IE534/CS 598 - Project Report

Show and Tell: A Neural Image Caption Generator

1. The Team

Name	NetId
Avinash Kadimisetty	avinash9
Rachneet Kaur	rk4
Daksh Saraf	dsaraf2
Ankit Kumar	ankitk3

2. Introduction

This project aims to build a generative model using latest techniques in Computer Vision and Machine Translation to describe an image. Previous works in the area of Image Captioning involved combining solutions of each of the sub-problems to generate a text description from an image. Since the goal of this project is not just getting a visual understanding of the image but also expressing the image in a language like English, advances in translation are quite useful. In the previous works related to Machine Translation, the translation was achieved through a series of individual tasks like translating each word separately, reordering words etc. but there is a better way of doing this using Recurrent Neural Networks. In RNN, an input sentence is first transformed into a compact representation using an encoder RNN and then this representation is used as an initial hidden state of a decoder RNN that outputs the translated sentence. In this project, a single joint model is constructed to generate a target sequence of words that describes the input image. The idea is similar to encoder-decoder RNN used in translating sentences except that a Convolutional Neural Network is used in place of the encoder RNN. CNN, which takes image as an input, is trained for an image classification task to generate a compact representation of the original image. This representation is then passed as an input to a decoder RNN that generates the sentences.

3. Architecture

In Machine Translation, the state-of-the-art models used a RNN to encode the given input sequence into a fixed vector representation and used this representation to decode it to a desired target sequence. The same process is used in captioning images as well. In the Show and Tell model, word embedding is used as a representation of the input image

and acts as an initial state of the LSTM. Here, LSTM is used as the decoder because they have shown state-of-the-art performances on tasks related to machine translation which also have the ability to deal with vanishing and exploding gradients which are frequently encountered in training RNNs. The input of the LSTM is tokenized by mapping each unique word to a distinct number. All the LSTMs share the same parameters. The LSTM memory size will be set to 512 dimensional. We also experimented with GRU units also along with LSTM units, and tried more layers for both the architectures. We used a batch size of 32, since while training for a larger batch size, we were facing out of memory issues on Blue Waters. We used the maximum sequence length for the predicted captions to be 25.

4. Data Processing

We worked with two datasets namely MSCOCO 2014 [5], and Flickr30k [6]. For each of the datasets, we created a vocabulary based on the captions of each of these datasets individually. We considered a word only if it appears at least 5 times in the dataset. Every caption is padded with start and end tags and non-frequent words are replaced with unknown tags. Once the vocabulary is built, each caption of the image is tokenized using the vocabulary and is used as the target caption for training the model. The MSCOCO dataset has ~80K training images, ~40K validation set images with at least 5 captions per image. The Flickr30k dataset has ~30K training images and ~1K validation set images. For data augmentation in each of the datasets, we first resize the images to 224 X 224 (since we used the pretrained Resnet models as our CNN architecture). Then we applied the following augmentation methods: Horizontal flip, Vertical Flip. All the images are normalized and then given as input to the model.

5. Hyperparameter Exploration

We tried various optimizers namely ADAM with varying initial learning rate = 0.01, 0.1, 0.0001 etc., SGD with momentum 0.9 and initial learning rate = 0.1, 0.01, etc. and an adaptive learning rate decay. We used the cross entropy loss function and for regularization, we used dropout. We also used early stopping to prevent overfitting of the test time loss function. Also, while inference sampling, we used greedy and beam search. We explored various beam sizes, namely 3, 5 and 7 for the optimal results. For the CNN architectures, we explored various pretrained architectures on the ImageNet dataset, namely Resnet50, Resnet101 and Resnet152. For the embedding dimension, we used 512 dimensions. For the hidden units for the RNN, we explored LSTM and GRU units with the no. of hidden units as 512, 1024 etc. Next, we tuned the no. of layers of the RNN structure. We tried 1, 3, 5, 7 and 10 layers for both LSTM and GRU layers with both greedy and beam search inferences. Approximately, we trained our models on MSCOCO datasets for ~25 epochs, with each epoch running for about 2.5 hours. For the Flickr30k dataset, we trained our models for ~50 epochs. The following plots show the variation of training and test loss for some of the architectures we trained.

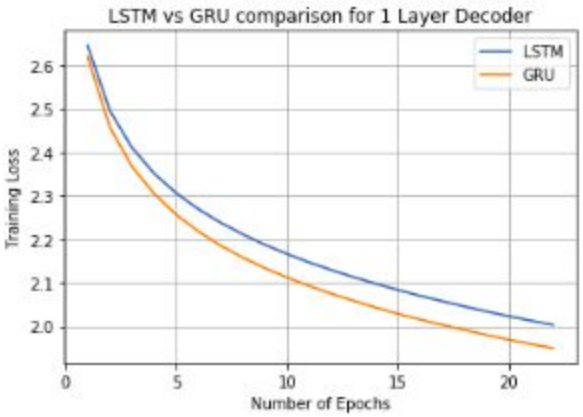


Figure 1: Comparing the loss function for 1 layer LSTM and GRU models for 23 epochs

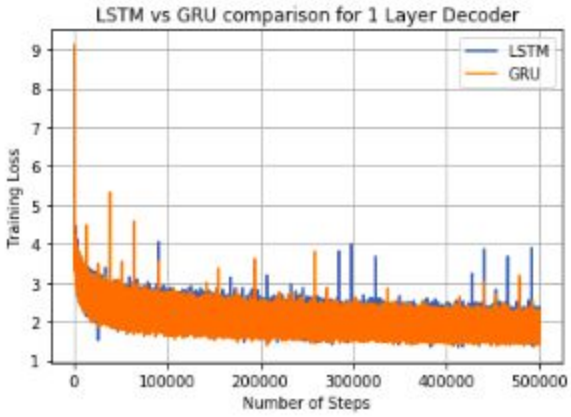


Figure 2: Step wise training loss of LSTM and GRU decoders

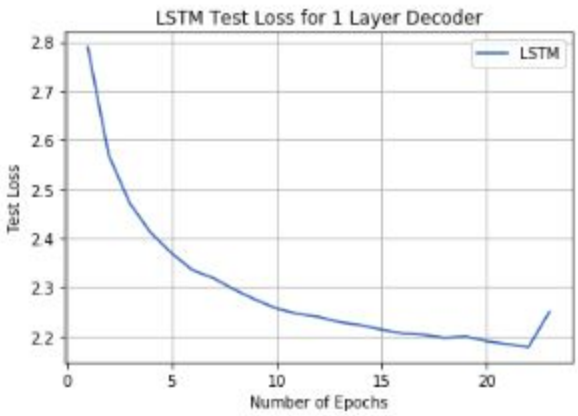


Figure 3: Test loss of LSTM Decoder with 1 layer

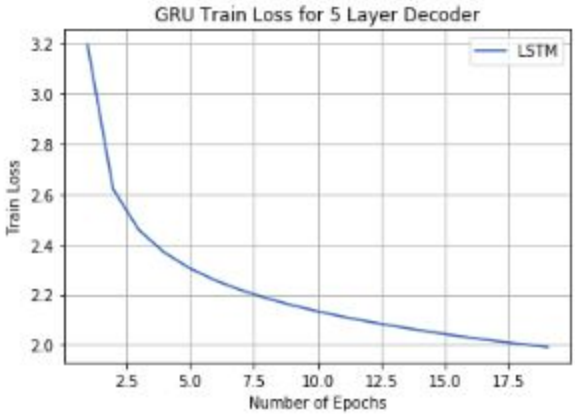


Figure 4: Test loss of GRU Decoder with 5 Layers

In Figure 1, we compare the decay of the loss function for the 1 layer LSTM and GRU models trained for 23 epochs. We can notice that the loss function for our GRU model decreases faster than the the LSTM model. We did similar experiments with multiple layer LSTM and GRU models as well, and observed a similar pattern.

Hence, we conclude that GRU models perform better than LSTM. Figure 2, we can notice a noisy stepwise loss as compared to epoch wish loss in Figure 1. In Figure 3, we notice that the testing loss function increases after ~23 epochs hence, we use early stopping to decide the optimal trained model. Figure 4 is the loss function of the best model we trained, i.e. GRU with 5 layers.

6. Evaluation metrics and Inference Sampling

We have evaluated our results on several metrics, namely BLEU 1 [3], BLEU 4, CIDEr [4] and ROUGE_L [8]. While testing, we used two methods to perform inference sampling namely greedy search and beam search. For beam search, we tried beam sizes 3, 5 and 7. Although the paper used a beam size of 3 we got best results with a beam size of 5. We observe that on an average Beam search improved the c1 BLEU score by 1 point.

7. Quantitative Results

The following table shows the quantitative results of some of the models we tried on MS COCO dataset. We initially calculated only BLEU score with and without beam search for the models. We later on also calculated the CIDEr and ROUGE_L scores for only the captions generated by the best model. The BLEU scores shown below are c1 scores i.e, each prediction caption is evaluated against each of the target caption against which it was predicted for. We also calculated c5 score by comparing the predicted captions with all the captions of the image and picking the best caption. We have only tabulated the results for Resnet50 with LSTM and GRU units, although we calculate the scores for Resnet101 as well. We get the best models with Resnet50.

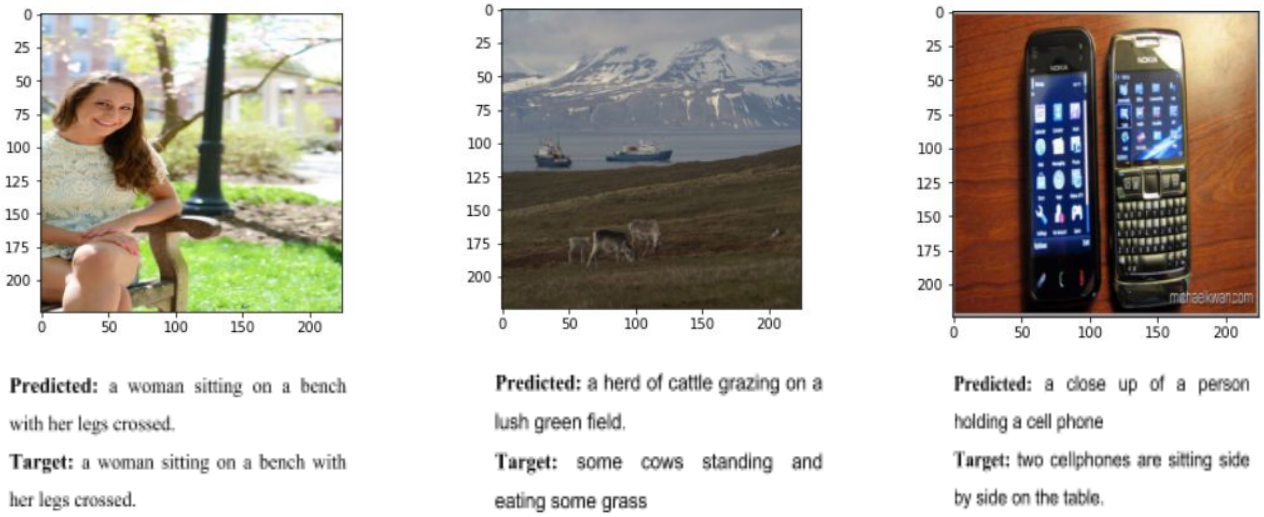
CNN Type	RNN Type	Number of Layers	Number of Hidden Units	Epochs	Testing Loss	BLEU1	BLEU4	BEAM_BLEU1	BEAM_BLEU4
Resnet50	LSTM	1	512	20	2.18	23.19	20.34	NA	NA
		3	1024	20	1.98	22.83	19.43	-	-
		5	512	20	2.17	23.79	20.4	24.61	21.4
		7	512	20	2.96	24.05	18.6	27.38	20.56
		10	512	5	4.65	24.83	21.01	25.41	21.18
Resnet50	**GRU**	1	512	63	1.51	30.12	20.79	30.53	21.02
		3	1024	20	1.62	30.02	20.27	-	-
		5	**512**	**20**	**1.91**	**30.26**	**21.39**	**30.86**	**21.94**
		7	512	20	2.09	28.05	21.41	29.5	20.33

For Flickr30k, we have used the same architecture that gave the best results on COCO dataset which is Resnet 50 Encoder with GRU Decoder of 5 Layers. The following table provides a comparison of our results with the results from Google on the MSCOCO and Flickr30k datasets. The scores below are c5 scores - we compare each caption against 5 of the captions of that image and pick the caption with best score.

Dataset	BLEU1 Our code | Paper	BLEU2 Our code | Paper	BLEU3 Our code | Paper	BLEU4 Our code | Paper	CIDEr Our code | Paper	ROUGE_L Our code | Paper
COCO	64.6 | 71.3	45.9 | 54.2	31.7 | 40.7	22.0 | 27.7	69.4 | 85.5	47.6 | 53.0
Flickr30k	53.9 | 63	34.8 | NA	22.0 | NA	14.3 | NA	26.5 | NA	39.5 | NA

8. Qualitative Results

For some of the MS COCO dataset test images, the predicted captions are as follows:



The first image is an example of a good caption along with a good BLEU score, the second image is an example of an image with a good caption along with a poor BLEU score. This occurs because the BLEU is not a great metric evaluating image captions, it looks at the 4-gram score and evaluates the quality of the caption if multiple identical words occur consecutively in a sentence which is not a very accurate indicator of a good caption. The last image is an example of a poor caption and a poor BLEU score.

9. Computational Resources used

We implemented the architectures discussed above with MSCOCO and Flickr30k datasets in Pytorch, with several other libraries, namely, numpy, NLTK, etc. and used Blue waters and Google Colab for the computational purposes. We used ~**873 GPU hours on Blue Waters and 42 GPU hours on Google Colab (Free GPU hours)** for our project. Before deploying the code on Bluewaters we tested our code on Colab to check for the correctness. The rough estimate of hours for different parts of the project listed as follows:

- Preparing and preprocessing of MSCOCO, Flickr30k data-sets.
  This required memory of about 200GB. *Hours used:* 5 hours
  The major tasks and the computation hours for the MSCOCO dataset are mentioned below:

- Training the last 2 layers (linear and batchnorm) of the Reset 50/101 and all the parameters of the 1/3/5/10 layer LSTM/GRU networks. *Hours used:* 260 hours
- Hyperparameter selection to obtain optimal results for the CNN and the RNN network, namely no. of hidden units (512/1024), learning rate (with and without decay), no. of hidden layers (1/3/5/10), optimizer selection (SGD with momentum/ADAM) in the LSTM/GRU network. *Hours used:* 430 hours
- Generating captions with images and inference sampling with greedy search vs beam search (with beam sizes 3, 5, 7) testing comparisons. *Hours used:* 140 hours

- Replicating the best model on Flickr30k, using the pretrained model of COCO on Flickr30k. *Hours used:* 80 hours

10. Contributions

- Avinash implemented the data loaders for MSCOCO, Flickr30k datasets, implemented the beam search for sampling captions and the testing code. He also worked on optimal hyper parameter tuning by increasing the number of hidden units, embedding length, types of RNN units (LSTM/GRU) and the number of RNN layers. Helped with writing the proposal, presentation and the report.
- Daksh implemented a learning rate scheduler for hyperparameter tuning and experiment with various encoders like Resnet101, Resnet50 and number of hidden units. Helped write the test code for testing the model and generated a script to generate images and captions as a end result. Helped with writing the proposal, presentation and the report.
- Rachneet integrated the training and testing code with Resnet50, Resnet101, RNN code in the pipeline. She also did hyperamater exploration with different layers of LSTM and GRU models. Helped with writing the proposal, presentation and the report.
- Ankit implemented the evaluation script to evaluate BLEU, CIDEr and ROUGE_L scores and extended COCO evaluation code [2] to generalized formats. Helped in the code for mapping caption and image ids for end output. Helped in the result section of report.

References

1. Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan. "Show and tell: A neural image caption generator." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156-3164. 2015.
2. COCO evaluation scripts https://github.com/tylin/coco-caption/tree/master/pycocoevalcap
3. Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: a method for automatic evaluation of machine translation." In Proceedings of the 40th annual meeting on association for computational linguistics, pp. 311-318. Association for Computational Linguistics, 2002.
4. Vedantam, Ramakrishna, C. Lawrence Zitnick, and Devi Parikh. "CIDEr: Consensus-based image description evaluation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4566-4575. 2015.

5. Chen, Xinlei, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO captions: Data collection and evaluation server." *arXiv preprint arXiv:1504.00325* (2015).
6. Plummer, Bryan A., Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models." In *Proceedings of the IEEE international conference on computer vision*, pp. 2641-2649. 2015.
7. Lin, Chin-Yew. "Rouge: A package for automatic evaluation of summaries." *Text Summarization Branches Out* (2004).
8. PyTorch tutorials https://github.com/yunjey/pytorch-tutorial

Code Submission:

We created training and testing scripts for 2 datasets namely:

- MSCOCO
- Flickr30K/8K

Inside each of the corresponding folders for the datasets, we have a train.py and test.py

Next we have evaluation scripts for the following metrics, in the folder Eval_Scripts:

- BLEU 1
- BLEU 4
- CIDer
- ROUGE_L

We submit .pbs files namely train.pbs and test.pbs required to run the train and test scripts respectively.

We also submit file to generate the qualitative results of images along with captions, called Qualitative_results.py