Automatic image captioning with CNN-RNN

Chenglong Lin [1] **Chenan Wang** [2]

Abstract

The ability to describe an image is important in many practical applications. In this project, an end-to-end automatic image captioning model was developed and evaluated. This model is a combination of a feature extractor and a recurrent natural language processing engine. We trained and evaluated our model using the Microsoft COCO dataset.

1. Introduction

Automatic image captioning is a prominent problem in machine learning and natural language field. It has received plenty of attention from many research scientists because of its practical applications, such as aiding visually impaired person and organizing unstructured images data. Generating a description of an image is not an easy task. It goes beyond object detection and classification problems. To solve this difficult task, a model consisted of a convolution neural network and a recurrent neural network with long short term memory was developed and evaluated. This model is based on Krunal Kshirsagar's work [1].

2. Design and Implementation

In this section, the design process and methodology is presented.

2.1. Datasets for training/testing

The dataset we used in this project is Microsoft Common Objects in Context (COCO) [2]. In this dataset, each data sample contains an image and 5 associated captions that describe the context of the image. This dataset contains around 122 thousand data samples, in which around 82 thousand of data samples are used to train our model, 40 thousand for testing.

A man sitting next to a boy on a couch holding a Wii game controller.
A man is sitting next to a little boy on a couch, as they stare at something straight ahead.
A man and a young boy sitting on a couch.
A boy plays the Wii as his father watches.
A man sitting by a boy holding a remote on a couch.

Figure 1. COCO data sample

2.2. Model architecture

Figure 2 shows the architecture of our automatic image captioning model. The model takes in an image as input and output a text description of that image. The convolution neural network is used to generate feature vectors from the spatial data in the images and the vectors are fed through the fully connected linear layer into the recurrent neural network to generate a sequence of words that describes the image.

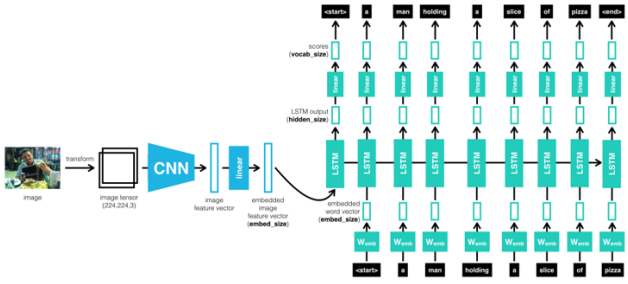


Figure 2. Model architecture

2.3. Image Pre-processing

To feed the colored image into the convolution neural network, prepossessing is required. Because the COCO data

[1]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA [2]Department of Computer Science, Northeastern University, Boston, MA. Correspondence to: Chenglong Lin <lin.chengl@husky.neu.edu>, Chenan Wang <wang.chena@husky.neu.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

set has all kinds of image sizes, to standardize the image, the images are resized and cropped into the size of 224 by 224. The images have a 50/50 chance to get a horizontal flip. Then, it was transformed into a 224 by 224 by 3 tensor. Finally, it was normalized using standard mean value and deviation value from ImageNet [3].

```
transform_train = transforms.Compose([
    transforms.Resize(256),                          # smaller edge of image resized to 256
    transforms.RandomCrop(224),                      # get 224x224 crop from random location
    transforms.RandomHorizontalFlip(),               # horizontally flip image with probability=0.5
    transforms.ToTensor(),                           # convert the PIL Image to a tensor
    transforms.Normalize((0.485, 0.456, 0.406),      # normalize image for pre-trained model
                         (0.229, 0.224, 0.225))])
```

Figure 3. Image pre-processing

2.4. Image encoder - Convolution neural network

In this project, the residual neural network (ResNet) architecture was chosen for capturing features from the image. To be specific, the ResNet-50 was used. The residual neural network was chosen because it was the winner of the ImageNet challenge in 2015 and was used for many computer vision tasks. Typically, as the neural network goes deeper, the back-propagation isn't feasible due to the vanishing gradient problem. Resnet architecture mitigates this problem by allowing an alternate shortcut path for the gradient to propagate through. Therefore, it allows us to train an extremely deep neural network [4].

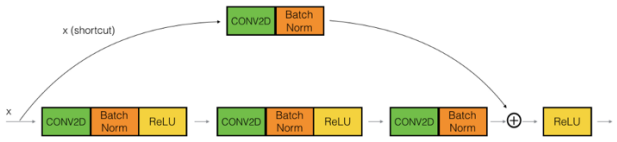


Figure 4. Resnet skipping connection

To save on training time, the ResNet-50 is initialized with pre-trained ImageNet weights. During the training, these weights are frozen, only the last fully connected layer is subjected to training. The original ResNet-50 architecture has a fully connected layer at the end with the soft-max activation for image classification. Since the objective of the image encoder is to extract image feature vectors from the image, it was removed. However, it is then replaced with a linear activation fully connected layer with output size matching that input size of the recurrent neural network decoder. In this project, the output of the encoder is 1 by 1 by 300 tensors. Figure 5 below is the structure of the image encoder.

Each position in the output feature vectors is the number represents the property that only the neural network knows.

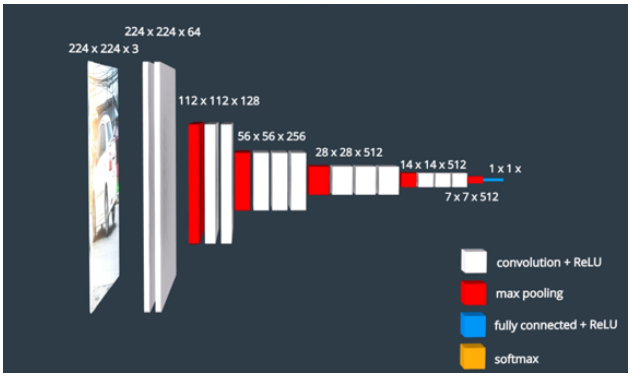


Figure 5. Image encoder structure

Depending on the length of the feature vector, the meaning of each number in each position of the feature vector could be different. These number, which indicates meaning, together becomes a vector representing the property of an image. We are going to introduce word embedding in the next section, it is really the same thing. Instead of images, we use embedding to represent a word, these vectors are only understood by the neural network.

2.5. Image decoder - Recurrent neural network

From the CNN network, we received many embedded image feature vectors. These feature embeddings would be inputs of RNN to generate words. This RNN is a one-to-many RNN, like the RNN in a music generator, we feed this RNN with one input, one tensor of feature embedding, and this RNN would return many outputs, many words to form a sentence. It turned out that the period symbols in the generated sentence were not working well, but we would see how it goes.

The RNN model consisted of many LSTM cells. LSTM stands for long short-term memory. LSTM is one of the attention models. This model was introduced by Hochreiter and Schmidhuber in 1997[5]. The other popular attention model, GRU, stands for gated recurrent units, was introduced by Kyunghyun Cho et al in 2014[6]. The formulas of these models were listed in figure 6[7].

From these formulas, we could see that they were using the same strategy: gating mechanism. GRU has two gates, used to preserve the state of each word in a sentence. LSTM has three gates, also used to memorize the state of the sentence. Thus, if we have a one-to-many RNN, in an output sentence, the words in the tail of the sentence would still correctly reflect the input.

Here, an LSTM cell was used to form an LSTM sequence. Each word would be generated by an LSTM cell. The

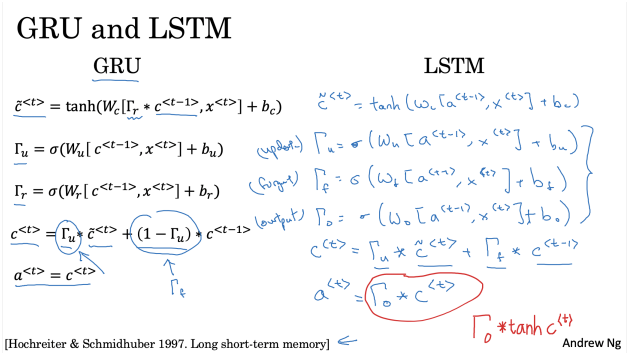


Figure 6. Attention models: GRU and LSTM, credit to Andrew Ng

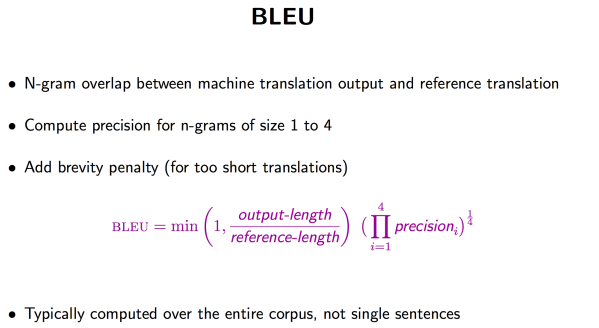


Figure 7. Formula of BLEU score

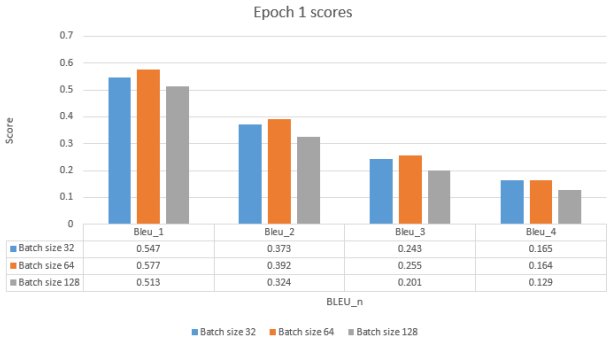
outputs of the LSTM sequence would be then fed into a linear layer. The outputs of the linear layer were many vectors, called scores. These scores represent each word. For each next word, these scores would be fed back into an embedding layer of the RNN, and gone through the LSTM sequence. In this manner, the decoder RNN received many feature embeddings as tensors from encoder CNN, and output many words that formed a sentence.

3. Evaluation Metric

We use the BLEU score, which is popular in machine translation, to evaluate the performance of the inference. BLEU score was a modified form of precision. In the testing set, we provided several candidate translations for each image. Then the BLEU score would compare how similar by matching the words between candidate sentences with a predicted sentence. BLEU score was an improvement from precision because it would not give a penalty if the predicted sentence uses more words than the candidate. The formula of the BLEU score with precision was shown in figure 7 [8]. The most frequently used BLEU score was BLEU-4, meaning 4-gram in the formula when calculating the BLEU score. The best possible score is 1, and the worse possible score is 0. In the experiment, we used the pycocoevalcap package to calculate the BLEU score between the original caption and the generated caption.

4. Result

Because of the huge dataset used in the training and the deep neural network structure, every epoch of the model took around 3.5 hours. This disabled us to do any substantial testing and evaluation of the model using cross-validation under the tight timing constraints. However, we were able to tweak couple hyper-parameters manually to obtain the locally optimal value with limited time frames. The hyper-parameters we examined were the number of epoch, batch size, and different BLEU evaluation metrics. All evaluations are done on the separate validation dataset.
From the figures below, you can see as the BLEU_n increase, the score decreases. This is because it's harder to find exact matching n-grams between the generated sentence and original image captions as n increases. Another thing to note from the figures is that the batch size during training does not affect the accuracy of the model. Also, as the number of epoch increases, we see an improvement in accuracy on the validation dataset until the model becomes over-fitted to the training data.



	Bleu_1	Bleu_2	Bleu_3	Bleu_4
Batch size 32	0.547	0.373	0.243	0.165
Batch size 64	0.577	0.392	0.255	0.164
Batch size 128	0.513	0.324	0.201	0.129

Figure 8. Epoch 1 BLEU score vs. batch size

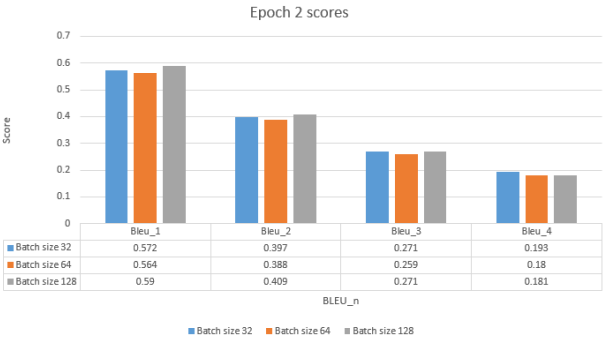


Figure 9. Epoch 2 BLEU score vs. batch size

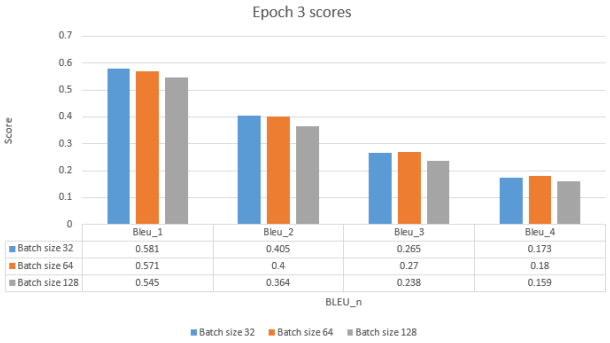


Figure 10. Epoch 3 BLEU score vs. batch size

5. Problems, analysis and future work

5.1. Problems

1. Compared with traditional machine learning, end-to-end learning requires fewer pipe-lines and does not require many tricks and comprehensive considerations. Even end-to-end learning largely simplifies the setup of the model, the drawback of this way, however, is that it requires a tremendous amount of data to train.

2. Despite using immense data for training, it also takes lots of time. In this experiment, we only had 1 GTX1080 GPU, and it took 3.5 hours to go over one epoch. It means in development, we need more time on the machine learning development cycle like in figure 11 [9].

5.2. Possible modifications in the future

1. We could change the CNN structure in transfer learning. In this experiment, we used ResNet-50, which was a pre-trained network. We could change this CNN structure, to make it deeper, or shallower.

2. We could change the RNN structure. We could use the GRU sequence instead of the LSTM sequence; we could also use a bidirectional RNN.

3. We could apply the drop-out strategy in the model.

4. We used adam optimizer in this experiment. We could use other optimizers like SGD, RMSProp, or momentum.

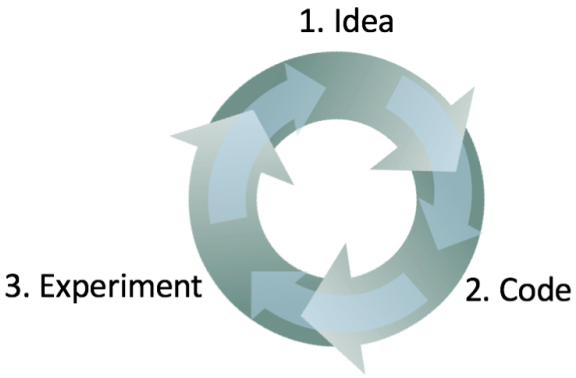


Figure 11. Machine learning development cycle

6. Conclusion

In this paper, we presented an end-to-end image captioning model. This model has a convolution neural network for image feature vector extraction, and it passes this extracted feature vector to an LSTM recurrent neural network for caption generation. After hours of training on the Microsoft COCO training dataset. We tested the model with the BLEU metric on a separate validation dataset. This model achieved a 0.18 BLEU_4 score on epoch 3. Though the result shows that the model did not achieve state-of-the-art performance in the industry, it was a fun project to learn and code on.

References

[1] K. Kshirsagar, "Automatic Image Captioning with CNN RNN," Medium, 21-Jan-2020. [Online]. Available: https://towardsdatascience.com/automatic-image-captioning-with-cnn-rnn-aae3cd442d83. [Accessed: 16-Apr-2020].

[2] "Common Objects in Context," COCO. [Online]. Available: http://cocodataset.org/home. [Accessed: 16-Apr-2020].

[3] n0obcoder, "Classification of Mobile Gallery Images in PyTorch," Medium, 24-Nov-2019. [Online]. Available: https://towardsdatascience.com/classification-of-

mobile-gallery-images-in-pytorch-8ba2d32ce2bf. [Accessed: 16-Apr-2020].

[4] Dwivedi, "Understanding and Coding a ResNet in Keras," Medium, 27-Mar-2019. [Online]. Available: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33. [Accessed: 16-Apr-2020].

[5] Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". arXiv:1406.1078 [cs.CL].

[6] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276.

[7] A. Ng, "Sequence Models," Coursera. [Online]. Available: https://www.coursera.org/learn/nlp-sequence-models. [Accessed: 16-Apr-2020].

[8] Koehn, Philipp. Statistical Machine Translation. Cambridge University Press, 2014.

[9] Amilkanthawar, Vivek. "Choosing a Deep Learning Framework." Medium, In Pursuit of Artificial Intelligence, 28 Feb. 2019, medium.com/in-pursuit-of-artificial-intelligence/choosing-a-deep-learning-framework-5669a85ebc3f.