Experimentation with Various Encoder-Decoder CNN-RNN Architectures for Image Captioning

Lucas Nguyen, Paul Pan
Department of Computer Science and Engineering
University of California - San Diego
La Jolla, CA 92093
`lnguyen.professional@gmail.com, jpan@ucsd.edu`

Abstract

A common adage in training neural networks says "Don't be a hero", implying it is easier to build neural networks from previous work than to train a model on ones own. Here we experiment with both approaches to see which one achieves better top-1 accuracy on our 20-category classification task. First we train a baseline model and tune the model by modifying architecture and hyperparameters, starting with 25% accuracy and improving to 45%. Next we train two state-of-the-art models, VGG16 and Resnet18, freezing all but the last layer with accuracies of 74.06% and 77.89%.

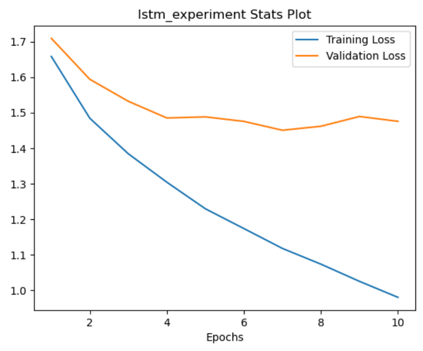
1 Introduction

In this study we perform image captioning with an encoder-decoder structure, where the encoder is a CNN and dense layer for images and words in a caption and the decoder is an RNN. We test several different decoder variations, including different RNN cells, different input architectures, and different hyperparameters. After this we caption images using a random generative and a deterministic method. RNNs use state to represent time; in our model, we use an LSTM cell inside the RNN. The LSTM uses various gates to memorize inputs, allowing it to memorize long term relationships in the sequence data.

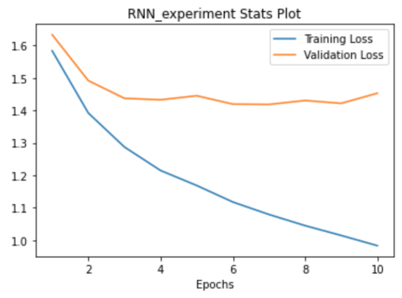
2 Related Work

3 Models

We used an encoder-decoder architecture with two different decoders for our classification task, a Baseline LSTM and vanilla RNN. Since our task is captioning, we encode the image to the embedding size using a pre-trained Resnet-50 model with the last linear layer with embedding size output units and learnable. For captions we one-hot each word then pass it through another linear layer with embedding size output units. For training the decoders take the encoded image as the first image then the sequence of encoded words as inputs for their next respective time steps in a process called teacher forcing. The decoders are both recurrent neural networks, where time is represented by state of the unit. However the LSTM differs from the RNN because it has gates that allow it to memorize certain data for future inputs. This allows the LSTM to model long term dependencies better than the RNN. Both decoders have a fully connected linear layer mapping the decoder to the vocabulary size. The model uses a linear layer on top of one-hot encoded vectors to embed words. We sample from the decoder by using the encoder embeddings as the first time step input, then we use the output of each of the time step and embed it to pass into the next time step to generate the next time embedding.



LSTM Loss Curves



RNN Loss Curves

Looking at the loss curves, compared to the LSTM, the RNN seems to perform about the same. This could be because our task does not require long term dependencies since captions are not very long, and once we have an encoded image representation the model can determine a subject and action. In other words the model doesn't need to decode the semantics of words to find their meaning, it just needs to group words together that somewhat describe features in the image and syntactically make sense. The RNN and LSTM both trained in the same amount of time, it seems like they did most of their training in the first epoch since loss started out below 2 which is notable. After this both experienced overfitting as validation straightened and training decreased. We think we might have changed learning rate too high from default on accident without realizing it however we did not intentionally change it.

The LSTM model achieved 1.49 vs the RNN which achieved 1.437 test loss. The RNN did about the same compared to the LSTM, which is likely because the captioning task doesn't require the model to memorize long term temporal relationships.

4 Caption Generation

With both our decoder models we generated captions using two methods. Noting that the decoder model outputs a vector of probabilities for each word in vocabulary depending on input, we tried a deterministic approach where we chose the word with greatest probability, and a stochastic approach where we sampled from the probability distribution to get the next word.

4.1 Deterministic Approach

Model	BLEU-1	BLEU-4
LSTM	43.5	2.78
RNN	55.5	6.53

Table 1: LSTM and RNN Deterministic Caption Generation Scores

First we implemented the deterministic caption generation. Surprisingly, the RNN was able to achieve the best BLEU scores. We're not sure why this is since they trained similarly, although it is worth noting that the RNN's encoder had all but the last layers weights frozen while the LSTM's encoders weights were all learnable.

4.2 Stochastic Approach

For stochastic caption generation, we altered the probability distribution with a parameter called temperature, τ which determines how deterministic the distribution is. When temperature is high, the distribution is more uniform (all words have a more equal chance) while when temperature is low it resembles deterministic sampling.

$$y_i = \frac{e^{\sigma_i/\tau}}{\sum_n e^{\sigma_n/\tau}}$$

Equation 1: Softmax Equation modified with Temperature

Score	0.2	0.7	1	1.5	2
BLEU-1	43.5	42	45	31.35	20
BLEU-4	2.8	2.5	3.9	1.47	1.12

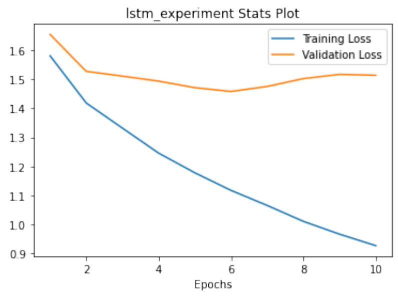
Table 2: LSTM Stochastic Caption Generation BLEU Scores

With temperature 1 stochastic caption generation performed better than deterministic generation. This is likely because with temperature 1 generation resembles a normal softmax which the model is trained and optimized on as compared to other temperatures.

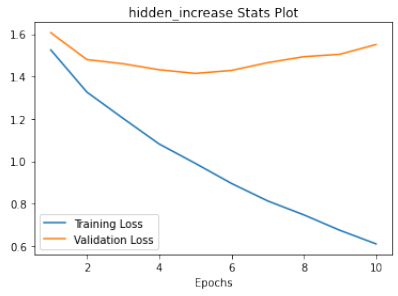
5 Model Tuning

5.1 Baseline vs Tuned LSTM

We fine-tuned the LSTM model by changing the embedding and hidden sizes to 2 times the baseline in separate experiments: (300 to 600 and 512 to 1024 respectively). With these hyperparameters, the model with increased embedding returned BLEU-1 and BLEU-4 scores of 31.5 and 1.45 respectively, while increased hidden size returned scores of 37.5 and 1.99. While increasing hidden size seems like a better choice, both results performed poorer than baseline. When looking at the final losses we noticed the training loss dropped below 1 for both experiments but test and validation loss were the same as baseline, implying we were giving the model more parameters to overfit which would not improve performance given our model was already not generalizing well.



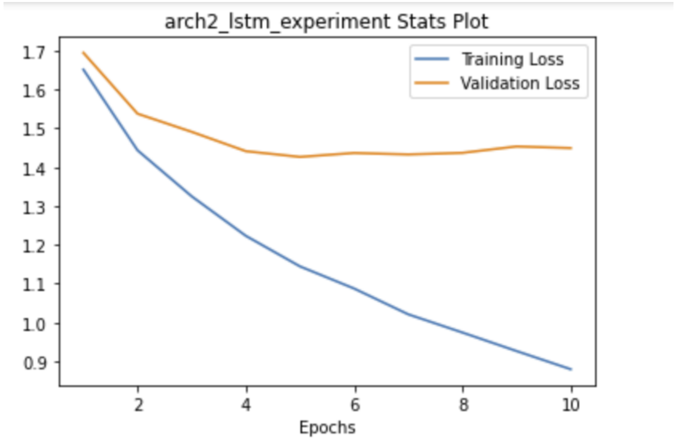
Doubled Embedding Size LSTM Loss Curves



Doubled Hidden Size LSTM Loss Curves

5.2 Baseline vs Alternative LSTM

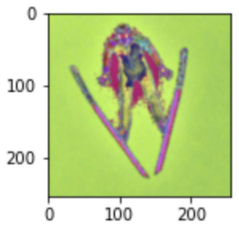
We also implement an alternative decoder architecture using an LSTM cell but inputting a concatenated vector containing the caption and image embedding each time step. As a result our input is now $(sequencelength, embeddingsize*2)$ instead of $(sequencelength+1, embeddingsize)$. The model achieves BLEU-1 and BLEU-4 scores of 14.63 and 0.68 respectively, which is bad compared to the original architecture. However since the loss seems to be the same as baseline. This is likely because the caption generation tended to be more difficult with the image features passed in on each forward pass due to the fact that there are extra unnecessary information in the input other than the output from the previous layer.



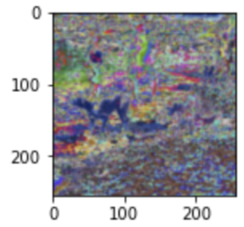
Alternative LSTM Loss Curves

6 Best Model Results

To demonstrate our results, we took our best model, the temperature 1 LSTM with default embedding and hidden size, and generated captions for a subset of the images, printing the generated captions, original captions and original image. We show 10 effective results and 10 poorly predicted results.

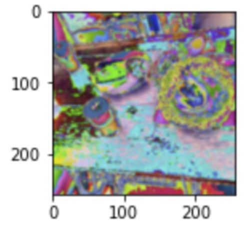


Original: a man performing a trick in the air with a pair of skis .
Generated: a black ski dusted storm clouds refueling exposure turnip tidal stunt inverted choir a black white sittting canadian

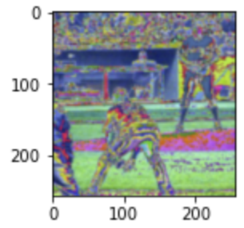


Original: people standing while there dogs play in the water and on the riverbank .
Generated: a man springtime ext watermelon raisins bison shoulders hooves dirt lot of propellor ruins cans coastline alongside other

Good Captioned Images

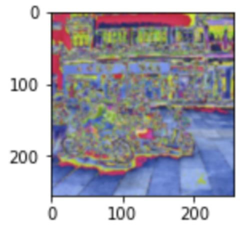


Original: a table with cups , a plate with cheese and tomato and a platter of bread .
Generated: a table set needed eggs cream deserts placed fierce coffee table dress-up dinner factory breakfast nook generating t



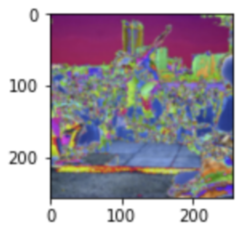
Original: a pitcher stands on the mound while the umpire , players , and crowd look on .
Generated: a stadium released vac catcher mitt eco-friendly baseball backside rings sheeps coastline teammates watching tv stuffs r

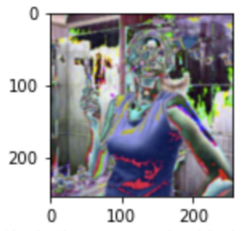
Good Captioned Images



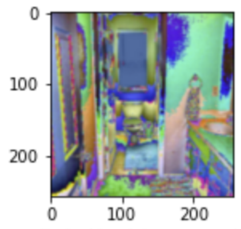
Original: a group of people in next to motor scooters , in front of a store with asian character writing on the awnings .
Generated: a busy intersection by some people crevasse chargers dressings os motorcycles spilled motorcycles events driz

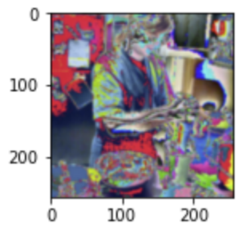
Good Captioned Images


Original: a young man jumps his skateboard over concrete obstacles in front of a crowd .
Generated: a man ma pasted whip kicking swiss herself raspberry appealing shared people file soldiers getting married

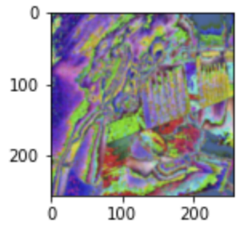

Original: a woman is in the bathroom , wearing black , and holding a toothbrush .
Generated: a man kneels hull controller somebody applauding partial backpacker waxing a man specially efficient container fr

Badly Captioned Images


Original: view of a bathroom vanity looking into a separate room with a toilet and window .
Generated: a sink retro baker toilet sheer overlooked faucets rummaging partner angeles nursery disrepair matches ray target rubik


Original: a woman near a messy kitchen counter holds a hand blender into a green drink .
Generated: a man 's shell combat wedding ball arrows accordion earbuds oblong ornate double typical vision lucky taco gri

Badly Captioned Images


Original: a woman has her elbows on the table and is looking at something on a laptop .
Generated: a car interrupted chunky lounging basil lorikeet headgear shriveled chalet cushions furred span limit sea kitten petted

Badly Captioned Images

7 Contributions

Lucas: I wrote the original architecture 1 and 2 before paul fixed it, hyperparameter tuned, implemented RNN

Paul: I implemented the architecture 2 and caption generation. Also, I implemented temperature of the generation and fixed the bugs presented in architecture 1, added caption padding, and visualization of the image.

We both helped each other debug.