COL774 Assignment 4

Suchith (2021AIZ8323) and Mridul (2021AIZ8322)

2 Non-competitive part

2.1 Pre-processing

- Building on the `starter_code.py`, the captions were first enclosed within START and END tokens, and then padded with PAD tokens to make them all fixed length. Let's call this fixed length ℓ.
- The captions were tokenized at word-level. Character-level tokenization with OCR task was considered, but we didn't have the right dataset to train an OCR like model.
- After tokenization, they were converted into one-hot vectors. There were 7739 tokens.
- Images were resized to fixed size. Thresholding the images to remove the background details and converting to grayscale was tried, it didn't help.
- In images, we used the EAST text detector in opencv to crop tightly to the text. The results were poor.
- Som in conclusion, the captions were tokenized, enclosed in start and end markers and padded to fixed length ℓ after which a vocabulary mapped it to integers which was then mapped to one-hot encodings. Images were resized to fixed width-height.

2.2 CNN Encoder

We made a simple CNN block to extract features from images comprising of convolutional layers and pooling layers. Default kernel size and stride size were used.

2.3 RNN Decoder

- The output of the CNN decoder was first fed into a linear layer to create attention weights. It was then multiplied with features output by encoder.
- This was then passed into two other linear layers which would output initial hidden state and cell state to be used by RNN decoder.
- We used `torch.nn.LSTM` in biderectional mode first, 2 layers deep. We soon realized a problem here: this only gave the final output. So, we switched to `torch.nn.LSTMCell` that would give out the hidden and cell states at each time step and it can be used to run it for as many time steps as needed.
- So, in the `torch.nn.Module.forward` method (inherited by our model) we had a `for` loop running for the length of the sequence which was the same as ℓ (see page 1).
- The hidden state at each step is seen as the unnormalized log probabilities over all possible words.
- We did teacher-forcing, just as asked for.

2.4 Loss

We used negative log likelihood loss to maximize the likelihood of each word at each time step for each example. So,

$$P_v(x^{(i)}) = [\text{softmax}(h_\theta(x^{(i)}))]_v,\ v \in V$$
$$LL(\theta; x, y) = \sum_{i \in M} \sum_{t=1}^{T} \sum_{v \in V} 1\{y_t^{(i)} = v\} \log P_v(x^{(i)})$$

where with abuse of notation, $h_\theta(x^{(i)})$ is used to mean the hidden state output by the rnn and time t by the RNN decoder. V is the set of all words, the vocabulary. M represents the mini-batch.

The parameters of the model, both encoder and decoder are then trained in an end-to-end fashion using SGD optimizer.

3 Competitive part

3.1 CNN Encoder

For the CNN Encoder, pretrained `RESNET101` was used.

3.2 RNN Decoder

For the decoder, GRU based decoder was used

3.3 Loss

Negative log likelihood loss at each time step was used just as in non-competitive part with regularization.