REPORT ON

CNN-LSTM system for image captioning

Team Name: SRK
Shubham Jha : MT2020514
Rahul Vishnoi : MT2020076
Kapil Mehta : MT2020520

UNDER GUIDANCE OF:
PROF. VISWANTH G.

Overview of project:

The goal of the project is to design a CNN-LSTM architecture for the application of image captioning. The dataset for the image captioning used is FLICKR8K, which is used both for training and testing our model. In order to check the performance of our model, BLEU score on test data is considered as the metric subject to the condition on total model size.

CNN-LSTM model:

The image captioning problem can be considered a one to many sequence prediction task. One of the most interesting and practically useful neural models comes from mixing different types of networks to create a hybrid model. Convolutional Neural Networks are designed to map input data to an output variable. CNN architecture is used to extract features from the image which are then fed into the LSTM architecture to output the caption.Long Short Term Memory networks or LSTM networks, which comes under Recurrent Neural Networks, are used for sequence prediction problems. LSTM networks are used as they encapsulate a wider sequence of words or sentences for predictions.

Image captioning:

Image captioning can be described as generating a human-readable textual description of an image. This could be viewed as an application of both the fields of Computer Vision and Natural Language Processing. The steps involved in the process is described in the image below:

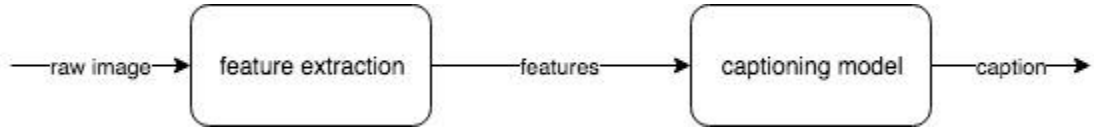

Fig. Image caption

Flickr8k dataset:

The dataset used for the task of image captioning is the Flickr8K dataset. The dataset could be described as a new benchmark collection for sentence-based image description and search, consisting of 8,000 images

that are each paired with five different captions which provide clear descriptions of the salient entities and events. The images were chosen from six different Flickr groups, and tend not to contain any well-known people or locations, but were manually selected to depict a variety of scenes and situations.
Flickr8k_images: Folder containing 8092 images in jpeg format
Flickr8k_text: Folder containing 5 text files described below:

- Flickr8k.token.txt - the raw captions of the Flickr8k Dataset . The first column is the ID of the caption which is "image address # caption number"
- Flickr8k.lemma.txt - the lemmatized version of the above captions
- Flickr_8k.trainImages.txt - The training images to be used in experiments
- Flickr_8k.valImages.txt - The validation images to be used in experiments
- Flickr_8k.testImages.txt - The test images to be used in experiments

BLEU score:

BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translation. A perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0. The score was developed for evaluating the predictions made by automatic machine translation systems. We take the geometric mean of the test corpus' modified precision scores and then multiply the result by an exponential brevity penalty factor. We first compute the geometric average of the modified n-gram precisions,pn, using n-grams upto length N and positive weights wn summing to one. Next, let c be the length of the candidate translation and r be the effective reference corpus length.We compute the brevity penalty BP, and then BLEU.

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

Proposed Model architecture:

The first step is to load the FLICKR8K images and convert these images into latent variables. For this, **Googlenet**, which is a trained model available in torchvision, was used as it is trained over a large dataset and able to detect objects. The images are preprocessed using resizing, recentering and normalizing, etc. as per the model Googlenet. Overall we get a **1024 latent variable** for all the images. This is given in latent_representationDf.

After generating the latent variables, we create a vocabulary dataset which will define the set of words that will be used for generating the caption. Now, using the caption file having 5 descriptions/captions for each image, we create a dictionary of tokens. The total tokens on the training dataset is very high and in order to reduce it, we set the condition that a token must repeat more than once. Once the tokens are now generated for each image, we now convert these tokens into index numbers for faster training of our model. We can set an index for each token and using these indexes we can also recover the word it is associated with. This is given in tokenized_descriptionsDf.

Now that the words of a sentence are encoded with index, we need to translate it using one-hot encoding which gives a vector of length equal to our vocabulary size. We use word embedding **word2vec** with the **Continuous Bag of Word** variant. This encoding represents words as numerical vectors that transpose word meaning similarities into vector similarities. Using CBOW, we predict a center word given the context word.

We then create the PyTorch model to predict this center word and train it. This word embedder is trained for 1000 epochs with Adam optimizer and learning rate set to 0.0001, and the loss function of Cross Entropy.

After training the model, we got a training loss of 2.408 . The word embedding is the weight matrix stored in the encoder embedding layer.

Now we train an LSTM model using latent representation of the image as input, and the model predicts the most likely word that will be produced after a set of given words in the caption. We train the LSTM model using the embedded dictionary and the image latent variable. And then we use validation set to check what the model is learning and verify it with the loss. Here, the number of epochs is 500, with Adam optimizer, dropout of 0.5

and Cross Entropy loss function. Hidden dimension is 32, number of layers is 2. After training, we get training loss of 3.32 and validation loss of 3.39.
After the training, now we put in images in test dataset to generate captions.

Results:

There are 1000 images in test dataset. We generate captions for 5 images on the test set. The image along with the caption the model generates is given the figure below:

Fig. Sample Test 1

Fig. Sample Test 2

Fig. Sample Test 3

Fig. Sample Test 4

Fig. Sample test 5

Now, the BLEU Score was calculated as an evaluation matrix for test image captions as reference corpus and the predicted caption of images as candidate corpus. When the LSTM model was trained for 500 epochs, the BLEU score results are given below:

```
BLEU_1 = bleu_score(candidate_corpus, reference_corpus, max_n = 4, weights = [1,0,0,0])
BLEU_2 = bleu_score(candidate_corpus, reference_corpus, max_n = 4, weights = [0.5,0.5,0,0])
BLEU_3 = bleu_score(candidate_corpus, reference_corpus, max_n = 4, weights = [0.3,0.3,0.3,0])
BLEU_4 = bleu_score(candidate_corpus, reference_corpus, max_n = 4, weights = [0.25,0.25,0.25,0.25])
print("BLEU_1: {},\n BLEU_2: {},\n BLEU_3: {},\n BLEU_4: {}".format(BLEU_1,BLEU_2,BLEU_3,BLEU_4))
```

```
BLEU_1: 0.5565337538719177,
 BLEU_2: 0.3538365066051483,
 BLEU_3: 0.24337860941886902,
 BLEU_4: 0.11593033373355865
```

In order to see the progress of predictions made by the model on test dataset, the table below summarizes the BLEU score for different epochs.

Number of epochs	BLEU-1	BLEU-2	BLEU-3	BLEU-4
150	0.3643	0.1998	0.1159	0.0371
500	0.5565	0.3538	0.2433	0.1159

As the number of epochs increase, our model does better job at predicting the caption of an image.

LSTM model summary:

Finally, in order to check the constraint on the model, we see the LSTM model summary. The architecture is given below:

```
print(lstm_model)
```

```
LSTM_fixed_embedding(
  (lstm): LSTM(128, 32, num_layers=2, batch_first=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (linear): Linear(in_features=1056, out_features=4430, bias=True)
)
```

The total number of parameters is 4711694 i.e. approximately 4.7 million. And the total size of the model is around 18 Mb. Thus, the constraint on the model size of 100 Mb is satisfied.