Image Caption Generator

- Ch.Sabareesh Reddy
19BCE7210

Contents

- Project Overview
- Objectives Achieved
- Implementation
- Results
- Timeline

Project Overview

What is Image Caption Generator ?

- The Objective of an Image Caption Generator is to automatically generate descriptive and accurate captions for images using advanced computer vision and natural language processing techniques.
- Develop a system that can generate captions for images without manual intervention. The system should analyze the visual content of the image and generate captions that accurately describe the objects, people, actions, and contextual details depicted.
- The Captions should be generated in a way that is consistent with human language use, and should be:

  “Grammatically correct but also Semantically meaningful”.

Objectives Achieved

Caption Accuracy:

- The system generates captions that accurately describe the content of the image, identifying objects, people, actions, and contextual details with a high level of accuracy.

Language Fluency:

- The generated captions exhibit fluency and coherence in human language use. They are grammatically correct, semantically meaningful, and align with natural language patterns.

Evaluation Metrics:

- The performance of the Image Caption Generator is evaluated using established metrics such as BLEU Score. The system achieves high scores on these metrics, indicating the similarity and coherence between the generated captions and human-annotated reference captions.

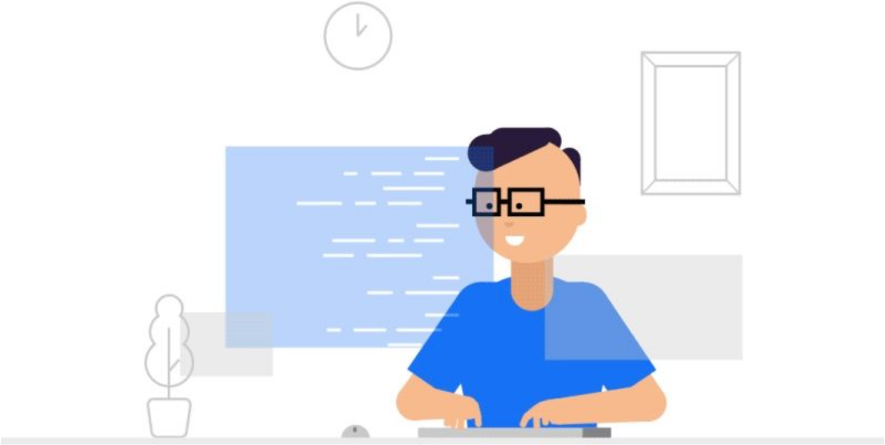
Captions Relevance:

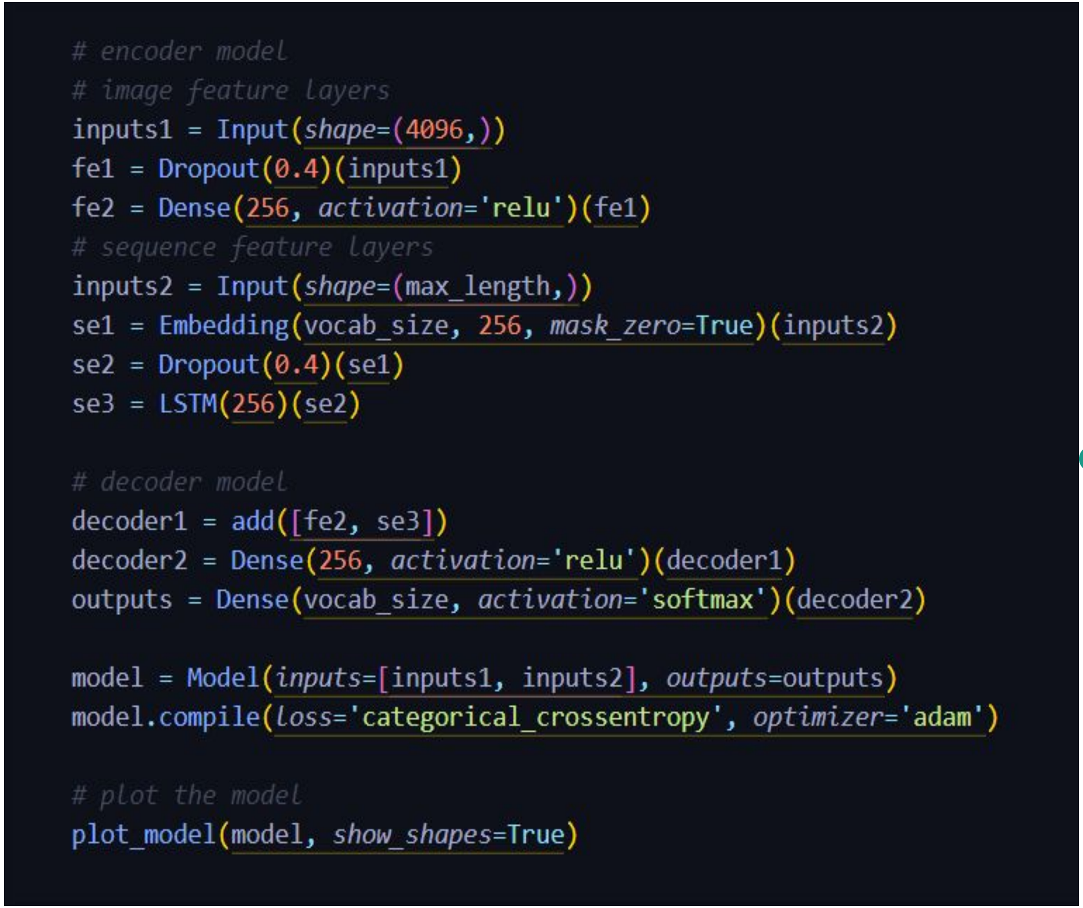
- The generated captions are relevant to the content of the image, accurately describing the main subject, objects, and actions depicted.
- The system avoids generating captions that are irrelevant or misleading, ensuring that the captions align with the visual information in the image.

Real-time Caption Generation:

- The system can generate captions in real-time, providing instantaneous descriptions as images are uploaded or processed.
- It showcases efficiency and responsiveness, making it suitable for applications requiring on-the-fly captioning.

Implementation
&
Results

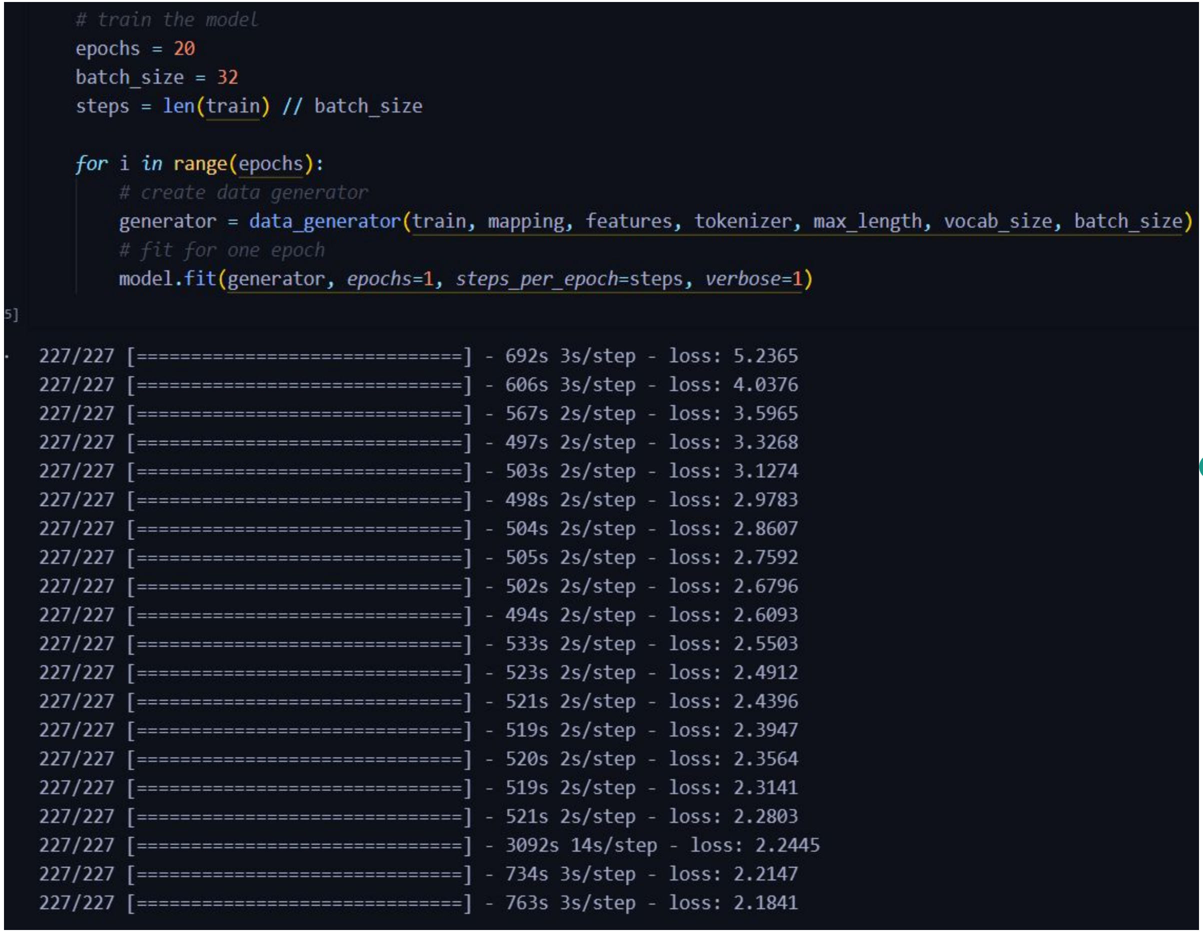




```python
# encoder model
# image feature layers
inputs1 = Input(shape=(4096,))
fe1 = Dropout(0.4)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)
# sequence feature layers
inputs2 = Input(shape=(max_length,))
se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
se2 = Dropout(0.4)(se1)
se3 = LSTM(256)(se2)

# 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)
```

Model Creation

Model

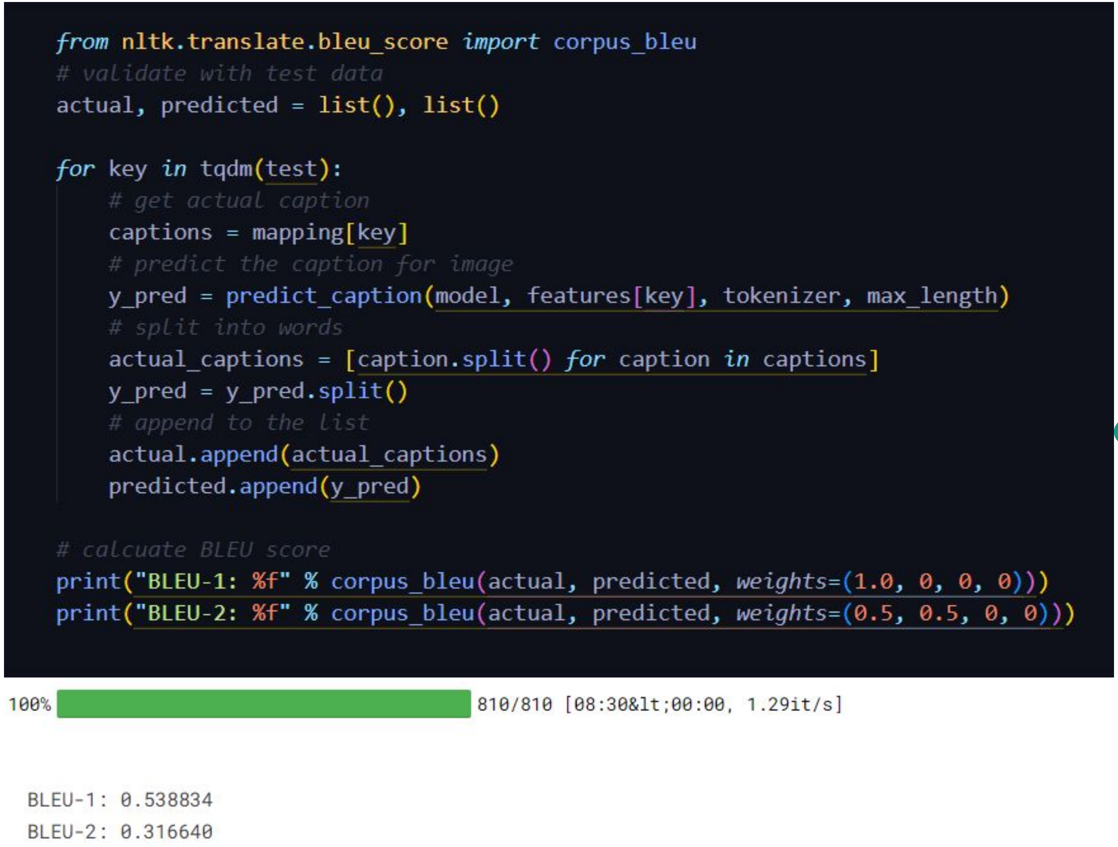


```
# train the model
epochs = 20
batch_size = 32
steps = len(train) // batch_size

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 [==============================] - 692s 3s/step - loss: 5.2365
227/227 [==============================] - 606s 3s/step - loss: 4.0376
227/227 [==============================] - 567s 2s/step - loss: 3.5965
227/227 [==============================] - 497s 2s/step - loss: 3.3268
227/227 [==============================] - 503s 2s/step - loss: 3.1274
227/227 [==============================] - 498s 2s/step - loss: 2.9783
227/227 [==============================] - 504s 2s/step - loss: 2.8607
227/227 [==============================] - 505s 2s/step - loss: 2.7592
227/227 [==============================] - 502s 2s/step - loss: 2.6796
227/227 [==============================] - 494s 2s/step - loss: 2.6093
227/227 [==============================] - 533s 2s/step - loss: 2.5503
227/227 [==============================] - 523s 2s/step - loss: 2.4912
227/227 [==============================] - 521s 2s/step - loss: 2.4396
227/227 [==============================] - 519s 2s/step - loss: 2.3947
227/227 [==============================] - 520s 2s/step - loss: 2.3564
227/227 [==============================] - 519s 2s/step - loss: 2.3141
227/227 [==============================] - 521s 2s/step - loss: 2.2803
227/227 [==============================] - 3092s 14s/step - loss: 2.2445
227/227 [==============================] - 734s 3s/step - loss: 2.2147
227/227 [==============================] - 763s 3s/step - loss: 2.1841
```

Training the Model with approximately 20 Epochs.

Training the Model

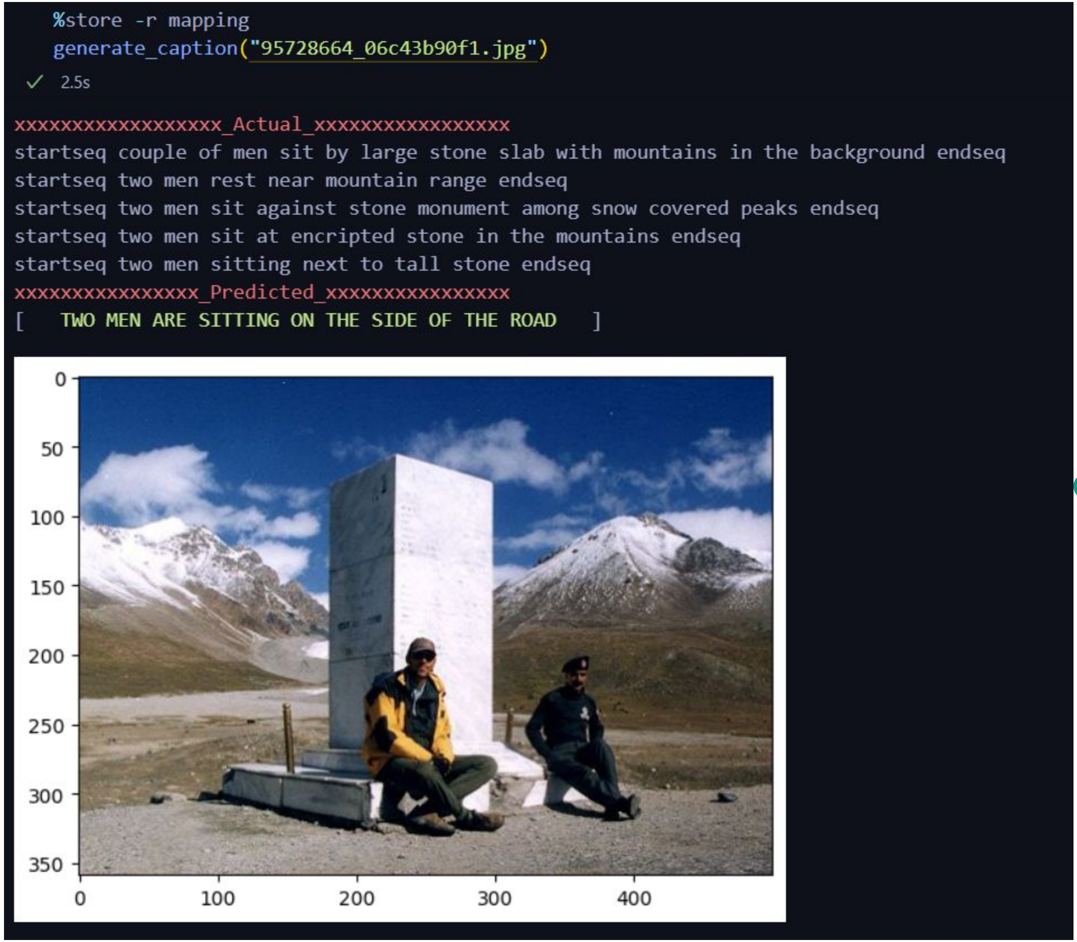


```
from nltk.translate.bleu_score import corpus_bleu
# validate with test data
actual, predicted = list(), list()

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)))
```

```
100%                                      810/810 [08:30&lt;00:00, 1.29it/s]

BLEU-1: 0.538834
BLEU-2: 0.316640
```

Analysing the Blue Score,
BLEU-1
BLEU-2

Calculating BLEU Scores (Accuracy)



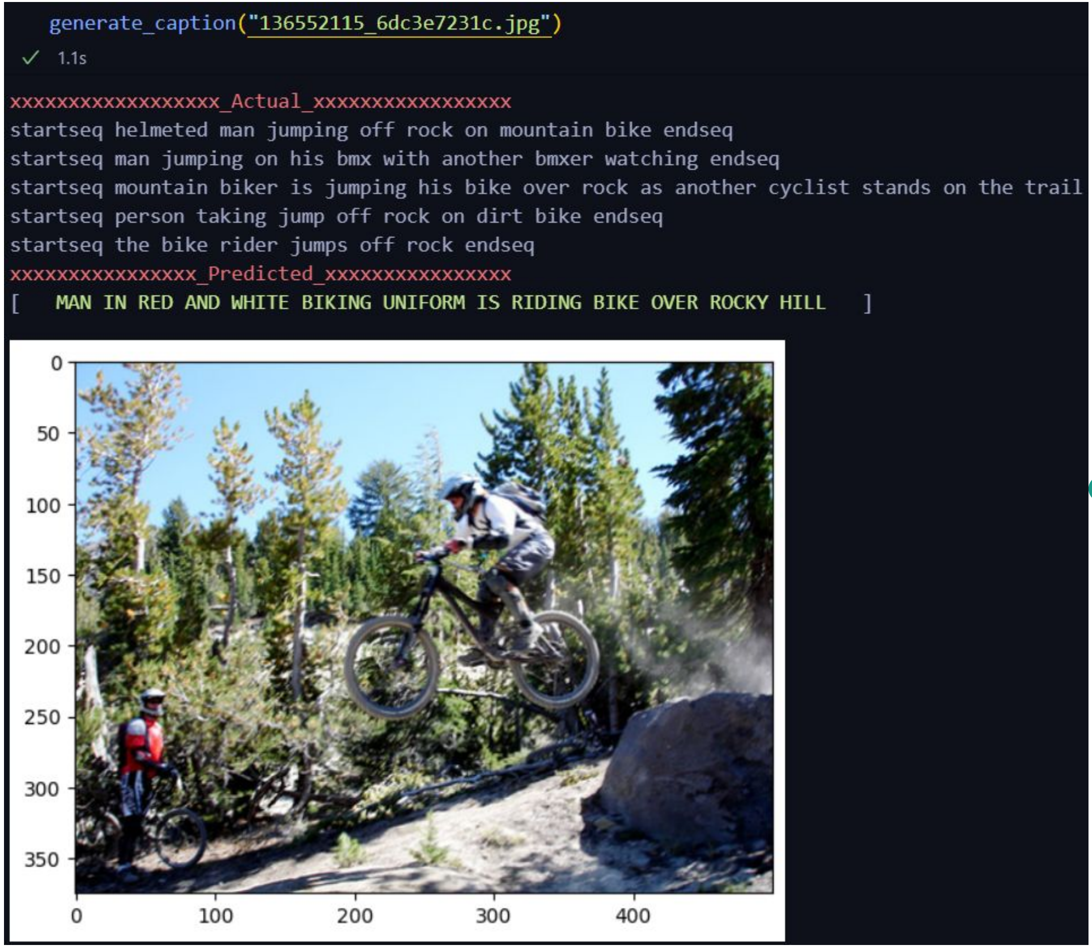
```
%store -r mapping
generate_caption("95728664_06c43b90f1.jpg")
```

✓ 2.5s

```
xxxxxxxxxxxxxxxxxx_Actual_xxxxxxxxxxxxxxxxx
startseq couple of men sit by large stone slab with mountains in the background endseq
startseq two men rest near mountain range endseq
startseq two men sit against stone monument among snow covered peaks endseq
startseq two men sit at encripted stone in the mountains endseq
startseq two men sitting next to tall stone endseq
xxxxxxxxxxxxxxxx_Predicted_xxxxxxxxxxxxxxxx
[   TWO MEN ARE SITTING ON THE SIDE OF THE ROAD   ]
```

Here are the results! Well polished, Short and Crisp Output for any Image provided.

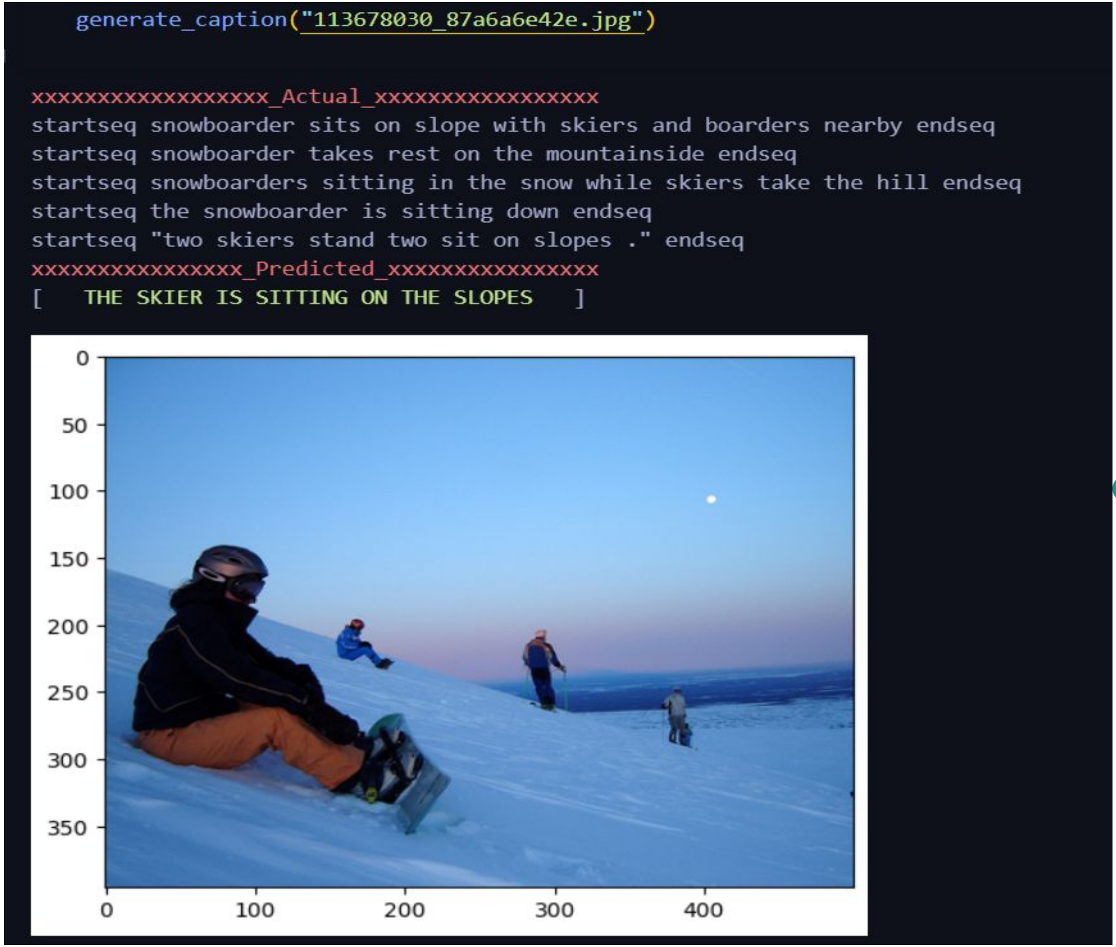
Caption Generation in no Time !!



```
generate_caption("136552115_6dc3e7231c.jpg")
✓ 1.1s

xxxxxxxxxxxxxxxxxx_Actual_xxxxxxxxxxxxxxxxxx
startseq helmeted man jumping off rock on mountain bike endseq
startseq man jumping on his bmx with another bmxer watching endseq
startseq mountain biker is jumping his bike over rock as another cyclist stands on the trail
startseq person taking jump off rock on dirt bike endseq
startseq the bike rider jumps off rock endseq
xxxxxxxxxxxxxxxxx_Predicted_xxxxxxxxxxxxxxxxx
[   MAN IN RED AND WHITE BIKING UNIFORM IS RIDING BIKE OVER ROCKY HILL    ]
```

Few Outputs from the Model

Caption Generation in no Time !!



```
    generate_caption("113678030_87a6a6e42e.jpg")


xxxxxxxxxxxxxxxxxx_Actual_xxxxxxxxxxxxxxxxxx
startseq snowboarder sits on slope with skiers and boarders nearby endseq
startseq snowboarder takes rest on the mountainside endseq
startseq snowboarders sitting in the snow while skiers take the hill endseq
startseq the snowboarder is sitting down endseq
startseq "two skiers stand two sit on slopes ." endseq
xxxxxxxxxxxxxxxx_Predicted_xxxxxxxxxxxxxxxx
[   THE SKIER IS SITTING ON THE SLOPES    ]
```

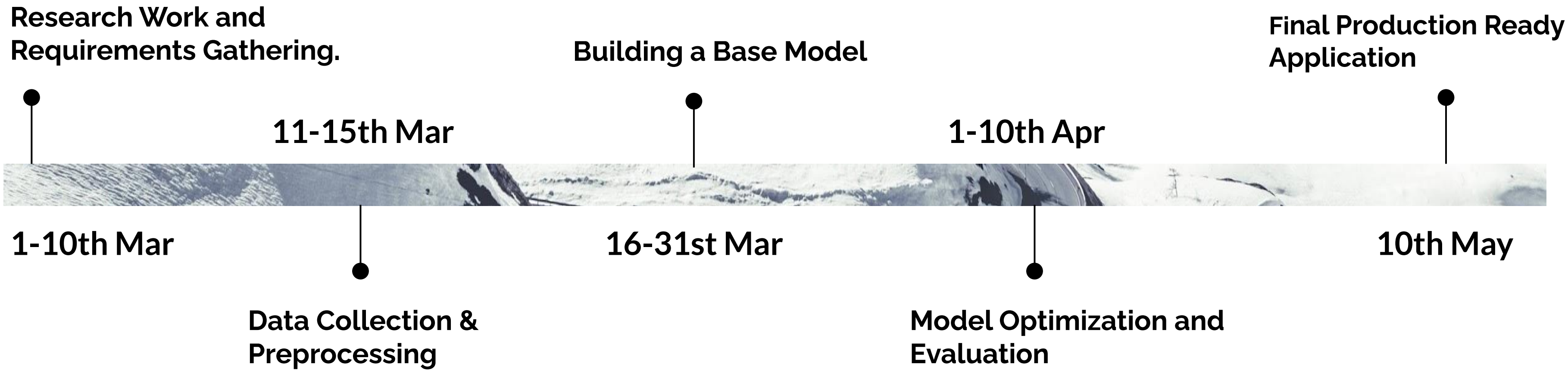
Few Outputs from the Model

Caption Generation in no Time !!

Time Plan

MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY
450 x 12

Timeline



1-10th Mar
Research Work and Requirements Gathering.

11-15th Mar
Data Collection & Preprocessing

16-31st Mar
Building a Base Model

1-10th Apr
Model Optimization and Evaluation

10th May
Final Production Ready Application

Thank You.