2017CS50417 - Ritvik Vij

2017CS10322 - Aditya Panwar

<u>Image Captioning in Italian</u>

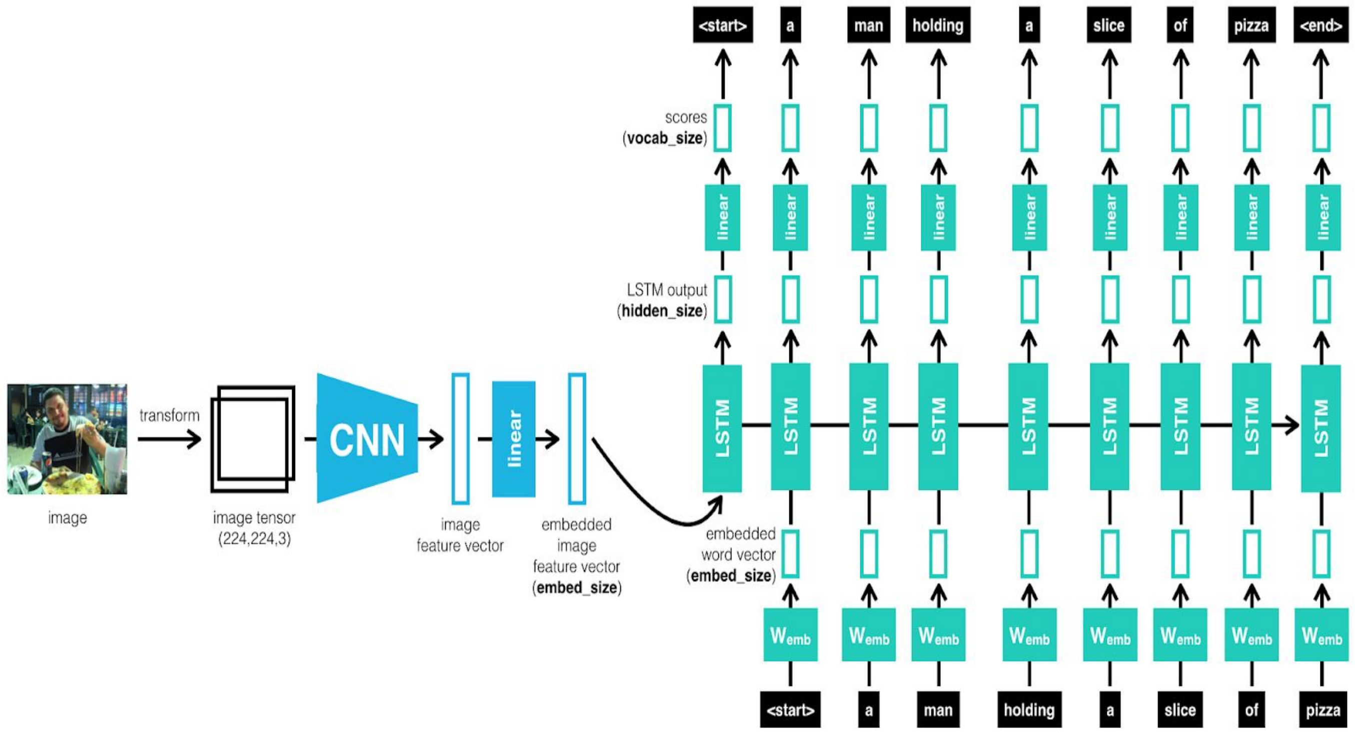
image
transform
image tensor (224,224,3)
CNN
image feature vector
linear
embedded image feature vector (embed_size)
embedded word vector (embed_size)
LSTM output (hidden_size)
scores (vocab_size)
LSTM
linear
Wemb
<start> a man holding a slice of pizza <end>
<start> a man holding a slice of pizza

Introduction

For the task at hand, we use a sequence to sequence model which uses a CNN-based encoder which passes out it's feature vector output to the RNN based Decoder. The RNN Decoder uses certain variations such as the LSTM and the GRU cell units.

Pre-Processing

For Preprocessing Images, we have used transforms function from torchvision, the smaller dimension of the image is first resized to 256 and a random crop of 224*224 is taken out of the image which is then sent onto the image batch from training. All Punctuations were kept intact.

Dataset Building

For building the vocabulary, we add to our vocabulary dictionary every new word from the captions that is encountered. For building token vectors which represent sentences as a list of numbers representing the dictionary key, we start it with the start token, fill in the words in the token vector and append the end token at the end of the sentence. Later on when these captions are sent as batches, we use torch’s pack padding function to pad captions to the same length of the max caption amongst them before being sent into the LSTM. We use a batch size of 64 which are chosen by shuffling in the dataset. The NUM_WORKERS used is 4 to parallelize data loading procedure.

ENCODER

We use two types of encoder architectures-one for the non - competitive part(scratch training) and another for the competitive part(pre-trained model).

1. We use standard alexnet architecture for scratch training. An embedding/linear layer is added in the end to convert output of the alexnet into the feature vector size we want. We have chosen embedding size to be 512.
2. As a pre-trained model, we use ResNet-50 and an embedding layer similar to the above part in the competitive part.

DECODER

The Decoder is a RNN Architecture which converts the embedded feature vectors into sequences(captions). We experimented with LSTM cells and GRU cells for this operation. GRU cells, however faster in implementation turned out to give lower accuracy.

While training, we concatenate the feature vector with the ground truth captions which is then passed on into the LSTM for training. The LSTM output is then embedded into vocabulary size so that for each caption, we have a list of scores of all words in the vocabulary per word in the caption.

TRAINING

The Hyperparameters we have used during training are as below :-

```
EMBED_SIZE = 512

HIDDEN_SIZE = 512   #Hidden Size in LSTM

HIDDEN_LAYERS = 1   #Number of hidden layers used in LSTM

NUMBER_OF_EPOCHS = 20

LEARNING_RATE = 1e-3

BATCH_SIZE = 64

NUM_WORKERS = 4

loss_function = nn.CrossEntropyLoss()

Optimizer = optim.Adam
```

PREDICTION

After training, we output the feature vector from the encoder as we did during training, this feature vector is passed onto the prediction function. States from the previous cell is sent as input to the next LSTM cell. At each step, the words in the past are used to predict the current word using this technique. At each step, we greedily choose the word amongst the vocabulary with maximum score. The sentence construction stops as soon as we find the end token.

CITATIONS :

1. https://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes06-NMT_seq2seq_attention.pdf
2. https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/
3. https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html
4. https://towardsdatascience.com/automatic-image-captioning-with-cnn-rnn-aae3cd442d83
5. https://arxiv.org/pdf/1411.4555.pdf