Visual Recognition Group Project 2

Group code : MOP

Team members

Mudit Goyal	Omkar Lavangad	Parithimalan A
(IMT2018045)	(IMT2018053)	(IMT2018055)

Goal : Design a CNN-LSTM system that can perform image captioning
Dataset: Flickr_8k dataset

Training Methodology

We have broken down the problem to various steps. We will go through the steps one by one.

Importing libraries

We are using **Keras** and **Tensorflow** libraries in this project.

Loading the captions from the dataset

1. We extract the captions from the "**Flickr8k.token.txt**" file.
2. This file contains the image name and five reference captions mapped to it.
3. We create a dictionary with the key as the image name and the value as the list of the reference captions.

Cleaning the captions

1. We convert all the words to lowercase.
2. We are removing the noise such as punctuations and special characters.

Loading the train images

1. We are importing the train images from " **Flickr_8k.trainImages.txt**".
2. We create a list of the train image names to use to get its corresponding captions.
3. We preprocess the captions by adding '**stseq**' at the beginning and '**enseq**' at the end of the sentence to each caption. This helps the model in identifying the beginning and the ending of a caption and when it should start writing or stop predicting the next word at the training phase.

Extracting features from the images using CNN

DenseNet201

1. We have used transfer learning with this model (trained on ImageNet dataset).
2. We pass our images through this model to get a feature vector of the size 1920.

ResNet50

1. We have used transfer learning with this model (trained on ImageNet dataset).
2. We pass our images through this model to get a feature vector of the size 2048.

Build and tokenize the vocabulary

1. We will build vocabulary from all train image captions.
2. We have defined a certain **threshold** which will limit the size of the vocabulary. **i.e.** only the words which occur in the set of train image captions more than the threshold value will be added to the vocabulary.
3. We have to tokenize the captions from the train set and get the word-to-index & index-to-word dictionaries from that.
4. The word-to-index dictionary is meant to represent the captions as a number to be used as inputs to the model.
5. The index-to-word dictionary is to convert the next word/prediction to the word form, as we know it.

Creating input and output sequences

1. For every caption we define input sequences of lengths 1 to its own length -1 and use the next word as the output sequence.
2. The input sequences are then padded to be made into the same length.
3. The output sequence is made into a vector of the size of the vocab and the word is one hot encoded.

Glove embedding

We have used "**glove.200d**" to create 200 dimension vectors for converting words to vectors.

Defining the model (Caption generator)

1. For our model we have two input layers, one is the image feature vector and the other is a set of input sequences for the captions which goes through a **LSTM** layer.
2. The output word is added to the input sequence and sent to the LSTM to get the next word.
3. This process is repeated till '**enseq**' is reached.

Training the model

1. We pass the image feature vectors, input sequences as the input layers to the model with the expected output as the corresponding output sequence.
2. For training we used **adam optimizer** and **categorical cross entropy** as the loss function.
3. We are training for 30 epochs for a batch size of 256.

Model Architecture

Model for feature extraction (CNN)

Transfer learning : It is used in machine learning, is the reuse of a pre-trained model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another.
For the model we have used **densenet201/resnet50** as the feature extractor.

Architecture details

Densenet201

Number of parameters : 18.32 million

Resnet50

Number of parameters : 23.58 million

Model for caption generation (LSTM)

1. For our model we have two input layers, one is the image feature vector and the other is a set of input sequences for the captions.
2. The image feature vector (size : 1920 for DenseNet / 2048 for ResNet) is passed through a dropout layer and a dense layer
   which results in an output vector of size 256.
3. The input sequences are passed through an embedding layer which uses the glove embeddings defined previously and is passed through a dropout layer and a dense layer which results in an output vector of size 200. This is then passed through a LSTM layer to get a vector of size 256.
4. The two output vectors are combined and passed through 2 more dense layer which results in a word, this word is then used in the LSTM input sequence to get the next word.
5. This process is repeated till '**enseq**' is reached.

   Model Summary

   Number of parameters :

   2.18 million (If we use DenseNet201 as the feature extractor)

   2.21 million (If we use ResNet50 as the feature extractor)

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, 34)]	0	
input_4 (InputLayer)	[(None, 1920)]	0	
embedding (Embedding)	(None, 34, 200)	506200	input_5[0][0]
dropout (Dropout)	(None, 1920)	0	input_4[0][0]
dropout_1 (Dropout)	(None, 34, 200)	0	embedding[0][0]
dense (Dense)	(None, 256)	491776	dropout[0][0]
lstm (LSTM)	(None, 256)	467968	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, 2531)	650467	dense_1[0][0]

Testing

We have used **greedy search** and **beam search(k=3)** for predicting the captions using the LSTM model.

BLEU Score for the predicted images

BLEU score in Python is a metric that measures the goodness of Machine Translation models. The BLEU score compares a sentence against one or more reference sentences and tells how well does the candidate sentence matched the list of reference sentences. It gives an output score between 0 and 1. We have used the **sentence_bleu()** function from the nltk library. For every image in the test dataset we have calculated the BLEU score and return the average of the scores as the BLEU score for the test dataset.

Experimentation details

Model	Total parameters	BLEU Score (greedy search)	BLEU Score (beam search)
DenseNet201 + LSTM	18.32 + 2.18 = **20.5 million**	**0.556**	0.542
ResNet50 + LSTM	23.58 + 2.21 = **25.79 million**	0.544	0.541

Remark : While creating the vocabulary from the training captions we limited it to words which have occured atleast 5 times (threshold=5) in the training corpus, because lowering it was causing the program to exceed the available RAM on colab.

Results on Subjective Images

As we have got better bleu score with model setup as **DenseNet201 + LSTM** (20.5 million parameters), we have attached results with subjective images for the same.

Image 1

```
# Greedy Search
caption = "man in black shirt is standing on the sidewalk with his back to the camera"
# Beam Search
caption = "man is sitting on bench next to fancy sculpture"
```

Image 2

```
# Greedy Search
caption = "man in black and white biking gear is kneeling on the sidewalk"
# Beam Search
caption = "group of people are riding bicycles on the street"
```

Image 3

```
# Greedy Search
caption = "group of people are gathered around building"
# Beam Search
caption = "group of people are gathered in front of fountain"
```

Image 4

```
# Greedy Search
caption = "two dogs are playing with ball"
# Beam Search
caption = "two dogs are playing with ball"
```

Image 5

```
# Greedy Search
caption = "man in black shorts and hat is standing on rock with his legs spread wide open"
# Beam Search
caption = "man is standing on rock with his hands in the air"
```