Python Implementation of RNN and LSTM for Image Captioning with PyTorch

Colin (Zirui) Wang*
Halıcıoğlu Data Science Institute
Department of Cognitive Science
University of California, San Diego
San Diego, CA 92093
zwcolin@ucsd.edu

Jerry (Yung-Chieh) Chan*
Halıcıoğlu Data Science Institute
University of California, San Diego
San Diego, CA 92093
ychan@ucsd.edu

Bingqi Zhou
Halıcıoğlu Data Science Institute
Department of Mathematics
University of California, San Diego
San Diego, CA 92093
biz024@ucsd.edu

Abstract

In this part of the assignment, we examine the ability of Recurrent Neural Networks and Long-Short Term Memory to deal with data that has temporal structure by using them to generate captions for images. We used the ResNet50 as encoder to turn images into features vectors along with word embedding vectors, and a RNN/LSTM as decoder to turn these vectors into captions word by word. Our key finding is that LSTM converges faster and performs better than traditional RNNs. We also found that embedding and hidden size determines the model's ability to predict accurately, and that temperature gives the model diversified the generated captions given the same image. Using the best parameters, we achieved a BLEU1 score of 68.9 (out of 100) and a BLEU4 of 8.9 (out of 100) for baseline LSTM, a BLEU1 score of 68.4 (out of 100) and a BLEU4 of 8.71 (out of 100) for vanilla RNN, and a BLEU1 score of 70.7 (out of 100) and a BLEU4 of 9.2 (out of 100) for the architecture 2.

1 Introduction

With the advancement of computing power in recent years, people are able to construct complex deep learning models to make predictions on sequential data. Using Recurrent Neural Networks (RNN), researchers are competing to make semantically informative and grammatically correct sentences. In this assignment, we used the COCO 2015 Image Captioning Task dataset [1], from the Common Objects in Context(COCO) repository, to test our models. COCO is a dataset created for advancing the state-of-the-art in object recognition. It contains common objects in their natural context. Here, we used this dataset to train, validate, and test our model in predicting captions. Because of limitations in computing power, we only used about $1/5^{th}$ of the original dataset.

In this part of the assignment in particular, we implemented the baseline LSTM model, a RNN model, and a modified LSTM model where we concatenated image features into inputs for every timestamp. We performed training, validation and testing on all three models to compare the performance of them. We experimented with different hyperparameters including learning rate, weight

*equal contribution

decay, scheduler step size, epoch, embedding size and hidden size, etc. We also experimented how the generation approach such as deterministic approach and stochastic approach (with different sets of temperatures) will affect the quality of generated sequence. Finally, we attempted fine-tuning by back-propagating errors into encoder convolutional layers.

In these experiments, we found that the modified LSTM performed the best, following by the baseline LSTM, and RNN at the end. We found that the scheduler is important in the training procedure for these models as the model validation loss often reaches its plateau and gets stuck if we didn't adjust the learning rate. We found that a larger embedding and hidden size usually led to smaller validation loss as well as better BLEU scores, but it also led to more overfitting due to the complexity of model. We found that although a deterministic approach will usually give us best BLEU scores, a stochastic approach with a temperature between 0.1 and 0.2 will give us better generated caption because of increased diversity and insignificant drop of BLEU scores. We found that changing the batch size also influence the model's performance as the proportion of $< pad >$, which is determined by the longest caption in a batch, is generally smaller when batch size is relatively small. While this small batch size increased training, validation, and test loss because of a reduced proportion of $< pad >$, it also gave the models the opportunity to learn more about captions themselves, therefore leading to even high BLEU scores in the end. Finally, we found that back-propagating errors into convolutional layers didn't actually give us a better results on the model's performance. Instead, using a pre-trained model with an unfreezed last linear layer gave the model best performance as our training dataset is relatively small for the encoder to perfectly fit and find patterns from scratch.

2 Related Work

David Rumelhart et. al [2] first proposed the Recurrent Neural Network in 1986. In their work, they used the word "internal hidden units" to describe the structure of RNN. They accomplished the task to predict temporal and sequential data by back-propagating errors in these recurrent internal hidden units. In 1997, Hochreiter and Schmidhuber et. al [3] invented Long Short-Term Memory model because they found that learning to store information over extended time intervals by recurrent backpropagation takes a very long time due to insufficient and decaying error backflow. Their LSTM model is able to bridge minimal sequential lags by enforcing constant error flow through constant error carousels within special units, which we now call the cell state.

In 2009, Graves and Schmidhuber [4] proposed a novel deep learning frame work with LSTM where they created a globally trained offline handwriting recognizer that takes raw pixel data as input and predicts sequential data. Their work showed that deep learning models can directly learn from the representations of data without any alphabet specific preprocess and prior knowledge. Li and Wu [5] proved that LSTM based acoustic modeling methods were shown to give the state-of-the-art performance on some speech recognition tasks over traditional deep recureent neural networks in 2014. At the same year, Sultskever et. al [6] proposed a novel variation of RNN, namely Seq2Seq, to enable any subtypes of RNN (vanilla RNN, LSTM, GRU, etc) to map sequences to sequences. They achieved the state-of-the-art performance on English-to-French translation tasks with their model.

3 Architecture for LSTM and Vanilla RNN

3.1 Description of Architectures

RNN (Recurrent Neural Network) is a class of neural network with an internal state that serves as memories. They're often used to handle sequential data with their memories, which enable them to process current input with the information collected in the past. LSTM (Long-Short-Term Memory) is a type of RNN architecture with an input gate, an output gate and a forget gate in each memory cell. Those gates enable a more dynamic use of memory. The input gate controls the input to the memory cell; The output gate controls the extent of the memory's effect on the unit's prediction; The forget gate decides if the memory remains in the cell. The architecture has shown excellent performance in various of application on sequential data such as speech and hand writing recognition.

The model is composed of two part: an encoder and a decoder. The encoder is a pretrained ResNet-50 model with the last layer substituted with a fully connected layer with size of our desired embedding size. Its purpose is to extract features from the image.

The first decoder that we experimented with is a single-direction LSTM model with input size as the embedding size, a fully connected output layer, and an embedding layer. The second decoder that we experimented with is a single-direction vanilla RNN model. The architecture of this model is exactly the same with LSTM model except that we replaced all LSTM units with RNN units. Table 1 shows our choice of loss function, optimizer, and scheduler. Table 2 shows our choice of best hyperparameters. Unless specified, this set of hyperparameters is used for experiments in following sections.

3.2 Obtaining Word Embeddings & Sampling Approach

The caption input is firstly mapped into indices, then one-hot encoded, and finally converted to the expected input dimension, which is the embedding size from the encoder, with an embedding layer. The encoded image and the caption will be inputted to LSTM sequentially following the "teacher forcing" training method. The outputs will then be converted to the dimension of the vocabulary size by a fully connected layer. While testing and predicting, the model will predict the caption word-by-word. The initial input is the encoded image. The second input will be selected based on the first output.

There are two ways of selecting an output, "deterministic" and "stochastic". The deterministic method always picks the word with the highest predicted probability from the output. For the stochastic method, we feed the output to a softmax layer and select the word based on the probability distribution, which is the output of the softmax layer. The process repeats until it reaches the maximum length set by us. The state of the LSTM/RNN layer is relayed through out the word-generating process.

CATEGORY	CHOICE
Loss Function	Cross Entropy Loss
Optimizer	Adam Optimizer
Scheduler	StepLR Scheduler

Table 1: Choice of loss function, optimizer, and scheduler

HYPERPARAMETER	VALUE
Batch Size	128
Rotation	FALSE
Horizontal Flip	FALSE
Vertical Flip	FALSE
Epochs	10
Learning Rate	1e-3
Eps	1e-8
Weight Decay	1e-4
AMSGrad	FALSE
Gamma	0.1
Early Stop	TRUE
Patience	4
Step Size	3
Hidden Size	2048
Embedding Size	2048
Encoder Unfreeze	FALSE

Table 2: Best set of hyperparameters

Using the above set of hyperparameters, we trained the baseline LSTM model and the vanilla RNN model. Figure 1 shows the training and validation loss curve for each model over 10 epochs

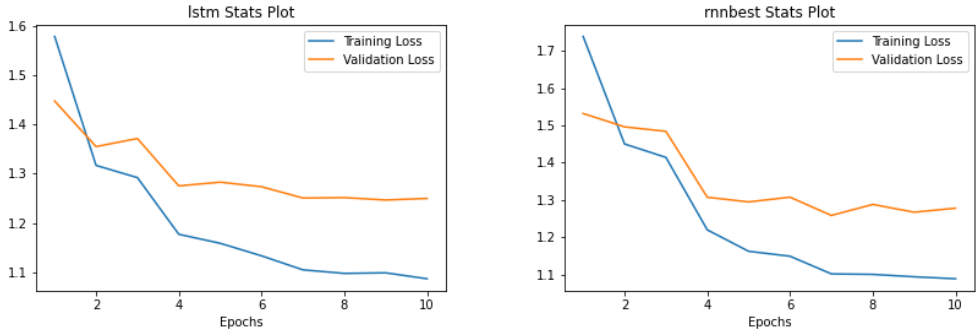


Figure 1: Comparison of training and validation loss over 10 training epochs between baseline LSTM model (left) and vanilla RNN model (right).

3.3 Discussion of Learning Curves

From the above figure, it shows that LSTM started with a lower loss but RNN model is able to drive its loss down as the learning rate decreased (specifically at epoch 3 when the scheduler decreased the learning rate for the first time). Both model's loss reached a plateau after seven epochs that cannot be further lowered by decreasing the learning rate.

Although the learning rate curve for both models looks similar, we noticed that RNN's learning curve is more unstable[1] while LSTM's learning curve is smooth. Based on our training statistics, both models need 7 epochs to achieve the best performance with our current set of hyperparameters, but LSTM was able to achieve a lower validation loss than vanilla RNN by a magnitude of 0.1 in the first epoch. So in terms of satisfactory performance[2], vanilla RNN needs more epochs to achieve the desired result.

We attributed RNN's performance at initial epochs to the opinion that vanilla RNN model learns slower on a big learning rate than the baseline LSTM. This is because it doesn't have the gates and the highway to prevent the gradients from exploding or vanishing when they were back-propagated to deeper layers relative to the output layer. With exploding gradients, vanilla RNN needs a smaller learning rate to balance out the large gradient. With vanishing gradients, vanilla RNN needs more epochs as to sufficiently update parameters in deeper layers relative ot the output layer. This explains why vanilla RNN model needs more epochs to achieve satisfactory performance than the baseline LSTM model.

4 Discussion of Results on Best Models

This section presents the result in terms of cross entropy loss on the test set, BLEU-1 and BLEU-4 scores against the reference captions in the test set with the deterministic approach for the best models of baseline LSTM and vanilla RNN. Both models used the best set of hyperparameters described in Table 2. The results are shown in Table 3. Their corresponding training & validation losses are shown in Figure 1 in the last section.

[1] Equivalently, worse than

[2] We define *satisfactory* as the magnitude of difference between the current validation loss and the lowest possible validation loss with the current set of hyperparameters for both models.

MODEL	TEST SET LOSS	BLEU1	BLEU4
Baseline LSTM	**1.247**	**0.689**	**0.0890**
Vanilla RNN	1.309	0.684	0.0871

Table 3: Comparison between the performance of baseline LSTM and vanilla RNN on the test dataset

4.1 Discussion on Test Set Loss

From Table 3, we can see that Baseline LSTM model slightly outperformed Vanilla RNN model in all three metrics. The LSTM model achieved a 4% lower test loss and slightly higher BLEU1 and BLEU4 score. We believe that the reason where the test set loss on the baseline model is lower than vanilla RNN model is that LSTM's model is more complex and its gradients are useful in deeper layers relative to the output layer during back-propagation, as the architecture of LSTM alleviates its gradients from vanishing or exploding. Subsequently, the baseline LSTM model is more accurate in predicting words at the beginning of the sentence and robust in building semantic and grammatical relationships between words.

4.2 Discussion on BLEU scores

We used nltk's functions to calculate BLEU scores. All bleu scores shown in Table 3 are the average of BLEU scores across each image in the test dataset, where a single BLEU score is generated by comparing the generated caption against all five captions for that particular image. Before we sending these inputs into calculation, both original captions and the generated caption have been properly tokenized using nltk's tokenization function to ensure consistency.

This lower test set loss in baseline LSTM model resulted in a higher BLEU4 score than vanilla RNN, where the LSTM model is more capable of generating consecutive words that are similar to the original referenced captions. On the other hand, the increment of BLEU1 score is higher only by a slight margin for the baseline LSTM. So, we concluded with our observations that while vanilla RNN is capable of generating the *correct* words, the sequence that this model generated is often not as grammatical as the baseline LSTM, resulting in a similar BLEU1 score but a lower BLEU4 score.

5 Experimentation with Generation Approach and Temperature

We used the best LSTM baseline model mentioned in Section 4. The hyperparameter settings can be found in Table 2. We experimented with the deterministic approach as well as stochastic approach with temperature from 0.1 to 1. Table 4 shows the BLEU1 and BLEU4 scores for this model with respect to different temperatures.

TEMPERATURE	BLEU1	BLEU4
1	0.485	0.026
0.9	0.536	0.032
0.8	0.576	0.041
0.7	0.606	0.05
0.6	0.632	0.059
0.5	0.651	0.067
0.4	0.666	0.076
0.3	0.677	0.081
0.2	**0.684**	**0.085**
0.1	**0.687**	**0.088**
DTM($\lim_{T\to 0^+} T$)	0.689	0.089

Table 4: Temperature Tuning for Baseline Model. Here, DTM means the deterministic approach, where no temperature is available. All BLEU scores have a max of 1 and a min of 0.

From this table, we found that BLEU1 and BLEU4 score goes up when temperature decreases, and these scores tend to saturate when the temperature reaches 0.2 and below. Based on our results[3] on the baseline model, the deterministic approach actually gives us highest BLEU1 and BLEU4 scores. This is because with a deterministic approach, the model can always get its most confident word based on inputs, and because our model achieves a relatively low validation loss, its confidence on word choices usually give us the most accurate outcome. Subsequently, the deterministic approach is able to get the highest BLEU1 and BLEU4 scores.

On the other hand, we could not conclude that just because the deterministic approach gives us the highest scores, it will help the model work well in reality. In fact, we believe that a stochastic approach with a temperature between 0.1 and 0.2 would give us even better results *in practice*. This is because the deterministic approach will always generate the same captions for the same image whereas the stochastic approach won't necessarily behave like that. Since the model samples from a softened distribution with stochastic approach, it makes the model more humanized and diverse in generating captions for each image. Between a temperature of 0.1 and 0.2, we not only ensure that the BLEU scores don't drop much (which guarantees accuracy), but also empowers the model the ability to express differently each time for the same image because of the sampling of a probability distribution. A temperature lower than 0.1 could make the model behave similar to the deterministic approach, while a temperature higher than 0.2 could make the model generate inaccurate image captions.

6 Comparison and Discussion of Fine-Tuned Model

We tried different set of embedding size and hidden size on the baseline LSTM model. For each change of hyperparameter, we recorded the train and validation loss at the 5th epoch. To examine the effect of different embedding size, we keep the hidden size at 2048, and to examine the effect of different hidden size, we keep the embedding size at 2048. All the other hyperparameters are kept the same as Table 2. Table 5 shows the lowest training and validation loss over the first 5 epochs with different embedding size, and Table 6 shows the lowest training and validation loss over the first 5 training epochs with different hidden size.

EMBEDDING SIZE	TRAINING LOSS	VALIDATION LOSS
256	1.307	1.376
512	1.277	1.357
1024	1.258	1.357
2048	**1.232**	**1.330**

Table 5: Lowest Training & Validation loss for different embedding size over 5 training epochs

HIDDEN SIZE	TRAINING LOSS	VALIDATION LOSS
256	1.429	1.465
512	1.369	1.437
1024	1.325	1.378
2048	**1.321**	**1.374**

Table 6: Lowest Training & Validation loss for different hidden size over 5 training epochs

From the above results, we found that the model tends to perform better when embedding size and hidden size are large. As a model with larger embedding and hidden size tends to be more complex, a side effect is overfitting when trained for more epochs. Regardless, we found that the validation

[3]We are sure that we implemented the deterministic approach exactly as what PA said by taking the maximum output at each step instead of using a softmax function, although the PA said that deterministic approach "usually doe not work well at all". More discussion about this can be found on Piazza Post @760

loss for a larger embedding and hidden size is still significantly lower than a model with smaller embedding and hidden size. Due to time and resource constraints, we only tested embedding size up to 2048 and hidden size up to 2048. Therefore, with results from Table 5 and Table 6, we use a combination of 2048 embedding size and 2048 hidden size to propose our model configuration. Keeping the same set of other hyperparameters, we trained this model along with the baseline model, which has 300 embedding size and 256 hidden size for 10 epochs. Table 7 shows the resulting test set loss, BLEU1 and BLEU4 scores for each model. Figure 2 shows the training and validation loss curve between these two models over 10 epochs.

MODEL	EMBED SIZE	HIDDEN SIZE	TEST LOSS	BLEU1	BLEU4
Baseline	300	256	1.404	0.560	0.035
Ours	2048	2048	**1.247**	**0.687**	**0.088**

Table 7: Comparison between baseline model and ours based on embedding size, hidden size, and resulting test loss, BLEU1 score, and BLEU4 score. BLEU1 and BLEU4 scores are based on a temperature of 0.1 using stochastic approach. All hyperparameters other than embedding size, hidden size, and temperature can be found on Table2

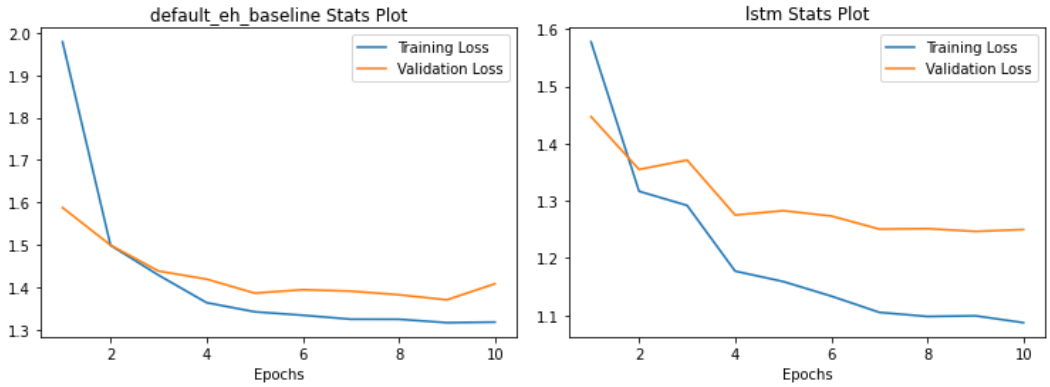


Figure 2: Comparison of training and validation loss over 10 training epochs between baseline model (left) and ours (right). *Notice that the range of y-axis is different between two figures.*

From the above results, we found that our model with larger embedding and hidden size learns to generate the correct output faster than the baseline model given the same initial learning rate and different training & validation loss at epoch 1 and 2. We found that our model is more sensitive to changes in learning rate than the baseline model when the model reached a plateau. We derived this conclusion because our model's validation loss always drops a relatively significant amount whenever the optimizer scheduler changes the learning rate whereas the baseline model's validation loss doesn't change much after the optimizer scheduler changes the learning rate. We believe that this difference occurs because larger embedding and hidden size give the model more space to learn from subtle features, which can be seen as a result of continuously decreased validation loss at smaller learning rates. We also found that our model performed slightly better in terms of BLEU1 score than the baseline model, but significantly better in terms of BLEU4 score. We believe that this is caused by difference in hidden size, where a larger hidden size could store more information on which exact word to generate and its relationship to the word before and after it. Therefore, our model can perform much better in BLEU4, which is a score based on 4-gram.

7 Implementation of Architecture 2

7.1 Structure

Architecture 2 is very similar with the LSTM model. Instead of feeding the encoded image at the beginning of the generation process, we feed it to LSTM in every steps by concatenating it with the embedded caption. The first input will be the embedded $< pad >$ concatenate with the image encoding. The model is expected to generate the token $< start >$. The output word will be concatenate with the same image encoding and feed into LSTM again. Although in this architecture the image encoding dimension doesn't have to match the caption embedding dimension, we set them the same in order to compare the model with baseline model.

7.2 Test Result

We test baseline model and architecture 2 with the default batch size, hidden size (LSTM), and embedding size. We keep other training details such as learning rate, l2, and data transformation the same as the best-performing baseline model. Notice that even both model have the same embedding size and hidden size, architecture 2 has a larger LSTM layer since the input dimension is doubled. Figure 8 shows the training and validation loss throughout the training process. Table 8 compares the testing statistic with the baseline model.

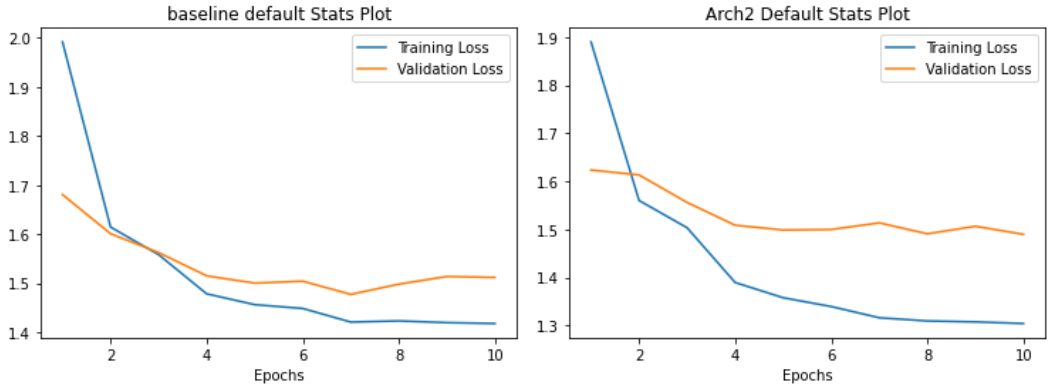


Figure 3: Training and validation loss over 10 training epochs for baseline model (left) vs architecture 2 under default size (right)

MODEL	TEST LOSS	BLEU1	BLEU4
Baseline	1.519	0.683	0.081
Architecture 2	**1.501**	**0.707**	**0.092**

Table 8: Comparison between baseline model and architecture 2. *We used a batch size of 64 instead of 128 here*, which reduced the proportion of $< pad >$ in each batch because this proportion will be generally determined by the longest caption within a batch. Therefore, the test loss is slightly higher than the loss in other sections, but at the same time, the model learned more substantial information, resulting in higher BLEU scores.

7.3 Discussion

Architecture 2 outperformed the baseline model in loss, BLEU1 score, and BLEU4 score. Architecture 2 feed the image to the LSTM layer in every generation step. This helps the generated caption focus on the image features. On the other hand, the baseline model can rely on the LSTM memory (state) to remember the image. We can expect that the captions generated by architecture 2 is more

related to the image itself instead of the output word at last timestamp. We think that this is the reason that architecture 2 outperform baseline model.

We notice that the losses in this experience are larger then those in previous sections. After experimenting with different sets of hyperparameters, it turned out that batch size is the reason that lead to a higher loss. A larger batch will consist more $< pad >$ tokens since we will pad all the sentences to the same length as the longest sentence in the batch. It's much more easier for the model to predict a $< pad >$ token as they always follows $< end >$ or $< pad >$ tokens. Therefore, the more $< pad >$ tokens there are in the dataset (batches) the lower the loss will be.

8 Visualization of Caption Generation Results

8.1 Good Predictions

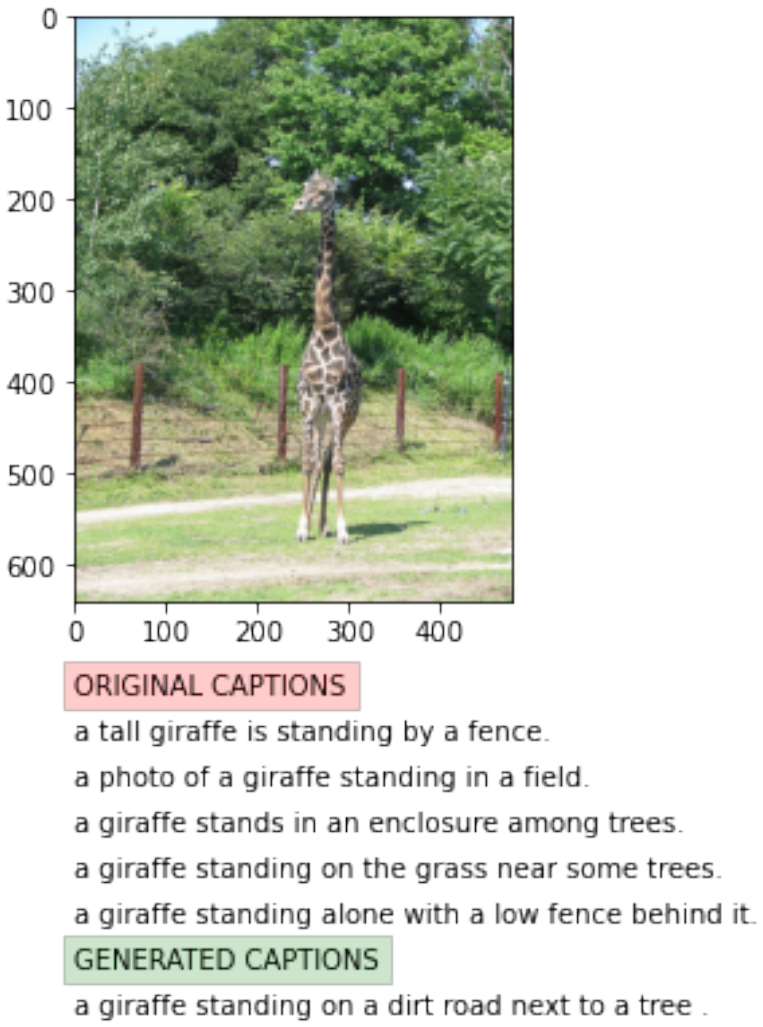


Figure 4: In this example, the model successfully captured details about the giraffe, the trees behind it, and the road (which is not even mentioned in the original captions). The only small deficit is that the model says a tree instead of some trees. We believe that this is likely caused by resizing the center crop, which lost some information in the original image.

ORIGINAL CAPTIONS

a hummingbird sits perched alone on a branch.

a small white and brown bird resting on a twig.

small long billed bird sitting on thin branch.

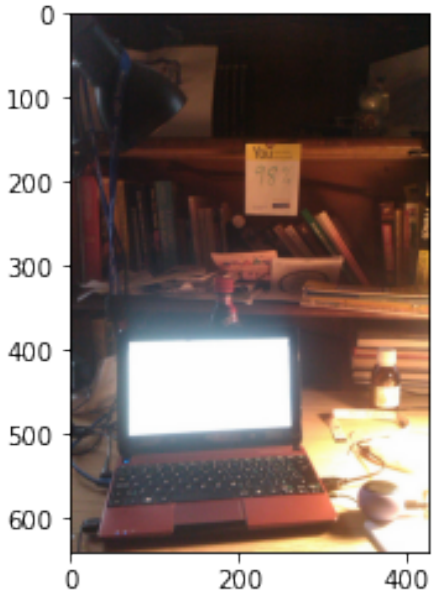
a small bird on a tree branch with a blurry background

a small bird sitting on a thin tree branch.

GENERATED CAPTIONS

a bird perched on a small branch in a tree .

Figure 5: In this example, the model successfully captured details about the bird, the perching action, the tree, and the small/thin/tiny branch.



ORIGINAL CAPTIONS

a computer is sitting on a desk with books.

an open laptop sits in front of a bookshelf.

a desk and bookshelf with a laptop computer, various books and a desk light.

a cluttered desk with a laptop computer turned on.

a computer is left on in front of a bookshelf

GENERATED CAPTIONS

a laptop computer on a desk next to a keyboard

Figure 6: In this example, the model successfully captured the laptop computer and the fact that it's on a desk. The only deficit is that the model probably didn't recognize that the keyboard is part of the laptop. Regardless, we'll consider this as a successful example.

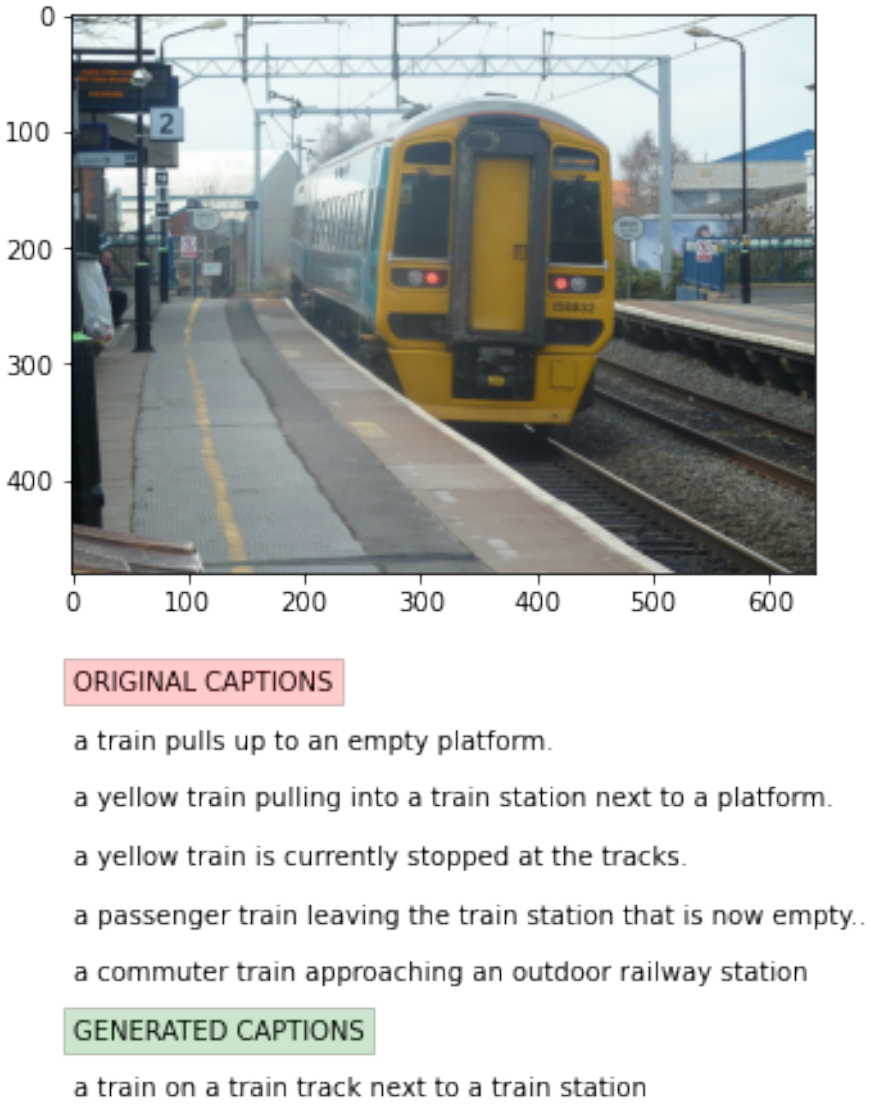
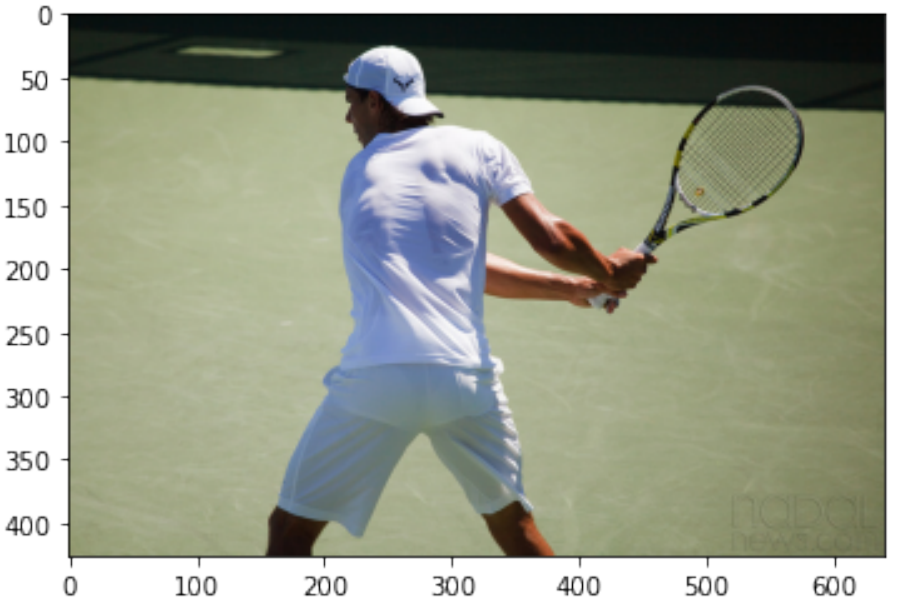


Figure 7: In this example, the model successfully captured all major components of this image, which are the train, the train track and the train station. The only deficit is the model failed to capture some of the adjective details about these objects, such as a commuter train or a yellow train.



ORIGINAL CAPTIONS

a man ready to swing his tennis racket

a man is playing tennis, dressed in white, about to swing his racket.

a tennis player prepares to swing with both hands.

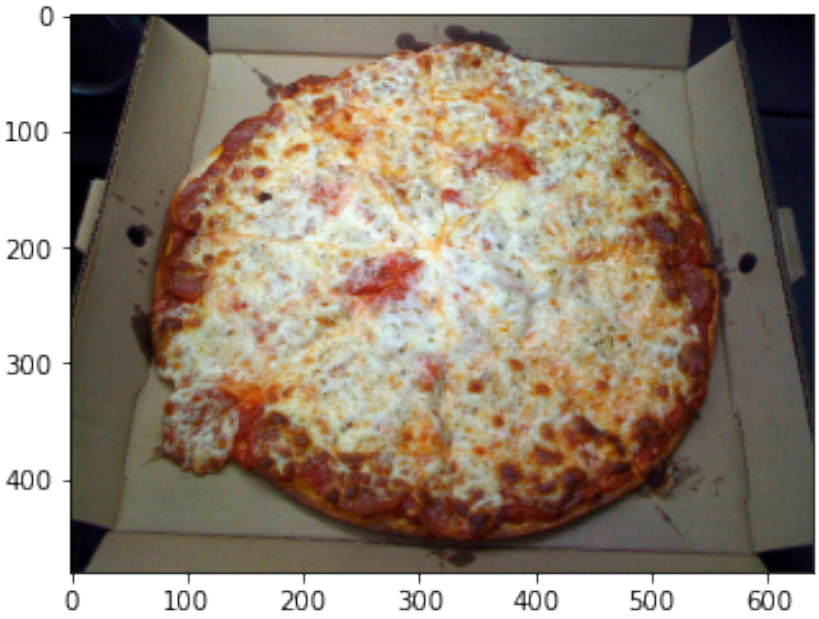
a man holding a tennis racquet while standing on a tennis court.

a man with a tennis racket getting ready to swing.

GENERATED CAPTIONS

a man holding a tennis racket on a tennis court .

Figure 8: In this example, the model successfully captured all major components of this image, which are man, a tennis court, and a tennis racket. Although the action is swing, in one of the original captions, the action is holding. So this example is considered successful as well.



ORIGINAL CAPTIONS

a pizza sitting on top of a pizza box covered in cheese.

a cheese pizza cut into eight pieces in the box.

a cheese pizza from a store in a box.

the pizza looks like it was ordered with extra cheese.

a large cheese pizza in a cardboard box.

GENERATED CAPTIONS

a pizza with cheese and cheese on top

Figure 9: In this example, the model successfully captured the pizza and the cheese on top of it. The model says cheese and cheese is probably because there is too much cheese :). The model failed to capture the box, but we believe that it's likely caused by resizing the center crop transformations where the box edge is ignored.

ORIGINAL CAPTIONS

a bathroom with a toilet, sink, and shower.

a full bathroom with a wicker laundry basket.

a little bathrood decorated with many colorful objects

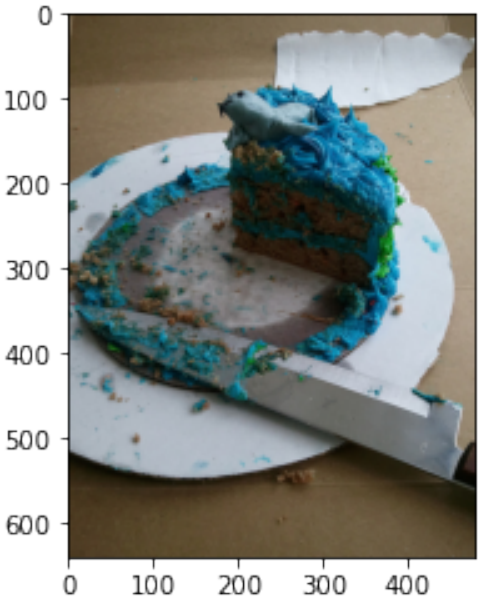
a small bathroom containing a toilet and sink.

bathroom containing a toilet, a sink and a wicker basket.

GENERATED CAPTIONS

a bathroom with a sink , toilet and a tub .

Figure 10: In this example, the model resembled the first original caption, which is very good!



ORIGINAL CAPTIONS

a mostly eaten cake sitting on a table next to a knife.

there is a small piece of bright blue cake left on a plate with a knife.

one piece left of a blue birthday cake.

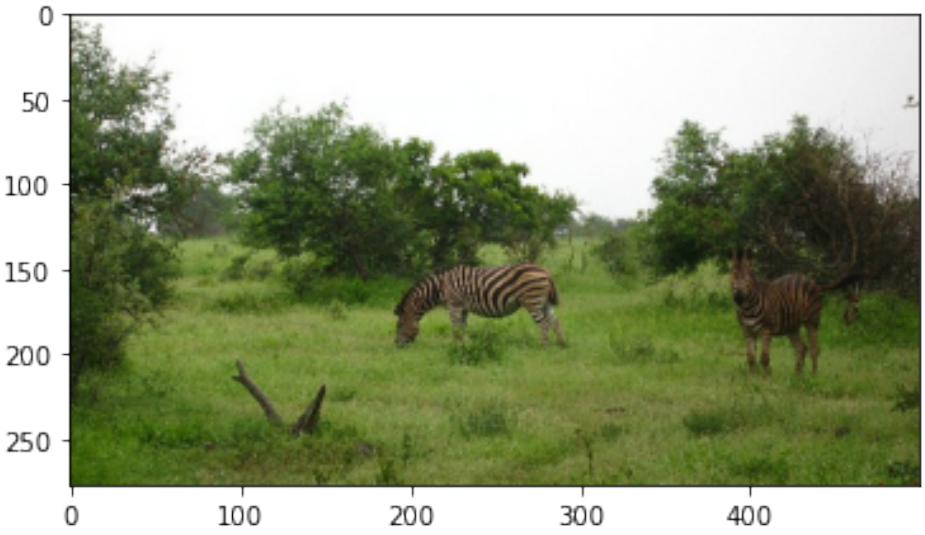
the last piece of blue frosted cake on a tray.

long knife and leftover cake on a platter.

GENERATED CAPTIONS

a white plate with a piece of cake on it .

Figure 11: In this example, the model captured the piece of cake and the white plate. The only deficit is that it missed the knife object. We believe that this is likely caused by the fact that there's some blue stuff on it, which make it hard for the encoder to encode this object accurately.



ORIGINAL CAPTIONS

a zebra standing in a field grazing as another looks alert.

a couple of zebras graze on some grass

some animals in a field with grass near bushes

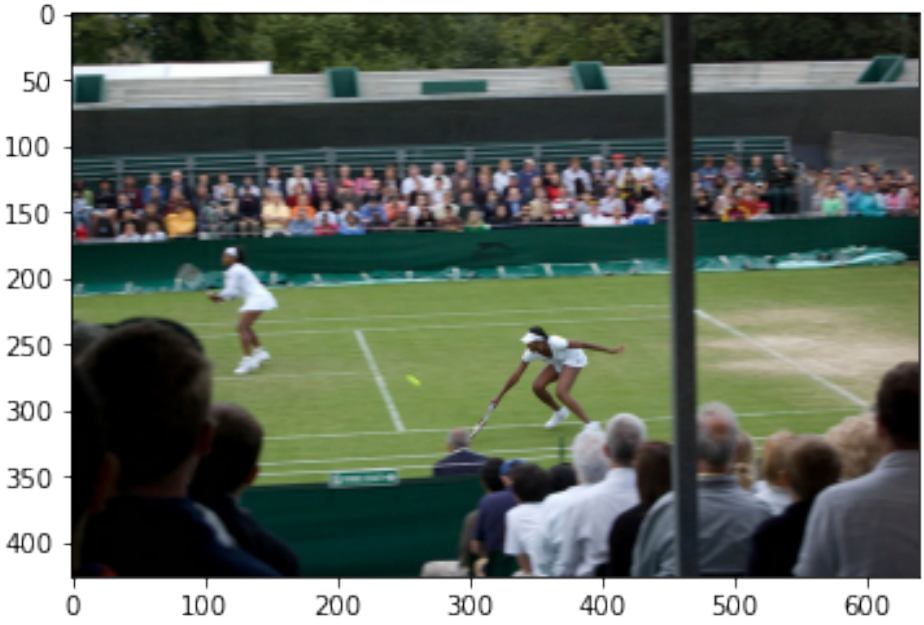
a couple of zebra standing on top of a lush green field.

two wild zebra eating grass in a safari .

GENERATED CAPTIONS

two zebras standing in the grass next to each other

Figure 12: In this example, the model captured two zebras, their states (standing), their relationships (next to each other), as well as the environment (the grass).



ORIGINAL CAPTIONS

two women in white dresses playing a game of tennis.

two women playing tennis while a crowd watches.

two women playing doubles tennis at a tennis match.

a couple of women on a court playing tennis.

two women on the same side of the tennis court, playing tennis.

GENERATED CAPTIONS

a crowd of people watching a game of tennis .

Figure 13: In this example, the model captured what game it is, the audience, and the fact that the audience is watching the game. The only deficit is that the model failed to recognize the major object, which is two women. Regardless, we consider this as a successful one.

8.2 Bad Predictions

ORIGINAL CAPTIONS

a white toilet sitting next to a white bath tub.

a marble wall is next to a tub and a toilet stands near the tub.

a bathroom with a boilet and marble enclosed shower stall.

a bathroom filled with bathroom furniture and decor.

an all white bathroom with a bathtub and a toliet.

GENERATED CAPTIONS

a toilet with a sink and a toilet .

Figure 14: In this example, the model recognized something as a sink whereas there is no sink. We believe that it might have recognized the bathtub as the sink.

ORIGINAL CAPTIONS

a couple of people that are flying some kites

people playing with kites in a park on a clear day.

a small girl flying a kite on a sandy road.

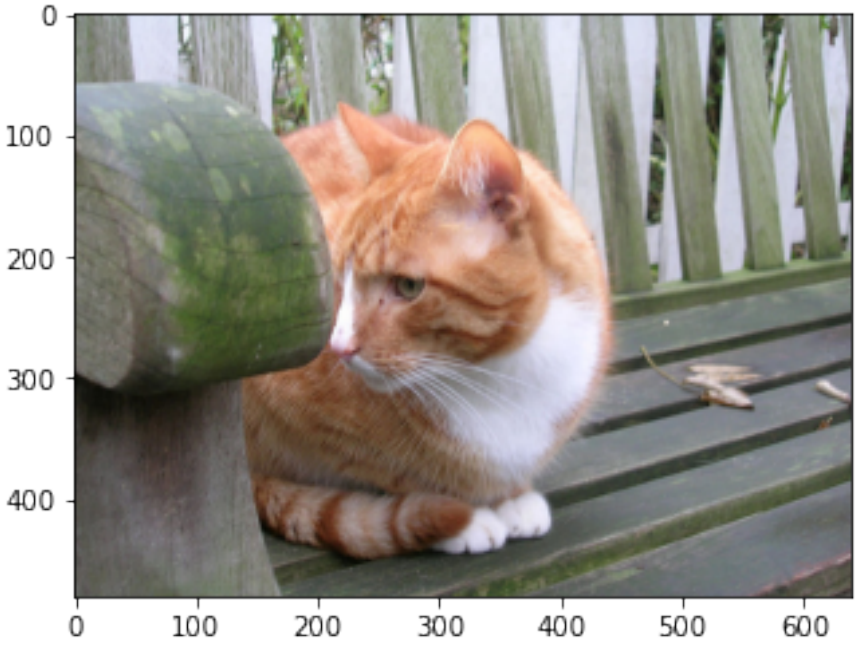
a woman playing with a kite hods it in front of her

a girl in a white shirt is flying a red kite

GENERATED CAPTIONS

a man flying a kite in a park .

Figure 15: In this example, the model recognized the person but predicted with the wrong gender although other features are correct (flying a kite, in a park).



ORIGINAL CAPTIONS

a cat sitting on a wooden bench outside.

a cat is sitting on a bench outside.

a cat with a concerned look sitting on a bench.

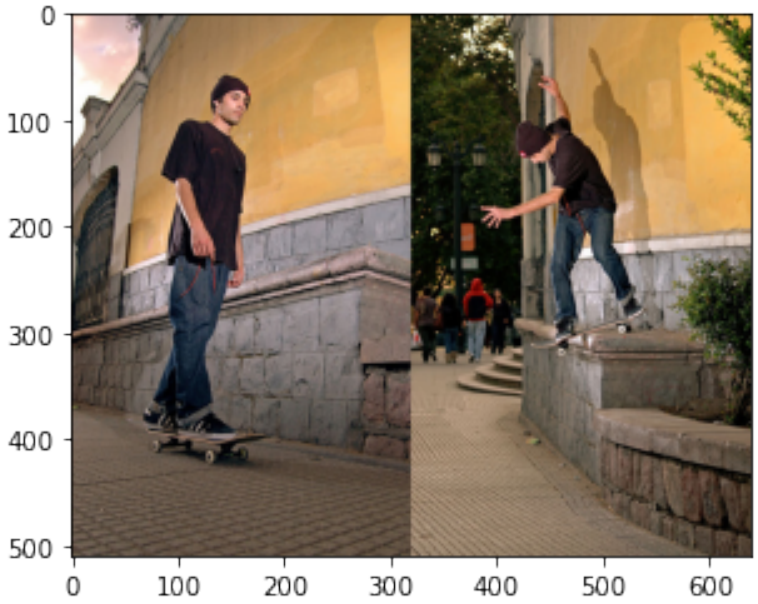
a cat sitting on a wood bench near leaves.

a large orange and white cat sitting on top of a wooden bench.

GENERATED CAPTIONS

a cat sits on a table with a plant looking over .

Figure 16: In this example, the model wrongly captured the word table and plant. This is likely caused by the generalization of surfaces and the fact that the bench rail is green



ORIGINAL CAPTIONS

a man riding a skateboard next to a yellow building.

a skateboarder balances on his skateboard, then balances on the board at the edge of a low wall.

a man performing skateboard tricks on the street.

a person is on a skateboard performing tricks off a wall.

a man riding on a skateboard down a side walk.

GENERATED CAPTIONS

a man and woman standing in front of a large wall .

Figure 17: In this example, the model didn't realize that these are two images instead of one. That's why it said there were two people. The misclassification of man as woman is probably because in the right image, the model recognized the hat as long hair.

ORIGINAL CAPTIONS

a mama and a baby duck are swimming in the water.

two birds are swimming along on a lake.

a couple of small birds in the water.

an adult duck and a duckling on a pond.

two ducks swimming in the water near each other.

GENERATED CAPTIONS

a small polar bear swimming in the water

Figure 18: In this example, the model captured that there are animals swimming in the water, which is correct. However, instead of saying ducks or birds, it said polar bears. We believe that the model said this because polar bear also has white skins and they can swim!

ORIGINAL CAPTIONS

a man standing in a kitchen holding a glass full of alcohol.

a young man stands in a kitchen full of alcohol

a man standing in a kitchen with several bottles on the counter.

a man standing in a kitchen holding a drink next to a bunch of bottles.

a man with a pony tail stands in a kitchen holding a drink.

GENERATED CAPTIONS

a man is standing in a kitchen with a sink .

Figure 19: In this example, the model captured that the man is standing in a kitchen, but there is no sink! We believe the model either assumes too much that a kitchen must have a sink or it sees that table surface as the sink.

ORIGINAL CAPTIONS

a man holding a tennis racquet on a tennis court.

a man taking a swing at a tennis ball

a tennis player is standing close to the net.

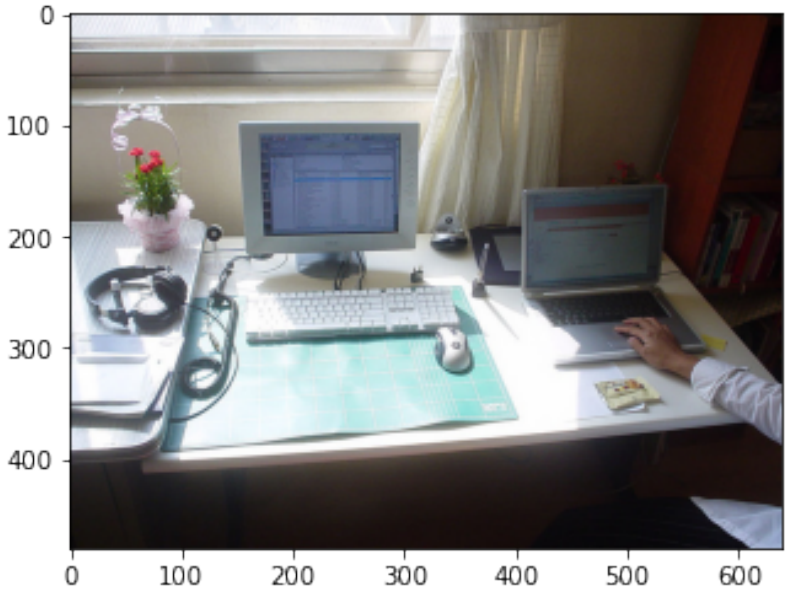
a man in black shirt and blue shorts playing tennis.

a man standing in the middle of a tennis court with a large crowd watching

GENERATED CAPTIONS

a couple of people playing tennis on a tennis court

Figure 20: In this example, the model recognized that the background is a tennis court and someone is playing tennis. However, it probably messed up with the audience where it says a couple of people playing that.



ORIGINAL CAPTIONS

two computers that are sitting on a desk.

a desk with a computer monitor, keyboard and mouse and a laptop computer.

there is a laptop computer next to a desktop computer.

a person uses a laptop on a desk with another computer.

a person typing on one computer with another adjacent.

GENERATED CAPTIONS

a desk with a laptop , a mouse , and a laptop .

Figure 21: In this example, the model probably misrecognized the desktop computer as a laptop, and because a laptop is not supposed to have a separate mouse, it further specified that a mouse is also on the desk.

ORIGINAL CAPTIONS

a woman holds a weiner in each hand.

the woman is holding up two large hot dogs.

a woman is holding two hot dogs in her hands

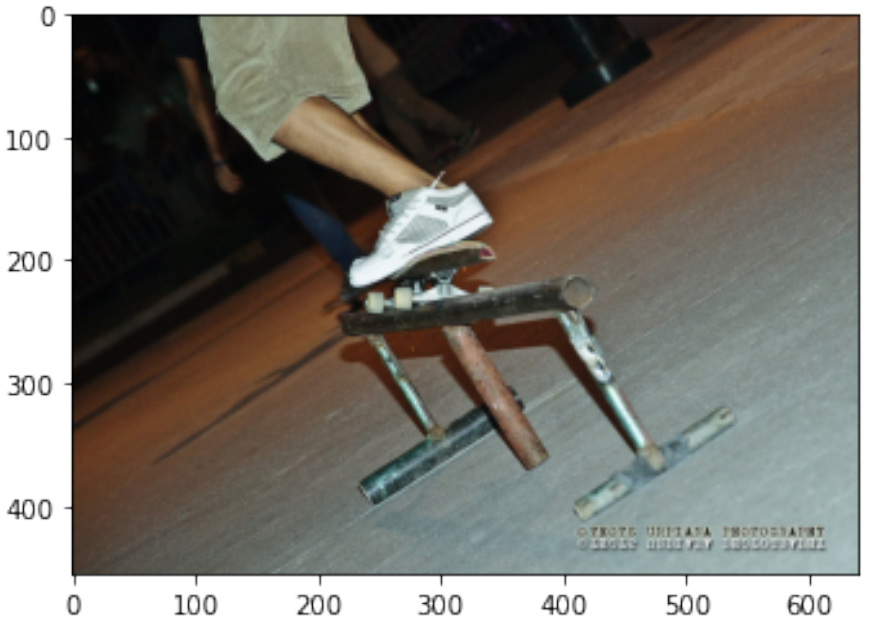
a woman holding two sausage rolls on a sidewalk

a man with sunglasses holding a hot dog in each hand.

GENERATED CAPTIONS

the man is holding a hot dog outside .

Figure 22: In this example, the model says a hot dog whereas there are two hot dogs. Also, it doesn't matter the model says it as a man or a woman because even the original captions have conflicts in this (i.e. the first four captions said woman where as the last captions said man)



ORIGINAL CAPTIONS

a man grinding his skateboard on a rail.

a man riding a skateboard on the side of a metal rail.

a man on a skateboard grinding on a pole

a young man doing an axle grind on a piece of pipe in a park.

a boy on a skateboard rail on a skateboard

GENERATED CAPTIONS

a man is riding a skateboard on a skateboard .

Figure 23: In this example, the model misrecognized the rail as another skateboard, subsequently it says that the man is riding a skateboard on a skateboard. However, it's reasonable because the rail looks like a big skateboard in its shape.

9 Team Contributions

TYPE	TASK NAME	Colin	Jerry	Bingqi
Report	Title	0.5		0.5
Report	Abstract	0.7		0.3
Report	Introduction	0.6		0.4
Report	Related Work	0.5		0.5
Report	Architecture	0.2	0.8	
Report	Discussion	0.2	0.8	
Report	BLEU	0.2	0.8	
Report	Temperature	1		
Report	Tuning	1		
Report	Arch2		1	
Report	Caption Generation	1		
Code	Train Pipeline	1		
Code	Train Pipeline Debug	0.7	0.3	
Code	Model Pipeline		1	
Code	Model Pipeline Debug	0.4	0.6	
Code	LSTM		1	
Code	LSTM Debug	1		
Code	RNN	1		
Code	RNN Debug	1		
Code	Arch2		1	
Code	Arch2 Debug	0.5	0.5	
Code	Readme	0.5	0.5	
Experimentation	Caption Generation	0.7	0.3	
Experimentation	Temperature	1		
Experimentation	Architecture 2		1	
Experimentation	Fine Tuning Encoder	0.3	0.7	
Experimentation	Size Tuning			1

Table 9: Team Contributions. Each row represents a portion of the assignment. Numbers in each entry represent individual percent contribution into corresponding portions.

References

[1] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," *arXiv:1405.0312 [cs]*, Feb. 2015. arXiv: 1405.0312.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[3] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[4] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," *Advances in Neural Information Processing Systems*, vol. 21, pp. 545–552, 2008.

[5] X. Li and X. Wu, "Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition," *arXiv:1410.4281 [cs]*, May 2015. arXiv: 1410.4281.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *arXiv:1409.3215 [cs]*, Dec. 2014. arXiv: 1409.3215.