Matthew Rogers

Homework 2 Report 10/18/24

GITHUB : https://github.com/mrog9/Deep_Learning.git

The convolutional neural network used had three layers: a 2d convolutional layer, a max pooling layer, and a 3d convolutional layer. The data had a shape of (1900, 3, 100, 100). In order to get this shape, each video was converted to a tensor with a (frame_length, 3, 100, 100) shape where the width and height were 100 pixels each. Since the frame_length was different for each video, the tensors were eventually padded to a length of 1900. The video then went through the first layer of the CNN model, a 2d convolutional layer. The reason why a 2d convolutional layer was used first was so that the network could start recognizing objects in each frame. Then the output of this layer was fed into a max pooling layer with a kernel size of (50, 50) and a stride of (50, 50), reducing the dimensions of each feature map to (2,2). The reason behind this layer was, since these were relatively simple videos, the model should only concentrate on a small number of areas in each frame. Then, the output of this layer was reshaped into a (1, 16, 1900, 2, 2) shape before proceeding into a 3d convolutional layer with the number of output channels equal to 10 and a kernel size of (100,1,1). This layer is supposed to see the temporal relationship with the "objects" located from the previous layers. The output shape was (1, 10, 19, 2, 2) and was reshaped into (19, 1*10*2*2) before being fed into a linear layer of 40 input nodes and 100 output nodes for classification. A RELU was used for the 2d convolutional layer and the 3d convolutional layer. A log_softmax was used for the final linear layer.

This CNN model was pretrained by using 100 different videos. These videos were upsampled to 1000 but, since each video was randomly transformed, the CNN was effectively trained with 1000 different videos with 100 different classfications. Each frame of each video could be randomly rotated by thirty degrees and horizontally flipped while each video was reduced by 30% of its original padded size where the frames used were randomly chosen. The model was pretrained using Adam optimizer with learning rate of 0.001 and cross entropy loss was calculated for each video in comparison with the target classification.

Before training the sequential-to-sequential encoder-decoder model, the vocab (word:idx) and rev_vocab (idx:word) was formed. These dictionaries were formed by taking the first caption of each video. The pretrained CNN was loaded and hooked on to this encoder-decoder model by removing the last linear layer (so there were 2d convolutional, max2d, and 3d convolutional layers). The encoder contained a single lstm layer with input size equal to 40 and hidden nodes equal to 50. The hidden layers for each part of the sequence, the final hidden layer, and the final contextual layer was fed into the Decoder. The decoder was composed of an lstm layer of input size equal to the vocab length and hidden nodes equal to 50. This layer was then connected to a linear layer which had an input size of 50 and output nodes equal to vocab length. This last layer was used to classify an individual word.

The sequence length of the input going through the encoder and the decoder was 19. This was going to be the length of my sentence. This meant that each caption or target sentence was transformed so that it had 19 tokens. The target sentence always started with <BOS> token, its words, an <EOS>

token, and padded with <PAD> to 19 tokens if needed. This also meant that decoder was stepped through 19 times. For the first iteration, a tensor of shape (1, vocab length) and zeroed was the initial input to the decoder along with the final hidden and contextual state from the encoder. The output was then put through linear layer to eventually get inference word. Attention and scheduled sampling was used afterwards. The sampling was scheduled to iterations of the training. It was made in such a way that it would be less likely to have scheduled sampling as the training proceeded. Attention was used to find a relative contextual state based on a sort of weight from the sequence hidden states of the encoder. The way this contextual state was calculated by taking the dot product of the sequential encoder hidden states and the transpose of the decoder's previous hidden state. Softmax of this dot product was taken to get attention weights and these weights were multiplied into each row of the sequential encoder hidden states. To get the contextual state, each column was then summed. This produced a (1,50) tensor that was used for the current step of the decoder. This process produced the probabilities of 19 sequenced words. In other words, the decoder outputted a (19, vocab length) tensor. The cross entropy loss was then calculated for the inference sentence and the target sentence, was backpropagated, and optimized using Adam with a learning rate of 0.001. The training used all 1450 videos and went through 10 epochs.

To evaluate the model, attention and beam search was used. Five best words were initially outputted in the decoder. Then for each word, a sequence was produced for the next three words. Then the best three sequences were chosen and the its previous word and the words from the sequence were concatenated. The process then continued until there were 3 sequences of 19 words. Attention was used for each time a single word was produced. The 100 videos in the testing directory were used for the evaluation. The BLEU evaluation score was not even used because it was noticed that the same sentence or a similar variation was outputted for each video. The model clearly was stuck in some kind of "local minimum" in the sense that the training or the model did not allow itself to be able to converge to "different" minimums for the corresponding video. There are three possibilities that were formed to come up for this result. The first one is some kind of programming error in the attention or beam search mechanism that did not allow for variation like what was intended. The second could have been the "cooling schedule" of the scheduled sampling. Perhaps the probability of sampling needed to imitate the shape of another function. In this particular training, the "cooling" was linear. It was based on the iteration of the training. It could have been based on 1 – log instead, allowing for targets to be used frequently for a more significant part of training and then tailor off to letting the model to rely more on its own. The third possibility is that the pretrained CNN was too simple and not robust enough to handle the more significant features of the videos. In the future, a more complex model convolutional neural network will need to be used and see if it provides more distinctive features to provide more variability in predicting caption.