ResNet based Image Caption Generator

Dayou Du
New York University
dayoudu@nyu.edu

Xin Chen
New York University
xc1113@nyu.edu

Abstract

Inspired by recent work in image caption and machine translation[12][13], we implement a generative model based on a deep recurrent architecture that can be used to generate natural sentences describing an image. We replace the inception network[10] in the original model with pre-trained Resnet-152[4] and VGG-19[9] from Torchvision package. We also show through visualization how the model is able to automatically learn and generate the corresponding words in the output sequence. Meanwhile, several different RNN model setups has been explored and compared. A beam search decoder has also been implemented to achieve a better performance. At last, we achieve a BLEU-4 score of 26.9 on MS COCO 2017 dataset[1], which is quite close to the 27.7 of NIC model in the original paper[12], and also surpass the Nearest Neighbor method (9.9) and the Human result (21.7).

1. Introduction

Automatically generating captions for images has became as a prominent interdisciplinary research problem in both academia and industry. It is a challenging but powerful task, for example it can help visually impaired people better understand the content of images on the web. The description must capture not only the objects contained in an image, but also must express how these objects relate to each other as well as their attributes and the activities they are involved in. Moreover, we need to decode those descriptions into natural language sentences.

Previous attempts have been proposed to combine existing models to solve the above sub-problems. In the area of natural language processing, an "encoder" would map raw inputs to feature representations, while "decoder" can take this feature representation as input, process it to make its decision, and produce an output. Here We follow the recipe, take an image I as the input and use a CNN model as feature extractor, in which the last fully connected layer is replaced to match the size of fixed-length word vector. It's nature to call the CNN part "encoder". Then a RNN "decoder" is trained to maximize the likelihood $p(S|I)$ of producing a target sequence of words $S = \{S_1, S_2, ...\}$ where each word S_t comes from a given dictionary, that describes the image adequately.

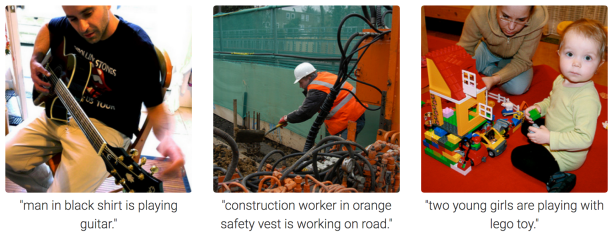
"man in black shirt is playing guitar."
"construction worker in orange safety vest is working on road."
"two young girls are playing with lego toy."

Figure 1. Image Captioning. *Source: Google Image*

We first freeze the pre-trained CNN models to train the RNN model, then unfreeze CNN part when the loss of RNN becomes stable. We also compare the performance of different "encoders" models such as Resnet and VGG, different word embedding size and RNN hidden size, finally we achieve a BLEU-4 score of 26.9 with a fine-tuned Resnet-152 "encoder" and beam search in inference stage.

2. Related Work

The problem of annotating images with natural language at the scene level has long been studied in both computer vision and natural language processing. Several methods have been proposed for generating image descriptions. Hodosh *et al.* [6] proposed to frame sentence-based image annotation as the task of ranking a given pool of captions. Recently many of these methods are based on recurrent neural networks and inspired by the successful use of sequence to sequence training with neural networks for machine translation[2]. One major reason image caption generation is well suited to the encoder-decoder framework [2] of machine translation is because it is analogous to translating an image to a sentence.

The first approach to use neural networks for caption generation was Kiros et al. [7], in which they constructed a joint multimodal embedding space using a powerful deep

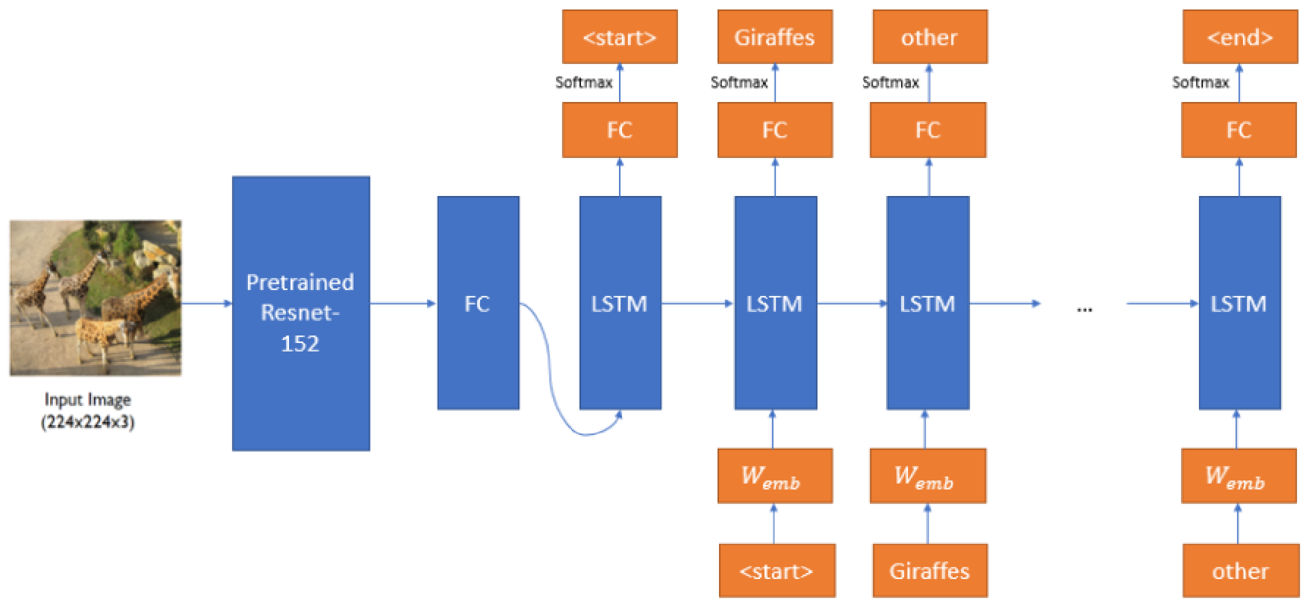

Figure 2. General CNN-Encoder/RNN-Decoder Model Structure

CNN model and an LSTM that encodes text. Vinyals et al. [12] combined deep CNNs for image classification with an LSTM for sequence modeling, to create a single network that generates descriptions of images. Xu et al. [13] proposed a model based on visual attention. The CNN-RNN approach has the advantage that it is able to generate a wider variety of captions, can be trained end-to-end, and outperforms the previous approach on the benchmarks.

3. Methods

3.1. Data Pre-processing

Microsoft Common Objects in Context (COCO) dataset [1] is a large-scale object detection, segmentation, and captioning dataset, in which every image has 5 captions. There are 118K images in our training set contains and 5K images in the validation set.

For the images, we use transforms operations in torchvision package for data preparation and augmentation. In the training process, we randomly crop and then resize and normalize the image to match the requirement of CNN model, while in the inference process (model would output the description), no randomness is needed thus we only need resizing and normalization. The randomly resizing is also helpful for us to load all captions of a single picture without worry about model divergence.

For the sentences, we first use Natural Language Toolkit (NLTK) for tokenization. Then add special words like $\langle start \rangle$, $\langle unk \rangle$ into the vocabulary set. Thresholds are also needed to discard some low-frequency words. Every word in the vocabulary set are mapped into a index so with the help of embedding layer, the input word could be transformed into a vector. The output part is quite similar to the input part instead that the generated vectors would be translated to a word.

Here, we do not use pre-trained embedding models such as GloVe since it will not result in a better performance. These pre-trained embeddings are trained with different objectives while our embedding matrix need to be trained with LSTM models and has different vocabulary set.

3.2. Models

A general CNN-Encoder/RNN-Decoder model structure is illustrated in Figure 2.

On the encoder side, the image is re-sized and fed into a CNN model, which will generate the image feature embedding. Then, a fully-connected(FC) layer is in the middle to transfer the image features into the Long short-term memory(LSTM)[5] input feature space.

On the decoder side, the very first input to the RNN network is the embedded image features. Then the train/inference process generally follows a typical language generator RNN network: a word embedding layer is used to compute the input word features. A log-softmax layer with a FC layer will compute the score of each word on each time-step, from the output of RNN networks.

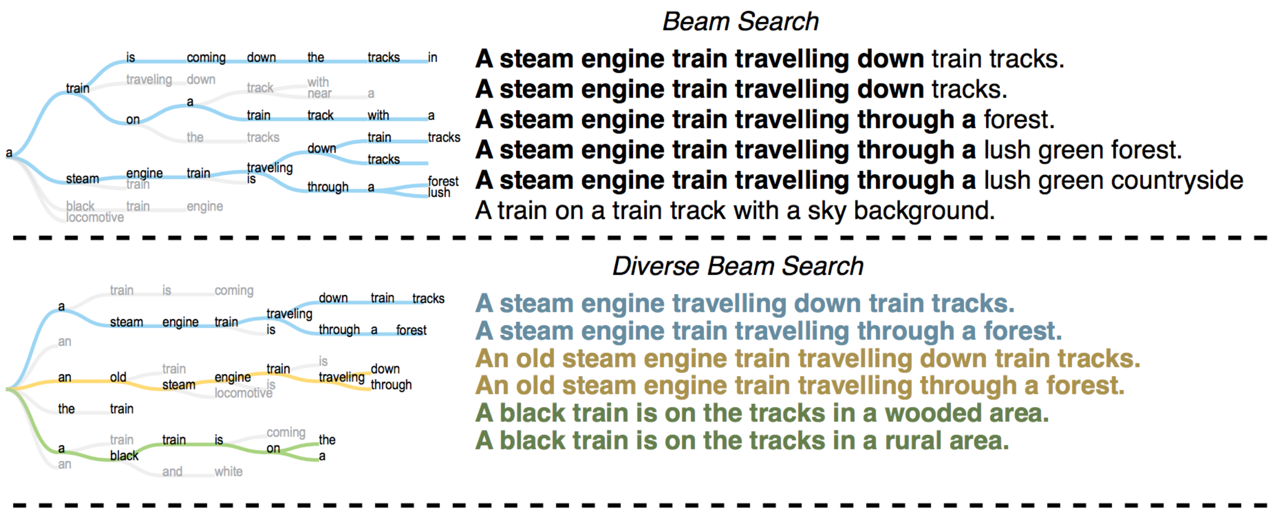
Beam Search
A steam engine train travelling down train tracks.
A steam engine train travelling down tracks.
A steam engine train travelling through a forest.
A steam engine train travelling through a lush green forest.
A steam engine train travelling through a lush green countryside
A train on a train track with a sky background.
Diverse Beam Search
A steam engine travelling down train tracks.
A steam engine train travelling through a forest.
An old steam engine train travelling down train tracks.
An old steam engine train travelling through a forest.
A black train is on the tracks in a wooded area.
A black train is on the tracks in a rural area.

Figure 3. Beam Search and Diverse Beam Search. *Source: Vijayakumar et al.[11]*

3.2.1 CNN encoders

In this project, we explored both Resnet-152 and VGG-19 as the base structures of CNN encoders. To speed up training process and achieve a better performance, we used the pre-trained models from torchvision and removed the very last classification layer. Instead, we used an extra fully-connected(FC) layer to produce the image embedding, which will be feed as the first input into the LSTM. The base CNN-network is pre-trained on Imagenet[3], which could provide us a strong and robust image feature generator.

3.2.2 RNN decoders

As suggested in [12], we used LSTM units to build our RNN model. To extend the original paper, we also tried different setups and compared their performances, such as different hidden sizes and different number of LSTM layers. The input of the first time-step is the embedded image features, then the others are the embedded word features. Note that the output of first time-step is unused, and the input at the second time-step will always be a $\langle start \rangle$ token. The outputs of LSTM will be fed into a FC layer followed by a log softmax to compute the probability of each token.

In training phase, the ground truth captions are shifted on the input side and output side. And the model criterion is implemented to maximize the probability of the correct description given the image by using the following formulation: $\theta^* = \arg\underset{\theta}{max} \sum_{(I,S)} \log p(S|I;\theta)$, where θ are the parameters of the model, I is an image, and S is its correct caption. Meanwhile $\log p(S|I)$ is given by calculate the joint probability over the tokens in S. i.e. $\log p(S|I) = \sum_{t=0}^{N} \log p(S_t|I, S_0, ..., S_{t-1})$

During inference, since the ground truth captions are unknown, the decoders would just start with the image embedding and a $\langle start \rangle$ token. The following inputs will be generated one-by-one from the previous time-step. Both the greedy search algorithm and the beam search algorithm are implemented to generate the whole sequences, and a comparison experiment was performed.

3.3. Train

As suggested in [12], we enabled a two-phase training mechanism to get better performances and speedup the training process.

In the first phase, a pre-trained CNN model is loaded from Torchvision package and the whole CNN model is frozen except the very last FC layer. This is because we don't want to disrupt the trained features in it. Meanwhile, by freezing the CNN we can also use a mini-batch size as large as 512 on a single NVIDIA TITAN V, which effectively boosts up the training process and helps the model performance. In phase 1 we start with a learning rate at 10^{-3} for 60 epochs, then decay to 2×10^{-4} and 5×10^{-5} each for 30 epochs.

In the second phase, the CNN is unlocked and will be joint fine-tune with the RNN part. In this phase we used a very small learning rate at 10^{-5} and trained for another 50 epochs. The gradient map in CNN part requires extra memory thus we can only use a batch size as 64 in phase 2. Due to the time limit and the graphic cards contention on the NYU Prince cluster, the second phase only lasted about 50 epochs.

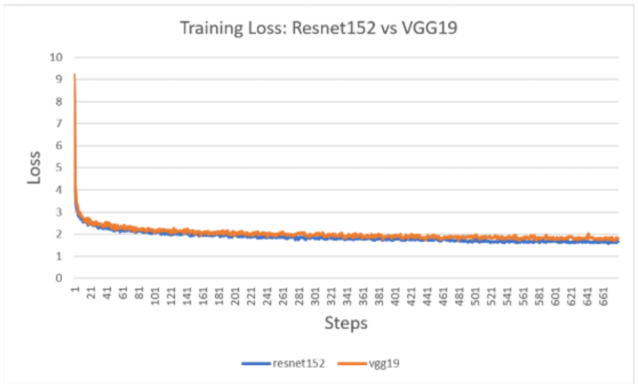

Figure 4. Training Loss: ResNet-152 and VGG-19

3.4. Inference

The inference procedure in RNN part is very different from the the procedure in training, since the ground truth captions are unknown. Thus to produce a caption from a image we will need a sequence generator like a greedy-search generator. Meanwhile, a beam-search generator is also implemented to achieve a better performance.

3.4.1 Greedy Search

A simple approximation is to use a greedy search that selects the most likely word at each step in the output sequence. To do this we simply use a max function on the vocabulary probabilities, right after the $log\_softmax$ layer. Then the token with the highest probability will be fed as the input of the next time-step.

3.4.2 Beam Search

Another popular heuristic is the beam search that expands upon the greedy search and returns a list of most likely output sequences. Instead of greedily choosing the most likely next step as the sequence is constructed, the beam search expands all possible next steps and keeps the k most likely, where k is a user-specified parameter and controls the number of beams or parallel searches through the sequence of probabilities.

To do this we start from the top k likely words after the $\langle start \rangle$ token, by using a $topk$ function. Afterwards, at each step, each candidate sequence is expanded with all possible next steps. Each candidate step is scored by multiplying the probabilities (negative $log\_softmax$ probability) together. The k sequences with the most likely probabilities - the smallest scores - are selected and all other candidates are pruned. The process then repeats until the $\langle end \rangle$ token or reach a max threshold. After all beams finish, we sort the candidate sequences using the scores and select the best one as our final result.

LSTM Hidden Size		512		1024	
Tune CNN		No	Yes	No	Yes
Greedy Search	ResNet152	23.7	24.2	22.9	23.3
	VGG19	21.6	22.0	21.9	22.4
Beam Search	ResNet152	26.2	**26.9**	-	-
	VGG19	-	-	24.1	24.5
Baselines	NIC	27.7			
	Random	4.6			
	Nearest Neighbor	9.9			
	Human	21.7			

Table 1. Comparison of Different Methods (BLEU-4)

4. Experiments

4.1. Dataset

We report BLEU-4 scores based on MS COCO, which consist of images and sentences in English describing these images. Each image has been annotated with 5 sentences that are relatively visual and unbiased. The training and testing split follows COCO official split, which contains 118K training images, 5K validation images and 41K test images. We use the official test split for our testing.

4.2. Metrics

BLEU [8] scores are designed for automatic machine translation evaluation. It is a form of precision by comparing n-grams of the candidate with the n-grams (up to 4-gram) of the references and count the number of matches. All scores are computed with the coco-evaluation code [1].

During the training process, we also use sentence **Perplexity** to capture the degree of 'uncertainty' a model has in predicting some text, helping evaluating the output sentences during inference.

4.3. Result and Discussion

4.3.1 Training Loss

Our loss function is the sum of negative log likelihood of the correct word at each step:

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

The training loss graph for phase 1 is shown in Figure 4:

- For ResNet-152, the LSTM setup is one layer, with 512 units. The loss gradually converged to about 1.4 to 1.5.

- For VGG-19, the LSTM setup is one layer, with 1024 units. The loss gradually converged to about 1.7 to 1.8.

4.3.2 Comparison on Different Model Setups/Methods

Our main precision results and the comparison between different methods, as well as the comparison to the baseline number provided in [12] are shown in Table 1. The best model we got is ResNet-152, with one uni-directional LSTM layer and 512 LSTM units. Both beam-search and phase 2 fine tune is applied. This model achieved about 26.9 BLEU-4 score on MSCOCO-Val2017. The result is very close to the NIC performance reported in the paper, and also much better than the traditional Nearest Neighbor approach and the human result given in [12].

Meanwhile, we can also identify how Beam Search, Phase 2 tuning and the number of LSTM units affect the model performance:

- Beam search generally gives about 10% performance boost on each model
- ResNet-152 is about 2 absolute BELU points better than VGG-19
- Fine tuning CNN (phase 2 training) doesn't show much improvement on the performance (generally less than 1 absolute BLEU points boost). A possible reason is that due to the time limit and the graphic cards contention on Prince cluster, the phase-2 training didn't last long enough in our experiments.

Besides the items listed in Table 1, we also tested the performance of two LSTM layers. Generally we found out that the two-layer version gives much worse performance than the one-layer versions, probably indicates that the two-layer version suffers from overfitting.

4.3.3 Good/Bad Examples

Generally speaking, our image captioning model is good at object detection and localization. Just like the results shown in Figure 5, the model can successfully detect the different objects, the number of objects, and also the relative position between the objects. But when the background is confusing or image is related to human's attention, it is easy to get the wrong result.

Besides the BLEU-4 scores, another problem we can identify from the generated captions is that they are lack of sentiments. Unlike the human-generated captions, which would usually use more adjective and adverbs, the machine-generated ones are mostly simple statements without emotions. This is typically not reflected in the metrics, but can be easily identified by a human judge.

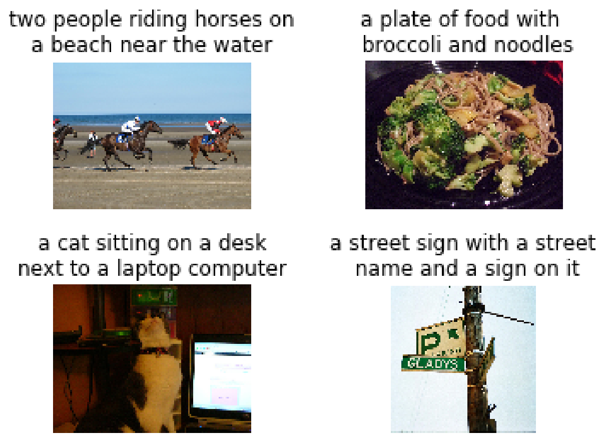

Figure 5. Examples of Good Results

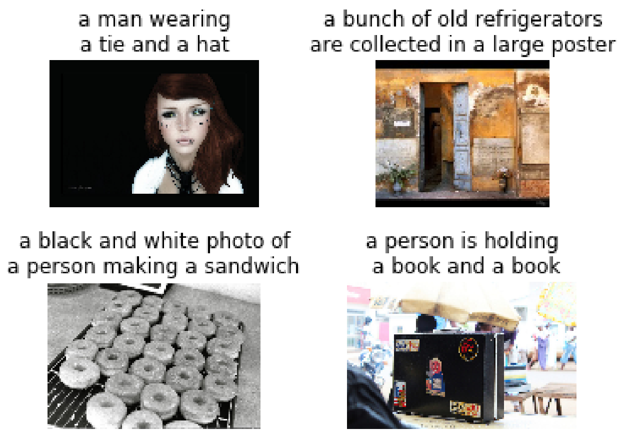
a man wearing
a tie and a hat
a bunch of old refrigerators
are collected in a large poster
a black and white photo of
a person making a sandwich
a person is holding
a book and a book

Figure 6. Examples of Bad Results

5. Conclusion

In this project, we re-implemented the Show and Tell model[12], meanwhile explored different CNN architectures such as ResNet and VGG and varies of RNN architectures. The result shows that our implementation can achieve the precision close the to the NIC approach reported in [12], and also better than the Nearest Neighbor methods and the Human result. Meanwhile, a comparison between varies of model setups is given, and also have a brief discussion on the advantage and disadvantage of the current model structure based on the observed examples.

6. Acknowledgement

The authors of this paper thanks Professor Rob Fergus, who gave an entirely amazing and excellent computer vision course in this semester. The authors would also like to thank NYU Prince cluster, who provide the free GPU computing resources required in this project.

References

[1] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013.

[7] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *International Conference on Machine Learning*, pages 595–603, 2014.

[8] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[11] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

[12] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[13] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.