Experiment Image Captioning with CNN and LSTM Neural Network

Dongze Li
Department of Cognitive Science
University of California, San Diego
La Jolla, CA 92093
dol073@ucsd.edu

Xiaoyan He
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093
x6he@ucsd.edu

Yunfan Long
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92093
yulong@ucsd.edu

Abstract

In this project, we created captions for a particular image from the data set: COCO 2015 Image Captioning Task by combining CNN and LSTM. To build our training, validation, and test sets, we randomly selected a small portion (1/5) of the original images, each of which contained five representative captions. We used these data sets to train our models using images from the training set, and we used the validation and test sets to calculate the loss, BLEU-1 and BLEU-4 scores. In the end, we were successful in developing an architecture that produces appropriate predicted captions that are similar to the actual captions for the image. We used custom CNN as encoder and LSTM as decoder to build the first model, and we utilized a pretrained Resnet-50 and LSTM to build the second model by transferring the knowledge from Resnet-50. According to our findings, our first model performed better once the hyperparameters hidden size and embedding size were modified. Our first model with custom CNN as encoder at best achieved BLEU-1 and BLEU-4 scores score of 48.502 and 1.986, and cross-entropy test loss of 1.508. Our second model performed better once the optimizer and learning rate were modified. Our second model at best achieved BLEU-1 and BLEU-4 scores of 50.267 and 1.542, and cross-entropy test loss of 2.308.

1 Introduction

Natural language processing is constantly expanding to include a wider range of uses, from text production to text translation. Recurrent neural networks are one of the more recent approaches in the field (RNNs). In the present period, RNNs have been applied to a wide range of sequential data and output generation-related challenges[4]. Long Short Term Memory(LSTM), however, was developed as a result of the necessity for models to be able to preserve their state for extended periods of time. It has been demonstrated that LSTM function well on sequential data having temporal structure, such as text data or video frame sequences[5]. Using an LSTM as a decoder network, we produce captions for images using a recurrent neural network that can take as input an image embedding from our custom CNN network as a encoder for model-1 and a pre-trained CNN encoder for model-2 respectively. Approximately 82,000 photos make up the training and validation sets, 3,000 images make up the test set, and the COCO (Common Objects in Context) dataset contains numerous captions for each image. We randomly select a portion (1/5) of these images to train our

model. We used custom CNN as encoder and LSTM as decoder to build the first model, and we utilized a pretrained Resnet-50 and LSTM to build the second model by transferring the learning from Resnet-50 to our model. We chose to generate captions using the stochastic instead of deterministic approach. Our attempts to change the hidden size & embedding size for model-1 and optimizer & learning rate for model-2 yielded better and ultimately best performance. We keeps all the other default hyperparameters like vocabulary threshold and number of epochs unchanged for both models. Our best model-1 ended up performing better than our best model-2. Our network architecture takes the word generated at the previous time step and the image embedding as inputs each time. The model were trained using "teacher enforcing", which use all previous actual words from caption alongside the current word as input. However, we generate output using all previous generated words as input. The BLEU-1 and BLEU-4 scores, a metric frequently used to evaluate machine translation models, will be utilized to assess the generated captions.

2 Related Works

We acknowledge the great help from our instructor Garrison Cottrell for briefly introduced and explained the model of CNN [2][3] and LSTM to us[4][5]. We also thanks for the book "Dive into Deep Learning" [1] which gives a lot of insight on model construction, and the ideas of how convolution layer detects the features of the input and how LSTM takes the sequence as input to learn caption generation. The dataset we used in this experiment, COCO (Common Objects in COntext)[7], gives a lot of good images to help us to understand the advantages and the shortages of our custom model. We also thanks for the PyTorch Library[8] provides useful function and method that saved a huge amount of time.

Here, we also thanks the previous works that also used COCO dataset for providing us a lot of insights. In the paper "Image Captioning with Deep Bidirectional LSTMs"[6], they also constructed the network by employing CNN as the encoder and LSTM as the decoder and also used COCO dataset for testing model. One different thing there is that they used bidirectional LSTM which increases the depth of non-linearity transition in different way. Their model achieved the better performance than our model which has BLEU-1 as 67.2 and BLEU-4 as 24.4. Through comparing between their model and ours, we think the bidirectional LSTM has a stronger ability to learning long term visual-language interactions. Also Yang Xian and Yingli Tian's work [9] gives us more ideas about how to deal with the situation when the training set contains imbalanced and noisy data. They introduced a novel self-guiding multimodal LSTM as the solution of such problem. By using the data whose textual description strongly correlates with the image content to train the model and using them as guiding information serves as additional input to the network along with the image representations when training the network with the rest data, their model shows the effectiveness with BLEU-1 above 40 and BLEU-4 above 3 with imperfect training set.

[1] A. Zhang, Z.C. Lipton, M. Li, A.J. Smola. Dive into Deep Learning. *arXiv:2106.11342 [cs.LG], Jul 2022*

[2] G. Cottrell. *Lecture 5: Introduction to Convolutional Networks*, lecture notes, Department of Computer Science, University of California San Diego, Oct 2022.

[3] G. Cottrell. *Lecture 6: Convolutional Networks, Part II*, lecture notes, Department of Computer Science, University of California San Diego, Oct 2022.

[4] G. Cottrell. *Lecture 7: Modeling sequences (Recurrent Networks)*, lecture notes, Department of Computer Science, University of California San Diego, Oct 2022.

[5] G. Cottrell. *Lecture 8: Generative Modeling with RNNs*, lecture notes, Department of Computer Science, University of California San Diego, Nov 2022.

[6] Cheng Wang, Haojin Yang, Christian Bartz, Christoph Meinel. Image Captioning with Deep Bidirectional LSTMs, Apr 2016.

[7] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR, abs/1405.0312, 2014.*

[8] A. Paszke. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019.

[9] Yang Xian, YingLi Tian. Self-Guiding Multimodal LSTM - when we do not have a perfect training dataset for image captioning, Sep 2017.

3 Model Description

3.1 Baseline Model Description

In the first model, we employ a customized CNN as the encoder and LSTM as the decoder. As shown in Table 1, The custom CNN architecture is constructed by a combination of convolution layer, batch normalization layer, non-linear Relu layer, and max pooling layers, which are detailed in the table below with input dimension, output dimension, kernal size, strides, and padding size for each layer. Note that NA here represents a specific feature is not applicable or not specified for a layer.

Table 1: CNN architecture

Layer	Input channels	Output channels	Kernel size	Stride size	Padding size
convolution 1	3	64	11	4	NA
batch 1 normalization	64	64	NA	NA	NA
Relu	64	64	NA	NA	NA
max pooling 1	64	64	3	2	NA
convolution 2	64	128	5	1	2
batch normalization 2	128	128	NA	NA	NA
Relu	128	128	NA	NA	NA
max pooling 2	128	128	3	2	NA
convolution 3	128	256	3	1	1
batch normalization 3	256	256	NA	NA	NA
Relu	256	256	NA	NA	NA
convolution 4	256	256	3	1	1
batch normalization 4	256	256	NA	NA	NA
Relu	256	256	NA	NA	NA
convolution 5	256	128	3	1	1
batch normalization	128	128	NA	NA	NA
Relu	128	128	NA	NA	NA
max pooling 3	128	128	3	2	NA
adaptive averagepool	128	128	1 * 1	1	NA
fully connected layer 1	128 * 6 * 6	1024	NA	NA	NA
fully connected layer 2	1024	1024	NA	NA	NA
fully connected layer 3	1024	300	NA	NA	NA

The RNN modified with a long short term memory cell, or LSTM , is the decoder in both of our model, gives the network "memory" and solves the issue of exploding and vanishing gradients that are generally present in multilayer RNNs. Our LSTM is constructed with 2 layers, embedding size of 300, and hidden size of 512, as shown in Table 2. Rather than using the term that the network generates as input to the next time step, our models were trained using "teacher forcing," which uses the actual words from the captions. The initial word is provided to the model during testing, and it is taught to output the next word in the caption recursively using previously generated words. When this word is input back into the network, the model will be able to create captions for previously viewed images. The model can generate captions either stochastically or deterministically. In the stochastic setting, words will be sampled from each time step's softmax probability distribution and smoothed with a temperature hyperparameter in order to produce novel predictions. In the deterministic setting, it greedily selects the word with the maximum probability at each time step. Here, we generate captions stochastically because we can use the the temperature parameter to control and fine-tune the sampling process from softmax probability distribution from the last layer. What's more, by setting the temperature to 0, we essentially get near deterministic distribution. Hyperparameters of our model includes: a vocabulary threshold indicating the number of times a word must appear in the caption before it gets abandoned, a hidden-size designating the dimensionality of the linear layer in the decoder network through which the word vectors pass, an embedding size designating the dimension to which we reduce one-hot encoded word vectors derived from

the captions, and finally, parameters pertaining to our mini-batch stochastic gradient descent procedure used during training (number of epochs, batch size, learning rate, choice of optimizer, etc.). Finally, BLEU scores and cross-entropy loss are used to assess our models. Table 2 below shows the default architechture of our LSTM network.

Table 2: Architecture of LSTM

Component	Value
Number of Layers	2
Hidden Size	512
Embedding Size	300

Despite models in Task 1 where we purposefully changed the hidden size and embedding size for fine-tuning our model, all other models uses the above architecture for LSTM unless otherwise specified.

Table 3: Default hyperparameters of LSTM

Hyperparameters	Value
Vocabulary Threshold	2
Number of epochs	10
Batch size	64
Learning rate	$5e^{-4}$
Optimizer	Adam
Lr Scheduler	Steplr

Table 3 above shows the default hyperparameters of our LSTM network. Despite models in Task 2 where we purposefully changed the learning rate and optimizer for fine-tuning our model, all other models uses the following hyperparameters for LSTM unless otherwise specified.

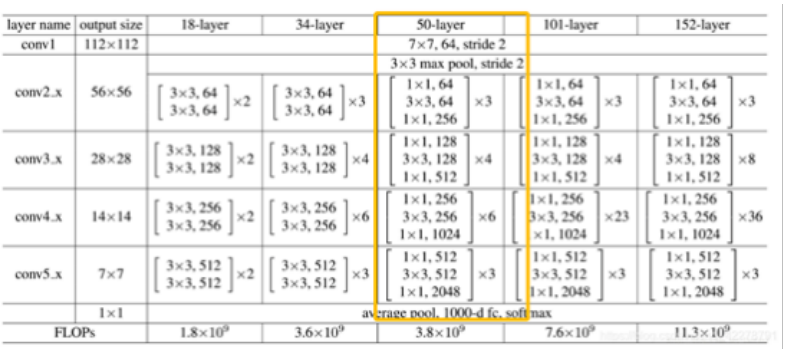
While it allows us the flexibility of adding more CNN layers to handle more challenging computer vision jobs, it also has a number of drawbacks. With more layers added, it has been found that training neural networks gets more challenging, and in some situations, accuracy decreases as well. After a certain point, the model's performance declines on both the training data and the testing data. Over-fitting did not cause this degradation. Instead, it can be the result of the network's initialization, an optimization function, or more crucially — an issue with vanishing or exploding gradients. ResNet was developed with the intent of solving this particular issue. Residual blocks are used in deep residual networks to boost model precision. The strength of this kind of neural network is the idea of "skip connections," which is at the foundation of the residual blocks. These skip connections operate in two different ways. First, they resolve the problem of the vanishing gradient by creating a different shortcut for the gradient to use. They also give the model the ability to learn an identity function. By doing this, it is made sure that the model's higher levels don't perform any worse than its lower layers. In summary, the layers learn identity functions much more quickly thanks to the residual blocks. ResNet hence reduces the number of errors while increasing the effectiveness of deep neural networks with more neural layers. To put it another way, the skip connections combine the results of earlier layers with the results of stacked layers, enabling the training of far deeper networks than was previously feasible.

The graph belowed (next page) shows the architechture of resnet-50 (square-selected in yellow):

3.2 Changes in Task 1

The embedding size, which indicates the length of the image feature vectors after being fed through our custom CNN encoder, and the hidden size, or the number of hidden units in the decoder network, are two crucial components on which we make changes to fine-tune our model. Our default model has hidden size of 512 and embedding size of 300, which can also be denoted as $hid_{512}emb_{300}$.

The bias-variance tradeoff of the embedding size makes it ideal for the vectors to be both small enough to enable effective computing and large enough to retain enough data for the network to produce an accurate caption. We first increased the embedding size from 300 to 500 because we thought the higher dimension, probably the better the capturing of information although more the time might be required for training. However, this model ($hid_{512}emb_{500}$) yield unsatisfactory result, with the bleu-1 score of 43.669, which is 0.858 lower than the



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; ×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 1: Architechture of Resnet-50

44.527 of default model. Decreasing embedding size from 300 to 100 ($hid_{512}emb_{100}$) gets a bleu-1 score of 47.429, which is 2.902 higher than that of our default model.

The hidden size functions as a step between sizing the vectors and must also have a value in order to be both useful and small enough to be used throughout a large number of time steps so that our model is neither flooded with parameters nor lack of necessary complexity. We first decreased the hidden size from 512 to 256 to check whether the default model has been potentially over-fitted. This model ($hid_{256}emb_{300}$) yields a bleu-1 score of 47.535, which is 3.008 higher than that of our original model. The 256 hidden unit seems to perform better than the default model. We then increase the hidden size from 512 to 1024. This model ($hid_{1024}emb_{300}$) turns out to be the best model, with bleu-1 score of 48.502 (about 4 higher than default model), bleu-4 score of 1.986, and test loss of 1.508 (0.061 lower than default model). The increase is impressive and it turns out to be the best model.

3.3 Changes in Task 2

Better optimizers are primarily concerned with being more efficient and faster but they are also frequently recognized for being more generalizable (less over-fitting) than others. It is probable that the optimizer we choose will have a significant impact on how well the model works. Our default model ($Adam_{5e^{-4}}$) has Adam as optimizer and learning rate of $5e^{-4}$, which yields bleu-1 score of 45.212. The bleu-1 score of our default model-2 better than our previous default model-1 by 0.685.

As a result, we conduct experiments by switching the optimizer from Adam to SGD. However, since SGD converge slower, it is important to find a suitable learning rate for SGD. Starting from $5e^{-5}$ all the way to $5e^{-0}$, we tried different learning rate to ensure our SGD properly converge. For learning rate $5e^{-5}$, $5e^{-4}$, our model with SGD optimizer get bleu-1 score of 0.199($SGD_{5e^{-5}}$) and 0.085($SGD_{5e^{-4}}$) respectively. Obviously, the learning rate is so low that our model is under-fitted. We then use $5e^{-3}$ $5e^{-2}$, $5e^{-1}$, and $5e^{0}$ as learning rate, which yield bleu-1 score of, 49.932($SGD_{5e^{-3}}$), 50.267($SGD_{5e^{-2}}$), 47.086($SGD_{5e^{-1}}$), and 45.134($SGD_{5e^{0}}$). This is much better, and it seems our model with SGD optimizer perform the best at learning rate $5e^{-2}$.

A model can perform better or worse given different learning rate. Therefore, we also conduct experiments using Adam optimizer with different learning rate. We first increased the learning rate from default $5e^{-4}$ to $5e^{-3}$. This model($Adam_{5e^{-3}}$) yields bleu-1 score of 44.539, which is 0.673 lower than that of our default model. We then decreased the learning rate from default $5e^{-4}$ to $5e^{-5}$. This model($Adam_{5e^{-5}}$) yields bleu-1 score of 49.007, which is 3.795 higher than that of our default model.

In conclusion, the $SGD_{5e^{-2}}$ model turns out to be the best. The bleu-1 score of best SGD model ($SGD_{5e^{-2}}$) is 1.26 higher than the best Adam model $Adam_{5e^{-5}}$.

4 Results

Table 4: Model changes in task 1

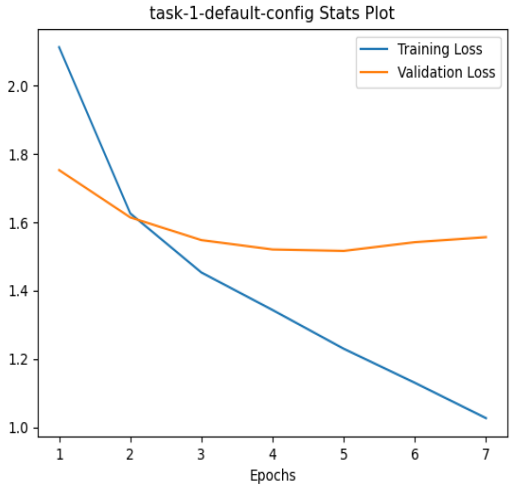
Model	Bleu-4	Bleu-1	Test Loss
Change Embedding			
$hid_{512}emb_{300}$ (Default)	2.156	44.527	1.569
$hid_{512}emb_{500}$	1.994	43.669	1.567
$hid_{512}emb_{100}$	1.568	47.429	1.580
Change Hidden Size			
$hid_{512}emb_{300}$ (Default)	2.156	44.527	1.569
$hid_{256}emb_{300}$	1.914	47.535	1.626
$hid_{1024}emb_{300}$	**1.986**	**48.502**	**1.508**

$hid_{512}emb_{100}$ worked well probably because the default model with embedding size 300 suffers from over-parametrization, thus its bleu-1 score is lower. This is reinfoeced by a further decrease in bleu-1 score when we increased the embedding size from 300 to 500 on model $hid_{512}emb_{500}$. $hid_{1024}emb_{300}$ worked better than $hid_{512}emb_{300}$ because our model need more hidden units to better learn complex representation.

Table 5: Model changes in task 2

Model	Bleu-4	Bleu-1	Test Loss
SGD			
$SGD_{5e^{-5}}$	0.023	0.199	9.300
$SGD_{5e^{-4}}$	0.019	0.085	5.500
$SGD_{5e^{-3}}$	1.551	49.932	3.078
$SGD_{5e^{-2}}$	**1.542**	**50.267**	**2.308**
$SGD_{5e^{-1}}$	1.520	47.086	1.581
$SGD_{5e^{0}}$	1.801	45.134	1.717
Adam			
$Adam_{5e^{-3}}$	1.693	44.539	1.619
$Adam_{5e^{-4}}$(Default)	1.637	45.212	1.517
$Adam_{5e^{-5}}$	1.561	49.007	1.719

$SGD_{5e^{-5}}$, $SGD_{5e^{-4}}$ didn't work well because SGD converges much more slower than Adam, which requires larger learning rate to converge in 10 epochs. $SGD_{5e^{-3}}$, $SGD_{5e^{-2}}$, $SGD_{5e^{-2}}$, and $SGD_{5e^{0}}$, in comparison, worked much better with $SGD_{5e^{-2}}$ being the best. $Adam_{5e^{-3}}$ and $Adam_{5e^{-4}}$ suffers from respectively relatively high learning rate, and hence risks overshooting during gradient descent. As a result, $Adam_{5e^{-5}}$ turns out to perform the best among Adam models.

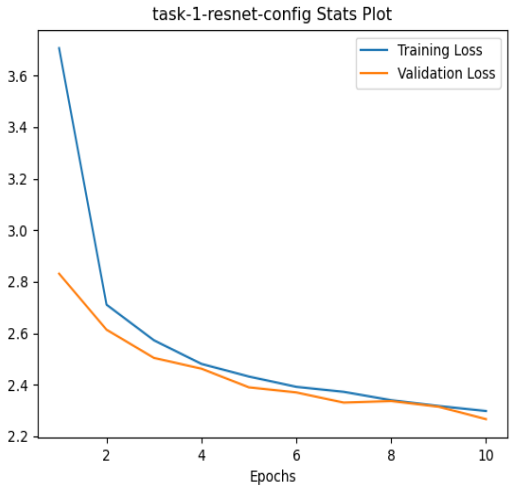


(a) Training and validation loss

Figure 2: Task 1 training performance for $hid_{1024}emb_{300}$

4.1 Task 1 - Individual Finding

In task 1, the training loss and validation loss both are decreasing at a decreasing speed. It takes only 2 epochs for our model to a reach validation loss of 1.6 - it converges pretty fast. However, validation loss starting to rise after the 5th epoch, which entails potential overfitting, given the test loss is still decreasing. This makes sense because we double the amount of the hidden units, which might lead to over-parametrization to our model.



(a) Training and validation loss

Figure 3: Task 2 training performance for $SGD_{5e^{-2}}$

4.2 Task 2 - Individual Finding

In the task 2, the training loss and validation loss both are decreasing at a relative constant but bumpy speed, with some fluctuations on validation loss. It takes about 10 epochs for our validation accuracy to drop only to 2.3. After that, the validation loss decrease at a relatively constant speed with a lot of oscillation. The near

constant decreasing of validation loss can be perceived as a sign of stable gradient update. It seems like after 10 epoch the curve for validation error is still decreasing.

4.3 Compare Two Models

As we can see, validation error for both model are decreasing but ultimately reaching two different validation accuracy (1.556 for $hid_{1024}emb_{300}$ and 2.266 for $SGD_{5e^{-2}}$). Model-1 starts off at a validation loss of 1.8 after epoch 1, while model-2 starts off at a validation loss of higher then 3. In addition, the validation loss of model-1 reaches 1.6 after only 2 epochs while it takes 10 epochs for model-2 while still far from achieving the same thing. Therefore, it can be observed that model-1 tends to converge much more faster. However, in the process of gradient descent, model-1 tends to have a much more smooth curve while model-2 has a relatively piece-wise, linear curve. This means the gradient update process of model-2 is in fact more stable in slope than that of model-1. Model-1 a bump at epoch 5, where the validation loss of each goes up, while model-2 did not. Since model-1 is much more smoothier than model-2, which oscillate up and down linearly. In this regard, model-1 might be better in terms of generalization as well as more robust to potential outliers since its harder for the validation loss of it to change as drastically as model-2. After 5th epoch, model-1 starts to suffer from overfitting with validation loss going up, while the validation loss curve of moodel-2 turns out to keep decreasing. It seems 512 hidden units instead of 1024 hidden units would be a better setting if we want to continue tuning our hyperparameter in the future to further reduce loss as model-1 starts to over-fit relatively early.

5 Generated Captions

5.1 Generated Good Captions in Task 1

Figure 4: Example of good caption generated by the best model in task 1
Original captions:
- a man in blue shirt and white shorts playing tennis.
- male professional tennis player swinging at a tennis ball.
- a tennis player swinging a racket at a ball.
- a man on a court swinging a tennis racket.
- a man hitting a tennis ball on a professional court.
Predict captions:
- a man holding a tennis racquet on a tennis court. (temp = 0.4)
- teh pressing archeological doubles attract alligator treatments sheath join weapon enclosure whipped sauteed tabby challenging showcasing cop televison pale bank (temp = 5)
- a man holding a tennis racquet on a tennis court. (temp = 0.0001)
- a man holding a tennis racquet on a tennis court. (deterministic)

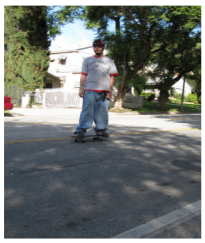


Figure 5: Example of good caption generated by the best model in task 1
Original captions:
- a man standing on a skateboard in the middle of the street.
- a young man riding a skateboard down a street.
- a skateboarder riding down the middle of the road in the shadows.
- a man is on a skateboard on the street.
- a man standing on his skateboard in the middle of the street
Predict captions:
- a man is riding a skateboard on the street. (temp = 0.4)
- hold collision style disinterested doughnut vegetated kingfisher space rises poultry desk coastal motorcyclists upcoming ajar create utilize thirty ripening 6 (temp = 5)
- a man is riding a skateboard on the street. (temp = 0.0001)
- a man is riding a skateboard on the street. (deterministic)

Figure 6: Example of good caption generated by the best model in task 1
Original captions:
- two young men sitting in front of an open laptop computer.
- some people are looking at laptops in a room
- a man showing something to a boy in a laptop sitting in a classroom.
- a child looks at a laptop with an adult.
- man and student studying something on a laptop.
Predict captions:
- a man is sitting on a couch with a laptop. (temp = 0.4)
- unboard full herds dangled iwth t-ball goggles strapped drawn seemed prey philippines eyes young hitched ceremony teresa calender railways filet (temp = 5)
- a man is sitting on a bench with a laptop (temp = 0.0001)
- a man is sitting on a bench with a laptop (deterministic)

5.2 Generated Bad Captions in Task 1

Figure 7: Example of bad caption generated by the best model in task 1
Original captions:
- people at the baggage claim area of an airport.
- three people standing before airport counters below airport signs.
- people standing at counters of booths being served
- the baggage delivery section of an air port.
- patrons are going to the shops of an airport.
Predict captions:
- a young man riding a skateboard down a ramp. (temp = 0.4)
- reflect pittsburgh snowboarder beanies crawfish dunkin prays unfinished chatting serve (temp = 5)
- a man is riding a horse in a field (temp = 0.0001)
- a man is riding a horse in a field (deterministic)

Figure 8: Example of bad caption generated by the best model in task 1
Original captions:
- an elephant looks at the camera while outside
- an elephant that is very close to the camera.
- a close up of an elephants face, with another one looking like its photobombing the picture
- an adult elephant with someone's foot on his side near a baby elephant.
- a closeup of an elephant near the water.
Predict captions:
- a man in a suit and tie standing next to a woman. (temp = 0.4)
- godzilla abstract cheers valleys struggling boxcar flowery kids jar being foam hunter domicile pizza-like congratulating unkept goes help decortive across (temp = 5)
- a man is sitting on a bench with a laptop (temp = 0.0001)
- a man is sitting on a bench with a laptop (deterministic)

Figure 9: Example of bad caption generated by the best model in task 1
Original captions:
- people in the water and parachutes overhead.
- many different sails flying over a large body of water.
- several gliders floating on the ocean next to an island.
- dozens of kite skiers out in the ocean
- parasails glide above the blue water of the lake.
Predict captions:
- a man and a woman sitting on a bench in a park. (temp = 0.4)
- monogrammed meatloaf pleasure (rom branded white lap frame chewing painted motocross attends keybard think jose intimate graduating spring hanger (temp = 5)
- a man is sitting on a bench with a laptop (temp = 0.0001)
- a man is sitting on a bench with a laptop (deterministic)

5.3 Generated Good Captions in Task 2

Figure 10: Example of good caption generated by the model in task 2
Original captions:
- a young man riding his skateboard on a ramp.
- a man in a blue and white check shirt doing tricks on a skateboard.
- a man skating a ramp at a competition.
- a skateboarder doing a trick in the air.
- a man skateboarding on a half pipe in front of judges.
Predict captions:
- a man on a skateboard is in the air. (temp = 0.4)
- brides district charms who shop well-kept heeled skulls bricked into crouches vegetation stems head demands containing lily seeking hula radishes (temp = 5)
- a man is riding a horse on a dirt road. (temp = 0.0001)
- a man is riding a horse on a dirt road. (deterministic)

Figure 11: Example of good caption generated by the model in task 2
Original captions:
- a man standing on a skateboard riding it down a sidewalk.
- a man wearing a helmet rides a skateboard
- a man on a skate board wearing a helmet and no shirt.
- an old man riding a skateboard down a street.
- a man in a helmet is riding a surfboard on the road.
Predict captions:
- a man is standing on a sidewalk with a skateboard. (temp = 0.4)
- motorboat message curl generously offices hood in windmills alto soccer choclate 2009 tuck officer garnished paper when direction drive free (temp = 5)
- a man is standing on a sidewalk with a skateboard. (temp = 0.0001)
- a man is standing on a sidewalk with a skateboard. (deterministic)

Figure 12: Example of good caption generated by the model in task 2
Original captions:
- a man and his dog sit at the park.
- a man sitting at a green bench next to a tall tree.
- a man sits on a bench with a dog laying at his feet.
- a man and his dog enjoy an afternoon in the park
- dog and his human walking partner relax in a park.
Predict captions:
- a man sitting on a bench wearing a tie. (temp = 0.4)
- boxing interact divers around horton paradise covered diorama gauze hazard poo unsuccessfully within bookshelves thickly crouched mexican blocking nad obstacles (temp = 5)
- a man is riding a bike down the street. (temp = 0.0001)
- a man is riding a bike down the street. (deterministic)

5.4 Generated Bad Captions in Task 2

Figure 13: Example of bad caption generated by the model in task 2
Original captions:
- a black and white photo of an old locomotive.
- a black and white photo of a train engine in a train depot.
- a train on the tracks parked in a station.
- a train sitting in a train station on top of railroad track.
- a large grey train inside of a train station.
Predict captions:
- a man is standing outside with his skateboard (temp = 0.4)
- telivision conductor pottery petals palace creativity these sand distance viet cappuccino lizard enjoying unit opened describe labeled slab breasts overcoat (temp = 5)
- a man is sitting on a bench with a laptop. (temp = 0.0001)
- a man is sitting on a bench with a laptop. (deterministic)

Figure 14: Example of bad caption generated by the model in task 2
Original captions:
- two women walking down the street holding umbrellas.
- view from behind of two women under umbrellas
- two people in dresses walking with umbrellas and bags.
- the back of two women holding up umbrellas and carrying bags.
- two people are walking away from the camera with umbrellas.
Predict captions:
- a man riding a skateboard next to a man on a skateboard. (temp = 0.4)
- skiies fighter netted guest athletes patiently t-ball ollie frontal ostrich wonderful cloth-covered stunts black shapped wildwood badge man quality patrol (temp = 5)
- a man is riding a bike down the street. (temp = 0.0001)
- a man is riding a bike down the street. (deterministic)

Figure 15: Example of bad caption generated by the model in task 2
Original captions:
- a single elephant standing in the sun in front of fence.
- the elephant stands in front of the fence in and trees.
- an elephant curls his trunk in a fenced area.
- an elephant inside an enclosure looking at the camera.
- an elephant swinging its trunk inside of a pen.
Predict captions:
- a person riding a bike near a parked car (temp = 0.4)
- the girafes barbershop posture abot attempt pews bellow electronic motorhome croissants dos pike protector drapes video dated ranger phones hauler (temp = 5)
- a man is riding a bike down the street. (temp = 0.0001)
- a man is riding a bike down the street. (deterministic)

6 Discussions

The result is relatively intuitive. The best model-1 $hid_{1024}emb_{300}$, which achieves the best bleu-1 score of 48.502 in task 1, has more hidden units to better learn complex representation than the other models. What's more, it takes advantage of 300 as its embedding size, which is known to provide excellent performance in various tasks without a lot of over-parametrization risk[9]. The worst model-1 $hid_{512}emb_{500}$, in contrast, has too large a embedding size and thus risk over-parametrization, and thus has the lowest bleu-1 score of 43.669 in task 1. The best model-2 $SGD_{5e^{-2}}$, which achieves the best bleu-1 score of 50.267 in task 2, has SGD optimizer that converges much slower than Adam, when accompanied with relatively large learning rate, together achieves high bleu-1 score. On the contrary, the worst model-2 $SGD_{5e^{-4}}$ didn't pair the learning rate properly with the SGD optimizer. As SGD optimizer converges very slowly, it needs a much larger learnning rate (i.e $5e^{-2}$) to converge. As a result, this model-2 gets a terrible bleu-1 score of 0.085, which means the model is not even close to convergence at all.

The deternministic approach would not work well because it simply takes the maximum at each step. The lack of flexibility makes it hard to tackle tasks like image caption generation, where a lot of randomness are involved. [11] For example, in this picture, the stochatic approach with temperature of 0.4 accurately predicts that the image is "a man riding skateboard in the air", while the deterministic approach predicts "a man is riding a horse on a dirt road", which doesn't make sense at all. The performance of stochastic generation is similar to deterministic generation at temperatures close to 0. When temperatures reached 0.4, stochastic generation outperformed deterministic generation by a large edge. Even higher temperatures, nevertheless, cause a sharp decline in performance. Higher temperatures signify a more regular distribution of the chosen phrases, which is consistent with our intuition. The deterministic scenario is similar at lower temperatures. As a result, the subtitles become absurd as the temperature rises much, and the model's performance plummets.

We can observe that the model performs admirably when the things that need to be identified are straightforwardly generalizable, such as when recognizing common objects and the image of a person. We can readily observe that certain salient objects or persons can be identified rather accurately in the good sample. Common context for the many categories of items seems to make object identification more accurate. Salient aspects of humans, like faces seen in the photograph or the people themselves, make it simple to recognize them. If the item is easily generalizable, the model can still recognize the prominent objects despite visual distortion.

The model does not perform as well, though, when there are more minute features to be considered. The model misidentifies a part of elephant as a man in Figure 8, cannot tell a locomotive from a skateboard in Figure 13, and sopmetimes cannot tell a woman apart from man in Figure 14. We can see that despite the model's best efforts, it has constructed something that does not match the image, claiming that there are two people on the skateboard rather than two woman with umbrella. We can see how the model may have confused contextual cues with real image information in the majority of these errors. Considering the part of the elephant as a shadow, the part may look like a man in black, but due to the existence of an elephant nose and teeth, the model gives the person a suit and tie as well. With the the train image, we can see that the model is likely to mistake one side of the train for a skateboard since a skateboard is more likely to be flat and rectangular. As a result, the model predicts that there is a skateboard rather than one side of a train. The bias in the original ResNet50 training dataset, which makes some predictions more likely in certain settings, might be a potential cause of these errors.

7 Contribution

Xiaoyan He, Dongze Li, and Yunfan Long equally distributed all the work in implementing the CNN and LSTM Architechture and carrying out experiments, tuning hyperparameter, and debugging. We took turns that one wrote a test and the other made the test pass in the process of completing this project. During this process, we met together in library, took turn to serve as the primary programmer when constructed CNN and LSTM, transfer the learning from Resnet-50, and conduct all the experiments together. During hyperparameter tuning, we all ran same amount of load during the two weeks.

Beyond the equally distributed works in implementing the CNN and LSTM network, every of us undertake some extra work in different areas:

Dongze Li: hyperparameter tuning of the hidden size for custom CNN in task 1 and implementing code for stochastic/deterministic caption generation.

Xiaoyan He: hyperparameter tuning of the optimizer and learning rate for LSTM in task 2 and implementing code for bleu-1 and bleu-4 metrics evaluation.

Yunfan Long: hyperparameter tuning of the embedding size for custom CNN in task 1 and and implementing code for loss calculation, visualization, and management/deployment of experiment result.

Note that we collaborate and help each other with debugging and testing in every aspect of this project along the way.

All of us contributed equally on writing the report.