COL 774: Assignment 4 (Semester I, 2021-22)

Sweta Mahajan(2021AIZ8356)
Subrat Kumar Swain(2021QIZ8247)

August 26, 2022

1 Introduction

The task in this assignment is to read noisy captions embedded in images. We will develop a model based on the Encoder-Decoder architecture. The encoder will take an image as input and generate a hidden representation of it. The hidden representation will then be fed to a decoder which will try to predict the caption embedded in the image. Here, in this assignment, we have used CNNs as encoders and LSTMs (with attention) as decoders.

The primary purpose of the encoder is to capture essential features of the image, which are required for the decoder to complete the specific task at hand. As the architecture used is Convolutional Neural Networks(CNN), it helped to construct salient features from an image. Subsequently, the decoder takes the hidden representation of the image and try to generate the text that it sees. For the purpose of seeing properly, we have implemented soft attention. The role of soft attention is to feed the decoder with a weighted image, focusing on the part where the text is present, rather than focusing on the entire image and treating all regions equally [4].

2 Data Pre-processing

The images which are fed to the encoder were resized to a fixed size i.e., (256,256) and the entries were normalised. For the captions, we extracted the raw captions from the tsv file and then padded the <start> and <end> token to mark the beginning and end of the caption as this information will be required by the transformer and then we added some <pad> tokens to bring all the captions to the same size. We created a vocabulary from the training captions and used it to predict the captions for the test images. We created a dictionary that converts words into indices which are used to get the word embedding which in turn is fed to the decoder.

3 Proposed Method and Architecture

3.1 Non-competitive

For the non-competitive part, the requirement was to create an Encoder-Decoder architecture. For the encoder part, we implemented VGG-19, from the first principle, without using any pre-trained weights. We used VGG over other resnet architectures because it was computationally heavy to train resnet. So, we used resnet in the competitive part of the assignment. For the decoder, we used RNN with LSTM cell in the hidden layer. We implemented a word-level LSTM.

Here, Fig [1] depicts the entire flow, starting from an image to generate a caption in the forward propagation. Firstly, we re-scaled an image(assuming batch size = 1, for simplicity) and converted it to a torch tensor which then passed through the CNN(the Encoder). We removed the fully connected layers for the CNN as we're not doing the

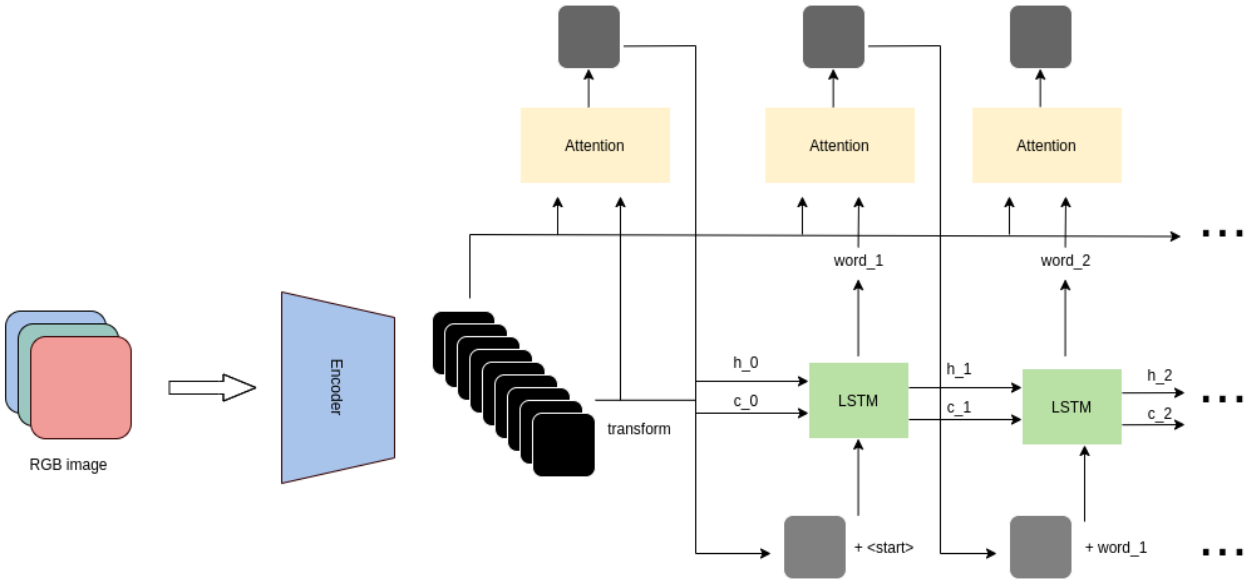

Figure 1: Encoder-Decoder pipeline with attention

classification. Hence, the shape of the output coming from the Encoder is (length * breadth * number_of_channels). We feed this output to two entities; the Decoder and the Attention network. Attention plays a very crucial role in determining the position of the text as we are not training with a dataset with a bounding box around the text inside an image. First, we transform the encoder output to generate the initial hidden state and initial cell state for the LSTM Cell[3]. Then, the attention layer will take encoder output and along with the previous hidden state, output to generate weights(alphas) for the entire image. At last, we concatenate the previously generated word along with the weighted image and feed it to the LSTM to generate the next pair of hidden and cell states and the next word. We trained the LSTM network with the teacher forcing method[2].

3.2 Competitive

For the competitive part, we were a bit flexible to experiment with different architectures because we used pre-trained weights for the encoder. So, we tried out, VGG-16, ResNet-50, ResNet-101 for the encoder. We used a learning rate of 0.1(very large compare to literature) that resulted in gradient exploding. And ResNet-101 was very big for 50,000 training data. For this, it was taking a lot of time and the convergence was also slow. For the decoder, we used the same LSTM with a little modification to the embedding and hidden layer size. The flow of an image, starting from input to the encoder to the generation of output words is the same as the non-competitive part Fig [1].

4 Evaluation & Results

For experimenting with different architectures, we made different configuration files containing information such as model name, epochs, batch size, learning rate, encoder dimension etc. We evaluated the fitness of a prediction with the BLEU score [1]. We ran those experiments and some are stopped in the middle and a few hit the early stopping criteria. We intentionally stopped some of them because of the gradient explosion. For a few of them, some technical issues happened. So, we didn't include them in our report. We only took the final run for a configuration and summarized it in the table below Table [1].

Table 1: Value of randomness for different output sizes(in Bytes)

Encoder Architecture	**Hyperparameters**	**Blue-Character**
VGG16	encoder_lr=0.01, decoder_dim=256, attention_dim=256, epochs=12	0.03
VGG19(non-competetive)	encoder_lr = 0.0001, decoder_dim = 512, attention_dim = 512, epochs=18	0.06
Resnet50	encoder_lr = 0.0001, decoder_dim = 512, attention_dim = 512, epochs=70	0.14
Resnet101	encoder_lr = 0.0001, decoder_dim = 512, attention_dim = 512, epochs=24	0.07

4.1 Observations

We used an attention layer in our network, that helps the algorithm to focus on the part of the image where the text is present, instead of looking at the image itself which contains misleading information or in some cases noisy images. Fig [2] and Fig [3]depicts two such images where our network focused on the portion where the text is present instead of focusing on the entire image.

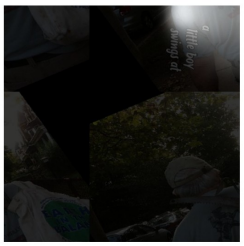

Figure 2: Text on right top(tilted)

Figure 3: Text on left top(straight)

Even though it's focusing on the correct part, there were a few cases where our model struggled to predict the correct text. For a round-shaped font, the model couldn't differentiate between "a" and "o" Fig [4]. The model is able to extract text from an image with high contrast background whereas it couldn't where there is a background with similar colour or background containing text Fig [5].

Figure 4: An is extracted as On

Figure 5: Text with low contrast background

5 Conclusion

We tried to make this report an empirical analysis of detecting and generating the captions embedded in an image. We tried different architectures of the encoder to find the optimal setting. Due to the limited dataset size and computational resources, hyperparameter optimization is not done to a full extent. With proper hyperparameter optimization, a better and more generic model can be developed.

References

[1] BROWNLEE, J. A gentle introduction to calculating the bleu score for text in python. https://machinelearningmastery.com/calculate-bleu-score-for-text-python/, 2017.

[2] BROWNLEE, J. What is teacher forcing for recurrent neural networks? https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/, 2017.

[3] OLAH, C. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

[4] XU, K., BA, J., KIROS, R., CHO, K., COURVILLE, A., SALAKHUTDINOV, R., ZEMEL, R., AND BENGIO, Y. Show, attend and tell: Neural image caption generation with visual attention, 2016.