IE598JS Project Report
Image Captioning Using Show and Tell Model

Ankit Rai*, Safa Messaoud†, Tarek Elgamal‡ and Gizem Tabak§
(*rai5, †messaou2, ‡telgama2, §tabak2)@illinois.edu

I. Introduction

In this project, we built a pipeline consisting of two models. The first model, Show and Tell (Im2Txt) [1] generates a caption from an image. The second model Txt2Im [2] produces an image from the generated caption. This experience has exposed us to the exciting area of multi-modal representation learning. Both models learn a shared feature representation across the image and natural language modalities and are trained to predict missing data in one modality conditioned on data from another modality.

A large number of image annotation models have been proposed recently that exploit the synergy between visual and textual modalities by learning the correspondence between image regions (or features) and keywords. Early work of neural network-based image captioning models include the multimodal RNN and LSTM [3][4]. In these methods, neural networks are used for both image-text embedding and sentence generating. These models were later extended with attention mechanism [5]. The closest work to the model we used is the work in [6]. Similar to show and tell, the model in [6] also include pretrained CNN on ILSVRC-2012-CLS image classification dataset and the last layer of the CNN is connected to an LSTM layer that predict subsequent words of the sentence. The core idea of the two models are similar, however, show and tell model is significantly better according to the first author of [6] as a result of better CNN and more careful engineering.

The project report is structured as follows: In Section II, we describe the Show and Tell model. Details of the implementation as well as the results are presented in Section III. Section IV is about the Txt2Im model. Information about our code can be found inn Section V.

II. The Show and Tell Model Description

The Show and Tell model [1] is "end-to-end" trained to generate a caption given an image by maximizing the likelihood of the target description sentence given the training image, as it is shown in Figure 1. The image is encoded using a convolutional neural network (Inception Net) into a rich fixed length vector representation. This representation is fed to a recurrent neural network with LSTM cells. The LSTM is trained to predict each word of the sentence as well as all proceeding words $p(S_t|I, S_0, \cdots, S_{t-1})$, where S_t represents each word as a one-hot vector:

$$x_{-1} = CNN(I)$$
$$x_t = W_e S_t$$
$$p_{t+1} = LSTM(x_t)$$

The loss function used in the model is the sum of the negative log likelihood of the correct word at each step as follows:

$$L(I, S) = \sum_{t=1}^{N} logp_t(S_t)$$

The above loss function is minimized with respect to all the parameters of the LSTM, the top layer of the image embedder CNN and word embeddings W_e.

During inference, we use beam search by iteratively considering the set of the k best sentences up to time t as candidates to generate sentences of size t + 1. In our experiments, we use a beam size equal to 20.

III. Implementation and Results

Show and Tell model consists of two networks stacked on top of each other: One is a convolutional neural network, and other is the sequential network, which is an LSTM in the paper [1]. Since the CNN expected to be used in this work is a very deep one, instead of training it from scratch, pretrained Inceptionv3 model is used and its checkpoints are fed

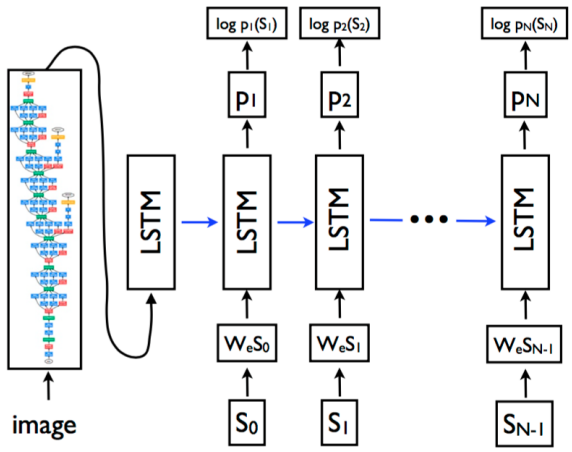

Fig. 1. Show and Tell Model for Caption from Image Generation **??**

TABLE I
DIFFERENT MODELS USED ON FLICKR8K HYPERPARAMETER TUNING

	Dropout	Learning Rate	Sequential Model (Nodes)	Optimizer (Batch Size)	Training Loss
BL	0.3	1	LSTM (512)	SGD (32)	2.4651
M1	0.5	3	LSTM (512)	SGD (32)	2.5575
M2	0.5	3	GRU	SGD (32)	2.3606
M3	0.3	0.01 (Decay 0.7)	LSTM (512)	Adam	2.7263
M4	0	1	LSTM (512)	SGD (32)	1.6204
M5	0.5	0.5	RNN	SGD (16)	3.256
M6	0.3	1	LSTM (256)	SGD (32)	2.5599
M7	0.3	0.01 (Decay 0.9)	LSTM (512)	Adam	2.5544

into the network as features before the last fully connected layer [**?**]. This way, we only needed to tune sequential model with relatively less training time.

The baseline Show and Tell model was trained for MSCOCO dataset with 82783 training, 40504 validation and 40775 test images. However, we used Flickr8k dataset with 6000 training, 1000 validation and 1000 test images. Since the content and the scale of these two datasets are different from each other, we expect to have different optimum hyperparameters. Thus, we tried different models derived from the baseline model. A few parameters we modified are the dropout probability, learning rate, different RNN types for sequential modeling and their intrinsic parameters such as number of nodes, and different optimizers. A more detailed introductions of the models we tried can be found in Table I. While finding the optimal model for our case, we trained the model on Flickr8k dataset for 15k iterations and calculated training loss. Then, we calculated BLEU scores and caption probabilities. We encountered issues with BLEU scores, which will be explained later in this section, so we used probability scores instead of BLEU score or other machine translation metric for model comparison.

For each model in Table I, we calculated BLEU scores with different n-grams. Basically, how BLEU score works is, it generates subsets of n words from a text and compares its "similarity" to the ones generated from the real text. It is usually used for machine translation evaluation and for corpus-level translations.

Based on these scores in Fig. 2, Model 3 (M3) and Model 7 (M7) seem to be the best performing ones based on BLEU scores. However, when we check a random image with generated captions, we noticed that their outcomes are noticeably less related to the image (Fig. 3). Noticing that they are the models with ADAM optimizer, we guess parameters used with ADAM were not appropriate for proper training. We tried ADAM optimizer with different hyperparameters such as learning rate, decay, number of decay epochs, etc. but we still could not find a model that is properly trained with ADAM.

When we check the captions for other images generated by Model 3 and Model 7, we noticed that they generate the same captions for all the images, which means they are not trained properly. However, their BLEU scores are still higher than other models. This implies that BLEU score is not a reliable metric in sentence-level caption comparison. Thus, we looked

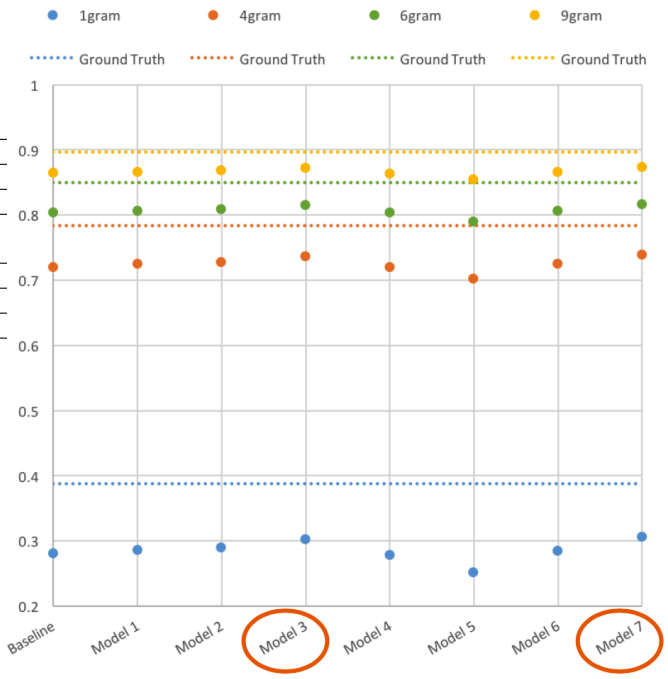

Fig. 2. BLEU scores of different models with different ngram lengths

for another comparison metric. Instead of using a machine translation metric, we decided to use log probabilities of the generated captions, since it was the actual objective function on which the models were trained. Thus, comparing the models on this objective function made sense. Log-probabilities of the captions can be thought as the confidence of the model in its generated caption. If it is high, the model has a high confidence with its generated caption, and it is likely to be close to the original caption. If it is low, it is closer to a random guess. A comparison of log-probabilities of each model can be seen in Fig. 4. Here, Model 4 performs the best among all 8 models. When we compare all the models with the baseline (Table I), it seems like Model 4, whose only difference with the baseline is not having dropout, topped the baseline model. This makes sense since the original baseline model given in the paper is trained on MSCOCO, and it is a bigger dataset

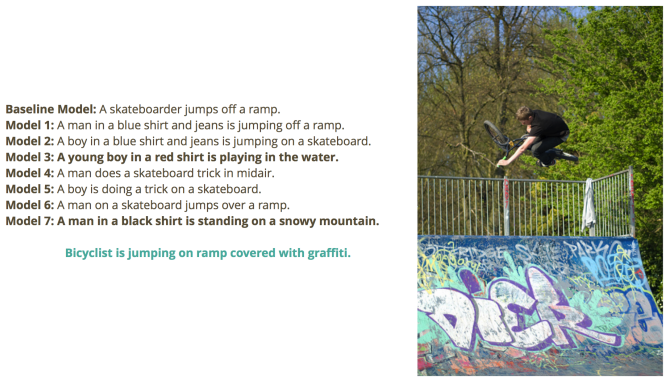

Fig. 3. Real (blue) and generated captions with all 8 models for the image (on the left)

TABLE II
TRAINING LOSSES, BLEU SCORES AND LOG-PROBABILITIES OF EACH MODEL

	Training Loss	BLEU Score	Probability (Confidence)
BL	2.4651	0.28	0.001722
M1	2.5575	0.286	0.000833
M2	2.3606	0.29	0.000583
M3	2.7263	0.303	0
M4	1.6204	0.279	0.002017
M5	3.256	0.251	0.001268
M6	2.5599	0.285	0.000415
M7	2.5544	0.306	0

compared to the Flickr8k data we used in this project. Since the baseline model had more samples, it had higher chance to overfit the data and it required regularization. Since our dataset is smaller than theirs, the model probably did not have enough data to overfit. So, removing dropout helped to improve results. After finding the best model, we trained it for a longer time, with 50k steps and generated captions.

Training results and some examples of the generated captions can be found in Appendix. Images included in the appendix are randomly selected, and even the ones that have relatively low log-probability value (Fig. 11, Fig. 13, Fig. 14), are somehow related to the image.

Another interesting outcome we noticed was the model performs noticeably good with dog images. Although we did not conduct a detailed analysis on the reason, we guess one reason might be there are too many dog images in the training set and the model became biased, and another is, dog image usually contain less unique actions, more similar settings, and represented with similar captions compared to the images that contain humans.

IV. EXTENSION: IMAGE GENERATION FROM CAPTION

We extended the image to caption generation pipeline with a model text2im that performs the reverse operation, namely generating an image from a caption. For this purpose we use the architecture introduced by Mansimov et al in [2]. The proposed model, shown in Figure 5 iteratively draws patches on a canvas c_t, while attending to the relevant words in the description $y = (y_1, y_2, .., y_N)$. First, a bidirectional RNN transform every word y_i in an m-dimensional vector representation h_i^{lang}. Next, a sentence representation s_t is computed at every time step through the weighted sum of the single words using alignment probabilities $\alpha_{1...N}^t$.

$$s_t = \alpha_1^t \cdot h_1^{Lang} + \alpha_2^t \cdot h_1^{Lang} + \cdots + \alpha_N^t \cdot h_N^{Lang} \quad (1)$$

The alignment probability α_k depends on the word representation h_k^{lang} and the current state of the generative model h_{t-1}^{gen}.

$$\alpha_t^k = \frac{exp(v^T \tanh(Uh_k^{Lang} + Wh_{t-1}^{gen} + b))}{\sum_{i=1}^{N} exp(v^T \tanh(Uh_i^{Lang} + Wh_{t-1}^{gen} + b))} \quad (2)$$

For image generation, a recurrent variational auto-encoder is used. During training, an encoder network (Inference RNN) determines a distribution $P(Z_t|Z_{1:t-1})$ following a normal distribution with mean $\tanh(W_\mu h_{t-1}^{gen})$ and variance $\tanh(W_\sigma h_{t-1}^{gen})$ over latent codes that capture salient information about the input data. A decoder network (generative RNN) receives a sample z_t from the code distribution and uses it together with the sentence s_t representation to condition its own distribution over images. The encoder network is removed during inference. At every time step, the write operator produces two Gaussian filters $F_x(h_t^{gen})$ and $F_y(h_t^{gen})$, which determine "where to write" and an image patch $K(h_t^{gen})$ which refers to "what to draw", as it is show in the equation below:

$$c_t = c_{t-1} + write(h_t^{gen}) + F_x(h_t^{gen})K(h_t^{gen})F_y(h_t^{gen})^T \quad (3)$$

To produce the final image x, every pixel x_i is set to $\sigma(c_{T,i})$, where c_T is the final canvas matrix. The model is trained to maximize a variational lower bound on the marginal likelihood of the correct image x given the input caption y. The hyperparameters for the model are described in Table III. We were not able to run multiple experiments with different hyperparameters as training for an epoch on BlueWaters took 15 hours. Figures 15, 16, 17 and 18 show examples for running the two models im2txt and txt2im back to back.

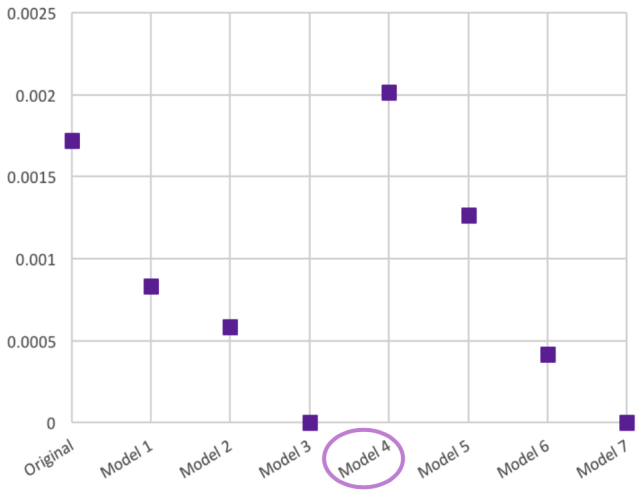

Fig. 4. Log-probabilities (confidences) of each model

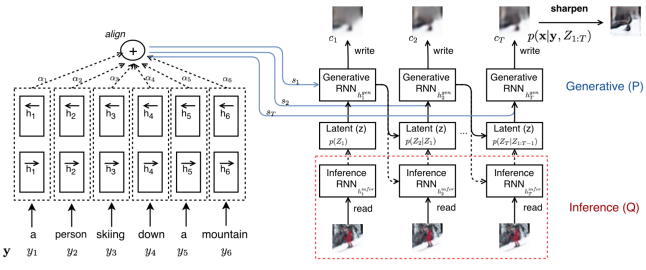

Fig. 5. Architecture for image from caption generation [2]

TABLE III
HYPER-PARAMETERS FOR TXT2IM ARCHITECTURE

Data Set	MS COCO
Optimizer	RMSprop
Parameter Initialization	sampled from $N(0, 0.01)$
Regularization	Gradient Clipping at 10
Learning Rate	0.001
Number of Epochs	6
Size	128
Vocabulary size	25323

V. CODE

We used Google's im2txt Github repository for the baseline[1]. We modified `show_and_tell_model.py` file for model changes, and `configuration.py` for hyperparameter tuning. We also modified `build_mscoco_data.py` to read and save Flickr8k dataset into train, validation and test subsets and also to save original captions. We wrote
`bw_training_script.py`,
`bw_training_script.sh`,
`bw_caption_script.py`,
`bw_caption_script.sh`,
`generate_pbs_captions.py`,
`generate_pbs_train.py`,
`submit_batch_captions.sh`,
`submit_batch_train.sh`,
`calculate_bleu.py`,
`parse_captions.py`
scripts to be able to run the project on BlueWaters in batch mode, parse generated captions and log-probabilities, and calculate BLEU scores.

For the Txt2Im model, we use the code provided by the authors of [2] [2]. As we planned to fine tune the model on Flickr8, we wrote scripts to process the images (dimensionality adjustment/ data augmentation) and dictionary generation from the caption:
`flickr_image_gen.py`,
`genereate_flick8_caption.py` ,
`genereate_flick8_dict.py`
Due to the long time it took to train the model on MSCOCO, we were not able to experiment with the Flickr8 dataset.

VI. CONCLUSION

In this project, we built a pipeline with two major blocks that we trained separately, namely "Show and Tell" which generates a caption from an image and Txt2Im which draws an image given the generated caption. The project exposed us to a wide variety of tools in the deep learning field, including soft attention, generative models, CNN, RNN and encoder-decoder which enable translation between different modalities. We also gained experience in Tensorflow and Theano.

REFERENCES

[1] Vinyals, O., Toshev, A., Bengio, S. and Erhan, D., 2015. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3156-3164).

[2] Mansimov, E., Parisotto, E., Ba, J.L. and Salakhutdinov, R., 2015. Generating images from captions with attention. arXiv:1511.02793.

[3] Chen, Xinlei and Zitnick, C Lawrence. Learning a recurrent visual representation for image caption generation. arXiv:1411.5654, 2014.

[4] Bahdanau et al. "Neural machine translation by jointly learning to align and translate." arXiv:1409.0473 (2014).

[5] Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." arXiv:1502.03044 2.3 (2015): 5.

[6] Andrej Karpathy, Li Fei-Fei. "Deep Visual-Semantic Alignments for Generating Image Descriptions." CVPR 2015.

[1] https://github.com/tensorflow/models/tree/master/im2txt

[2] https://github.com/emansim/text2image

APPENDIX

Fig. 6. A dog swimming in water. (p=0.019199)

Fig. 7. Two men play basketball. (p=0.050603)

Fig. 8. The surfer is riding a wave in the ocean. (p=0.028798)

Fig. 9. A group of people in a blue and white boat. (p=0.000110)

Fig. 10. Two dogs play in the grass. (p=0.008489)

TABLE IV
FINAL TRAINING RESULTS OF DIFFERENT MODELS

	Dropout	Learning Rate	Sequential Model (Nodes)	Optimizer (Batch Size)	Training Loss	BLEU Score	Probability (Confidence)
BL	0.3	1	LSTM (512)	SGD (32)	2.4651	0.28	0.001722
M1	0.5	3	LSTM (512)	SGD (32)	2.5575	0.286	0.000833
M2	0.5	3	GRU	SGD (32)	2.3606	0.29	0.000583
M3	0.3	0.01 Decay 0.7	LSTM (512)	Adam	2.7263	0.303	0
M4	0	1	LSTM (512)	SGD (32)	1.6204	0.279	0.002017
M4 (50k)	0	1	LSTM (512)	SGD (32)	2.3384	0.288	0.002147
M5	0.5	0.5	RNN	SGD (16)	3.256	0.251	0.001268
M6	0.3	1	LSTM (256)	SGD (32)	2.5599	0.285	0.000415
M7	0.3	0.01 Decay 0.9	LSTM (512)	Adam	2.5544	0.306	0

Fig. 11. The man in the white shirt is sitting at a table. (p=0.000042)

Fig. 12. A white dog runs along the beach. (p=0.011407)

Fig. 13. A girl in a pink shirt is walking along a wooden bridge. (p=0.000004)

Fig. 14. A man in a red shirt is standing in front of a white truck. (p = 0.000014)

Fig. 15. A black and white dog swims in the water.

Fig. 16. A brown dog is running through the grass.

Fig. 17. A man in a red jacket is skiing down a snowy hill.

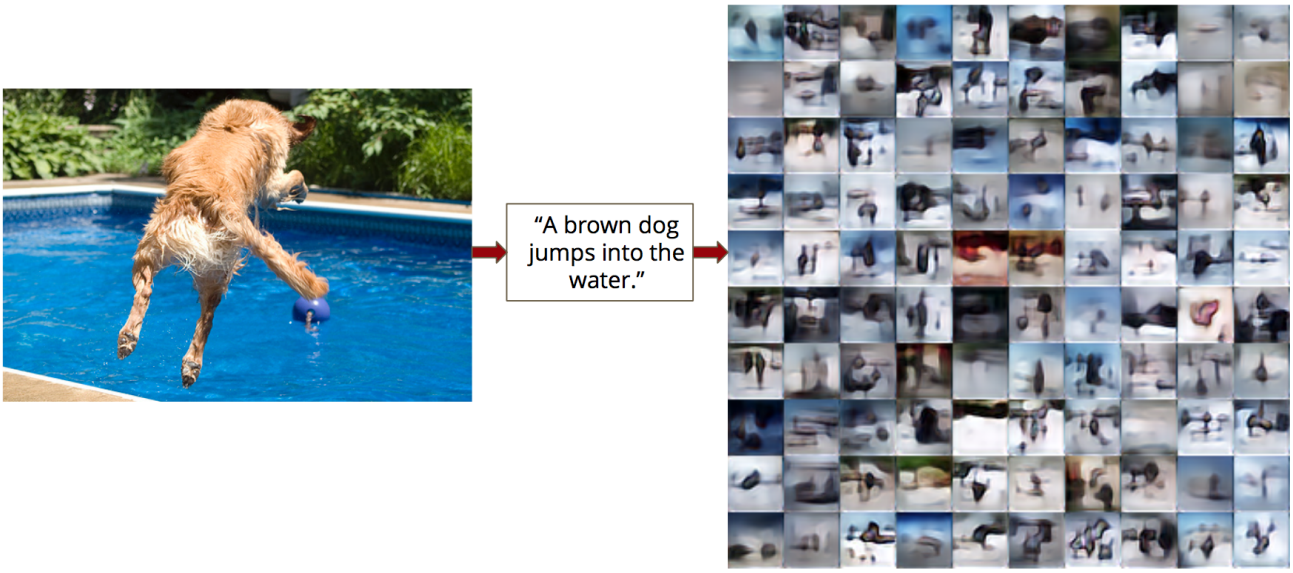

Fig. 18. A brown dog jumps into the water.