COCO Captions

Authors: Hieu Luu, Michael Ruddy, Joseph Samuel

Abstract

In this project we were able to train a model to caption images using the COCO data set. We trained 4 models. The first 2 used a custom CNN to to encode the image and an LSTM to decode it. The difference between the two is that the second had a different model architecture. The accuracy for the models was bleu1 = 46.08% and bleu4 = 1.93% for the first model and bleu1 = 44.24% and bleu4 = 1.65% for the second model. The other two models had the RESNET50 model as the encoder and the same LSTM, the difference between the 2 here is that the second model has different hyper parameters. The accuracy for the models was bleu1 = 63.73 and bleu4 = 6.54 for the first model and bleu1 = 61.54 and bleu4 = 5.74 for the second model. A insight we saw was that the model started to learn the word 'a' and other common words first.

1 Introduction

Common datasets such as CIFAR-10 motivate the problem of a computer learning to classify images. But often, it is important to observe an image and recognize what is happening in the image, not just what object is in it. What if a computer learned to do something more than just image classification like image captioning? The COCO (Common Objects in Context) dataset provides more than 330,000 images and 5 captions per labelled image to teach a computer to do so. The approach we took to solve this problem was to use a convolutional neural network (CNN) and a long short-term memory (LSTM) model in an encoder-decoder architecture. Using the CNN to extract features in the image and the LSTM to generate the captions from them gave notable results that this report will detail.

2 Related Work

2.1

Holger Caesar,Jasper Uijlings, Vittorio Ferrari. COCO-Stuff: Thing and Stuff Classes in Context. University of Edinburgh. Link

We chose this paper because the authors had a similar goal as us in classifying images and recognizing individual details of images such as backgrounds, objects, and assigning context to them. This paper focuses on using image classification for "Stuff" which is really just background object that can be extremely variable and vague like the sky or walls or grass. The researchers who published the paper used the COCO and VGG16 model in order to do semantic segmentation on the images.

2.2

Patterson, G., Hays, J. (2016). COCO Attributes: Attributes for People, Animals, and Objects. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9910. Springer, Cham. Link

Patterson's work emphasized more heavily on objects which are essential to generating good captions, since most of the images in the data set have objects in them. The Researchers in the paper created

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Economic Labeling Algorithm (ELA) which creates annotations/ object attributes for the various object instances in the COCO data set. This was done in order to create more specific and attribute oriented captions for models to train on while using the COCO data set

2.3

In addition the following Pytorch libraries aided us in designing our models.

functional

tensors

cuda

All the PyTorch websites were used to search up documentation for the multitude of function and methods we had to use in order to create the CNN-LSTM

3 Models

3.1 Network Architecture

We used an encoder-decoder architecture using a CNN and an LSTM.

Our CNN architecture consists of the following layers of which we have provided: input size, output size, stride size, kernel size, activation function, and amount of padding

Layer	Input	Output	Stride	Kernel	Activation	Padding
conv1	3	64	4	11	ReLU	x
maxpool1	64	64	2	3	x	x
conv2	64	128	x	3	ReLU	2
maxpool2	128	128	2	3	x	x
conv3	128	256	x	3	ReLU	1
conv4	256	256	x	3	ReLU	1
conv5	256	128	x	3	ReLU	1
maxpool3	128	128	2	3	x	x
avgpool	128	128	x	x	x	x
linear	128	300	x	x	ReLU	x

Our LSTM architecture consisted of 2 layers with an embedding size of 300, hidden unit size of 512,

The ResNet-50 model is well-renowned for image recognition. As a separate experiment, we replaced our CNN with a pre-trained ResNet-50, removing its last layer, and replacing it with our own trainable linear layer with 2048 inputs and 300 outputs.

3.2 Changes to Architecture

We experimented with the effects of changing some of our architecture. The changes we made were the number of hidden units in the LSTM going from 512 to 600 and we increased embedding size from 300 to 500. The reason we made this change was because we wanted to see how important the hidden size and embedding size were to the accuracy the LSTM model could achieve. We also wanted to see if we could improve the accuracy a sizeable margin if we increased these two aspects of the model architecture. Our initial hypothesis was that this would improve accuracy since we are increasing network complexity and the number of features the mode takes into account. But, this was wrong. We predict that this was caused by the added complexity leading to the model being worse at generalizing and over-fitting to the data.

3.3 Changes to Hyperparameters

For the second task, the hyperparameter changes we made were the optimizer going from Adam to Stochastic Gradient Descent and decreasing the learning rate from 0.01 to 0.001. The reason we made these changes was because we wanted to see which optimizer was better at getting the model to converge, while also understanding if changing the learning rate had a significant effect in time to convergence. Also wanted to know if this hyper-parameter combo would improve accuracy and efficiency.

4 Results

4.1 Bleu1 ad Bleu4 Scores

Models	Bleu1	Bleu4
Task1 Original	46.08%	1.93%
Task1 Altered	44.29%	1.65%
Task2 Original	62.73%	6.54%
Task2 Altered	61.54%	5.74%

4.2 Loss Curves for Best Models

4.2.1 Custom CNN-LSTM Model

Our original custom CNN-LSTM model (model without any architecture changes) performed better than the model with the changes to the CNN-LSTM hidden units and embedding size. Test set loss = 1.595.

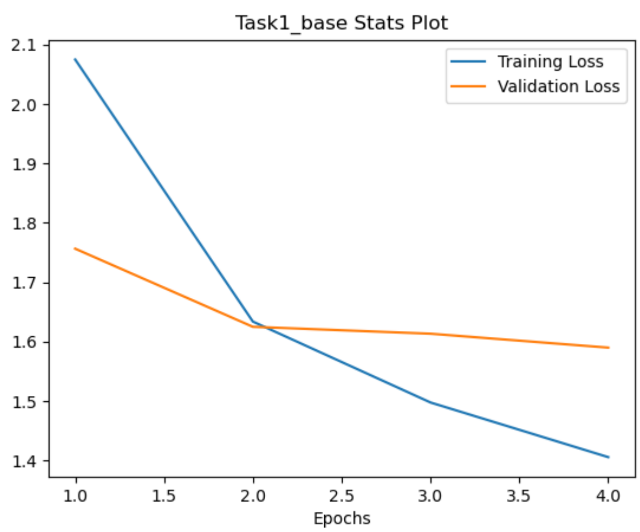

Figure 1: Custom CNN-LSTM Model with default architecture.

4.2.2 ResNet50-LSTM Model

Our ResNet50-LSTM model with a lower learning rate and lower weight decay performed better. Test set loss = 1.438.

4.2.3 Plot Observations

Observing both plots (Custom CNN and ResNet50), the models rapidly reduce the training loss, but at the risk of overfitting, given enough epochs because the validation loss plateaus. This demonstrates why early-stopping was useful because if the models continued to train, they would greatly memorize the training instead of generalizing more.

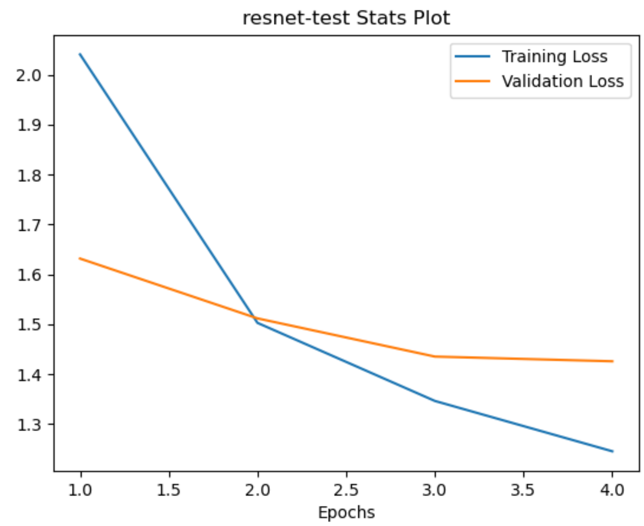

Figure 2: ResNet50-LSTM Model with default hyper parameters.

5 Captions

Figure 3: Task1 Good Image 1
Deterministic: a man is riding a surfboard on a wave
Stochastic temp = 0.4: a man riding a surfboard on top of a wave.
Stochastic temp = 0.001: a man is riding a surfboard on a wave
Stochastic temp = 5: starving formal branches paid tiring navel aids meat league raises place milkshakes these sheltered minimally wedged ringed one bite load

Figure 4: Task1 Good Image 2
Deterministic: a man is riding a surfboard on a wave
Stochastic temp = 0.4: a white plate with a sandwich and chips on it.
Stochastic temp = 0.001: a man is riding a surfboard on a wave
Stochastic temp = 5: bounds spewing businessman system explosion for rest surrounding diced wrangling trellis caped airborn mounts gathered salt kearny poling scatters lineup bulldozer

Figure 5: Task1 Good Image 3
Deterministic: a man is riding a surfboard on a wave
Stochastic temp = 0.4: a man riding skis down a snow covered slope.
Stochastic temp = 0.001: a man is riding a surfboard on a wave
Stochastic temp = 5: sightseers peach wooden marshmallow typing coaster pattern flushed almond ramekin night serving spiky dollops footing shells lighted dials upright record

Figure 6: Task1 Trash Image 1
Deterministic: a man is riding a surfboard on a wave
Stochastic temp = 0.4: a man is eating a slice of pizza
Stochastic temp = 0.001: a man is riding a surfboard on a wave
Stochastic temp = 5: knocking in the renovated bowl its mariners perform imagery spooky ashtray wreckage forks heads groupe souffle throw pine serene ripe remaining

Figure 7: Task1 Trash Image 2
Deterministic: a man is riding a surfboard on a wave
Stochastic temp = 0.4: a man in a blue shirt holds a teddy bear.
Stochastic temp = 0.001: a man is riding a surfboard on a wave
Stochastic temp = 5: seating shines vantage bicycle crutches rise cabinets gay curbed hers samples cabinets corralled platforms shared ground illuminated illustrations areas cant

Figure 8: Task1 Trash Image 3
Deterministic: a man is riding a surfboard on a wave
Stochastic temp = 0.4: a man with glasses and a tie in his pockets.
Stochastic temp = 0.001: a man is riding a surfboard on a wave
Stochastic temp = 5: packed cucumbers seems mechanics partitioned going changing wispy bushes statuette parasailing surfaces collision body seeing dive beach-goers white topped nesting

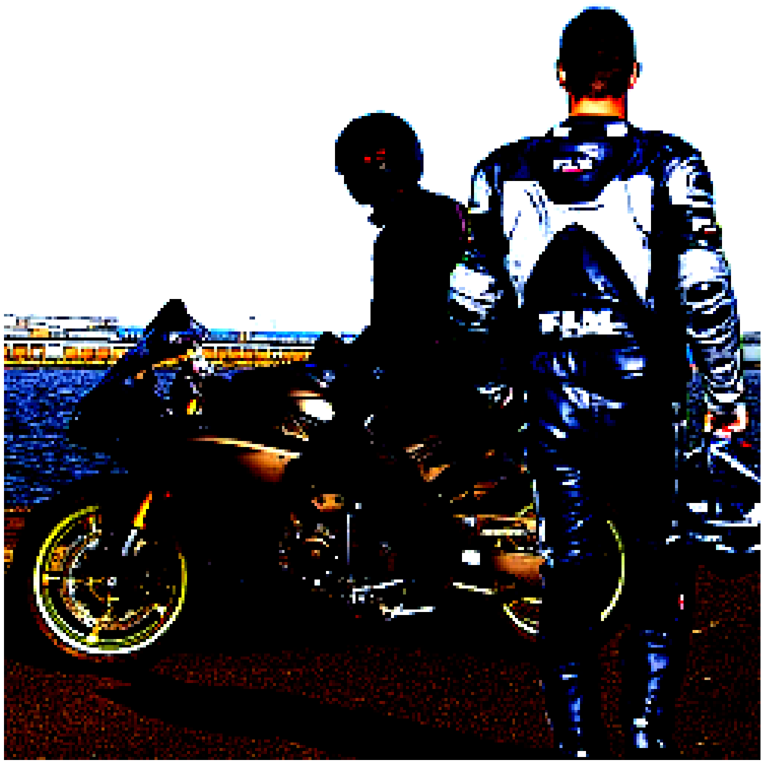

Figure 9: ResNet Good Image 1
Deterministic: a man riding a motorcycle down a street.
Stochastic temp = 0.4: a man riding on the back of a motorcycle .
Stochastic temp = 0.001: a man riding a motorcycle down a street .
Stochastic temp = 5: salsa almost ericsson carved squid parked maple moderately wallet salon trimmed sawdust tricks well soccer dipping challenging advising wild cardiff

Figure 10: ResNet Good Image 2
Deterministic: a bathroom with a sink, mirror and a mirror.
Stochastic temp = 0.4: a bathroom with a toilet and a tub.
Stochastic temp = 0.001: a bathroom with a sink , mirror and a mirror.
Stochastic temp = 5: surveys seem google stored moored sideline removed disarray stopped towing creme changed whine dull peacefully data looking aer meow throughout

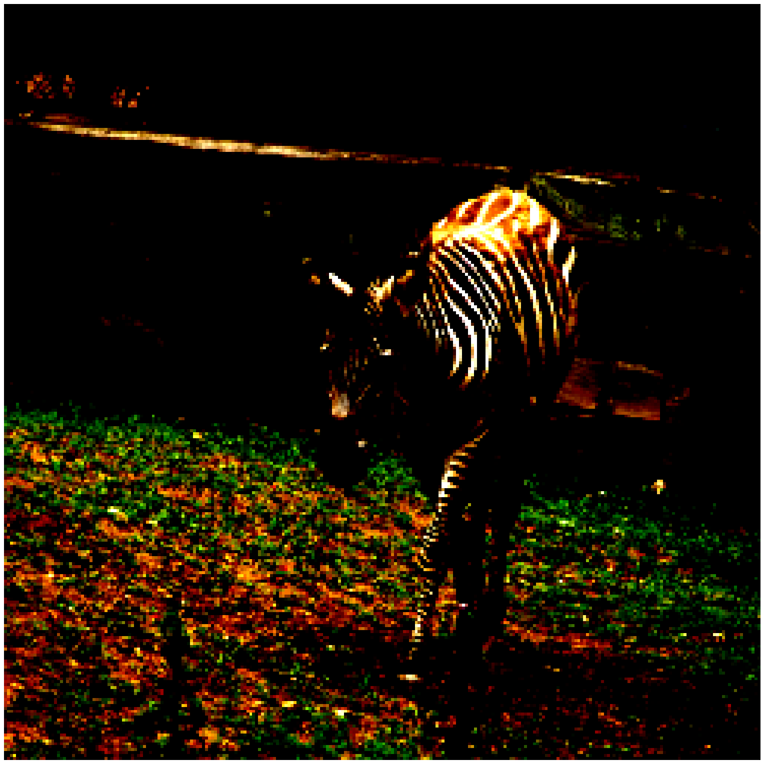

Figure 11: ResNet Good Image 3
Deterministic: a zebra standing in the grass near a tree.
Stochastic temp = 0.4: a zebra standing on a lush green field .
Stochastic temp = 0.001: a zebra standing in the grass near a tree
Stochastic temp = 5: clasping first stagecoach creeping lean enthusiastically her different radio feed college paddleboarding evergreens treading form kid serious descend duke irises

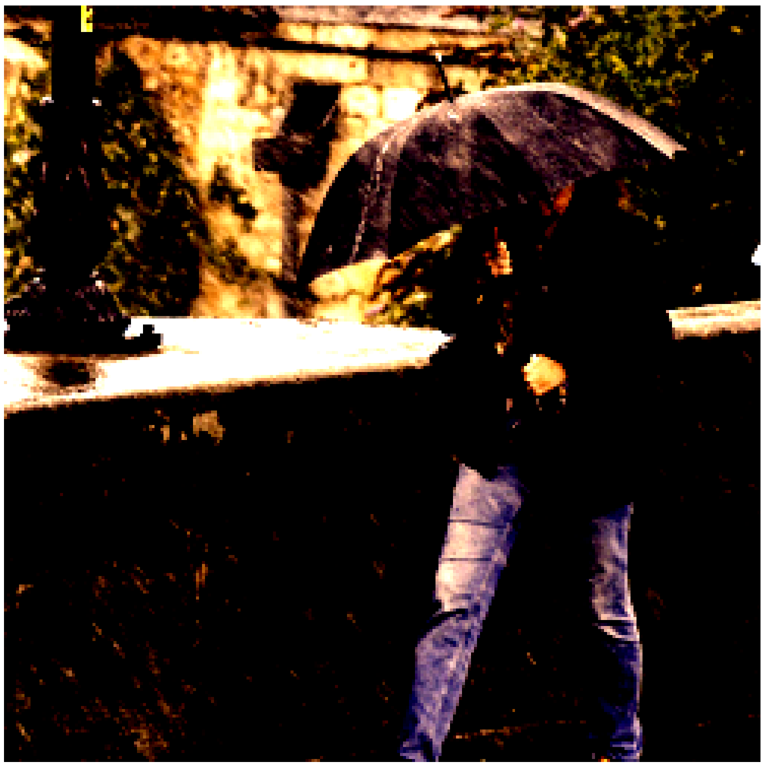

Figure 12: ResNet Bad Image 1
Deterministic: a man is walking on the sidewalk with a dog in the background.
Stochastic temp = 0.4: a street sign and a street sign on a pole.
Stochastic temp = 0.001: a man is walking on the sidewalk near a bench.
Stochastic temp = 5: kiwi racer grape project fountain casing columbus logos pieces abut camel like supplied inches human crown denotes oxygen pointer novels

Figure 13: ResNet Bad Image 2
Deterministic: a man riding a horse with a dog on the back of it.
Stochastic temp = 0.4: a plate of hot dogs and a sandwich on a plate.
Stochastic temp = 0.001: a man riding a horse with a dog on the back of it.
Stochastic temp = 5: mouths pomegranates sprinkled bent heads treat british gymnasium cylinder welcome aeroplanes treat marathon remain barrier storefront boarding break underway well-worn

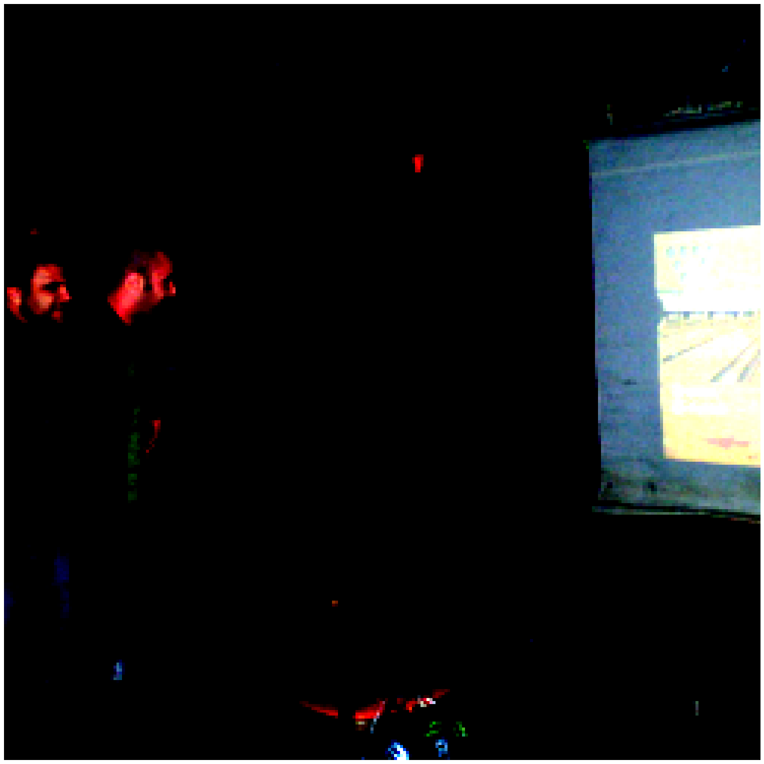

Figure 14: ResNet Bad Image 3
Deterministic: a man is sitting on a couch with a laptop on his lap.
Stochastic temp = 0.4: a large elephant with its eyes open in the air
Stochastic temp = 0.001:a man is sitting on a couch with a laptop on his lap.
Stochastic temp = 5: watchdog permanent measure bald standing pines documents identically drums fat sanitary vapor turned regulation marking signal knocked anchors brightly streetlights

6 Discussion

We trained 4 models in total the first two were used a Custom CNN as the encoder. The last two used the RESNET 50 as the encoder. The accuracy for the models with the Custom CNN encoder was bleu1 = 46.08% and bleu4 = 1.93% for the first model (base model) and bleu1 = 44.24% and bleu4 = 1.65% for the second model (model with architecture changes). The accuracy for the models was bleu4 = 6.54 and bleu1 = 62.73 for the first model (base LSTM with Resnet Encoder) and bleu4 = 5.74 and bleu1 = 61.54 for the second model (LSTM with Resnet Encoder and hyper parameter changes). The best model we had was the base model from task2 which was the model with the Resnet50 encoder with the default LSTM (512 hidden units and 2 layers). We think the this model did the base because the Renset 50 is a highly accurate CNN that is able to pull out multiple features which means the LSTM gets a better representation and details of the image, helping the captions be more accurate. For example if the CNN does not accurately identify /represent an image there is no chance for the captions to be accurate. Our worst model was the Task1(Custom CNN Encoder) with Architecture. In this model we increased the number of hidden units to 600 and the embedding size to 500. We believe this was the worst because it lead to our model over fitting to the training, which is seen because our training loss was a little over 1.3 while our validation was still at 1.6. We believe deterministic doesn't work well because it can lead to the low variability and the same captions being outputted since some words have a higher probability and they will always be chosen, this means many of the words will never be used. For us the default hyper parameters with the Adam optimizer and momentum of 0.9 (lr_scheduler.ExponentialLR) were the best hyper parameters. In fact for both tasks are models converged by epoch 4, which is when early stopping kicked in. Furthermore when we changed the hyper parameters for both models the accuracy actually ended up decreasing. When we made the temperature really big the words in the captions became very random and didn't have any structure or correlation to the image. When we made temperature really small the captions became more uniform and started outputting the same words for different unrelated images. Both having a really high and low temperature had a negative impact on accuracy. We believe this is the case because when we increase temperature we we lower the difference in probability between the different possible output words leading to a more random caption, while for a really low temp the exact opposite happens which leads to certain words being chosen over and over again.

7 Team Contributions

Joseph Samuel:
For this project I assisted in the coding and development of every function in the modelfactory.py and every function in the experiment.py. I helped run the models (task1 base and task1 with architecture changes).I wrote the abstract, 3b, discussion, and some of the captions.

Michael Ruddy: For this project I assisted in the coding and development of every function in the modelfactory.py and every function in the experiment.py. I helped run the task 1 base model and helped set up for task 2. I wrote the related works on the report. I wrote our loss function and troubleshot our LSTM.

Hieu Luu: For this project I assisted in the coding in the modelfactory.py functions and generate captions and run in the experiment.py. I helped run the models (task2 base and task2 with architecture changes).I wrote, architecture changes, discussion, and half the captions.