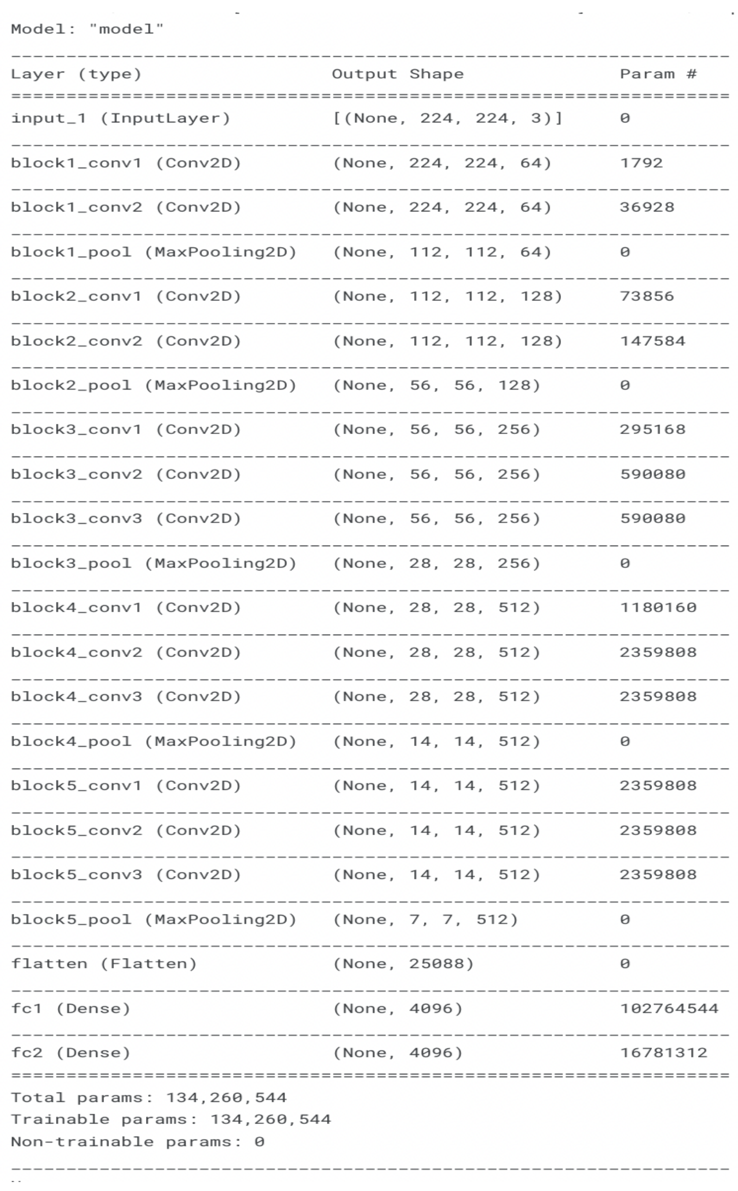
- **Transfer Learning with VGG16:** The VGG16 model is utilized as a feature extractor in transfer learning. The pre-trained weights of VGG16 are frozen during training to preserve the learned features while fine-tuning the model for our specific task.
- **LSTM Training:** The LSTM network is trained to generate captions based on the extracted features. The model is trained using a combination of the image features and corresponding caption sequences.

Evaluation Metrics:

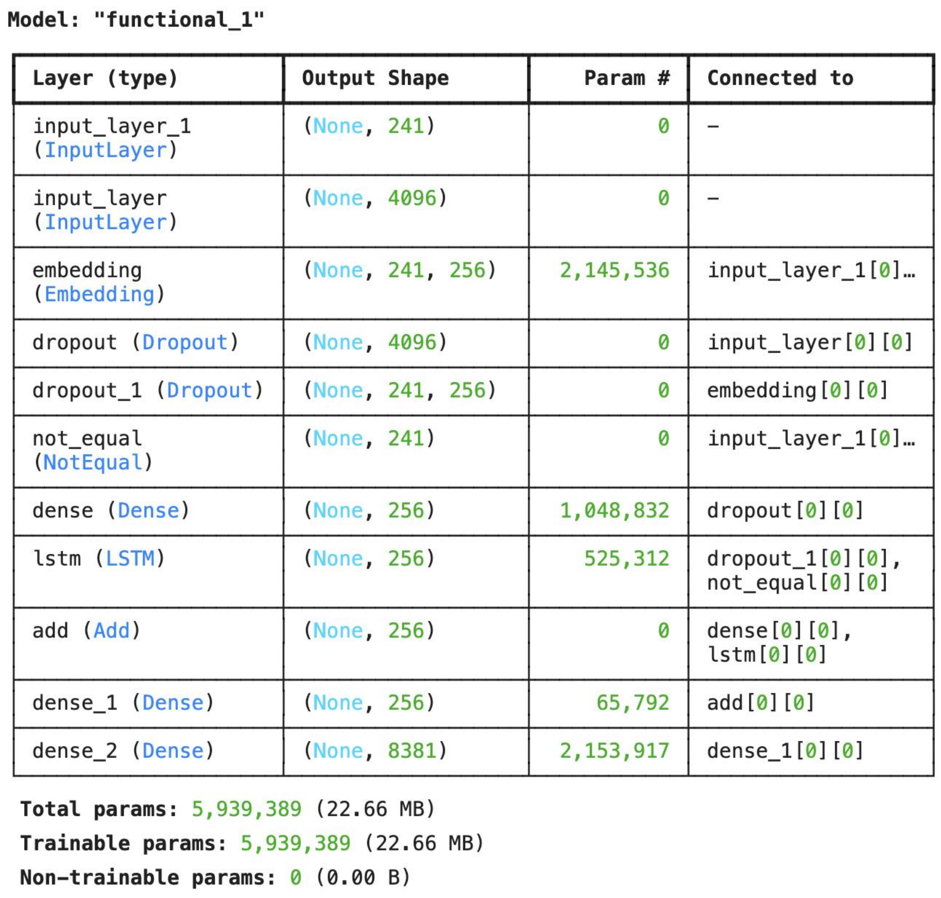
- **BLEU Score:** The quality of generated captions is assessed using the BLEU (Bilingual Evaluation Understudy) score, a metric commonly used for evaluating the similarity between generated text and human-generated reference captions.
- **ROUGE-L:** Another evaluation metric employed is ROUGE-L, which measures the overlap between the generated captions and reference captions based on the longest common subsequence.
- **CIDEr:** CIDEr (Consensus-based Image Description Evaluation) is utilized to assess the consensus between generated and reference captions based on multiple reference captions.

Results and Discussion:

VGG16 Layer where Image is passed to extract features, output of fc1 layer is passed to the decoder model.



```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```



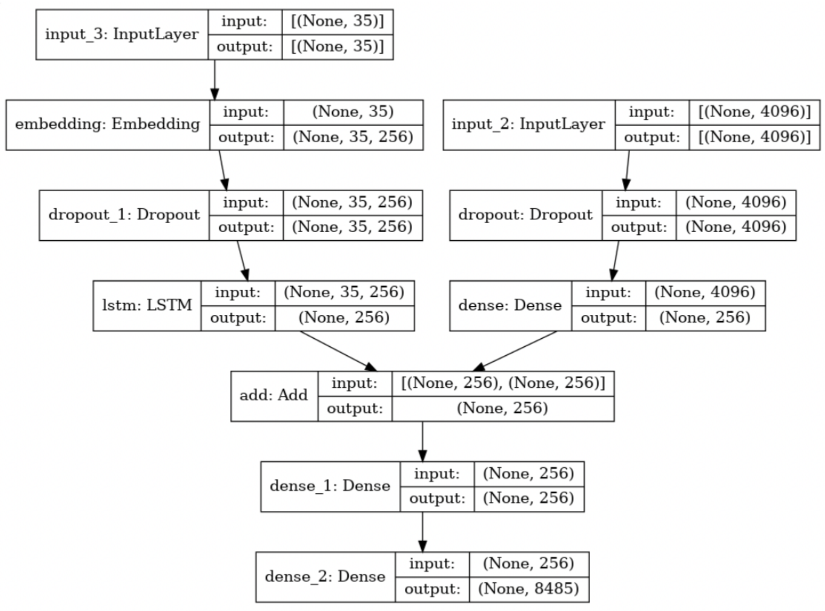
Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 241)	0	-
input_layer (InputLayer)	(None, 4096)	0	-
embedding (Embedding)	(None, 241, 256)	2,145,536	input_layer_1[0]…
dropout (Dropout)	(None, 4096)	0	input_layer[0][0]
dropout_1 (Dropout)	(None, 241, 256)	0	embedding[0][0]
not_equal (NotEqual)	(None, 241)	0	input_layer_1[0]…
dense (Dense)	(None, 256)	1,048,832	dropout[0][0]
lstm (LSTM)	(None, 256)	525,312	dropout_1[0][0], not_equal[0][0]
add (Add)	(None, 256)	0	dense[0][0], lstm[0][0]
dense_1 (Dense)	(None, 256)	65,792	add[0][0]
dense_2 (Dense)	(None, 8381)	2,153,917	dense_1[0][0]

Total params: 5,939,389 (22.66 MB)
Trainable params: 5,939,389 (22.66 MB)
Non-trainable params: 0 (0.00 B)

The combined outputs of Input Layer1 and Input Layer is passed to the LSTM layer.

Low-Level diagram of the model


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

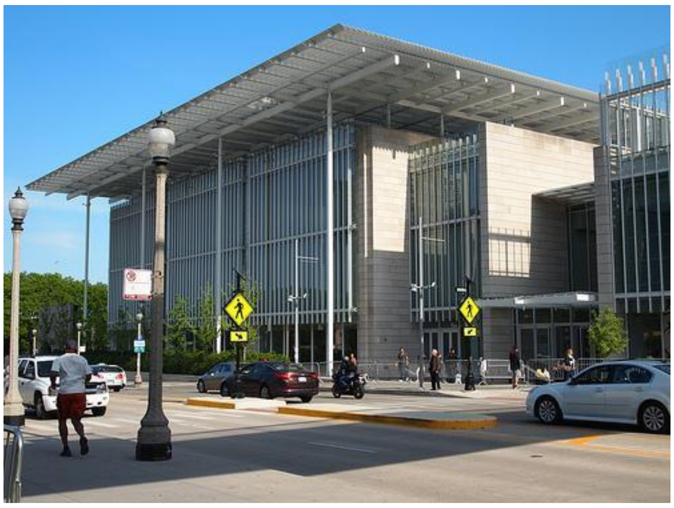
Metrics:

Metric	Score
Bleu_1	0.2522371140894812
Bleu_2	0.12824783418647498
Bleu_3	0.06528198951362799
Bleu_4	0.033707025889910584
METEOR	0.12778075883893764
ROUGE-L	0.23065914083018077
CIDEr	0.08159744079954015

Sample Output:

Actual: large building with bars on the windows in front of it there is people walking in front of the building there is street in front of the building with many cars on it

Predicted: the building is made of glass the building is white the roof of the building is white the windows on the side of the building are white there are vehicles on the side of the building the vehicles are all in the parking lot there are bunch of windows on the sidewalk the buses are all white



Decoding Methods:

Beam Search

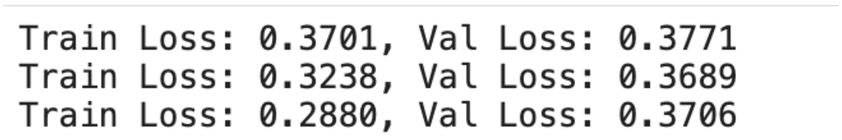
- A technique widely employed in natural language processing tasks such as caption generation.
- Beam search is chosen here to mitigate the issue of exhaustively exploring all possible combinations of words, which could be computationally expensive.
- Beam search efficiently explores the search space by considering only a limited number of candidate captions at each step, reducing computational overhead.

- By considering multiple candidate captions simultaneously, beam search can generate more diverse and contextually relevant captions compared to greedy decoding.
- In caption generation tasks, there may be multiple valid captions for a given image. Beam search helps in handling this ambiguity by exploring different possibilities.

Algorithm:

- Initialize the beam with a single tuple (start_token, 1.0), where start_token denotes the start of the caption.
- Iterate over a predefined maximum length of the caption. At each step, generate candidate captions for each partial caption in the current beam.
- For each partial caption in the current beam:
    - Convert the caption to a sequence of token indices using the tokenizer.
    - Pad the sequence to match the maximum caption length.
    - Use the model to predict the probability distribution over the vocabulary for the next word given the image and the partial caption.
- Choose the top beam_width candidates based on their scores, which are calculated as the product of the previous score and the probability of the next word.
- Check if any of the candidate captions in the current beam end with the end token. If found, stop the beam search process.
- If the end token is found in any of the candidate captions or the maximum caption length is reached, terminate the search.
- Finally, select the caption with the highest score from the current beam as the predicted caption.

Losses:



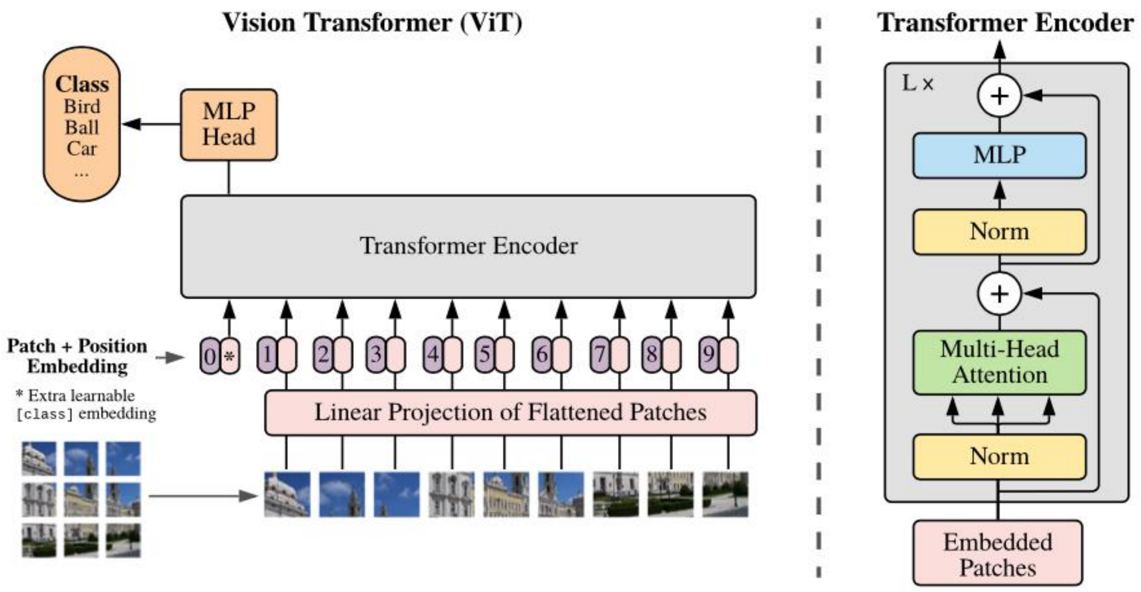
```
Train Loss: 0.3701, Val Loss: 0.3771
Train Loss: 0.3238, Val Loss: 0.3689
Train Loss: 0.2880, Val Loss: 0.3706
```

Part B:

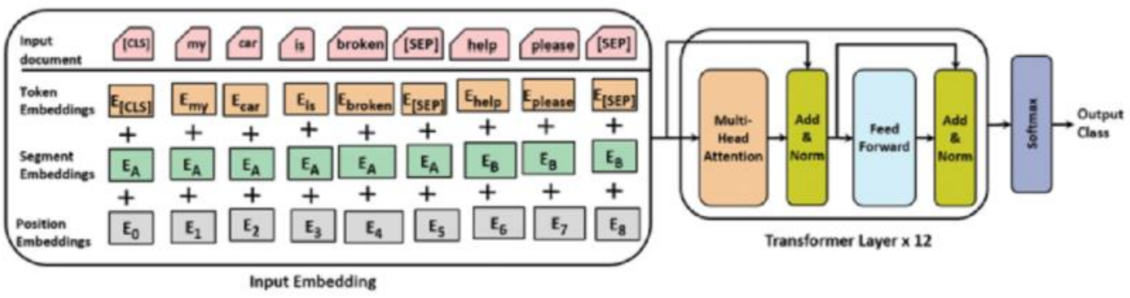
Introduction: The objective of this project is to develop a deep learning model capable of generating captions for images. To achieve this, design a transformer-based encoder decoder model using a Vision Transformer (ViT) as the image encoder and a text decoder.

Methodology: Here's a detailed methodology for designing a Vision Encoder-Decoder model using a Vision Transformer (ViT) as the image encoder and a BERT-based text decoder, along with diagrams for each model

1) **Vision Transformer (ViT) Model:** In ViTs, images are represented as sequences, and class labels for the image are predicted, which allows models to learn image structure independently. Input images are treated as a sequence of patches where every patch is flattened into a single vector by concatenating the channels of all pixels in a patch and then linearly projecting it to the desired input dimension.



2) BERT-based Text Decoder: BERT-based text decoding typically refers to the process of generating natural language text using the Bidirectional Encoder Representations from Transformers (BERT) model or its variants. It uses self-attention mechanisms to capture dependencies between words and learns contextual representations.



Process Overview:

Step 1: Data Preparation:
Load the image-caption dataset from CSV files (train.csv and test.csv) using pandas.
Resize images to a fixed size and convert them into pixel values.
Tokenize captions using a BERT tokenizer for input preparation.

Step 2: Image Encoding with Vision Transformer (ViT):
Initialize a ViTFeatureExtractor to extract features from images.
Process images with the ViT model to obtain image embeddings.

Step 3: Configuration of BERT-based Text Decoder:
Initialize a BERT tokenizer for caption tokenization and input preparation.
Define the architecture of the BERT-based text decoder, specifying layers, attention heads, etc.

Step 4: Training the Vision Encoder-Decoder Model:
Train the integrated model using the prepared data loaders and specified training parameters.
Monitor training progress using metrics like loss and accuracy.

Step 5: Text Generation and Evaluation:
Generate captions for test images using the trained model.
Evaluate the generated captions using metrics such as CIDEr, ROUGE, and BLEU scores to assess performance.

Evaluation Metrics:

BLEU Score: The quality of generated captions is assessed using the BLEU (Bilingual Evaluation Understudy) score, a metric commonly used for evaluating the similarity between generated text and human-generated reference captions.

ROUGE-L: Another evaluation metric employed is ROUGE-L, which measures the overlap between the generated captions and reference captions based on the longest common subsequence.

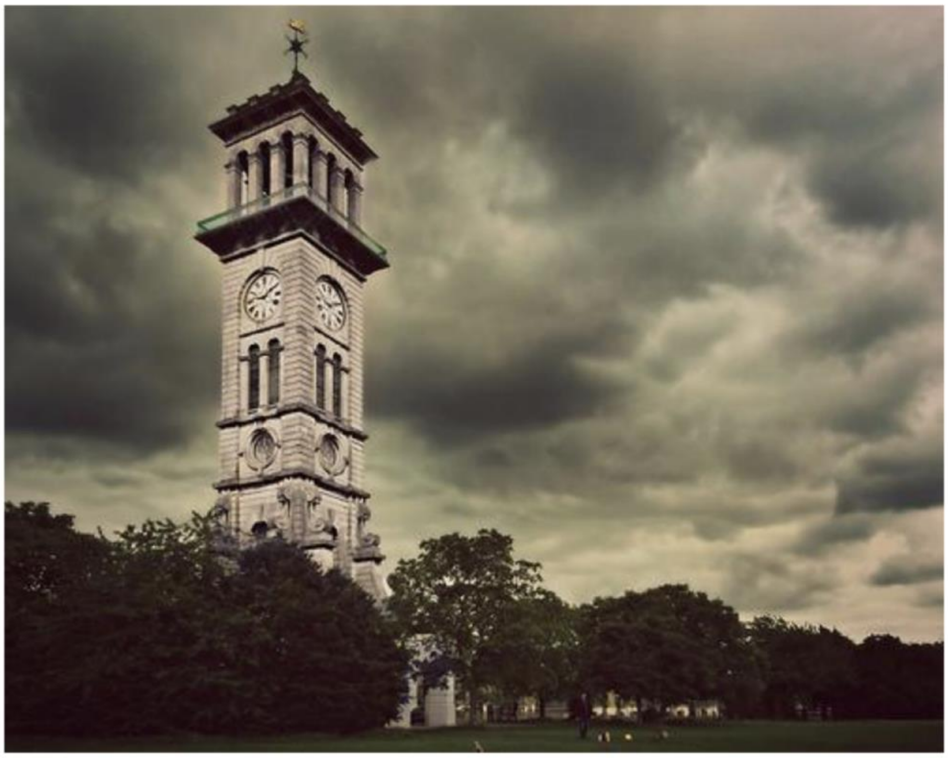
CIDEr: CIDEr (Consensus-based Image Description Evaluation) is utilized to assess the consensus between generated and reference captions based on multiple reference captions.

Metrics:

Metric	Score
Bleu_1	0.25937557803694905

Bleu_2	0.15038985265627905
Bleu_3	0.08846355165297
Bleu_4	0.05178538775034147
METEOR	0.11855599354496447
ROUGE-L	0.27481777131428153
CIDEr	0.13037723333767648

Predictions:



Actual Caption: The sky is gray and cloudy. The clock tower is visible. The tower has compass on its top, and has two clock faces. Below of the tower is row of trees and a field. On the field a person is walking.

Predicted Caption: a building is old fashioned. the building is brown in color. the clock on the tower is big. the clockface is white. the sky is very blue.