COL774: Machine Learning
Assignment 4

Devanshi Khatsuriya, 2019CS10344
Anjali Sharma, 2019CS50422

Reading (Noisy) Captions Embedded in Images

In this assignment, we develop a deep learning model that learns to read english captions that are embedded in images (but may or may not correctly correspond to the background image). We implement the following Encoder-Decoder architecture in *Pytorch* to model this problem:

- **Encoder:** The encoder is a Convolutional neural network that takes as input images of variable sizes and maps them to 84 length feature vectors. An input image of dimensions $(H \times W \times 3)$ is first resized and transposed to $(3 \times 256 \times 256)$. The exact specifications of remaining architecture of the encoder and decoder for the non-competitive part and the competitive part are mentioned below.

  - **Non Competitive Part**
    First a convolutional layer with 3 input channels, 6 output channels and 4×4 kernels with stride 2 is applied. This gives $(6 \times 127 \times 127)$ images. This is followed by a *ReLU* layer and a max pooling layer with 2×2 kernels with stride 2. This gives $(6 \times 62 \times 62)$ images. After that, a convolutional layer with 6 input channels, 16 output channels and 5×5 kernels with stride 3 are applied followed by the same *ReLU* and max pooling layers. This gives $16 \times 10 \times 10$ images to which a third convolutional layer of 16 input and output channels and 5×5 kernels with stride 1 is applied. This is again followed by the same *ReLU* and max pooling layers resulting in images of size $(16 \times 3 \times 3)$.
    Finally, the images are flattened to 1 dimension and two linear layers (each followed by a *ReLU* layer) are applied to first map to 120 and then to 84 length feature vectors.
  - **Competitive Part**
    In the competitive part, we make use of the pretrained $ResNet50$ Model to encode the images. We remove the last fully connected layer of the $ResNet50$ Model, then apply a linear layer to map to 84 output features. After that, we apply a *ReLU* layer followed by a batch normalization layer.

- **Decoder:** The decoder is a LSTM recurrent neural network. First, all captions are converted to 10-length token representations using a vocabulary that maps each word to an integer token. The *start* and *end* tokens are added to each token vector, and *pad* and *unk* tokens are also added where required. Each token is converted to 84-length embedding, so each caption can be represented as 10×84 size embeddings. Then, the 84-length encoded image features are concatenated with this caption embeddings (except the last embedding). This is done to make use of the *Teacher Forcing* method in training, which provides the ground truth at each time step for the next prediction. A dropout layer with probability 0.5 is applied, that randomly makes some of the features 0. This is given as input to an LSTM with 1 layer having input feature size 84 and hidden

layer feature size 168. This is then passed into a linear layer that maps output features of size 10×168 to $10 \times$ vocab size. This final output represents the predicted probability distribution (not normalized) of each word of the caption over the vocabulary words.

Training Strategy

As mentioned in the Decoder section, we make use of the *Teacher Forcing* strategy to train. We use cross-entropy loss between the predicted distribution for caption words and the actual caption words as the loss function. We use the Stochastic Gradient Descent method to optimize the loss function.

We trained 3 epochs on 50,000 train images with batch size 32. We used a learning rate of 0.1. The loss decreased from around 8 to 3 after training. The loss can be further optimized by hyper-parameter tuning.

Inference at Test Time

To predicted the caption embedded on a given test image, we encode the image to its features using the Encoder and then make use of the Decoder's LSTM and linear layers to generate words one by one by feeding the previous outputted word to get the next word.

We have used a Greedy Search method to make the predictions. We choose the word with the highest probability to be in the predicted caption, and also use it to generate the next word. An alternative method would be to use the Beam Search method, which keeps track of k- best words at each index. So, in general, it can give captions that have higher probability according to the trained model compared to Greedy Search.

Evaluation Scores

We report the BLEU (BiLingual Evaluation Understudy) scores (evaluated by computing the character-level BLEU scores for each predicted caption with respect to the corresponding target caption and taking mean over all examples) of the trained non-competitive and competitive models on a test set of 5000 images.

- **Non Competitive Part: 0.139**
- **Competitive Part: 0.139**