Image Caption Generation – INFO7390

Team Pixel (Team 8)

Team Members:
Purvang Jayesh Thakkar
Pratiksha Milind Lavhatre
Rashika Moza

INTRODUCTION

Without the deep learning and AI algorithms, this problem was very different to solve even for computer scientists because normal computers cannot understand the images as humans do. With the aid of Deep learning, we can solve this problem easily.

Image caption generator is an interesting task where both computer vision techniques and natural language processing techniques are explored to recognize the context of an image and outline them in a natural language like English.

PROBLEM DESCRIPTION

Image Caption Generation involves predicting a sequence of words(sentence) from an input image. It is essentially an advanced image classification problem with a lot of potential applications like:

- Aiding visually impaired people who rely on sounds and texts to describe a scene
- Targeted marketing on social media applications like Facebook and Instagram
- A slightly (not-so) long term use case would be explaining what happens in a video, frame by frame

OBJECTIVE

The Objective of our project is to build a working model of image caption generator by using CNN (Conventional Neural networks) with LSTM (Long short-term memory), RNN and GRU.
In simple words, our goal is to build a model which takes an image as input and outputs its caption.

DATASET

We are using Flickr8K dataset for image caption generation available in Kaggle
[https://www.kaggle.com/ming666/flicker8k-dataset] .

Flickr8K.zip comprises of two folders:
- Flickr8K_Images: Contains a total of 8000 jpeg images of different shapes and sizes. We are using 6000 images for training, 1000 images for testing and 1000 images for validation
- Flickr8K_TextData: Contains text files describing the images in the train, validation and the test sets. Each image has a total of 5 captions i.e. a total of 40000 captions

The dataset consists of images gathered from different Flickr groups, so the Flickr8K dataset contains a variety of images depicting scenes and situations. The images are currently of different dimensions i.e. 415x502, 500x332, 333x500 etc.

METHODOLOGY

In this project we deal with two types of data namely text and images.

Various pre-processing steps are applied to the captions and images as discussed below:

Text pre-processing

• Tokenized each caption and converted all the words to lower-case so that the model would not treat words like "Hi" and "hi" differently

• Removed alpha-numeric characters as they won't hold much significance in the data

• Removed punctuation marks from the description, as predicting even the punctuation correctly would be a deviation from the main motive of this project

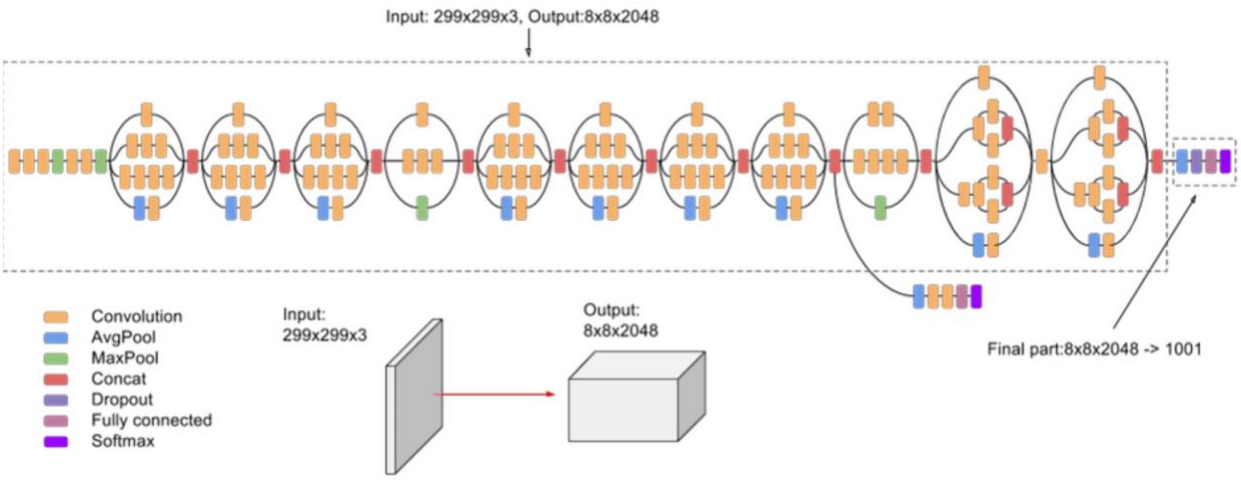
• Added two tokens namely startseq and endseq as the prefix and postfix for each caption respectively, to help our model recognize the start and end of each description during training and predicting phases

• Filtered out unique words from the training captions and represented each word by a unique integer
Since all descriptions are of variable lengths, we calculated the maximum length description which was 34 in our case. While creating a word vector for each description we map each word with its corresponding integer value and pad zeros to the remaining length of vector to convert it into a fixed length vector of size 34

Image pre-processing and feature vector generation

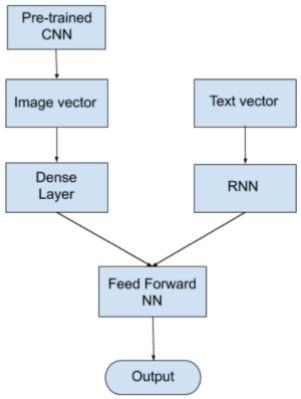
Since the input to the neural network should be an image vector, we first resized all the images to a fixed dimension of 299x299x3 using OpenCV, and then converted each image into a fixed-length vector by employing transfer learning. We used a pre-trained InceptionV3 CNN model.

InceptionV3 model was trained on Imagenet dataset for the purpose of classification. Since our aim is to generate a fixed-length vector of size 2048, we removed the last softmax layer from the pretrained model and extracted a 2048 length vector for each image. Each image along with its corresponding feature vector is stored in a pickle file. We have maintained 3 different pickle files for training, validation and test sets, for computational efficiency.



Inception V3 Architecture

MODEL ARCHITECTURE



• In this model we are using an encoder-decoder architecture
• The 2048 image vector generated from the pretrained CNN model is fed into a dense layer with 256 nodes, to obtain a fixed length image vector of 256
• The 34-length word vector is fed into an embedding LSTM/GRU/SimpleRNN layer with 256 nodes to generate a 256 fixed length word vector
• The 256-length image and word vector are considered as the encoder outputs
• Decoder model adds both the encoder outputs and feeds it into dense 256 layer
• The output layer is a dense layer with 8763 nodes (size of the vocabulary) followed by a SoftMax layer, which will output probabilities for each word in the vocabulary.

We used three different recurrent neural networks to generate captions, namely Long Term-Short Term Memory, Simple Recurrent Neural Network and Gated Recurrent Unit.

Training the Neural Network- LSTM

```
filepath = 'model-ep{epoch:03d}-loss{loss:.3f}-val_loss{val_loss:.3f}.h5'
checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min')
epochs=20
model = model_LSTM_FFNN(maxLendesc,vocab_size)
for i in range(epochs):
    model.fit([X1_train, X2_train], y_train, epochs=1, verbose=2, callbacks=[checkpoint], validation_data=([X1_test, X2
    model.save('modelLSTM_' + str(i) + '.h5')
```

```
Model: "functional_5"
```

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 34)]	0	
input_2 (InputLayer)	[(None, 2048)]	0	
embedding (Embedding)	(None, 34, 256)	1940224	input_3[0][0]
dropout (Dropout)	(None, 2048)	0	input_2[0][0]
dropout_1 (Dropout)	(None, 34, 256)	0	embedding[0][0]
dense (Dense)	(None, 256)	524544	dropout[0][0]
lstm (LSTM)	(None, 256)	525312	dropout_1[0][0]
add (Add)	(None, 256)	0	dense[0][0] lstm[0][0]
dense_1 (Dense)	(None, 256)	65792	add[0][0]
dense_2 (Dense)	(None, 7579)	1947803	dense_1[0][0]

```
Total params: 5,003,675
Trainable params: 5,003,675
Non-trainable params: 0
```

Training the Neural Network- RNN

In [48]:
```
filepath = 'modelRNN-ep{epoch:03d}-loss{loss:.3f}-val_loss{val_loss:.3f}.h5'
checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min')
epochs=20
model = model_RNN_FFNN(maxLendesc,vocab_size)
for i in range(epochs):
    model.fit([X1_train, X2_train], y_train, epochs=1, verbose=2, callbacks=[checkpoint], validation_data=([X1_test, X2
    model.save('modelRNN_' + str(i) + '.h5')
```

```
Model: "functional_7"
```

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, 34)]	0	
input_4 (InputLayer)	[(None, 2048)]	0	
embedding_1 (Embedding)	(None, 34, 256)	1940224	input_5[0][0]
dropout_2 (Dropout)	(None, 2048)	0	input_4[0][0]
dropout_3 (Dropout)	(None, 34, 256)	0	embedding_1[0][0]
dense_3 (Dense)	(None, 256)	524544	dropout_2[0][0]
simple_rnn (SimpleRNN)	(None, 256)	131328	dropout_3[0][0]
add_1 (Add)	(None, 256)	0	dense_3[0][0] simple_rnn[0][0]
dense_4 (Dense)	(None, 256)	65792	add_1[0][0]
dense_5 (Dense)	(None, 7579)	1947803	dense_4[0][0]

```
Total params: 4,609,691
Trainable params: 4,609,691
Non-trainable params: 0
```

Training the Neural Network- GRU

In [49]:
```
filepath = 'model-ep{epoch:03d}-loss{loss:.3f}-val_loss{val_loss:.3f}.h5'
checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min')
epochs=20
model = model_GRU_FFNN(maxLendesc,vocab_size)
for i in range(epochs):
    model.fit([X1_train, X2_train], y_train, epochs=1, verbose=1, validation_data=([X1_test, X2_test], y_test))
    model.save('modelGRU_' + str(i) + '.h5')
```

```
Model: "functional_9"
```

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	[(None, 34)]	0	
input_6 (InputLayer)	[(None, 2048)]	0	
embedding_2 (Embedding)	(None, 34, 256)	1940224	input_7[0][0]
dropout_4 (Dropout)	(None, 2048)	0	input_6[0][0]
dropout_5 (Dropout)	(None, 34, 256)	0	embedding_2[0][0]
dense_6 (Dense)	(None, 256)	524544	dropout_4[0][0]
gru (GRU)	(None, 256)	394752	dropout_5[0][0]
add_2 (Add)	(None, 256)	0	dense_6[0][0] gru[0][0]
dense_7 (Dense)	(None, 256)	65792	add_2[0][0]
dense_8 (Dense)	(None, 7579)	1947803	dense_7[0][0]

```
Total params: 4,873,115
Trainable params: 4,873,115
Non-trainable params: 0
```

RESULTS

We used three different recurrent neural networks to generate captions, namely Simple Recurrent Neural Network, Gated Recurrent Unit and Long Term-Short Term Memory. **We observed that LSTM was performing the best out of all the networks.**

Bilingual Evaluation Understudy Score (BLEU) FOR PERFORMANCE EVALUATION

- BLEU is a metric for evaluating a generated sentence to a reference sentence
- BLEU score lies between 0 and 1, where a perfect match results in a score of 1 and a perfect mismatch results in a score of 0
- It counts the matching n-grams in the generated and the reference description, where 1-gram or unigram would be each token and a bigram comparison would be each word pair
- The n-gram comparison is made regardless of word order
- BLEU ensures that the all the occurrences in the reference text is considered

BLEU Score for LSTM model:

In [34]:
```
model = keras.models.load_model('modelLSTM_19.h5')

evaluate_model(model, test_dict, encoded_test_v3, tokenizer, maxLendesc)
```

```
BLEU-1: 0.624825
BLEU-2: 0.396498
BLEU-3: 0.290238
BLEU-4: 0.149033
```

BLEU Score for RNN model:

In [56]:
```
model = keras.models.load_model('modelRNN_19.h5')

evaluate_model(model, test_dict, encoded_test_v3, tokenizer, maxLendesc)
```

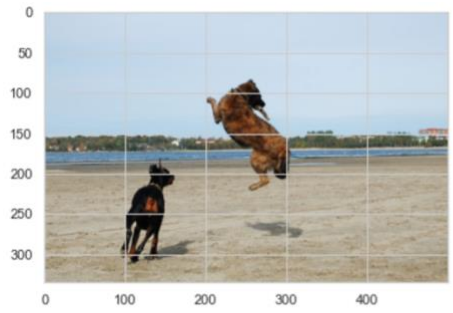
```
BLEU-1: 0.496459
BLEU-2: 0.293420
BLEU-3: 0.211384
BLEU-4: 0.103328
```

BLEU Score for GRU model:

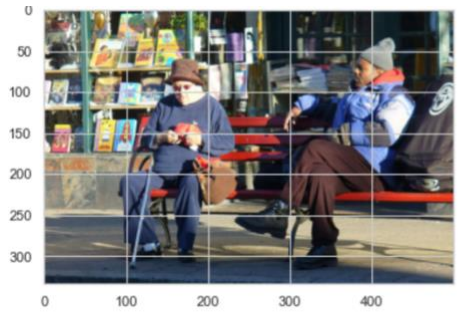
In [57]:
```
model = keras.models.load_model('modelGRU_19.h5')

evaluate_model(model, test_dict, encoded_test_v3, tokenizer, maxLendesc)
```

```
BLEU-1: 0.611734
BLEU-2: 0.385810
BLEU-3: 0.285952
BLEU-4: 0.156869
```

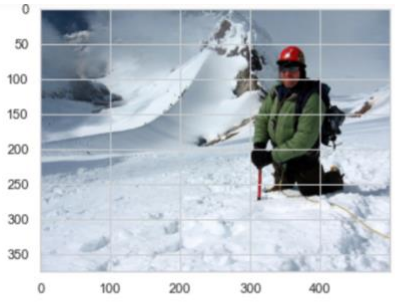
CAPTION GENERATED BY MODELS:



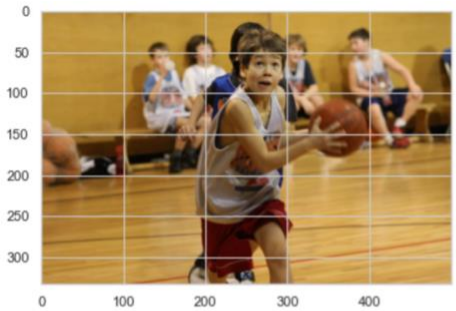

```
Actual Prediction:
['startseq black dog is running along the beach beside brown dog in midair catching stick in its mouth endseq', 'star
tseq black dog runs on the shore while light brown dog jumps up to catch stick endseq', 'startseq brown dog leaps int
o the air next to black dog on sandy bay endseq', 'startseq on beach black and brown dog runs brown dog jumps endse
q', 'startseq two dogs are playing together endseq']
----------------------------------------
Our Prediction
startseq two dogs are running on the beach endseq
```



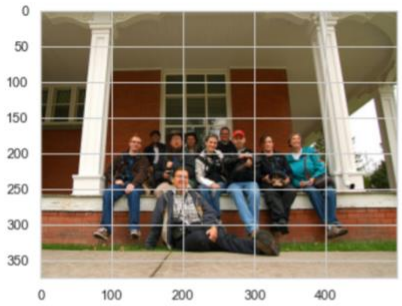
```
Actual Prediction:
['startseq an older man is sitting on red bench with younger man endseq', 'startseq an older person is sitting on red
bench next to black man with children bookstore behind them both endseq', 'startseq person sits on red bench to eat w
hile another watches endseq', 'startseq two men sit on bench endseq', 'startseq two people sitting on red bench in fr
ont of books endseq']
----------------------------------------
Our Prediction
startseq man in black shirt and woman sitting on the street endseq
```



```
Actual Prediction:
['startseq man crouched on snowy peak endseq', 'startseq man in green jacket stands in deep snow at the base of mount
ain endseq', 'startseq man kneels in the snow endseq', 'startseq man measures the depth of snow endseq', 'startseq mo
untain hiker is digging steaks into the thck snow endseq']
Our Prediction
startseq skier in the snow is skiing down the snow endseq
```



```
Actual Prediction:
['startseq boy with red shorts is holding basketball in basketball court endseq', 'startseq young boy is playing bask
etball in an official game endseq', 'startseq the boy is playing basketball endseq', 'startseq the boy runs with the
basketball while looking up endseq', 'startseq the children are playing basketball indoors endseq']
----------------------------------------
Our Prediction
startseq basketball player in the orange jersey is running endseq
```




```
Actual Prediction:
['startseq group of people are gathered on white pillared porch for photo endseq', 'startseq group of people are sitt
ing in front of red brick and white trim building endseq', 'startseq group of people are sitting on the porch of bric
k building endseq', 'startseq group of people gather in front of red house endseq', 'startseq group of ten people pos
ing outside of classicstyle building endseq']
Our Prediction
startseq group of people are sitting on bench in front of building endseq
```

REFERENCES:

1) https://cs.stanford.edu/people/karpathy/cvpr2015.pdf
2) https://arxiv.org/abs/1411.4555
3) https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/
4) https://thinkautonomous.medium.com/rnns-in-computer-vision-image-captioning-597d5e1321d1
5) https://machinelearningmastery.com/calculate-bleu-score-for-text-python/