Individual Final Report

Introduction

Among the topics in Machine Learning, Computer Vision draws our interest, this leads our group to the study of image caption with the COCO dataset. With a encoder-decoder model based on the tutorial on Computer Vision, we were able to caption the images and generate text that describes the images. And by calculating the BLEU score, we were able to evaluate our resulting captions.

Description of Individual Work

- Read through the paper *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention,* and understand the architecture of the model.
- Research on the LSTM concept including its architecture and formulas
- Focus on the statistical interpretation of the Attention Mechanism in the paper and do further research on the algorithms
- Running Tensorflow image captioning tutorial to understand the overall training and evaluating process
- Summarizing the model architecture

What I have learned from the research

About LSTM

The Long-Short-Term-Memory (LSTM) includes three gates: input gate, forget gate, and output gate. All three gates apply a sigmoid transformation function with a input of

- Xt: input to the current timestamp.
- U: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- W: the weight matrix associated with hidden state

Forget Gate:

- $f_t = \sigma(x_t * U_f + H_{t-1} * W_f)$

Input Gate:

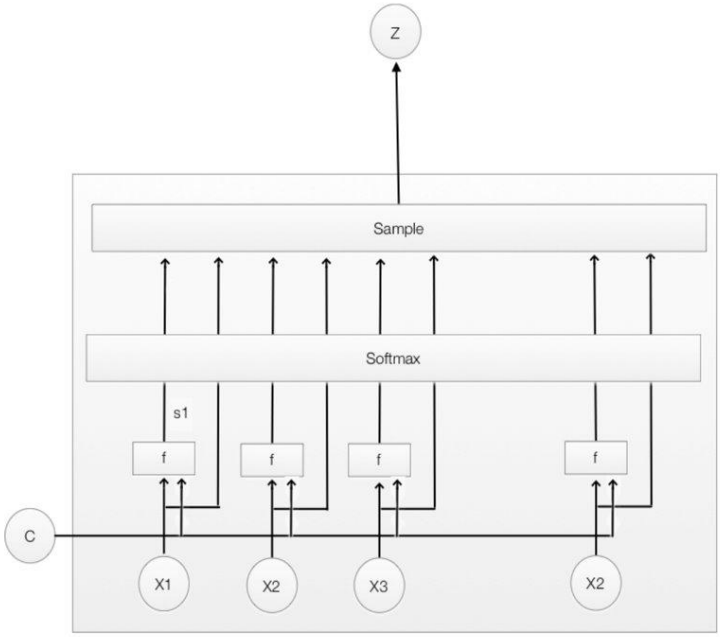
- $i_t = \sigma(x_t * U_i + H_{t-1} * W_i)$

Output Gate:

- $o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$

returning a value of 0 or 1 to implement whether the information is useful.

Stochastic Hard Attention



For Hard attention, our context vector is interpreted as a random variable, denoted as

$$\hat{\mathbf{z}}_t = \sum_i s_{t,i}\mathbf{a}_i.$$

As shown in this graph, it is sampled after the softmax transformation of the attention scores.

St is a location variable while St,i is an indicator one-hot variable where it is equal to 1 when the i-th location is used to extract image features. And ‘a’ is the image feature we are extracting

The loss function is printed below as Ls.

$$L_s = \sum_s p(s \mid \mathbf{a}) \log p(\mathbf{y} \mid s, \mathbf{a})$$

Since s is a discrete random variable and is infeasible to be computed, the algorithm takes the variational lower bound of the log-likelihood of observing word y given the feature ‘a’. We can then derive the weight by the gradient with the formula

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N}\sum_{n=1}^{N}\left[\frac{\partial \log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a})}{\partial W} + \log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a})\frac{\partial \log p(\tilde{s}^n \mid \mathbf{a})}{\partial W}\right]$$

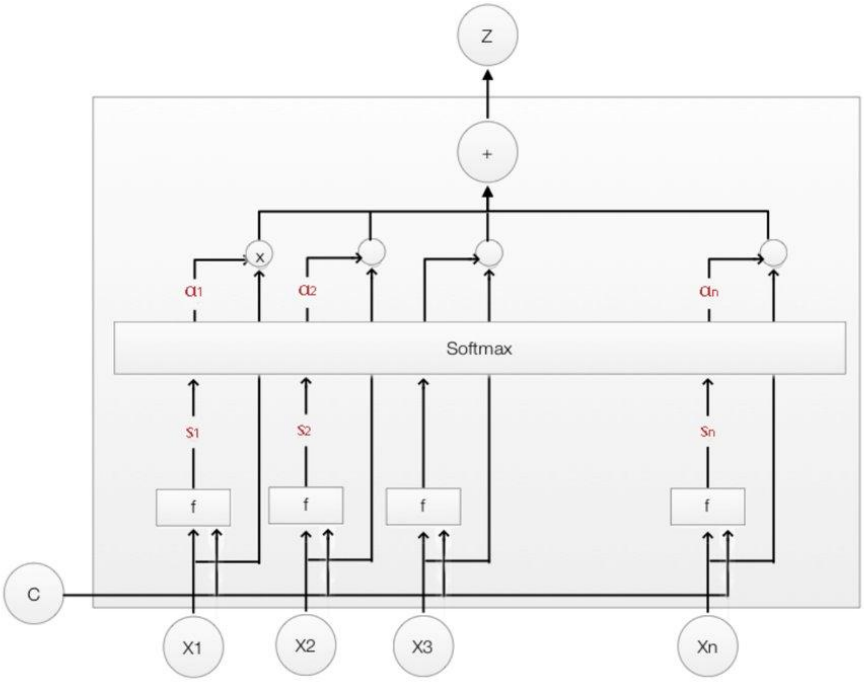
where sn follows a multinoulli distribution with a sample rate of alpha i.

Deterministic Soft Attention

Soft attention is what we used in the decoder in our algorithm. Z instead of being a random variable, is a differentiable equation:

$$\mathbb{E}_{p(s_t|a)}[\hat{\mathbf{z}}_t] = \sum_{i=1}^{L} \alpha_{t,i}\mathbf{a}_i$$

As shown in the figure,



the features x1 as well as the previous hidden state with their weights are first fed into a tanh function to compute score, formulated as:

$$s_i = \tanh(W_c C + W_x X_i) = \tanh(W_c h_{t-1} + W_x x_i)$$

which is a measurement of how much attention is applied to each x.
Then we apply a softmax transformation to the scores to compute the attention weight, implying the probability distribution for each feature.

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})}.$$

Finally we sum up the weighted features to get z.Since the activation functions are differentiable, it is easier to apply backpropagation to the network.

Summary of Model architecture

The model uses a Convolutional Neural Network as an encoder to extract features from the input images and feed them into a Long-Short-Term-Memory (LSMT) Network to calculate a context vector, which is then fed into the RNN decoder, generating a predicted word according to the input convolutional feature at each time.
The CNN encoder transforms a raw image data into a 14*14*512 feature map, as the feature map goes through the flatten layer, it is then transformed into a sequence of annotation vectors. Finally, by concatenating the annotation vectors, we get the image feature, which is passed to the LSMT as a input.
The LSMT includes a function calculating attention score, a calculation of attention weights as well as a calculation regarding the context vector.
The formula for generating attention score is denoted as

$$score_{t,j} = v_a^T * tanh(U_a * h_{t-1} + W_a * h_j)$$

which contains a hyperbolic tangent transformation of the sum of the multiplication of hidden state ht-1 from the decoder of the previous step with its weight and the output feature map from the CNN encoder with its weight. This score implies how important the jth pixel located in the input image is.
After this, the scores are transferred into an attention weight for the jth pixel at time t with a SoftMax function:

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})}.$$

Indicating the probability distribution for each feature map, where the sum of attention weight for all pixels within a single prediction step equal to 1.
Finally, the LSTM network calculates a context vector Ct, by simply summing up the multiplication of each feature and its attention weight calculated in the previous step, formulated as

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j$$

This step combines the extracted features back into an entire image with different weights applied on the blocks.
The last state of the model is an RNN decoder. This decoder is a GRU layer followed by two dense layers. It first takes the embedding dimension through an embedding layer and concatenates it with the context vector calculated by the LSTM to get the input batch size, timesteps and features. Then, this concatenated vector is passed to the GRU to get the output and state. As the output went through two dense layers, it can be finally translated into a predicted word.

Insights on the Attention Mechanism

We could have applied the Hard attention Method to the model since the paper implies that the prediction from hard attention has a higher BLEU score. However, it requires sampling and averaging the result with the Monte Carlo simulation. In contrast, the entire Soft Attention model is differentiable, making it computationally easier to compute the gradient. In this case, we prefer using a soft attention model.

References

Hui, Jonathan. *"Soft & Hard Attention"*, 15 Mar. 2017, https://jhui.github.io/2017/03/15/Soft-and-hard-attention/.

Makarov, Artyom. "Image Captioning with Attention: Part 1." *Medium*, Analytics Vidhya, 14 Dec. 2020, https://medium.com/analytics-vidhya/image-captioning-with-attention-part-1-e8a5f783f6d3.

Sarkar, Subham. "Image Captioning Using Attention Mechanism." *Medium*, The Startup, 15 June 2021, https://medium.com/swlh/image-captioning-using-attention-mechanism-f3d7fc96eb0e.

Sexena, Shipra. "LSTM: Introduction to LSTM: Long Short Term Memor." *Analytics Vidhya*, 16 Mar. 2021, https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/.

Xu, Kelvin, et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." *ArXiv.org*, 19 Apr. 2016, https://arxiv.org/abs/1502.03044.