CS 838 Assignment-3

Summary and analysis of the paper:
"Show and Tell: A Neural Image Caption Generator"

Siddhant Garg, Goutham Ramakrishnan, Varun Thumbe

Part-I Paper Summary:

1) Goal of the paper:

The goal of this paper is to design an end to end system to generate captions for an image. This is a non trivial problem as the system not only needs to identify the objects in an image but also needs to understand the relation these objects have with each other as well as their attributes and the activities they are involved in. Additionally this relation needs to be expressed as a natural language caption and this necessitates the requirement of a language model in addition to a visual understanding system.

2) Key Ideas:

Previous approaches to image captioning used a trivial approach by break this into two sub-problems and solving them individually. This paper was the first to introduce an end-end system for this problem which outperformed the state of the results at that time. Given an image *I* and the corresponding caption statement $S = \{S_1, S_2, \cdots, S_n\}$, the idea is to maximize the likelihood $P(S \mid I)$, where $S_1, S_2, \cdots, S_n$ are words in the caption for the image.

When expressed as the above stated likelihood maximization problem, this resembles the machine translation problem wherein we need to convert a text from a source language to a target language. In this case, the goal is to maximize $P(T|S)$ where S is a sentence in the source language and T is a sentence in the target language. The source sentence can be passed to an encoder RNN to get a fixed length representation which can be used as the initial hidden state of a decoder RNN that generates the target sentence. Since RNN face the problem of vanishing and exploding gradients in sequence predictions tasks, an LSTM network is used instead as the decoder.

Extending the above idea directly to image captioning has some shortcomings since images and captions belong to different domains. And hence the challenge is to convert them to a similar mathematical representation and fine tune their representations accordingly which is done through the proposed model as described in the next section.

3) Proposed Model:

Words are represented mathematically through fixed dimensional word embeddings. Image features are best extracted through convolutional neural networks and hence a ResNet-152 model with pre-trained weights is used to get the feature representations. The parameters of the last fully connected(fc) layer of the ResNet architecture are modified so that its output embedding and the word embeddings for the captions have the same dimensions.

The loss function is a sum of negative log likelihood of predicting the correct word at each step which is then back-propagated using stochastic gradient descent to update all the parameters of the decoder network. Only parameters of the last fc layer of the encoder CNN is updated, along with word embeddings and the parameters of the LSTM network. The weights of all the other layers of the CNN encoder are not updated. Experimentally it was observed that initializing the word embeddings from a pre-trained corpus did not have a major effect on the training attributing to the fact that the semantic space where the image and word embeddings have similar semantic meaning is an altogether different vector space.

For doing inference, the paper proposes two techniques. The first one is using a Beam-Search decoder where the best k sentences upto time t are considered as candidates to generate sentences of size t+1 instead of having a single token being used to predict the next token word. The other method uses a sampling trick (greedy decoding) where the feature vector extracted from the image through the encoder CNN is used to predict the first word token in the caption and the word embedding of this predicted token is used to predict the next token word in a sequential manner. The decoding is terminates when a special end of sentence (EOS) token is outputted by the LSTM decoder. The github code we are experimenting on uses the second method of greedy decoding.

4) Experiments:

The experimental results were carried out on 5 datasets for image captioning: Pascal VOC 2008, Flickr8k, Flickr30k, MSCOCO and SBU. The evaluation metric used for the image captions were the BLEU score, CIDER and METEOR score. Some experimental decisions involved initializing the CNN encoder with pre-trained ImageNet weights, initializing the word embeddings with pre-trained weights, using dropout regularization and ensembling. All experiments were carried out with a fixed size (512) of the LSTM and word embeddings. The model fares well when compared to human raters on the MSCOCO dataset and also beats the state of the art image captioning models on the other datasets. However it when evaluating the captions using human raters on the other datasets, the model fares much more poorly. Further experiments are carried out on using transfer learning to a neural image captioning model over Flickr8k and Flickr30k datasets by increasing the size of the training data (thereby improving the captioning results) and also using a trained model on MSCOCO for inference on the PASCAL dataset. Further the authors carry out qualitative experiments by discussing the diversity of the captions generated and ranking the generated captions and comparing them with human evaluation thereby suggesting the shortcomings of some evaluation metrics like BLEU. Another interesting qualitative analysis undertaken by the authors is to study the closest word embeddings of the generated captions for an image to understanding other possible output captions.

5) Findings and Conclusion of the paper:

The main contributions from the paper include providing an end-to-end trainable neural network system that can automatically view an image and generate captions using a convolution neural network that encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence. Experiments carried out on different datasets show the robustness of NIC in terms of qualitative and quantitative evaluations, using either ranking metrics or BLEU. Another interesting observation made is the improvement of metric scores of image captioning on increasing the training data size. Further extensions to this work involves providing unsupervised text or images to improve the captioning scores or to involve an attention mechanism for the same.

Part-II Experimentation with code:

We have used the code provided by https://github.com/yunjey for our experiments.

1) Implementation:

Dataset:The code is implemented on the MSCOCO dataset which has around 80000 training images and 40000 validations images. For loading the data, the pycocotools library is used which is a module written by the repository owners. The entire code can be structured as 5 modules as follows:

a) Building Vocabulary:

A separate class named Vocabulary stores the mapping between words and keys(integers) and vice versa. Caption labels from the training set are used to build the vocabulary. The captions are loaded using pycocotools and tokenized using the nltk module .The tokens which have a total count less than a threshold(4 in our case) are dropped and others are added to the vocabulary object. Special words like '<pad>', '<start>' and '<end>' are also added to this object. Finally this object is dumped onto a pickle file to be used later as the vocabulary.

b) Resizing:

The training set images are loaded, resized to [256 x 256] and then saved in another folder.

c) Model:

Two PyTorch nn.Module classes are created for the model: Encoder-CNN and Decoder-RNN. Encoder-CNN uses a pretrained ResNet-152 model with its last layer fc layer modified so that the output has dimensions equal to dimension of the word embeddings. There is also a batch-normalization layer added at the end in Encoder-CNN. In the Encoder-CNN only the parameters of the last fc layer are updated while all the other parameters are fixed to their pre-trained values.

The Decoder CNN code is more involved mainly because of the difference in the way training and inference is done. An additional sample function is created for this class to be used for the inference. This module has three main components. First an embedding layer that converts the caption words to its corresponding embedding, a LSTM layer that converts these to hidden embeddings and a fc layer that converts the hidden embeddings to a vector having size equal to the number of vocabulary words.

Design parameters used by the model are as follows:

Embedding size: 256
LSTM hidden embedding size: 512
Vocabulary size: 9956 (after rejecting tokens with total count<4)
Batch size for training: 128
Number of epochs for training: 5
Number of LSTM layers: 1

During training, the caption words are passed through the embedding layer. These word embeddings and the image feature embedding are concatenated together and passed to the LSTM network and the loss corresponding to each output word is back-propagated. However, since each caption in a batch can have different lengths, the captions are padded to the maximum sequence length in that batch. To optimize computations, pack_padded_sequence is used which stacks the caption words of all the batches together. The output of this pack_padded_sequence function is fed to the LSTM network to calculate the predicted captions across all batches.

d) Training:

The resized data that was created using the Resizing module is loaded and data augmentation techniques like Random crop, Normalization etc are performed on the image to convert its dimensions to 224*224 (Input dimensions required for ResNet-152). The encoder and decoder model are initialized with the design parameters described above. The loss function used by the model is the cross entropy loss function and it is optimised using Adam optimizer(torch.optim.Adam). The loss is computed between the each target caption word and predicted caption word across the whole batch. The target vector is also formed by stacking all caption words using pack_padded_sequence function. The loss is then back-propagated to update the trainable parameters. The model is continuously saved after every 1000 updates.

e) Inference:

The inference code is written for predicting the caption of a single image where the image is normalized similar to training (using the same mean and standard deviation parameters). During inference, we take the image feature embedding and pass it to the LSTM network to compute the first caption word. The embedding of this caption word is used as input to the LSTM network to compute the next word caption and so on. The process is repeated until max_seq_length(parameter given by the user) number of words are predicted or the ‘<end>’ word token is predicted whichever comes first. This represents the sampling method(greedy-decoding) for inference that we discussed in the paper summary section.

2) Modification to Inference code for our experiments:

The validation images folder was not being used by the original code repository. So, we decided to use it as the test set (our images). The experiments we carried out are as follows.

- We ran the resize code on the validation images dataset of MSCOCO to convert them to 256*256 dimensional images and saved it another folder.
- The inference code already available in the code repository does the inference for a single image. We created a new script that does the inference in batches(batch_size=128) rather than for a single image. We used this script to compute captions on the resized images of validation dataset of MSCOCO and stored these captions in a separate json file.
- We wrote a script to compute the BLEU score of the predictions given the json files of predicted and ground truth captions and ran it on the captions we saved for the MSCOCO validation dataset images.
- We wrote a script to compute captions on some of our own images(taken from the internet) to compare its results to the ones we have on images from the MSCOCO dataset.

Results:

The pretrained model provided in the code repository has only been trained for 5 epochs over the MSCOCO train dataset (encoder-5-3000.pkl and decoder-5-3000.pkl). We conjectured before running inference on new images that this model will not perform as well as the model reported in the paper for image captioning which achieves a BLEU-4 score of 0.272 on the MSCOCO test images. Since the aim of the assignment was to utilise a pre-trained model, we used this inferior pre-trained model towards running inference on the MSCOCO validation images and the extra images extracted by us from the internet. The average bleu scores obtained on the validation set images of MSCOCO dataset are as follows:

Average BLEU-1 score is 0.361
Average BLEU-2 score is 0.105
Average BLEU-3 score is 0.038
Average BLEU-4 score is 0.016

Qualitative experiments on additional images derived from Google:

1)

Predicted caption: a young boy eating a piece of cake with a fork .
Analysis: Good Result
This result is pretty good as the only thing the model predicted wrong is the piece of cake instead of noodles. The interesting thing to note is the fact that the model is able to recognize that the boy is young even though it wasn't able to recognize noodles (the token noodles is present in the vocabulary of the model). This may be explained from the fact that there are a lot of images of a young boys in the COCO dataset and not enough images of noodles as an object of concern in the images. As a result the model couldn't learn the high level feature details of noodles and henceforth couldn't predict it in the captions.

2)

Predicted caption: a small dog laying on a wooden table .
Analysis: Bad Result
The 'rabbit' and 'carrot' words are both present in the vocabulary of the model, but still the model did not predict either of them.

Qualitative experiments on additional images derived from MSCOCO:

1)

Predicted Caption: a group of people standing next to each other holding a cell phone.
Analysis: Good Result
In this case, the model predicts the information about holding cell phones incorrectly but correctly captures the information of a group of people standing next to each other.

2)

Predicted Caption: a man is preparing to eat a pizza .
Analysis: Good Result
In this case the model correctly captions the image.

3)

Predicted Caption: a bathroom with a toilet and a sink .

Analysis: Good Result

In this image the model also predicts a sink in the image. Maybe it interprets the fixture above the toilet as the sink. But other than that this is a good prediction.

4)

Predicted Caption: a close up of a pair of scissors on a table .

Analysis: Bad Result

This image can be considered as a bit of out of context image compared to other COCO images, since bikes generally don't have a clock as its tyre. Thus it is not very unexpected for the model to not perform well on such outlying images.

3) Conclusion:

In general we find that the results on the COCO dataset images are a bit more appealing mainly because it was being trained on the context specific to the dataset and a majority of the images can be successfully labelled using the vocabulary that the model currently has. Thus if an object like a clock appears multiple times in the training dataset, it is expected to appear a proportion number of times in the test dataset as well since a random split is used to generate the test-train split. Inferring captions on images which have frequently occurring objects similar to train images gives good captioning results while images which contain objects which are not common in the training corpus give extraneous results.

Contributions towards Assignment 3:

All three team members have an equal contribution towards this project, details of which are listed below:

- Read and understanding the paper: Varun, Siddhant, Goutham [Equal]
- Downloading images and formatting them: Goutham
- Script for BLEU score: Varun
- Script for captions on additional images: Siddhant
- Report writing: Varun, Siddhant, Goutham [Equal]