How to Win any Captioning Challenge: Using Encoder-Decoder Architecture to Caption Images with Convolutional and Recurrent Networks

Shubham Kulkarni
CSE Department
UC San Diego
La Jolla, CA 92092
skulkarn@ucsd.edu

Michael Wu
CSE Department
UC San Diego
La Jolla, CA 92092
m5wu@ucsd.edu

Abstract

We present a more advanced and accurate method of classifying images using a convolutional neural network. The aim of our project was to tune a model that could properly caption images.

The dataset used in our experiment was a portion of the Common Objects in Context (COCO) set containing 85,000 images and 425,000 captions.

The experiment was separated into 3 main stages: a baseline LSTM model, an RNN model, an LSTM model with tuned dimensions, and finally an LSTM model that passed the images at every timestep rather than just the first.

The first was the baseline model. This used ResNet50 to encode images into an lower dimension representation, and then ran them as well as caption data through a teacher forced LSTM to train the model. During testing, we used the model to generate captions on its own, and score generated captions against expected captions using BLEU scores - a common NLP measure of evaluating output quality. Our baseline LSTM model obtained a BLEU1 score of 67.05 and BLEU4 score of 7.78.

Next, we tried replacing the LSTM in our encoder with a traditional RNN. Keeping all other hyperparameters constant, this new model attained a test BLEU1 score of 65.31 and BLEU4 score of 7.12.

We then finetuned the embedding and hidden sizes of our LSTM decoder to achieve a BLEU1 score 67.91 and BLEU4 score of 9.24 with an embedding size and hidden size of 1024 neurons each.

We then used a new model where the input image was passed at each timestep, keeping the dimensions of 1024 neurons on both the embedding and hidden units. This model acheived a BLEU1 score 68.22 and BLEU4 score of 9.16.

1 Introduction

We used a series of advanced machine learning techniques to build and train our model.

1.1 Dataset

As mentioned in the Abstract, our dataset was a subset of the COCO image repository, using 85,000 images and 425,000 captions (5 captions per image). The dataset was split into a training set of 82,000 images and a test set of 3,000 images.

The images themselves are non-uniform in size. Some of the images are vertical while others are horizontal. Additionally, they were not consistent in terms of input channel count - most images are RGB, with a few grayscale images.

We dealt with these issues by reshaping them to all be 256×256 square images as ImageNet models expect, and changing them to all be 3 input channel RGB images (grayscale images are replicated across all 3 channels). We also normalized our images with the ImageNet mean and standard deviations (mean of $[0.485, 0.456, 0.406]$ and standard deviation of $[0.229, 0.224, 0.225]$ across the three channels.

1.2 Baseline Model

The baseline design for our neural network has two main components: the encoder and the decoder.

The encoder is an extension of ResNet50, a pre-trained convolutional neural network. We replaced the final layer with a trainable linear layer. We used the encoder to embed images into a lower dimensional representation. Words in captions were embedded into the same dimensional space.

The decoder is an LSTM module, a type of recurrent neural network. LSTMs have the ability to retain their internal hidden state well using gates to determine when the memory is cleared and overwritten.

All models were trained using a concept called teacher forcing, which uses the target values as input in the next timestep instead of the output our model generates. It helps recurrent models converge faster by avoiding a double penalty for incorrect words.

We generated captions in two ways: deterministically, where we take the argmax over the output, and stochastically, where we sampled from a probability distribution generated by applying a softmax to the outputs.

1.3 Vanilla RNN network

In this stage, we replaced the LSTM in our decoder with a traditional RNN. We did this to determine whether the gate on the LSTM was having a significant effect on accuracy. Since our RNN got only slightly lower results, we feel that the gated cell state memory gives an advantage, but not as significant as we originally thought.

1.4 Tuned Model Dimensions

We also tried finetuning the embedding and hidden sizes of our LSTM decoder. This helped us achieve a BLEU1 score 67.91 and BLEU4 score of 9.24 with an embedding size and hidden size of 1024 nodes each. We think this did well because it gave the model more information to work with at the embedding layer, and the hidden layer had a better ability to discriminate.

1.5 New Architecture

Finally, we tried passing the image at every timestep to the decoder to see if that would improve our model's accuracy by making more information accessible at every stage. This gave us a BLEU1 score of 68.22 and BLEU4 score of 9.16. We believe that the model did better because of the access to the image embedding at all timesteps, making it easier to "look" at the image again for the next word's prediction.

2 Related Work

To better understand the pre-trained models we were using, we found some background information on them. We learned about ResNet50 from the original research paper submitted by Microsoft Research [1].

We also looked at the various architectures and skimmed a paper titled Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) [2] that illustrated the various architectures and how they differ with some good visualizations showing what functions were applied when. Some useful slides on these concepts were lecture 10, slides 28-32.

The first really useful paper was Long-term Recurrent Convolutional Networks for Visual Recognition and Description [4]. This paper from Jeff Donahue et al used an CNN + LSTM combination to generate image captions, very similarly to our PA. They applied this combined architecture to three different tasks: activity recognition where they received a sequential input and generated a single output, image description which involves a single input and sequential output (like our PA), and video description which involves both sequential input and sequential output. When working with image description, they provided the image to the model at each timestep like Architecture 2. They used a deeper model for the recurrent part of the architecture - 4 LSTMs of 1000 hidden units each. They acheived some interesting results: on the COCO2014 test set they achieved a BLEU1 score of 62 (lower than our best score) but a BLEU4 score of 22 (considerably higher than our score). Their selected generated captions also look quite reasonable.

The second interesting paper was Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN) [3]. This paper from Junhua Mao et al used an CNN + RNN multimodal combination to generate image captions. However, their recurrent network was not fed with image data at any timestep - the image data is only used at the final multimodal layer prior to softmax. Some key differences between this paper and our model were that they used an RNN as opposed to an LSTM. Addditionally, they used ReLU activation rather than the regular sigmoidal activation for the RNN itself. This helped them combat any exploding gradient issues. Additionally, the multimodal layer got the input of the RNN, the output of the RNN, and the output of the convnet to predict a word, kind of like a residual net gets both the output and input of the layer before it as an identity function. Finally, their model was provided both the RNN output and the CNN encoded image at each timestep for prediction. They acheived some interesting results: on the COCO2014 test set they achieved a BLEU1 score of 67 (very close to our best score) and a BLEU4 score of 25 (considerably higher than our score).

PyTorch documentation proved extremely useful for us to understand where in the training code we could make changes to most effectively improve our model. This included the docs for RNN / LSTM, the docs for PyTorch's Embedding module, the docs for torchvision's pretrained ResNet50 model and how to use them, and of course the docs for nltk's sentence bleu score utilities.

We used official documentation from Matplotlib's pyplot and imshow to help us generate the plots for loss and accuracy results, as well as to visualize images to see if our generated captions were reasonable.

3 Models

Our models can be separated into two main components: an encoder and a decoder.

All of our models use the same encoder: a convolutional neural network that embeds input images into a lower dimensional space. We use the pre-trained model ResNet50 to achieve this task; however, we replace the final softmaxed output layer with a trainable linear layer. We wanted to use the encoder as a tool to represent images in a lower dimensional feature space, so we didn't want the final output layer to be softmaxed between 0 and 1. We encoded words using PyTorch's built in Embedding module. It takes words and turns them into one-hot encoded vectors with the same dimensionality as the vocabulary size, then applies a trainable linear layer to map them into a lower dimensional feature space. The dimensions of ResNet50's output and the Embedding's output were identical so the model could take in either inputs interchangeably.

Where these models differ is the implementation of the decoder. The aim of the decoder is to use the feature vector (passed from the encoder) to generate a caption for the input image one word at a time. We used teacher forcing to speed up decoder training. Teacher forcing uses the target word as the input of the next timestep instead of the output generated by our model; in other words, $x_{t+1} = \text{target}_t$ instead of y_t.

The two models we built implement the decoder network in two unique ways to see which one can generate more accurate and understandable English-friendly captions.

3.1 Vanilla RNN

This model uses a vanilla RNN as the decoder module. RNNs work by combining the hidden state from the previous timestep with the input at the current timestep, and then applying a non-linear activation to the sum of these two. Our RNN used the sigmoid activation. This dependence on the last hidden state allows the model to make predictions that depend on the previous state, which is very similar to how languages construct phrases and sentences.

Our best RNN model had the following default hyperparameters:

- Embedding size = 300
- Hidden size = 512
- Epochs = 10
- Learning rate = $0.0005 = 5 \times 10^{-4}$
- Batch size = 64

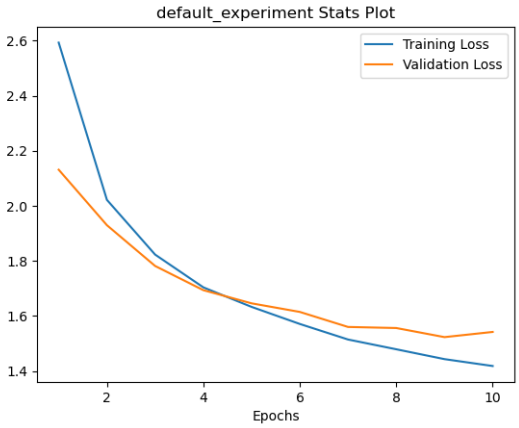


Figure 1: Cross Entropy loss for RNN model (Teacher Forcing). Hyperparameters: embedding size = 300, hidden size = 512, 10 epochs, learning rate = 5e-4, batch size = 64, Adam optimizer with default β.

The model got a BLEU1 score of 65.31 and BLEU4 score of 7.13, which is decent for the relatively small training set it received.

This model had rather solid performance. From how the validation loss curve decreases pretty consistently with training loss at each epoch, it did not overfit the training data. The learning rate could likely have been a little higher, since the training loss is decreasing in more of a linear fashion than an exponential fashion as it should.

3.2 Baseline Model - LSTM

For this experiment, the decoder is an LSTM model that uses the encoded feature vector to generate a caption for the input image one word at a time. The LSTM model has some marked advantages

over vanilla recurrent nets because of its use of gates to modulate its internal cell memory. The gates also help it avoid exploding and vanishing gradient issues.

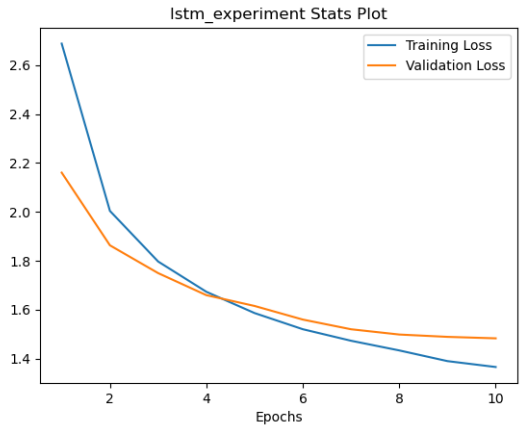


Figure 2: Cross Entropy loss for LSTM model (Teacher Forcing). Hyperparameters: embedding size = 300, hidden size = 512, 10 epochs, learning rate = 5e-4, batch size = 64, Adam optimizer with default β.

Our best LSTM model had the following default hyperparameters:

- Embedding size = 300
- Hidden size = 512
- Epochs = 10
- Learning rate = $0.0005 = 5 \times 10^{-4}$
- Batch size = 64

As we can see in the graphs, the baseline LSTM and vanilla RNN models share similar shapes of learning curve; however, the baseline learning curve seems to be shifted down (i.e., we can imagine vanilla RNN loss = baseline LSTM loss + some positive constant k). In addition, the RNN model took 10 full epochs to get to a validation loss of 1.52, whereas the baseline LSTM model obtained that loss in just 7 epochs. This means that the vanilla RNN likely needs more epochs to achieve a satisfactory performance compared to the baseline model.

4 Results for Best Models

The following table shows our final loss results after 10 epochs of training (for each model).

Table 1: Final Loss Values

Loss\Model	**Vanilla RNN**	**LSTM**
Training	1.418	1.366
Validation	1.542	1.483
Test Cross-Entropy	1.532	1.478

Much like the differences in the loss learning curves, the baseline LSTM model had a slightly lower cross-entropy loss on the test set than the vanilla RNN. The LSTM model was likely better at making predictions because its complex memory management technique allowed it to better remember what salient features the image had, which allowed it to build more accurate and useful captions using that feature representation.

5 Deterministic Caption Generation

Deterministic caption generation takes the highest-valued output from the decoder to generate captions at each timestep. Using this form of caption generation:

- The baseline LSTM model gave a BLEU1 score of 67.21 and BLEU4 score of 7.85.
- The vanilla RNN model gave a BLEU1 score of 65.55 and BLEU4 score of 7.23.

In both scoring criteria, our baseline LSTM model performed better than the vanilla RNN model. Because validation loss, test loss, and overall performance tend to be closely correlated when examining machine learning models, this performance disparity likely has the same causes as the differences we have observed thus far in learning curves and test cross entropy loss. It may just be another effect of LSTM's ability to retain more intricate information about the image better to generate higher-scoring captions.

6 Temperature Experiments

In this experiment, we generated captions stochastically by sampling from a probability distribution produced from a weighted softmax of the decoder outputs:

$$y^j = \frac{\exp(o^j/\tau)}{\sum_n \exp(o^n/\tau)}$$

where $\tau \geq 0$ is the "temperature" value that determines the variance in the probability distribution. Smaller values of τ causes this process to behave like deterministic caption generation, whereas larger values of τ makes the probability distribution more and more uniform.

Table 2: BLEU Scores by Temperature

Temperature	**BLEU1 score**	**BLEU4 score**
0.01	**67.23**	7.84
0.05	67.19	**7.91**
0.1	67.04	7.78
0.2	66.53	7.57
0.7	57.85	4.29
1.0	45.50	2.18
1.5	18.51	0.99
2.0	6.32	0.49

We used a variety of temperatures ranging from very small ($\tau = 0.01$) all the way to very large ($\tau = 2.0$) to see how different temperatures would affect caption generation. For the most part, smaller temperatures led to better results, with 0.01 and 0.05 being the best for BLEU1 and BLEU4 scores respectively.

We believe this is because the test set is actually very similar to the training set that we used. Usually, low temperature generation does not generalize well to new data when the model is not very sure of its predictions, and higher temperatures are used to artificially flatten the model's predictions to be more uniform.

However, since our datasets were so similar, more deterministic forms of generation (i.e., lower temperatures) were able to generalize to the test set very well. High temperature captions were only able to make very inaccurate captions because of the flattened curve essentially acting like a random uniform distribution.

We found that very low temperatures (between $\tau = 0.01$ and $\tau = 0.05$) worked marginally better than deterministic generation. This could be because deterministic generation does not account for how sure the model is of its prediction, whereas stochastic sampling allows us to use the sureness of a model to decide which word to predict next.

7 LSTM Finetuning

Our next experiment was to tune the size of the LSTM model to obtain a better result.

Our best finetuned LSTM model had the default hyper parameters of the following:

- Embedding size = 1024
- Hidden size = 1024
- Epochs = 10
- Learning rate = $0.0005 = 5 \times 10^{-4}$
- Batch size = 64

We chose to increase the embedding size in the hopes that it would deliver additional information about the image to the model, which the model could then use to make more accurate captions.

We also increased the hidden unit size so that there would be more information preserved from the last timestep. This would hopefully allow the model to have a better memory of what words it had previously predicted, and make more logically sound captions.

The model quickly converged to a validation loss of around 1.4, and then hovered there while the training loss kept dropping.

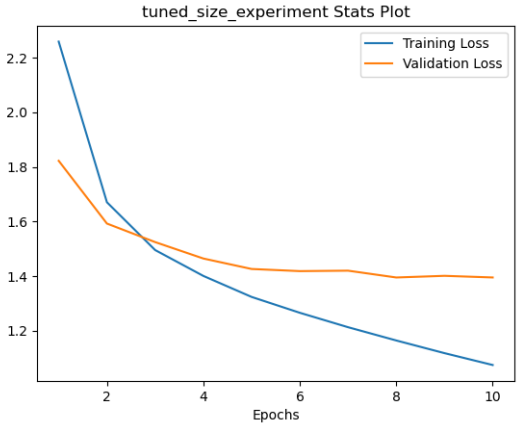


Figure 3: Cross Entropy loss for Finetuned LSTM model (Teacher Forcing). Hyperparameters: embedding size = 1024, hidden size = 1024, 10 epochs, learning rate = 5e-4, batch size = 64, Adam optimizer with default β.

The following table shows this model's end performance compared to that of the baseline LSTM model:

Table 3: Final Scores: Baseline vs Finetuned LSTM

Statistic\Model	**Baseline LSTM**	**Finetuned LSTM**
Test Loss	1.478	1.38
BLEU1	67.05	67.92
BLEU4	7.78	9.24

Compared to the baseline model, this model did a lot better. BLEU1 jumped by almost a full point, and BLEU4 jumped by 1.5 points. The cross entropy loss also dropped by 0.1, a 6% improvement

over the baseline. We believe that our goal of giving the model more information to work with was quite successful since this model performed so much better than the baseline.

We did try a variety of other values. For the most part, increasing the embedding size too much actually lowered the model's performance and also came with a significant increase in training time, likely due to too many features being given to the model and not enough training points.

8 New Architecture

We also improved the LSTM decoder by including the embedded image in the input at every time step. Adding the input image allowed the decoder to double-check that the next generated word fits into the context of the image. This new architecture made additional small improvements in performance.

We ran this new architecture with both the original hyperparameters and with our tuned set of dimensions.

This time, the input to the model was actually doubled at each time step, even though the embedding size was the same. This is because we concatenated the teacher-forced input with the image before passing it to the LSTM decoder.

We also padded all the captions at the very beginning since the model now expected both an image and a word at every timestep.

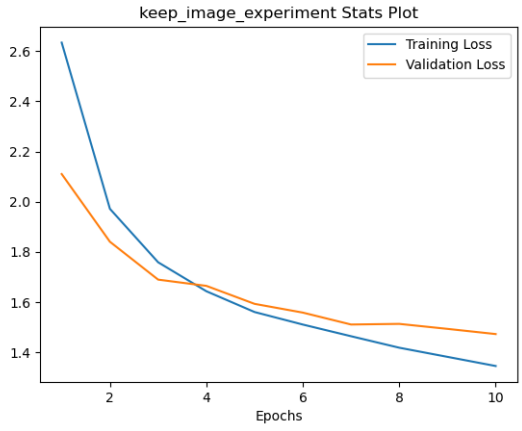


Figure 4: Cross Entropy loss for LSTM model with Image Repetition (Teacher Forcing). Hyperparameters: embedding size = 300, hidden size = 300, 10 epochs, learning rate = 5e-4, batch size = 64, Adam optimizer with default β.

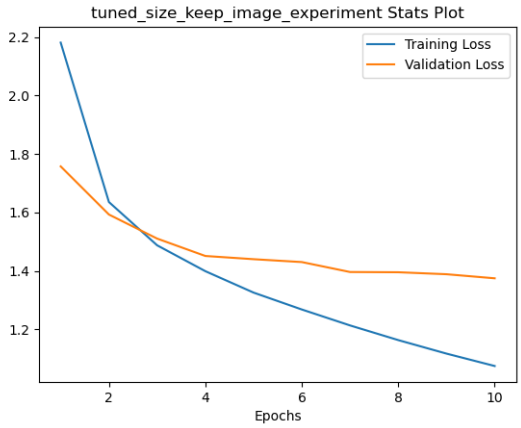


Figure 5: Cross Entropy loss for Finetuned LSTM model with Image Repetition (Teacher Forcing). Hyperparameters: embedding size = 1024, hidden size = 1024, 10 epochs, learning rate = 5e-4, batch size = 64, Adam optimizer with default β.

From the loss graphs, we see that they actually look very similar to their first architecture counterparts. The final results are do exhibit some differences though:

Table 4: Final Performance of New LSTM Architecture

Statistic\Model	**Baseline**	**Boosted (no tuning)**	**Baseline (with tuning)**	**Boosted (with tuning)**
Test Loss	1.478	1.473	1.38	1.395
BLEU1	67.05	68.17	67.92	68.22
BLEU4	7.78	8.43	9.24	9.16

In the table, Architecture 1 is Baseline and Architecture 2 is Boosted. Without finetuning, the loss values are very close (1.473 vs 1.478), but the BLEU scores are much better with the image provided at each time step - BLEU1 went up by more than 1 point and BLEU4 went up by 0.7 points. With finetuning, the loss value with architecture 1 is actually lower, but the BLEU1 score is better for architecture 2.

Overall, it seems like this architecture performs very similarly to architecture 1, with higher BLEU scores even if loss values are not significantly improved.

It seems like providing the image at each timestep is helping the model make better predictions by using the image directly in word generation to make sure each word is contextually consistent with its place in the image.

9 Generated Captions Visualization

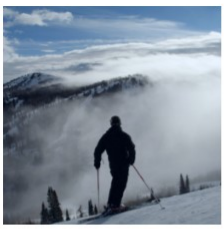
For this experiment, we used our best model (LSTM model with finetuned dimensions) to generate captions for images in the test set. We're going to use some of these images to explore what this model does well, as well as address some of the shortcomings of the model.

9.1 Good captions

First let's look at some of the successful captions our best model generated.

```
Actual captions:
 [['a', 'man', 'wearing', 'a', 'wet', 'suit', 'riding', 'the', 'wave'], ['a', 'person', 'riding', 'a', 'wave', 'on', 'top',
'of', 'a', 'surfboard', '.'], ['a', 'surfer', 'rides', 'a', 'small', 'wave', 'in', 'the', 'ocean', '.'], ['a', 'man', 'ridin
g', 'a', 'wave', 'on', 'a', 'blue', 'ocean'], ['a', 'young', 'man', 'ridding', 'a', 'small', 'wave', 'on', 'a', 'surfboard',
'.']]
Predicted caption:
 ['a', 'man', 'riding', 'a', 'wave', 'on', 'a', 'surfboard', '.']
score: (100.0, 100.0)

6 / 235 loss: 1.31 bleu1: 67.88 bleu4: 7.98
```



```
Actual captions:
 [['a', 'man', 'riding', 'skis', 'down', 'a', 'snow', 'covered', 'mountain', '.'], ['a', 'man', 'with', 'two', 'poles', 'sky
ing', 'on', 'snow'], ['a', 'skier', 'standing', 'in', 'the', 'snow', 'looking', 'at', 'the', 'view'], ['the', 'person', 'i
s', 'skiing', 'down', 'the', 'hill', 'and', 'it', 'is', 'foggy', 'outside', '.'], ['the', 'skier', 'is', 'at', 'the', 'top',
'of', 'the', 'slope', '.']]
Predicted caption:
 ['a', 'man', 'riding', 'skis', 'down', 'a', 'snow', 'covered', 'slope', '.']
score: (100.0, 71.42857142857143)
```

Figure 6: First 2 good caption predictions.

It's interesting that both of these images have relatively small subjects compared to the background, and yet the decoder was able to identify the correct action in the image. For instance, the decoder correctly predicted that it was a man riding a wave, not just that there was a wave in the image.

```
Actual captions:
 [['a', 'man', 'wearing', 'a', 'usa', 'shirt', 'skiing', 'in', 'the', 'sun', '.'], ['a', 'person', 'on', 'skis', 'in', 'a',
'race', 'with', 'usa', 'on', 'his', 'shirt', '.'], ['a', 'professional', 'skiier', 'moves', 'swiftly', 'along', 'the', 'cour
se', '.'], ['a', 'professional', 'skier', 'competing', 'on', 'a', 'winter', 'olympics', 'ski', 'slope'], ['a', 'man', 'ridin
g', 'skis', 'down', 'a', 'snow', 'covered', 'slope', '.']]
Predicted caption:
 ['a', 'man', 'riding', 'skis', 'down', 'a', 'snow', 'covered', 'slope', '.']
score: (100.0, 100.0)
```

```
Actual captions:
 [['a', 'road', 'that', 'has', 'a', 'bunch', 'of', 'bikers', 'driving', 'down', 'it'], ['men', 'who', 'are', 'on', 'police',
'motorcycles', 'driving', 'down', 'the', 'street', '.'], ['a', 'number', 'of', 'people', 'riding', 'motorcycles', 'on', 'a',
'city', 'street'], ['a', 'large', 'group', 'of', 'motorcycles', 'driving', 'in', 'a', 'parade', 'with', 'parade', 'onlooker
s', '.'], ['a', 'group', 'of', 'police', 'officers', 'on', 'motorcycles', 'in', 'a', 'parade', '.']]
Predicted caption:
 ['a', 'group', 'of', 'people', 'riding', 'motorcycles', 'on', 'a', 'street', '.']
score: (100.0, 42.857142857142854)
```

```
Actual captions:
 [['a', 'couple', 'of', 'people', 'flying', 'kites', 'on', 'top', 'of', 'a', 'beach', '.'], ['some', 'people', 'are', 'flyin
g', 'a', 'kite', 'at', 'the', 'beach', '.'], ['people', 'standing', 'and', 'some', 'sitting', 'on', 'the', 'sand', 'of',
'a', 'beach', 'with', 'a', 'kite', 'flying', 'and', 'water', 'in', 'the', 'distance', '.'], ['a', 'couple', 'of', 'people',
'fly', 'a', 'kite', 'on', 'the', 'beach', '.'], ['a', 'couple', 'flies', 'a', 'kite', 'on', 'a', 'large', 'beach', '.']]
Predicted caption:
 ['a', 'group', 'of', 'people', 'on', 'a', 'beach', 'with', 'a', 'kite', '.']
score: (90.9090909090909, 25.0)
```

Figure 7: 3 more good caption predictions.

Next up, we see the model taking a variety of complex backgrounds and successfully identifying what images it is seeing. We want to point out how the model was able to see the motorcycles in the second image and the kite in the third image.

The second image was very busy, with many different details that could have been identified instead, but the model correctly focused on the motorcycles for the caption.

The third image also had a risk of not noticing the kite at all since it is such a small component of the image, so we were thoroughly impressed that the model was able to pick it up. Additionally, the lighting of the image makes it a little difficult to identify the setting as a beach (the sand is underexposed and the sky is washed out), but the model was still able to identify the beach, likely from the way the water intersects with non-water portions of the image.

```
Actual captions:
 [['a', 'herd', 'of', 'sheep', 'grazing', 'on', 'a', 'hill', 'side', 'near', 'a', 'tree', '.'], ['several', 'sheep', 'standi
ng', 'in', 'the', 'grass', 'near', 'a', 'tree', '.'], ['a', 'couple', 'of', 'sheep', 'graze', 'on', 'some', 'grass'], ['a',
'small', 'herd', 'of', 'sheep', 'grazing', 'in', 'a', 'grassy', 'field', '.'], ['while', 'in', 'the', 'immediate', 'foregrou
nd', 'juts', 'a', 'gnarled', 'tree', 'branch', ',', 'the', 'majority', 'of', 'the', 'view', 'consists', 'of', 'a', 'an', 'ex
panse', 'of', 'short', 'grass', 'dotted', 'with', 'a', 'few', 'longer', 'tufts', 'and', 'a', 'number', 'of', 'scattered',
',', 'grazing', 'sheep', '.']]
Predicted caption:
 ['a', 'herd', 'of', 'sheep', 'grazing', 'in', 'a', 'field', '.']
score: (100.0, 66.66666666666666)
```

```
Actual captions:
 [['commercial', 'jet', 'passing', 'overhead', 'on', 'bright', 'cloudless', 'sky', '.'], ['a', 'large', 'aircraft', 'in', 't
he', 'blue', 'sky', 'by', 'itself'], ['a', 'ground', 'view', 'of', 'an', 'airplane', 'in', 'the', 'sky', '.'], ['a', 'jet',
'plane', 'is', 'flying', 'across', 'the', 'sky', '.'], ['a', 'jetliner', 'flying', 'through', 'a', 'light', 'blue', 'sky',
'.']]
Predicted caption:
 ['a', 'large', 'jetliner', 'flying', 'through', 'a', 'blue', 'sky', '.']
score: (100.0, 16.666666666666668)
```

```
Actual captions:
 [['a', 'bird', 'that', 'is', 'perched', 'on', 'a', 'branch', '.'], ['a', 'small', 'bird', 'perched', 'on', 'a', 'branch',
'with', 'leaves'], ['a', 'side', 'view', 'of', 'a', 'bird', 'sitting', 'on', 'a', 'branch', '.'], ['a', 'small', 'bird', 'si
ts', 'on', 'a', 'branch', 'in', 'a', 'tree', '.'], ['a', 'small', 'bird', 'perched', 'on', 'top', 'of', 'a', 'wooden', 'bran
ch', '.']]
Predicted caption:
 ['a', 'bird', 'perched', 'on', 'a', 'branch', 'in', 'a', 'tree', '.']
score: (100.0, 85.71428571428571)
```

Figure 8: 3 more good caption predictions.

We liked these images because they showcased the model's ability to pick up minute details as well as its ability to generalize to new contexts.

First, we have the herd of sheep. The image is almost entirely taken up by the tree or the hill, and the sheep are very small spots in a very large image. The model still correctly figured out that the small white dots were important features of the image and correctly generated a caption that included discussion of the sheep in the field.

The other impressive captioning was the bird. This is an extreme closeup of the bird, and the scope of the image does not make the background easily identifiable as a tree. However, the model was able to use context clues like bird in a green-brown environment to figure out that the bird was on a tree branch.

```
Actual captions:
 [['a', 'cat', 'is', 'sitting', 'on', 'a', 'white', 'toilet', '.'], ['a', 'cat', 'sitting', 'on', 'a', 'toilet', 'with', 'th
e', 'lid', 'closed', '.'], ['black', 'and', 'white', 'photograph', 'of', 'a', 'cat', 'on', 'a', 'toilet', '.'], ['a', 'blac
k', 'and', 'white', 'photo', 'of', 'a', 'small', 'cat', 'sitting', 'on', 'a', 'toilet', '.'], ['a', 'tabby', 'cat', 'sittin
g', 'on', 'the', 'toilet', 'in', 'a', 'bathroom', '.']]
Predicted caption:
 ['a', 'cat', 'sitting', 'on', 'a', 'toilet', 'in', 'a', 'bathroom', '.']
score: (100.0, 71.42857142857143)
```

```
Actual captions:
 [['the', 'clock', 'shown', 'above', 'has', 'someone', "'s", 'name', 'on', 'it', '.'], ['a', 'clock', 'protruding', 'from',
'the', 'side', 'of', 'a', 'building', '.'], ['a', 'building', 'with', 'a', 'clock', 'coming', 'out', 'of', 'it'], ['a', 'lar
ge', 'clock', 'hanging', 'off', 'the', 'side', 'of', 'a', 'building', '.'], ['a', 'clock', 'hanging', 'from', 'a', 'store',
'with', 'letters', 'instead', 'of', 'numbers', 'on', 'it', '.']]
Predicted caption:
 ['a', 'clock', 'on', 'the', 'side', 'of', 'a', 'building', '.']
score: (100.0, 50.0)
```

Figure 9: Final 2 good caption predictions.

Finally, we come to these last two good captions. We felt these captions both showed the model's ability to generalize. The images are both grayscale, which makes it especially impressive since only a very small portion (less than 0.1%) of the training set images was grayscale. It was interesting to see how the model was able to pick out salient details like the cat, even when it did not have the additional information given from color contrast.

9.2 Bad captions

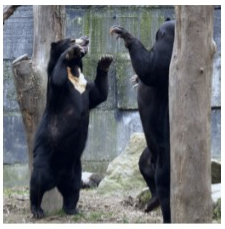
Now let us examine some of the incorrect captions our best model generated.

```
Actual captions:
 [['a', 'silver', 'motorcycle', 'with', 'many', 'lights', 'parked', 'on', 'the', 'street', '.'], ['a', 'flashy', 'motorcycl
e', 'parked', 'on', 'the', 'side', 'of', 'the', 'street', '.'], ['old', 'style', 'motorcycle', 'with', 'many', 'lights', 'pa
rked', 'on', 'the', 'street', '.'], ['a', 'motorcycle', 'with', 'more', 'than', 'the', 'usual', 'number', 'of', 'headlight
s', '.'], ['a', 'black', '-and-white', 'photo', 'of', 'a', 'motorcycle', 'with', 'many', 'headlights', '.']]
Predicted caption:
 ['a', 'red', 'fire', 'hydrant', 'sitting', 'on', 'a', 'sidewalk', 'next', 'to', 'a', 'building', '.']
score: (30.769230769230766, 1.0000000000000004)
```

Figure 10: 1 bad caption prediction.

This first bad caption is quite self explanatory - there is no red in this image, much less a whole fire hydrant.

```
Actual captions:
 [['a', 'couple', 'of', 'men', 'playing', 'a', 'game', 'of', 'frisbee', '.'], ['two', 'men', 'on', 'opposite', 'teams', 'jum
ping', 'for', 'a', 'frisbee', '.'], ['two', 'men', 'are', 'reaching', 'to', 'catch', 'a', 'frisbee', '.'], ['two', 'men', 'p
laying', 'a', 'game', 'of', 'frisbee', 'in', 'a', 'field', '.'], ['the', 'men', 'are', 'playing', 'a', 'game', 'of', 'frisbe
e', 'on', 'the', 'grassy', 'field', ',']]
Predicted caption:
 ['a', 'man', 'is', 'holding', 'a', 'baseball', 'bat', 'at', 'a', 'baseball', 'game', '.']
score: (33.33333333333333, 1.1111111111111112)
```



```
Actual captions:
 [['a', 'couple', 'of', 'black', 'bears', 'standing', 'next', 'to', 'two', 'trees', '.'], ['two', 'adult', 'black', 'bears',
'are', 'standing', 'on', 'their', 'hind', 'legs', '.'], ['a', 'couple', 'of', 'bears', 'that', 'are', 'standing', 'up'], ['t
wo', 'black', 'bears', 'stand', 'on', 'their', 'hind', 'legs', '.'], ['a', 'picture', 'of', 'two', 'black', 'bears', 'walkin
g', 'to', 'each', 'other', 'claws', 'up', '.']]
Predicted caption:
 ['a', 'bear', 'is', 'standing', 'in', 'the', 'grass', 'with', 'a', 'large', 'elephant', '.']
score: (25.0, 1.1111111111111112)
```

```
Actual captions:
 [['a', 'happy', 'woman', 'about', 'to', 'eat', 'a', 'slice', 'of', 'pizza', '.'], ['a', 'woman', 'smiles', 'as', 'she', 'ho
lds', 'a', 'plate', 'of', 'food'], ['an', 'asian', 'woman', 'smiles', 'as', 'she', 'received', 'pizza'], ['woman', 'smilin
g', 'receiving', 'a', 'plate', 'with', 'a', 'slice', 'of', 'pizza', '.'], ['the', 'older', 'woman', 'smiles', 'as', 'she',
'holds', 'a', 'plate', 'with', 'a', 'slice', 'of', 'pizza', '.']]
Predicted caption:
 ['a', 'man', 'is', 'holding', 'a', 'hot', 'dog', 'in', 'a', 'bun', '.']
score: (27.27272727272727, 1.2500000000000007)
```

Figure 11: 3 bad caption predictions.

We found the first caption interesting because we had actually seen an image with a baseball player and it predicted the same caption. We feel that maybe this was a sign that the model was not learning very well as much as it was just memorizing that certain features map to certain words.

The second caption is wrong because there is no elephant. We thought that the model might have seen the gray in the wall and concluded that it was an elephant.

We found the third caption interesting because it might represent a bias in our training set. The lady had very short hair, which the model may have learned to identify as a man. A more diverse training set could combat this, at least in the gender identification.

```
Actual captions:
 [['there', 'are', 'there', 'men', 'standing', 'on', 'top', 'of', 'surf', 'boards'], ['three', 'people', 'in', 'suits', 'ar
e', 'standing', 'on', 'top', 'of', 'surf', 'boards', ',', 'in', 'the', 'sand', '.'], ['three', 'businesspeople', 'stand', 'o
n', 'surfboards', 'on', 'the', 'beach'], ['two', 'men', 'and', 'a', 'woman', 'wearing', 'suits', 'on', 'surf', 'boards', 'i
n', 'sand'], ['three', 'people', 'are', 'posing', 'on', 'surf', 'boards', 'on', 'a', 'beach', '.']]
Predicted caption:
 ['a', 'man', 'in', 'a', 'red', 'shirt', 'is', 'playing', 'a', 'game', 'of', 'frisbee', '.']
score: (30.769230769230766, 1.0000000000000004)
```

```
Actual captions:
 [['many', 'people', 'holding', 'umbrellas', 'walking', 'across', 'a', 'street'], ['a', 'bunch', 'of', 'people', 'walking',
'on', 'the', 'street', 'with', 'umbrellas', '.'], ['people', 'crossing', 'the', 'street', 'in', 'a', 'cross', 'walk', 'holdi
ng', 'umbrellas', 'in', 'the', 'city', '.'], ['shoppers', 'carrying', 'umbrellas', 'against', 'the', 'sun', 'in', 'an', 'ori
ental', 'city'], ['the', 'people', 'are', 'walking', 'the', 'streets', 'with', 'their', 'umbrellas', 'up', '.']]
Predicted caption:
 ['a', 'group', 'of', 'people', 'standing', 'next', 'to', 'a', 'man', 'in', 'a', 'park', '.']
score: (35.61388764008908, 0.9259610786423165)
```

```
Actual captions:
 [['a', 'woman', 'holding', 'a', 'hotdog', 'and', 'reaching', 'for', 'ketchup', 'at', 'a', 'food', 'cart', '.'], ['woman',
'at', 'a', 'hot', 'dog', 'cart', 'holding', 'a', 'hot', 'dog', 'and', 'a', 'ketchup', 'bottle'], ['two', 'young', 'women',
'order', 'hotdogs', 'from', 'a', 'vendor', 'in', 'this', 'black', 'and', 'white', 'street', 'scene', '.'], ['two', 'women',
'ordering', 'hot', 'dogs', 'at', 'a', 'hot', 'dog', 'stand', '.'], ['a', 'woman', 'buying', 'something', 'from', 'a', 'foo
d', 'cart', 'on', 'a', 'sidewalk', '.']]
Predicted caption:
 ['a', 'group', 'of', 'people', 'standing', 'around', 'a', 'table', 'with', 'a', 'cake', '.']
score: (33.33333333333333, 1.1111111111111112)
```

Figure 12: 3 bad caption predictions.

The first caption is interesting because that looks like a crazy image that wouldn't reasonably encountered in the real world. Perhaps the lack of training data that looked like that meant the model did not know how to interpret the situation and made a very bad caption for it. It looks like the model is not generalizing to totally new images very well.

The second caption is wrong because there is no park. The model might have picked up on the trees at the side of the road and extrapolated those to be part of a park.

The third caption is interesting because this is another occurrence of a caption that was correct for a different image being predicted again, now for an incorrect image. This seems to reinforce our belief that perhaps the model just links features to key words rather than considering the image as a whole.

```
Actual captions:
 [['an', 'elephant', 'raises', 'its', 'trunk', 'to', 'grab', 'some', 'twigs'], ['an', 'elephant', 'grabbing', 'a', 'branch',
'with', 'his', 'trunk', '.'], ['an', 'elephant', 'with', 'an', 'open', 'mouth', 'lifts', 'his', 'trunk', '.'], ['an', 'eleph
ant', 'using', 'trunk', 'to', 'eat', 'leaves', 'on', 'a', 'tree', '.'], ['an', 'elephant', 'rubbing', 'up', 'against', 'a',
'tree', 'and', 'pulling', 'on', 'branches', '.']]
Predicted caption:
 ['a', 'giraffe', 'standing', 'in', 'the', 'grass', 'near', 'a', 'fence', '.']
score: (20.0, 1.4285714285714295)
```

```
Actual captions:
 [['a', '787', 'airplane', 'is', 'landing', 'at', 'the', 'airport', '.'], ['an', 'airplane', 'is', 'getting', 'ready', 'to',
'land', 'at', 'the', 'airport', '.'], ['a', 'airplane', 'in', 'the', 'sky', 'flying', 'above', 'a', 'building'], ['blue', 'a
nd', 'white', 'airplane', 'flying', 'over', 'radio', 'tower'], ['a', 'blue', 'a', 'white', 'plane', 'flying', 'through', 'th
e', 'air', '.']]
Predicted caption:
 ['a', 'large', 'jetliner', 'sitting', 'on', 'top', 'of', 'a', 'runway', '.']
score: (30.0, 1.4285714285714295)
```

```
Actual captions:
 [['a', 'small', 'bird', 'sitting', 'on', 'top', 'of', 'a', 'plastic', 'cup', 'of', 'water', '.'], ['plastic', 'cup', 'wit
h', 'yellow', 'and', 'white', 'bird', 'perched', 'on', 'the', 'rim', '.'], ['a', 'yellow', 'and', 'black', 'bird', 'perche
d', 'on', 'a', 'cup', 'of', 'water'], ['a', 'bird', 'perched', 'on', 'the', 'edge', 'of', 'a', 'water', 'cup', '.'], ['blac
k', ',', 'white', ',', 'and', 'yellow', 'bird', 'sitting', 'on', 'the', 'edge', 'of', 'a', 'plastic', 'cup', '.']]
Predicted caption:
 ['a', 'person', 'holding', 'a', 'banana', 'in', 'a', 'hand', 'holding', 'a', 'banana', '.']
score: (25.0, 1.1111111111111112)
```

Figure 13: 3 bad caption predictions.

The first caption is interesting because it is the second time we have seen an animal identified incorrectly. It is strange that it identified a giraffe since there is a lack of color, but understandable since lots of giraffe images involve them eating leaves from tall trees.

The second caption is interesting because this was a common occurrence. The model actually repeatedly thought that jetliners were always sitting on runways, and scored quite poorly on many jetliner images. The jetliner on clear blue sky in the Good Captions section was one of the only times the model actually correctly captioned images of flying planes. Again, this supports our hypothesis that the model would not generalize very well to new contexts.

The third caption is interesting because it is very understandable why the model thought it was a banana - there is a smidge of yellow on the bird. However, the sentence is almost entirely nonsensical, and we could not figure out how the model generated this caption for this image.

9.3 Discussion

It seems like our model does a pretty good job at images that are similar to ones it has already seen. It was able to correctly identify and caption the sheep on the field, the surfer, the motorcycle parade, and other images that have a lot of information to generate a caption with. On the other hand, it had numerous occurrences of mixing up its animals, as well as inability to generalize sometimes because it would lock itself into a previously seen caption by the first or second prediction.

We are quite proud of our model's performance overall.

10 Team contributions

Michael - I implemented the Experiment object and ran some of the experiments with Shubham. I was responsible for keeping the style of the code consistent across the whole project to improve readability. I wrote up about half the report and made sure syntax and organization stayed consistent across Shubham's and my writings.

Shubham - I implemented the model object, and connected all of its components together. I also worked with Michael to implement the Experiment object. We ran the experiments together, and merged the results. I also helped Michael write the other half of the report, and verified the mathematical figures and generated captions.

11 References

1. ResNet-50 Description
   https://arxiv.org/pdf/1512.03385.pdf
2. LSTM Architecture
   https://www.bioinf.jku.at/publications/older/2604.pdf
3. Long-term Recurrent Convolutional Networks for Visual Recognition and Description
   https://arxiv.org/pdf/1411.4389v2.pdf
4. Deep Captioning with Multimodal Recurrent Neural Networks (M-RNN)
   https://arxiv.org/pdf/1412.6632.pdf