COCO Encoder and Decoder Image Captioning

Yifei Wang
Department of Data Science
University of California San Diego
yiw014@ucsd.edu

Rui Zhang
Department of Data Science
University of California San Diego
r2zhang@ucsd.edu

Abstract

In this programming assignment, we studied the problem of image captioning, an approach that allows computers to generate descriptive text when it "sees" an image. Such techniques empowered by the recurrent network offer us a robust way of analyzing image data. With the use of LSTM(Long short term memory), our model achieved a decent quality for the generated caption within a relatively short period of training time. Our analysis and the results of our experiments are presented in this paper. We found that the best text generation quality was achieved when we downsized our embedding dimension and hidden unit size; This allows the model to achieve 68.34 BLEU-1 score and 8.526 BLEU-2 score. We also noticed that the network is prone to overfitting and underfitting if we do not choose the network's size carefully. In addition to studying how the architecture of the neural network affects the performance of the model, we also experimented with and analyzed the effect that changing temperature will have on the model's performance. We found that by setting the temperature to 0.1 the model could have the best text generation quality. Setting the temperature too high or too low yields poor text generation quality.

1 Introduction

With the advent of deep neural networks, the field of computer vision has been revolutionized. Image captioning is one of the fundamental areas of computer vision, which tries to provide model-generated descriptions for images automatically. It not only required the model to recognize salient objects in an image, understand their interactions but also to caption the images using natural language, which makes itself very challenging [3].

In this paper, we will use Microsoft COCO (Common Objects in Context) as the image captioning dataset [4]. In the COCO dataset, per-instance segmentations are used to label the objects, in order to facilitate precise object localization. The dataset contains photos of 91 objects types that would be easily recognizable by a 4-year-old child. With 2.5 million labeled instances in 328k images, the COCO dataset was accomplished by using extensive crowd worker involvement via novel user interfaces for category detection, instance spotting, and instance segmentation [4]. Many interesting computer vision and image captioning work has been done using COCO. For instance, Sean Bell1 and his colleagues utilized Inside-Outside Net (ION), an object detector that exploits information both inside and outside the region of interest on COCO to use skip pooling to extract information at multiple scales and levels of abstraction and concluded that their proposed architecture is particularly effective at improving detection of small objects [1]. For our purpose, we will use one-fifth of the COCO dataset, where the training set contains around 82k images with roughly 410k captions while the test set has around 3k images with almost 15k captions.

In order to generate captions for the COCO images, we used an encoder-decoder architecture in this paper: the encoder took the image as input and encoded it into a vector of feature values. This

was then passed through a linear layer for providing the input to the LSTM. A pre-trained convolutional network ResNet 50 with transfer learning was used as the encoder and an LSTM model was used as the decoder. We would also explore how Recurrent Neural Networks (RNN) could process data that has a temporal structure by experimenting with using the RNN cell and comparing it to the LSTM cell as used by the decoder. We also incorporated a third encoder-decoder model which combines the feature vector from our pretrained encoder and the word embeddings; and then passes this concatenate feature to the LSTM model at each time step.

2 Related Work

Image captioning has been a heated topic in the deep learning area. Existing approaches have two major issues: it is difficult to determine which elements of the captions are more important to the image during the training phase (encoder), and the objects or sceneries are occasionally misidentified during the caption generation phase (decoder).

In "Reference based LSTM for image captioning", the authors adopt R-LSTM to address the misrecognition issue in the caption generation phrase [2]. In their approach, the words in a caption are given different weights in the training phase based on their relevance to the corresponding image. When calculating the loss, a term with a greater relevance score indicates that it is more important to describe the image and is given a higher weight value. As a result, the model may learn more about the caption, such as what the main items are, which qualities are significant to them, and how they interact with one another. During the generation phase, they use the consensus score to combine the nearest neighbors of the input picture as references [2]. Their proposed R-LSTM model could alleviate the misrecognition by incorporating the nearest neighbor into the generation phrase, and their produced caption could to some extent imitate the habit of human cognition [2]. Their generation with reference and weighted training innovations could shed light on how to improve our proposed CNN and LSTM combined model further.

Lun Huang and his colleagues present an Attention on Attention approach (AoA) for the encoder-decoder attention-based image captioning, and they extend the conventional attention mechanisms to determine the relevance between attention results and queries and address the irrelevant attention issue [3]. They apply AoA to both the image encoder and the caption decoder of our image captioning model, and provide a comprehensive analysis of this AoANet [3]. Their paper helps us better understand the encoder-decoder structure and its possible flaws and solutions.

3 Methods

3.1 Baseline LSTM Model Description

Our baseline model consists of two parts: one is the encoder part, which is described in detail in section 3.3, and the other is the decoder part. The Encoder part transforms an input 3D tensor to a lower-dimensional embedding whereas the decoder part takes in the embedding and generates the descriptive text. In this case, we used a pretrained CNN model to encode the image, and use LSTM RNN to decode the embedding. Since LSTM RNN is better at capturing the long-term dependencies and relations in sequence data, we can expect that the image captioning model that is built based on LSTM will outperform its Vanilla RNN counterpart, provided that all the hyperparameters remain the same during the comparison.

3.2 Vanilla RNN Model Description

The vanilla RNN captioning model still has the same basic structure as that of the LSTM captioning model. But it is using RNN cells as its decoder, which means that this model is less capable of capturing long-term relations than the LSTM model is. In fact, we did find a significant performance boost when we switched to LSTM from the simplest vanilla RNN model. The results will be presented in the next subsection. We were able to further improve the performance when we concatenated the picture embedding with the word embedding and fed the elongated vector to our decoder. This is discussed in the next section.

3.3 The Model with the Image Encoding Given at Each Timestep

In this model, the model architecture is still the same as the one with LSTM as its decoder. However, the feature encoded at each time step changes: Instead of only passing in the image feature at time step one, the input embedding for every step now becomes the concatenation of the feature vector and the word embedding, which emperors the model as our input feature become more informative than it is previously.

We saw a performance boost when we apply this model to the image captioning problem, which will be discussed later in section 4.

3.3 Discussion on the encoder part

The encoder plays an important role in our image captioning model: It tells our RNN model what to generate by feeding the information that it extracts from the picture. The quality of the input information can significantly affect the quality of the decoder's output. Ideally, the essential information displayed on the picture should be preserved when the picture is converted to a vector, and the use of CNN as a feature extractor benefits the training greatly as it produces good feature embedding.
The architecture of convolutional neural networks is called RESNET50, which is a kind of CNN that applies residual connection in its architecture. Because of the use of residual blocks, the encoder could be very deep and powerful; The network can go very deep without too much suffering from vanishing gradient thanks to the "skipped" connection. Encoder preserves the crucial information that would otherwise be lost had we just directly converted the picture to vector.
Since we are under tremendous resources constraint, the encoder is pretrained before we use it. If time and resources permit, one can definitely implement his/her own feature extractor in whatever way that he/she likes. In our case, however, the pretrained feature extractor is already good enough. There is only one hidden layer that has trainable parameters, which the model will use to produce embedding that can feed into the decoder part.

3.4 How to Sample Output from Decoder

The sampling process is crucial to our image captioning task; When our model finishes training, we want the model to generate a caption when some input is fed into the encoder. Such generation tasks could be random or deterministic.

The model can generate the results deterministically: after the input feature is put into the decoder, our RNN model will generate a softmax distribution over all possible vocabularies that have been involved during the training. We will choose the word that has the largest softmax probability at each time step, and use its embedding as the input to the next time step. As one can clearly see, such an approach is totally deterministic; That is, if we try to caption the picture a thousand times, the caption will always be the same and never change.

On the other hand, we can let the model generate text in a relatively "random" way via sampling over the given softmax distribution. The randomness during the text generation stage is controlled by the term "temperature"; The higher the temperature is, the less deterministic our result will be. Additionally, choosing the right temperature is very important as it will influence the quality of the outcome, which we will discuss more in the following sections of the experiments.

3.5 How our model obtains word embedding

Technically speaking, the model gains the word embedding through the help of the PyTorch library, which outputs a representation of the word being encoded and reduces its dimensionality

to a size that is acceptable to our model (we can regard the embedding size as a hyperparameter of our model, and we should carefully tune that).

3.6 How do we tune the parameters, part I: How to understand the hyperparameters

Here are the parameters that we tuned during our experiments: Firstly we can tune the number of epochs for the training stage. This parameter is very critical to the success of our model: if the number of epochs is too much, then the model will probably suffer from overfitting. Otherwise, our model will underfit the data since the model doesn't have enough training time to fully adjust its weights. Secondly, we also tuned the learning rate, which, obviously, affects the optimization process. Thirdly, we tune the model's size: Here the model size means the embedding size and hidden units sizes. Lastly, model architecture is also critical to the success of the task. In this programming assignment, we tried three kinds of models structured as mentioned above: Vanilla RNN, LSTM RNN, LSTM RNN with different input feature embedding. Readers will see more of its performance in the following experiment section.

3.7 How do we tune the parameters, part II: range for tuning the hyperparameters

We use 4 workers in parallel, and with use of RTX2070Ti, the typical training time for 10 epochs is less than 1 hour(55 minutes on average), which allows us to do lots of experiments.

Table 1: Hyperparameters Search Range

Hyperparameters	Range
number of epochs	10—30
learning_rate	1e-5—-1e-2
hidden_size	200—-700
embedding_size	150—-450
temperature	Deterministic, 0.01-0.8

The best hyperparameters for each of the models can be seen in the next section.

4 Results

4.1 Best hyperparameters for each model

First of all, the best set of hyperparameters for the **architecture 2 (LSTM + improved feature embedding)** is the following:

Table 2: Best Architecture 2 Hyperparameters

Hyperparameters	Values
number of epochs	10
learning_rate	5e-4
hidden_size	300

embedding_size	150
temperature	0.1

The test performance is as follows:

```
Test Performance: Loss: 1.8078368949813965, Bleu1: 68.3476386867569, Bleu4: 8.526229283991666
```

The best hyperparameters for **Vanilla RNN**:

Table 3: Best Vanilla RNN Hyperparameters

Hyperparameters	Values
number of epochs	10
learning_rate	5e-4
hidden_size	512
embedding_size	300
temperature	0.1

The test performance is as follows:

```
Test Performance: Loss: 1.8073932513896143, Bleu1: 67.05130250488027, Bleu4: 8.014653469791678
```

The best hyperparameters for **LSTM**:

Table 4: Best LSTM Hyperparameters

Hyperparameters	Values
number of epochs	10
learning_rate	5e-4
hidden_size	300
embedding_size	300
temperature	0.1

The test performance is as follows:

```
Test Performance: Loss: 1.8073932513896143, Bleu1: 67.05130250488027, Bleu4: 8.014653469791678
```

4.2 Model Performance Plot

(a) The LSTM Model

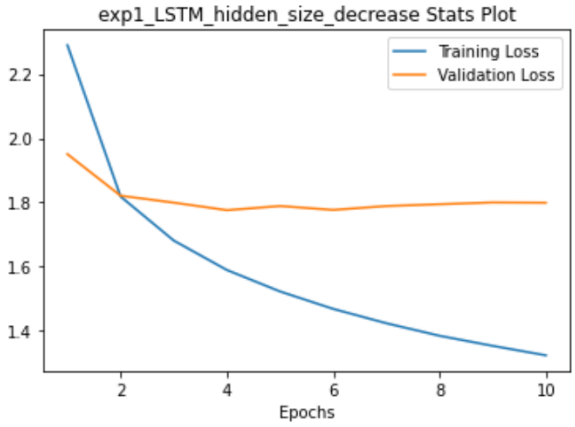

Figure 1: Training Loss and Validation Loss vs Number of Epochs for LSTM

Final loss achieved:

```
Epoch: 10, Train Loss: 1.3224594601491768, Val Loss: 1.7986111590300748,
```

(b) The Vanilla RNN Model

The cross-entropy loss on the test set for this vanilla RNN model is 1.91, and we show the training loss and the validation loss vs the number of epochs for the model in the below figure 2:

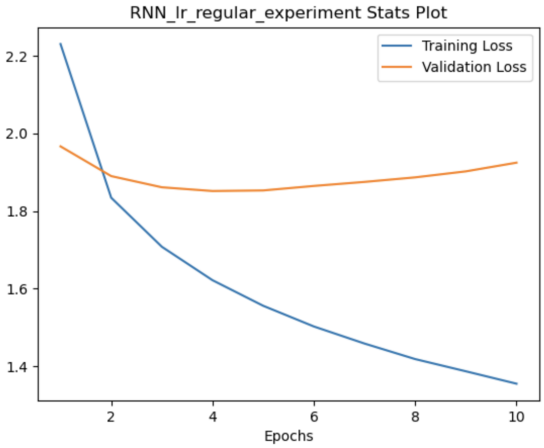

Figure 2: Training Loss and Validation Loss vs Number of Epochs for Vanilla RNN

Final loss achieved:

```
Epoch: 10, Train Loss: 1.3577651676381446, Val Loss: 1.9142488195168925,
```

(c) The Model with the Image Encoding Given at Each Timestep

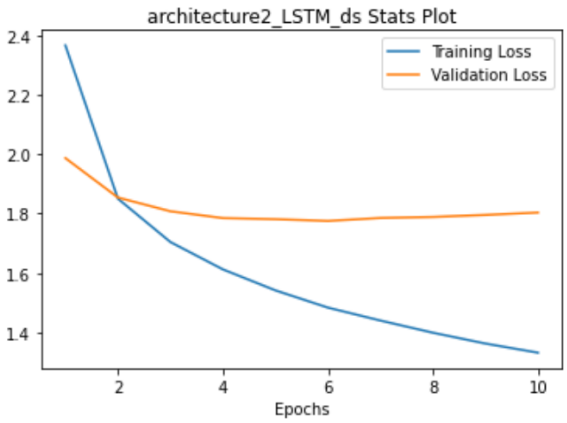

Figure 3: Training Loss and Validation Loss vs Number of Epochs for Vanilla RNN

Final loss achieved:

```
Epoch: 10, Train Loss: 1.3321367729679918, Val Loss: 1.8035327794469
```

4.3 Test loss, BLEU-1, and BLEU-4 scores

For our best LSTM baseline model,

```
Test Performance: Loss: 1.8073932513896143, Bleu1: 67.05130250488027, Bleu4: 8.014653469791678
```

For our best Vanilla RNN model, the BLEU-1 score is 63.26, and the BLEU-4 score is 6.40:

```
Test Performance: Loss: 1.8073932513896143, Bleu1: 67.05130250488027, Bleu4: 8.014653469791678
```

For our best model with the image encoding given at each timestep

```
Test Performance: Loss: 1.8078368949813965, Bleu1: 68.3476386867569, Bleu4: 8.526229283991666
```

4.4 Good and Bad Caption Images Analysis

In this section, we present the outcomes of our image captioning experiments. We used our best model of each architecture to generate captions. As expected, each model will sometimes produce good captions and bad captions, and both of which will be shown. In addition to that, the effects of changing temperature imposed on the quality of the generated text are analyzed. With the same picture, we experiment with extremely low temperature, reasonable temperature and high temperature. Readers could see the effects quite intuitively,

(a) The Baseline LSTM Model

One should notice that even though the caption generated is a good representation of what actually happens in the image, it can still be slightly different from the actual caption, which,occasionally, captures the very minor detail of the picture. Nevetheless, despite the noticeable differences between the predicted caption and actual caption, most caption generated by our model make sense to us and is usable for further analysis.

It is also not uncommon that our image captioning model only generates partially correct text information. It seems that the model only understands part of the image. How to improve the model is the main theme that we will discuss in the later section.

Figure 4: Caption example 1 for Baseline

Table 5: Generated Captions Comparisons For Figure 4

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.001	a large elephant standing on a sandy beach. (good)	1. an elephant playing with water at a watering hole with his trunk 2. An elephant's front legs are in the water and back legs are out of the water . 3. an elephant is going to the river to drink water 4. an elephant standing with its front feet in the water . 5. an elephant stands with its front feet in shallow water .
Deterministic = True	a group of elephants standing next to each other . (bad)	
Temperature = 0.4	a group of elephants walking a watering hole . (good)	
Temperature = 5	addressing arched palm kinds blooms exit rearing stacks straighten respective on crooks peeled whole pillows indoors topped male center course. (disaster)	

Figure 5: Caption example 2 for Baseline

Table 6: Generated Captions Comparisons For Figure 5

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.001	a herd of sheep grazing in a field . (very good)	1. three look alike dogs are herding sheep into a pile . 2. small group of sheep being herded by three border collies . 3. These sheep are being watched by three dogs . 4.a group of sheep in a field with three dogs close by . 5. a group of sheep surrounded by three dogs
Deterministic = True	a herd of sheep grazing in a field . (very good)	
Temperature = 0.4	a herd of sheep standing on top of a lush green field . (very good)	
Temperature = 5	gypsy performance coin-operated cupboard off shrimp hundred pedestal out engine airliners surrounding pepper philly lovingly tavern hardwood ply portions provided (disaster)	

Figure 6: Caption example 3 for Baseline

Table 7: Generated Captions Comparisons For Figure 6

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.001	a laptop computer sitting on a desk next to a laptop . (very good)	1. an open laptop computer with a cat laying on it . 2. a cat lays its head on the laptop keyboard .
Deterministic =	a laptop computer sitting on a desk	

True	next to a laptop . (very good)	3. a cat rests its head on the keyboard of a laptop computer . 4.a computer that is sleeping on a computer . 5. a black cat sleeping with its head on a laptop .
Temperature = 0.4	a laptop computer sitting on top of a laptop . (bad)	
Temperature = 5	danishes springs groomsmen wings bed approaches next , diagonal around broccoli rockwell could cross peers castle designer praising smoking (disaster)	

Figure 7: Caption example 4 for Baseline

Table 8: Generated Captions Comparisons For Figure 7

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.001	a woman is holding a tennis racket in her hand . (very good)	1. a woman standing on a tennis court holding a racquet . 2. a picture of a female tennis player . 3. a woman in a tennis outfit holds a racket . 4.a female tennis player holding a tennis racket . 5. She is well prepared to participate in the tennis
Deterministic = True	a woman is holding a tennis racket in her hand . (very good)	
Temperature = 0.4	a woman holding a tennis racket in the air . (BAD!)	
Temperature = 5	zoo elderly stretching female laundry facing churning items	

	christmas instructional motels tarp sunlight airway break studies bathtubs through repeal hanging (disaster)	match .

Figure 8: Caption example 5 for Baseline

Table 9: Generated Captions Comparisons For Figure 8

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.001	a herd of sheep grazing in a field . (good)	1. a herd of sheep grazing on a hillside near a tree . 2. several sheep standing in the grass near a tree . 3. a couple of sheep graze on some grass 4. a small herd of sheep grazing in a grassy field . 5. While in the immediate foreground just a gnarled tree branch , the majority of the view consists of an expanse of short grass dotted with a few longer tufts and a number of scattered , grazing sheep .
Deterministic = True	a herd of sheep grazing in a field . (good)	
Temperature = 0.4	a sheep grazing in a grassy field near a bush . (Better, more detail!)	
Temperature = 5	frisbee small chasing info countryside tag themed appearance resolution bought greenwich world earth heir greenery umpire watches language characters power (Nonsense as expected)	

Figure 9: Caption example 6 for Baseline

Table 10: Generated Captions Comparisons For Figure 9

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.001	a yellow and blue train on a track . (Bad; Wrong color description)	1. a yellow and red trains engine on its track and trees and signs . 2. a red locomotive stopping at a railway station . 3. The train engine is different from the ones we have in the USA . 4. a train is stopped next to a brick walkway at a train station . 5. a red and yellow train at a platform
Deterministic = True	a train that is on the tracks near a building . (Bad)	
Temperature = 0.4	a train is pulling into a station . (Not good)	
Temperature = 5	railroad something state us rather ooze they opposing manatee above fearless crt horn cartoon-like happily entering cameraman pompom sheds roses (disaster)	

(b) The Vanilla RNN Model

We could see from our experiments that the Vanilla RNN Model performs poorly on photos that lack a main object, photos with multiple main characters, and photos with multiple confusing objects as the background given the characters constraints.

The Vanilla RNN Model performs well on photos with clear and unconfusing background, photos with unconfusing main characters and photos with true colors.

When the temperature is close to 0 (0.001 in our experiment), the caption generated by the model is almost the same as the deterministic approach, meaning that the model would almost always generate the token with the maximal probability in the softmax distribution.

When the temperature is as large as 5, the caption generated by the model does not make sense, because all tokens in the dictionary were given almost the same weight, so the model would randomly pick tokens and produce sentences. A high temperature (above 1) makes the model less confident.

When the temperature is appropriate (between 0 and 1 and not close to 0 or 1), the captions generated by the model are not too deterministic and could match most of the actual captions.

We show 3 images of bad captioning and 3 images of good captioning under our different proposed temperatures and whether the model is deterministic.

Bad captioning:

Figure 10: Bad Caption example 1 for RNN

Table 11: Generated Captions Comparisons For Figure 10

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.4	a car driving down a street next to a street.	1. black and white photo of cars parked on the street. 2. large town of parked cars on lot and on street. 3. an old, black and white street scene is pictured. 4. a bunch of old cars that are parked in a lot and on the street. 5. an old photo of a street view with many cars parked.
Deterministic = True	a car driving down a street next to a street.	
Temperature = 5	corn cleaned luggage front rabbit printing strolls slow pendant mussels hover face blob parade ; pitchers collapse oval designating	
Temperature = 0.001	a car driving down a street next to a street.	

Figure 11: Bad Caption example 2 for RNN

Table 12: Generated Captions Comparisons For Figure 11

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.4	a flower arrangement in a vase on a table.	1. a wall with different types of decorations of art pieces. 2. a mosaic of vases hung on a wall. 3. assorted vases hanging on a wall next to each other. 4. glass is hanging on the white wall all bunched together. 5. a white wall displaying art that look like vases.
Deterministic = True	a table with a vase of red apples.	
Temperature = 5	entrance drawing crammed giraffe campus dimly futon tent coupled happen participants equipment ahs spotlessly assume yellow granola removing part compartment	
Temperature = 0.001	a table with a vase of red apples.	

Figure 12: Bad Caption example 3 for RNN

Table 13: Generated Captions Comparisons For Figure 12

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.4	a group of people standing around a patio with a wooden bench.	1. a couple of people are sitting on a boat. 2. two men sit near a sofa on a boat. 3. fancy passenger seating on an otherwise ordinary boat. 4. a couple of men that are sitting down. 5. people sitting near boats and a leather sofa.
Deterministic = True	a man and woman sitting on a bench with a dog in the background.	
Temperature = 5	floored these reptile daytime ton brochures bubba bd car selection gathered router filled larch meatball aluminum plump pack public miniature	
Temperature = 0.001	a man and woman sitting on a bench with a dog in the background.	

Good Captioning:

Figure 13: Good Caption example 1 for RNN

Table 14: Generated Captions Comparisons For Figure 13

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.4	a man is standing on a snowboard in a snow covered slope.	1. a young man riding skis down the side of a snow covered slope. 2. somebody is gotten in the peaceful of the picture. 3. a man that is standing on ski's in the snow. 4. person riding down snowy
Deterministic = True	a man riding a snowboard down a snow covered slope.	
Temperature = 5	sealed speared octopus grip tanks guided tours cloths . wades outfielder brindled seperated leafless awning moustache photo warped assemble	

	service	hill on a pair of skis. 5. a skier is skis down a snowy hill on a sunny day.
Temperature = 0.001	a man riding a snowboard down a snow covered slope.	

Figure 14: Good Caption example 2 for RNN

Table 15: Generated Captions Comparisons For Figure 14

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.4	a man is flying a kite on the beach.	1. in this scene we see a person flying a kite with a flag attached. 2. a man flying a kite at the beach. 3. colored kites sail in the sky above a sandy beach. 4. a person flying a colorful kite on a beach. 5. man on ocean beach flying several kites on windy day.
Deterministic = True	a man is flying a kite on the beach.	
Temperature = 5	tattered refilling kitty banana jumps wit eps ginger scene high rims blowup checkered reference pinned flower written simple chunks joe	
Temperature = 0.001	a man is flying a kite on the beach.	

Figure 15: Good Caption example 3 for RNN

Table 16: Generated Captions Comparisons For Figure 15

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.4	a group of men playing a game of soccer.	1. some kids are playing in a baseball game 2. some baseball players are playing baseball on a field. 3. a group of young men playing a game of baseball. 4. a group of people play a game of baseball together. 5. a number of baseball players on a field.
Deterministic = True	a group of men playing soccer on a field.	
Temperature = 5	tiger malnourished splashing chariots couple feet beards cheese footstool car towering intended soccer hideous readings dock graham upturned mounds birch	
Temperature = 0.001	a group of men playing soccer on a field.	

(c) The Architecture 2 LSTM Model

Notice that in this experiment alone we discover an interesting thing: 1) unlike previous experiments, which indicates that lower temperature performs better than higher temperature does(of course the temperature itself must be reasonable), this experiment shows that lower temperature makes the model perform worse than the model will in a slightly higher temperature.

We also set temperature=0.8 as high temperature; This is different from previous experiments where we set temperature=5. The reason is that one can certainly expect the output of the model when temperature is too high: The output would be a mess if we set high temperature. The readers can see how temperature affects the quality of text generation when we set temperature to be 0.8.

It is kind of surprising to see that 0.8 seems to be the best at generating coherent and informative English sentences. But It sometimes just gives the wrong caption, meaning it fails to detect the most critical information presented in the picture.

Please see the example captions generated below.

Figure 16: Caption example 1 for Architecture 2 LSTM

Table 17: Generated Captions Comparisons For Figure 16

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.1	a man talking on a cell phone while talking on a phone . (Bad; Wrong gender)	1. caucasian girl talking on a cell phone in public . 2. a woman at a restaurant tries to hear her cell phone . 3.the woman is sitting down holding her phone to her ear 4. a woman looks distressed as she talks on her phone . 5. a girl is trying to have a conversation on her cell phone in a noisy bar .
Deterministic = True	a woman talking on a cell phone while talking on her phone . (Bad; poorly written sentence)	
Temperature = 0.4	a woman talking on a cell phone while talking on her phone . (Not good; same as previous one)	
Temperature = 0.8	a pretty young lady talking on her cell phone . (Very good)	

Figure 17: Caption example 2 for Architecture 2 LSTM

Table 18: Generated Captions Comparisons For Figure 17

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.1	a person is flying a kite in the air . (Wrong)	1. skiier jumping with skis split in a v-shape over the olympic circles 2. a skier is high up in the air over a snowy hill . 3. a person on skis is in the air over the olympics ' sign on the snow below them . 4. an olympic games skier on a jump sails past the olympic symbol . 5. a skier does a big jump on a white snowy slope .
Deterministic = True	a person is flying a kite in the air . (Bad)	
Temperature = 0.4	a person on a board rides on the side of a wave (Completely wrong)	
Temperature = 0.8	a person on skies is up in the air . (Mediocre; Not too bad)	

Figure 18: Caption example 1 for Architecture 2 LSTM

Table 19: Generated Captions Comparisons For Figure 18

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.1	a person is standing on a skateboard on a skateboard (NOT BAD)	1. a man grinding his skateboard on a rail . 2.a man riding a skateboard on the side of a metal rail . 3. a man on a skateboard grinding on a pole
Deterministic = True	a person is standing on a skateboard on a skateboard . (NOT BAD)	

Temperature = 0.4	a wooden bench with a wooden bench and a small wooden table . (Completely wrong)	4. a young man doing an axle grind on a piece of pipe in a park . 5. a boy on a skateboard rail on a skateboard
Temperature = 0.8	a small wooden bench with some green and yellow and a skateboard . (bad; But it did detect the skateboard)	

Figure 19: Caption example 1 for Architecture 2 LSTM

Table 20: Generated Captions Comparisons For Figure 19

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.1	a train traveling down tracks next to a platform .	1. a long double decker passenger train going along a track 2. silver train crossing the road on tracks . 3. an amtrack california train running on the track 4. a long train traveling across a road on train tracks 5. the train is travelling down the tracks of the road .
Deterministic = True	a train traveling down tracks next to a platform .	
Temperature = 0.4	a train traveling down train tracks next to a platform .	
Temperature = 0.8	a passenger train is parked at a platform	

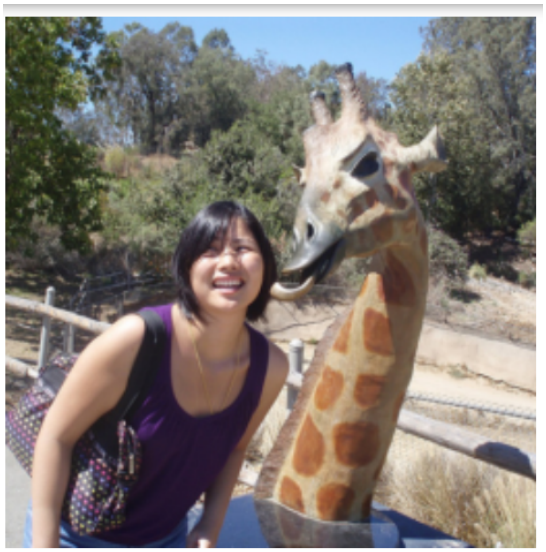

Figure 20: Caption example 1 for Architecture 2 LSTM

Table 21: Generated Captions Comparisons For Figure 20

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.1	a woman is holding a frisbee to her hand (Bad)	1. a young woman poses by a giraffe statue which appears to give her a kiss . 2. a woman standing in front of a giraffe . 3.the giraffe is licking the woman 's face . 4. The young woman visiting the zoo is posing for a photo with a statue of a giraffe . 5.a lady in a purple shirt leaning up to a giraffe .
Deterministic = True	a woman is standing on a wooden bench . (Bad)	
Temperature = 0.4	a woman is holding a frisbee to her hand . (Completely wrong)	
Temperature = 0.8	a young woman is holding a game with her hand up to keep the giraffe . (Mediocre; Not too bad)	

Figure 21: Caption example 1 for Architecture 2 LSTM

Table 22: Generated Captions Comparisons For Figure 21

Hyperparameter	Predicted Captions	Actual Captions
Temperature = 0.1	a stop sign with a stop sign and a stop sign . (Accurate but bad quality)	1. a freshly plowed street is a winter wonderland 2. a stop sign with lots of snow on the ground 3. a red stop sign sits in the snow along suburban streets . 4. a stop sign stands along a snow filled street . 5. a snow covered but recently plowed street in the city .
Deterministic = True	a stop sign with a stop sign and a street sign . (Accurate but bad quality)	
Temperature = 0.4	a stop sign that has been covered in graffiti . (Bad ? but not too bad)	
Temperature = 0.8	a stop sign with some graffiti on it (Bad? but not too bad)	

5 Discussions

Our proposed Encoder-Decoder model is built on a CNN that encodes a picture into a compact representation, followed by an RNN that creates sentences based on the learnt image attributes. When tried on numerous photographs, it performed admirably. It created quite accurate captions for the photographs. However, the source of the input image was equally essential in feature extraction and hence caption development. Certain photographs are not adequately identified, and we discovered that there is still room for development.

The architecture 2 LSTM model performs the best in our experiment. The model architecture is still the same as the one with LSTM as its decoder. However, the feature encoded at each time step changes: Instead of only passing in the image feature at time step one, the input embedding for every step now becomes the concatenation of the feature vector and the word embedding, which emperors the model as our input feature become more informative than it is previously. This new architecture could not only better incorporate the word embedding and the features from

ResNet50, but also incorporate the word embeddings at each time stamp, and therefore give satisfying results.

It seems that the vanilla RNN model performs the worst. It is using RNN cells as its decoder, which means that this model is less capable of capturing long-term relations than the LSTM cell is. When a vanilla RNN network is exposed to long sequences or phrases, it tends to lose the information because it can not store the long sequences and it focuses only on the latest information available at the node. LSTM with input gate, forget gate, and output gate could offer a better solution than RNN cell when it comes to long-term memory and image captioning that spans many time steps.

The deterministic approach does not work well because the next token generated at each step would always be the token with the maximal probability from the softmax distribution of all possible words in the word dictionary. In this way, the model will always predict the same caption with the maximal probability for the images, and this greatly reduces the variability of the model. The deterministic approach could hardly allow the encoder-decoder model to generalize to other unseen images to produce sensible image captioning compared to the sampling from the softmax distribution approach.

We did see duplication of sentences across different temperatures when generating the caption. This is because some words in our softmax distribution have particularly large probability compared with the others. As a consequence of that, no matter how we change the temperature,(as long as the temperature is reasonable), the resultant sampled text is not quite different at all.

One interesting finding is that, though the LSTM model with architecture 2 out performs LSTM model with architecture 1 and Vanilla RNN model in terms of BLUE scores, the quality of caption is not necessarily much better than that generated by previous two models. As one can see that they still generate the captions that are simply not correct. The reason could be that we only have limited training instances. If our model is not exposed to sufficient training cases, then it may not be able to generalize well to the unseen data. Deep neural networks require lots of data in order to perform well. So, to further improve our model, we should be able to collect as much data as possible.

We can also further improve our model by tuning the parameters that we did not get a chance to tune in this programming assignment. If more time and resources are permitted, we can optimize over the choice of optimizers, the choice of encoders, the number of hidden layers used in our decoder, and how we initialize the parameters of our model.

For reference, we also analyzed and summarized the outcome in the previous section. So, for the grading purpose, you could go back to the previous section in case you miss some points when grading our work. Thanks.

6 Team contributions

Rui Zhang: Abstract + Methods + Baseline LSTM experiments

Yifei Wang : Introduction + Related Work + RNN experiments

Collaboratively done: All the code+ Architecture 2 Model Experiments + Discussion

References

[1] Bell, S., Zitnick, C. L., Bala, K., & Girshick, R. (2016). Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2874-2883).

[2] Chen, M., Ding, G., Zhao, S., Chen, H., Liu, Q., & Han, J. (2017). Reference based LSTM for image captioning. In *Thirty-first AAAI conference on artificial intelligence*.

[3] Huang, L., Wang, W., Chen, J., & Wei, X. Y. (2019). Attention on Attention for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 4634-4643).

[4] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

[5] Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). *Dive into deep learning*. arXiv preprint arXiv:2106.11342.