Final Project - Image Captioning

Simran Nayyar, Aditi Krishnakumar, Dina Dehaini, David Wang
LIGN 167

Introduction

Image captioning is becoming an increasingly relevant tool in today's world. Text to speech processing has gained a lot of popularity for its convenience and image-to-speech processing is the next phase of furthering this convenience. It has many implications such as increasing safety for texting with Apple Carplay (by preventing people checking their phones to look at images) or increasing accessibility for visually-impaired people in reading articles or other image-related works. The main goal of our project is to build a model that, given an image, generates novel sentences describing the overall image as well as the relationships between objects. To do so, we implemented and compared the performances of three variations of encoder-decoder models. For the encoder, we used a pre-trained convolutional neural network (CNN) that extracts relevant features of provided images. For the decoder, we chose to experiment with the following three models: GRU, LSTM, and Transformer. In the end, we compared the BLEU score and validation loss of each model to evaluate the final results. With this project we hope to expand our understanding of the human language and contribute to the convenience and accessibility that image captioning brings.

Dataset

We utilized the MS-COCO dataset which is a common dataset used for machine learning tasks such as object segmentation. The dataset has over 300k images but for the purposes of this project and the GPU restrictions we had, we only used 10,000 images. Each image in this dataset has around five corresponding reference captions that describe the image. The corresponding captions were our reference points for evaluating (both qualitatively and through our BLEU score metric) whether our model was generating accurate predictions.

Model

Image Preprocessing

Our preprocessing of the images and captions from the MS-COCO dataset was duplicated from "Image Captioning with Visual Attention: Tensorflow Core" tutorial which started by assigning image paths with their relative captions in a dictionary structure which was shuffled. We then decreased the size of our data to the first 10,000 images in this shuffled dictionary in order to be able to train our models with our limited memory and space resources. We then created two lists for both the images and captions and used a pre-trained CNN, InceptionV3, to resize and preprocess the images to normalize the pixels and fit them to the format of images the InceptionV3 model was trained on. The model was then used to extract image features and store the resulting vector.

Caption Preprocessing

The captions also underwent standard and nonstandard preprocessing. Prior to vectorization, we ensured that the "<start>" and "<end>" tokens would be preserved. *Adapt* was then used to split words from the captions and create a dictionary with the top 5,000 most frequent words. The captions were then tokenized using word indices from the previously created dictionary and output sentences were limited to 50 words. Finally, we created mappings for words to indices and indices to words.

The data was split into training and validation sets with an 80/20 split and we created a tf.data dataset of batch size 64, buffer size 1000, embedding dimension of 256, and 512 units to train the model.

Pre-trained Model and CNN

We used InceptionV3, which is pre-trained on ImageNet, to extract features from the images. The output layer (8*8*2048) was the last convolutional layer of the InceptionV3 architecture. We passed all the images through the network and stored the

resulting vectors in a dictionary. We then reduced the shape of the vectors to (64, 2048) and passed this through our encoder model ("Image Captioning with Visual Attention:Tensorflow Core.").

The CNN encoder consists of a single Dense layer with an output dimension equal to the embedding dimension (256). We then passed the output through a ReLU function. The output is then ready to be fed to the decoder model ("Image Captioning with Visual Attention : Tensorflow Core.").

Loss function

We first pad the real captions so that all sentences are equal in length. Then we use a TensorFlow function to compute the cross-entropy loss between the real and prediction captions. Then the zero-padded real captions' tensor is casted into the same type as the loss and multiplied to the loss. In the end, the loss function returns the mean of loss ("Image Captioning with Visual Attention : Tensorflow Core.").

Decoder Overview

The features outputted from the encoder, the hidden state, and the decoder input are fed to the decoder model which is initialized with embedding dimension = 256, units = 512, vocabulary size = 5,000. For each batch (size = 64), the hidden state is initialized to zero to represent the fact that the captions of different images are unrelated to each other. The decoder input is initialized to be the '<start>' token at the beginning of a sentence. The decoder model, which is explained in further detail below, returns predictions and the decoder hidden state. The predictions are of shape (batch size * maximum length, vocab size). The decoder hidden state is passed back into the model and the predictions are used to calculate the loss. After the current loss is calculated and totaled, 'teacher enforcing' is applied to determine the next decoder input. Essentially, this means that the target word or ground truth is passed as input rather than the output of the previous time step. This process is repeated for the rest of the sentence. We then calculate the gradients and apply it to an Adam optimizer. Finally, we conduct backpropagation and return the current loss and total loss.

Bahdanau Attention

To improve the performance of our models, we used the Bahdanau Attention Mechanism which allowed us to produce more accurate caption predictions by focusing on more relevant information of the input sentences. We first calculate the attention scores using the formula, $a(s_{t-1}, h_i) = v^T \tanh(W_1 h_i + W_2 s_{t-1})$ where W_1 and W_2 are the two weight matrices, s_{t-1} is the hidden decoder state at the previous time step, and h_i is the annotation focusing around the ith word. Then we normalize all attention scores into weight values by applying softmax on the first axis. Finally, we multiply the attention weights and the encoder's hidden states to compute the context vector, which is then fed to the decoder to compute the final output (Cristina).

GRU

The decoder part of our model essentially translates the features extracted from the images by the encoder into natural sentences. The GRU attends over the image to predict the next word.

The hidden state (initialized to zero) and the encoder output (features from images) are passed to the Bahdanau Attention layer. This layer returns the context vector of shape (batch size, hidden size) and the attention weights which reflect how important the hidden decoder state at the previous time step and the current annotation are in generating the current state. The decoder input is then passed to the embedding layer which converts each word to a dense vector representation with embedding dimension equal to 256 and vocabulary size equal to 5000. This output is concatenated with the context vector generated by the attention layer and passed into the GRU layer (units = 512, return_sequences=True, return_state=True, recurrent_initializer='glorot_uniform').

GRUs are considered to be a variation of LSTMs as they both have similar architecture and deal with the vanishing gradient problem. However, the GRU only contains two gates (update gate and reset gate). The update gate determines how much of the information

from previous time steps should be passed along to the future. The reset gate determines how much of the previous information should be 'forgotten'. Using the two gates, the network is able to pass along the relevant information without washing out the new information each time and thus avoids the vanishing gradient problem (Kostadinov).

The output predictions of the GRU layer are then passed through a dense layer of output dimension equal to 512 (units), and then another dense layer of output dimension equal to 5000 (vocabulary size). The decoder finally returns the predictions, the hidden state, and the attention weights.

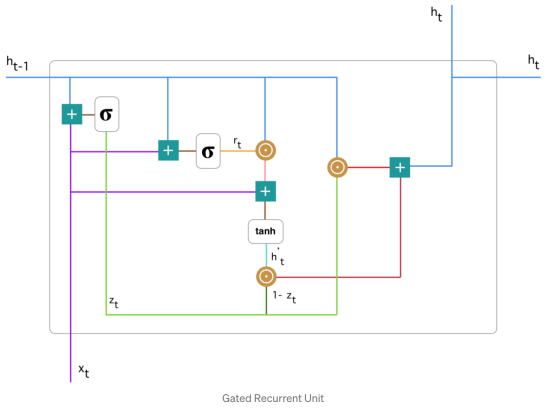


Diagram of GRU cell (Kostadinov)

LSTM

The second model we tested was a Long Short-Tem Memory (LSTM) network. We maintained the same overall architecture, but simply changed the GRU layer to an LSTM (units = 512, return_sequences=True, return_state=True, recurrent_initializer='glorot_uniform'). This model

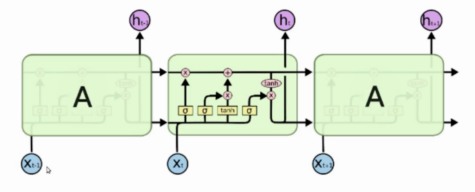


Diagram of the LSTM network (Saxena)

was created to prevent not only the vanishing gradient issue that is present with traditional RNNs, but also their inability to be able to remember long-term dependencies. Thus, a LSTM differs from a GRU because it has three gates as opposed to the GRU's two gates. The three gates of an LSTM are its forget gate, its input gate, and its output gate. LSTMs also have a hidden state (short term memory) and a cell state (long term memory) of both their current and previous timestamp. The role of the forget gate is to forgo irrelevant information and this is equivalent to the network deciding whether the cell state of the previous timestamp should be utilized or not. The input gate determines the importance of newly inputted information - if the information is relevant, it will be added to the current cell state. Lastly, the output gate generates the final product and this is calculated as the softmax of the current hidden state (Saxena).

Transformer

Finally, we tested a transformer network model. Transformers can also use an encoder/decoder architecture similar to the other ones we tested but the way they "learn" and train on data allows them to outperform recurrent neural networks like GRUs and LSTMs in sequence problems and natural language processing (NLP) tasks. What makes transformers so good at these kinds of problems is their attention mechanisms which allow the transformer to receive input sequences, in this case sentences or captions, in parallel so all the words can be passed in simultaneously without any recurrence or convolution (Vaswani, Ashish, et al.). They rely on a multi-headed self-attention mechanism to learn dependencies to accomplish this parallelization which prevents

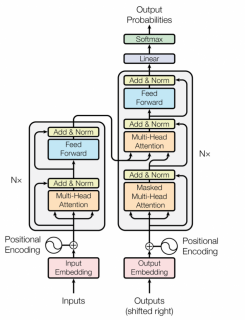


Diagram of Transformer (Vaswani, et al.)

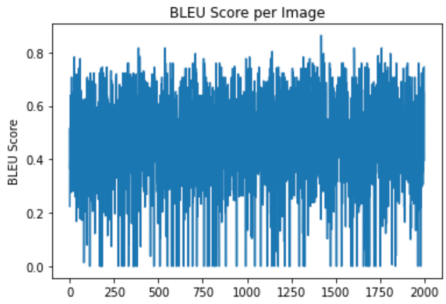
long-term dependency issues often faced by RNN architectures and results in the ability to glean more information about the relationship between words. The transformer's encoder consists of the multi-headed self-attention mechanism and a fully connected feed-forward network. This block produces an attention vector for each word containing information about that word's relationship to other words in the sentence. The transformer's decoder contains an attention block which performs word mapping and discovers dependencies between word vectors. It also contains a classifier layer and softmax to produce such probabilities. Our transformer model was based on (Gautam). It contained 4 layers, an input embedding size of 512 units, an upward projection size of 2048, 8 heads, rows, and columns, and a dropout rate of 0.1.

Results

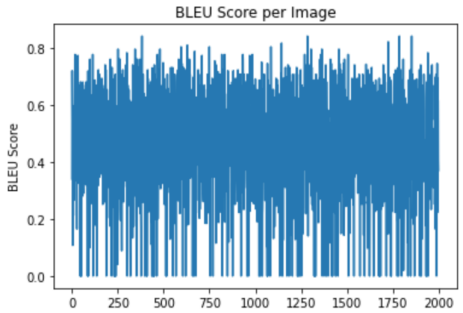
We utilized three different metrics to determine the accuracy of each model we tested. First was a qualitative observation. This was our baseline measure to simply see whether the models were outputting plausible captions for their corresponding pictures. We also compared the predicted caption to the "real" caption provided by the COCO dataset. Since we are looking to have the captions be as "human-like" as possible, it did not matter to us if the predicted caption was not an exact match to the real caption. If it was a statement a human would use to describe the image, we classified it as an accurate caption. All three captions fared well in this section and had similar results. Below is an example from our LSTM model:

The second metric we used was a BLEU score. BLEU score or the BiLingual Evaluation Understudy score measures the similarity of a machine-predicted text to its reference caption. This score works by first comparing individual portions of the sentence and then using an average value over the entire sentence and the values lie between 0 (no matches) and 1 (exact match) (Breitenbach). We calculated the BLEU-4 scores for each image in the testing set for each model. The BLEU-4 score utilizes 0.25 weights when calculating. The GRU model had an average score of 0.4625, the LSTM model had one of 0.4565, and the transformer had a score of 0.6936. We were initially a little concerned at how low the scores for the GRU and LSTM were considering that our qualitative observations indicated that the models were captioning correctly; however, after looking into the limitations of the BLEU score we found that oftentimes even if the predicted caption is correct from the human eye, the score will be low because the score is very much a literal comparison (Breitenbach). From this metric, we observed that the transformer model performed the best in producing exact matches to the reference captions provided by the dataset. We also plotted the BLEU score of each model:

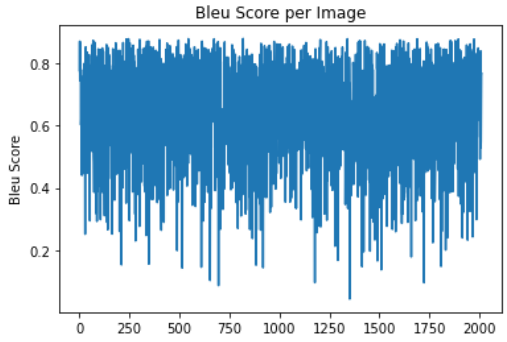
GRU



LSTM

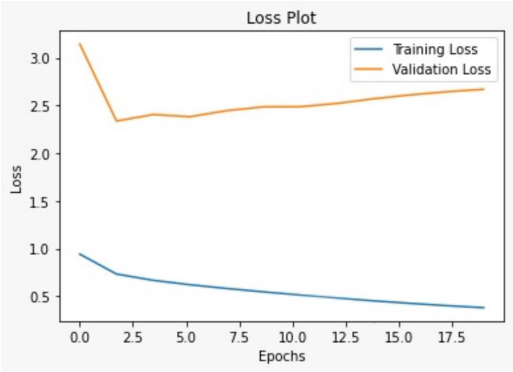


Transformer

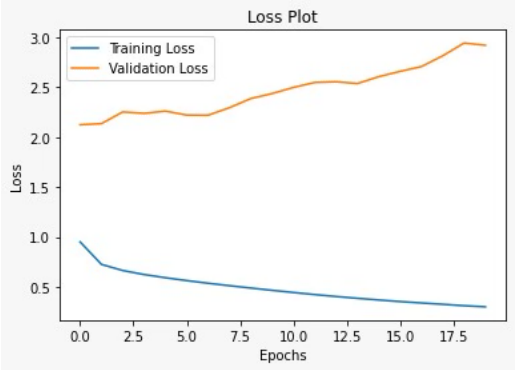


Finally, as a third metric we utilized the validation loss of the test set to determine the accuracy of each model. This metric determines how well a model fits the novel test data. Here are the loss curves of each model:

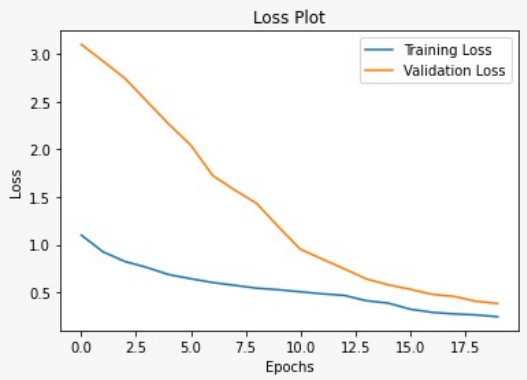
GRU



LSTM



Transformer



As shown by the graph, the GRU and LSTM models both seem to be overfitting because their validation loss is much higher than the training loss. However, if we were to compare the different models, the GRU model had a steep decrease in its validation loss initially, whereas the LSTM model rose steadily from the first epoch. The GRU model also seems to perform better than the LSTM and the Transformer model outperforms both models.

Discussion/Conclusion

What we Learned

In completing this project and working with different architectures to accomplish the same task, we observed several (phenomena) that mostly aligned with our predictions going in. Based on the research we conducted on the different models, we had predicted a GRU-based decoder would produce better captions than an LSTM based decoder and that a transformer-based model would outperform them both. Given that our results seem to validate this assumption, it seems what we can take away from this project is how the structures of different models perform better in image captioning tasks and other similar NLP problems. In addition to the increased computational efficiency of a GRU, it also has the advantage of the update and reset gates which allow the model to retain older information and pass down more relevant information to produce better predictions (Kostadinov). Moreover, we now understand that transformers tend to perform better because their design allows parallel training. This means they skip recursion and process captions as a

whole, allowing them to "learn" relationships between words and capture dependencies (Vaswani, et al). Transformers are attractive options for such semantic tasks and seem to be replacing RNN models in NLP (Wolf, Thomas, et al.) due to their multi-headed self-attention mechanisms and positional embeddings which provide more information about the relationship between words and prevent long-term dependency issues.

Limitations and Future Improvements

The validation loss in comparison to the training loss clearly indicates an overfitting problem. We additionally observed that the models did not perform as well when we imported our own images from sources outside of the validation set. This indicates poor generalizability. In the future, we would avoid this issue by using a larger dataset. This issue could also be resolved by getting training data from varied sources rather than a single dataset. Currently, due to limited GPU power, we faced challenges with increasing the size of our dataset and the number of epochs during training. With higher processing power, we would be able to combat this issue and potentially achieve better results. We could also improve our GRU and LSTM models by utilizing dropout regularization which is an effective way to reduce overfitting. We did implement this in our transformer model which resolved the issue, however we were unable to implement this with the GRU and LSTM due to time constraints.

Our BLEU score metric is also not very accurate. We used one caption for each image as a reference, however, we could potentially increase the accuracy of our BLEU scores by using multiple captions as references. We did implement a solution in our code, but we were unable to demonstrate the results due to the GPU constraints.

It would also be interesting to compare different pre-trained CNN models in our encoder to see how this affects our results. We would compare the current InceptionV3 to a ResNet or VGGnet (as used in the Stanford paper we were referencing).

Works Cited

Karpathy, Andrej, and Li Fei-Fei. “Deep Visual-Semantic Alignments for Generating Image Descriptions.” *Cs.stanford.edu*, https://cs.stanford.edu/people/karpathy/cvpr2015.pdf.

“Automatic Image Captioning Using Deep Learning.” *Analytics Vidhya*, 26 Aug. 2021, https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/.

Breitenbach, Sina. “MT for Beginners: What Is Bleu and What Is Wrong with It?” *Lengoo Blog*, Lengoo Blog, 20 July 2021, https://www.lengoo.com/blog/mt-for-beginners-what-is-bleu-and-what-is-wrong-with-it/.

Brownlee, Jason. “A Gentle Introduction to Calculating the BLEU Score for Text in Python.” *Machine Learning Mastery*, 18 Dec. 2019, https://machinelearningmastery.com/calculate-bleu-score-for-text-python/.

Cristina, Stefania. “The Bahdanau Attention Mechanism.” *Machine Learning Mastery*, 16 Oct. 2021, https://machinelearningmastery.com/the-bahdanau-attention-mechanism/.

Gautam, Tanishq. “A Guide to Use Transformers Using Tensorflow for Caption Generation.” *Analytics Vidhya*, 20 Jan. 2021, https://www.analyticsvidhya.com/blog/2021/01/implementation-of-attention-mechanism-for-caption-generation-on-transformers-using-tensorflow/.

“Image Captioning with Visual Attention ： Tensorflow Core.” *TensorFlow*, https://www.tensorflow.org/tutorials/text/image_captioning.

Kostadinov, Simeon. “Understanding GRU Networks.” *Medium*, Towards Data Science, 10 Nov. 2019, https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

Saxena, Shipra. “LSTM: Introduction to LSTM: Long Short Term Memor.” *Analytics Vidhya*, 18 Mar. 2021, https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/.

Vaswani, Ashish, et al. “Attention Is All You Need.” *Arxiv.org*, Dec. 2017, https://arxiv.org/pdf/1706.03762.pdf.

Wolf, Thomas, et al. “Transformers: State-of-the-Art Natural Language Processing.” *ACL Anthology*, https://doi.org/10.18653%2Fv1%2F2020.emnlp-demos.6.