Simple Image Caption Report

Zetian Yu, zy1546 Jiaming Liu, jl10986

May 14, 2021

Abstract

This project [2] follows the paper to implement an state-of-art approach of image caption. This project focus on exploring the performance under different kinds of hyper parameters such as drop out rate, optimizer and using different model to extract the feature from picture.

1 Project description

Over the last few years, the Image Caption generation draws a lot of attention in the field of Artificial Intelligence. Image Caption was designed to free people form manually adding captions to images, beyond the basic functions, it will not only be useful for providing a decent description of image for some visually impaired people when they read pictures but also be potentially useful for real-time video description generation

Due to the challenge of developing a natural language sentence associated with the specific image, It needs not only the method of computer vision to understand the content of the image, but also the language model in the field of natural language processing to translate the understanding of the image into words in the correct order. Thus, the initial approach which applies the object detection technique to get the feature of the image and then use it to generate the caption turns out to be insufficient for distinguishing an image from the others.

The state-of-art approach provides a new prospective to solve the problem by using the deep learning based technique, which learn features automatically from training data and thus can handle the large and diversity set of photos. In our project we will take advantage of CNN for feature learning and followed by a RNN for caption generating. In detail, the model will treat all images and sentences as the same form of features, train them by concatenating the image and one word at a time to predict the next word until the end. Following the paper, our team mean to explore different feature extractors and the various options while building the image caption model beyond the original experiment proposed by the paper.

To observe the performance, our team will utilize the BLEU score and the loss-epoch curve for evaluation among different set of the experiments.

2 Approach

Our project utilize the natural language generation model which will generate a sentence based on the previous information using a recurrent neural network(RNN). This is a classic problem in the Nature language processing(NLP) field. In detail, The model inputs a given prefix and predicts which word in the vocabulary is likely to follow.

As we can see on the Figure.1, the language model predict the sentence word by word based on the previous text sequence.

Then apply the same idea to the image where the model will treat the image as the prefix and predict the caption word by word. In other word, the neutral language model's prediction will take the image features as prefixes and predict the next word base on previous information.

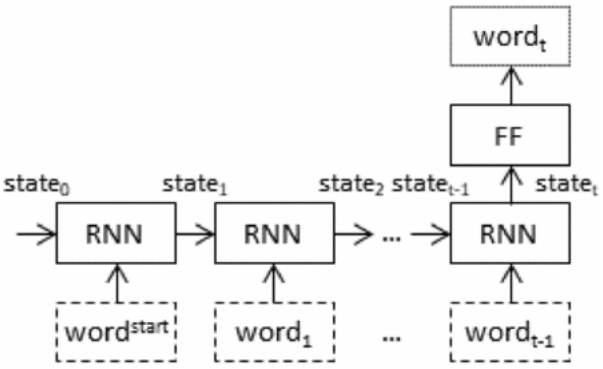

Figure 1: RNN-based neural language model

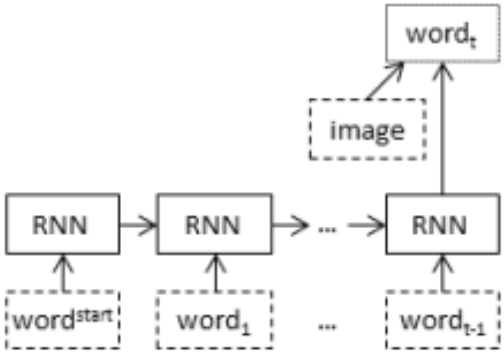

Figure 2: Merge architecture

The algorithm also contain three main network, text sequence processor, image feature extractor, and feed forward decoder. We will discuss further in section 2.2 through 2.4.

2.1 Overall architecture

A recent article shows there are four architecture to introduce the image features into the RNN: Inti-inject, Pre-inject, Par-inject and Merge. The difference between them is Inject architecture will put the image feature into the linguistic model layer and the merge architecture will only put image features into image caption prefix encoding process. the Merge architecture does not involve image feature vector anywhere in the RNN part, the image is introduced into the model only after the text prefix is encoded by the RNN and this late binding will prevent the image representation being modified every time step, thus yields the best performance as the paper suggested.

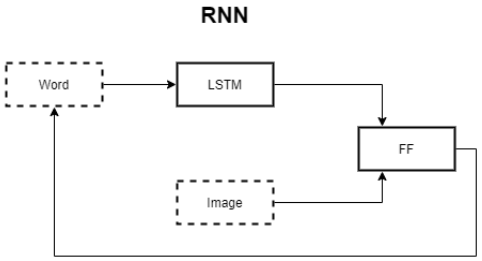

Figure 3: Image Caption Deep learning Model architecture

The overall architecture contains 3 main parts. First, the Text Sequence Processor, in this part, the image is left out waiting to be input into the RNN, so that the network will handle the caption in the RNN with LSTM first which is just the same as the pure linguistic model. Second, the Image Feature Extractor, which extract the image features through a CNN and encode them to be a prefix of the caption sequence. The final part, Feed Forward Decoder, which merges both encoded image feature input and the encoded text input together and pass to feed-forward NN to predict the output. Then use the predicted next word together with the previous information as input to the model for generating the next word until reaches the end.

2.2 Text Sequence processor

The Text Sequence Processor requires a cleaned and pre-processed or manually tuned word input since the smaller and the more expressive our vocabulary is the faster and better the model will train. Thus this part contains two main processes, one is for cleaning the input text and tuning the vocabulary, second is encoding the text through a word embedding layer followed by a LSTM RNN layer.

Fitting a machine learning or deep learning model with raw data will introduce lots of noise including the

typos or spelling mistakes; punctuation like commas, quotes, question marks; hyphenated descriptions; and so on. Since punctuation and upper/lower cases does affect the meaning of the description. The cleaning/manually tuning in our image caption model plays an important role for all description patterns are declarative. With those conditions, cleaning text can eliminate the noise of the model also make the caption closer to a human generated sentence.

The cleaning processor cleans the text with four steps: convert all words to lowercase; remove all punctuation; remove all words that are one character or less in length (e.g. 'a'); remove all words with numbers in them.

After cleaning we can get all the valid word vocabulary and the max length of the description for further use. Then we need to add the start sequence and end sequence symbol to the word sequence. For RNN we need those symbols to be the indicator of starting prediction and ending prediction.

Next step is to encode the word so that it can be input into our model, which we can utilize the Keras Tokenizer function to map each word in the vocabulary into an unique number in one-hot representation. Representing an idealized probability distribution with 0 values at all word positions except the actual word position, which has a value of 1.

After the cleaning and the encoding, the text sequence can be input into the word embedding layer, in this layer, the input vectors that represent the known words prior to being fed to the RNN consist of vectors that have been randomly initialized. and the layers are trained as part of the network in order to find the best representation of the word for the task. Then RNN will pass the embedded word vector to LSTM layer for a longer information memorization to produce a vector to represent the sequence.

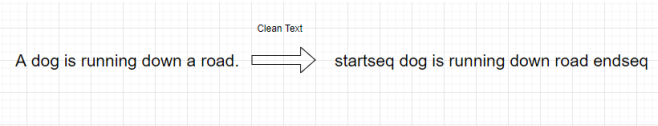

Figure 4: example of text sequence before and after the cleaning process

2.3 Image feature extractor

Image feature extractor is a CNN used to extract the features of the image, since our model is a large model and processing each photo through the network every time when we need to test a new language model configuration is time consuming. Thus, implementing a VGG model to extract the features only once and then save the feature to file for later usage will be an optimization that allow us to train the model in less memory and cost less time. Our project choose to use VGG model which is already pre-trained on the ImageNet dataset. We pre-process the photos with the VGG model without the last output layer and use the extracted features predicted by this model as image feature input to the FF decoder.

We remove the last layer from the original model since we do not need to predict a class for the image but only need the internal representation of it. The output vectors with length of 4096 are the extracted feature of the image as figure5&6 shown below. Different CNN model will also affect the performance of the caption generation, since the features extracted from the images should be as expressive as possible.

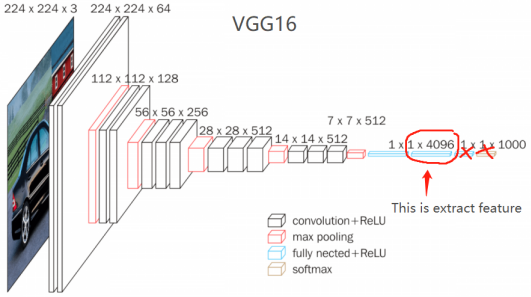

Figure 5: VGG16 feature extractor

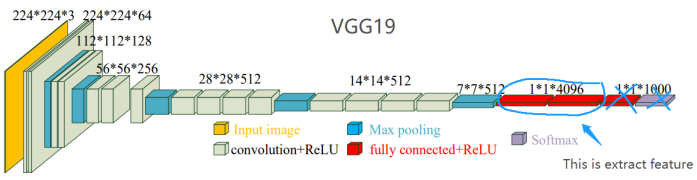

Figure 6: VGG19 feature extractor

2.4 Feed forward decoder

Once the image and the caption prefix have been forged into same length vectors and then was merged together, the next step is to use them to predict the next word in the caption. The mixed vector will then pass through a feed-forward layer with a softmax activation function that outputs the probability of each possible following word in the vocabulary, picking up the highest possible word in the distribution to be the next word that comes after the selected prefix. an example as shown in the figure below.

X1	X2(text sequence)	y (word)
photo	startseq,	a
photo	startseq, a,	dog
photo	startseq, a, dog,	stands
photo	startseq, a, dog, stands,	on
photo	startseq, a, dog, stands, on,	the
photo	startseq, a, dog, stands, on, the,	beach
photo	startseq, a, dog, stands, on, the, beach,	endseq

Figure 7: example of caption generation algorithm

3 Evaluation Metric

To evaluate how good our prediction is, we will use BLEU score which stands for Bi-Lingual Evaluation Understudy. BLEU score is a popular and cheap way to automatically measure the performance of the model by comparing compares the machine's translation with the human generated sentences or the reference translations. in our project we use BLEU score to compare the generated caption with the gold label in the Flicker8k_text using the 1/2/3/4 cumulative n-grams model. A score closer to one will indicate a better performance.

4 Implementation detail

4.1 Data set

In our final project, we use a "Flicker8k_DataSet" dataset and "Flickr8k_text ". In "Flicker8k_Dataset", it contains a total of 8092 images in JPEG format with different shapes and sizes. Of which 6000 are used for training, 1000 for test and 1000 for development. In "Flickr8k_text", it contains text files describing train_set ,test_set. Flickr8k.token.txt contains 5 captions for each image i.e. total 40460 captions. The reason why we chose this dataset is that the dataset is small in size, so the model can be trained easily on low-end computer. All data are properly labelled. There are five captions provided for each image. And the dataset is available free online.

4.2 Test environment Setup

Our experiment is deployment on GCP with follow configuration.
Tensorflow: 2.4
Keras: 2.4.3
Memory: 13GB
GPU: NVIDIA V100
CPU: 2

4.3 Model Structure

This is the model structure visualization generated from the code. The figure should be input to the CNN and extracted as a length 4096 vector. After applying a dropout layer, it should be dense to a length 256 vector. For the text sequence input, it should be the length of vocabulary size, which is 34 in our project. After applying embedding, we also apply the dropout layer. Then follow by the LSTM layer and padding to a length of 256 vectors to be matched with the image feature vector length. Then concatenate two same length vectors together to feed into the decoder and use the softmax layer to predict the next word.

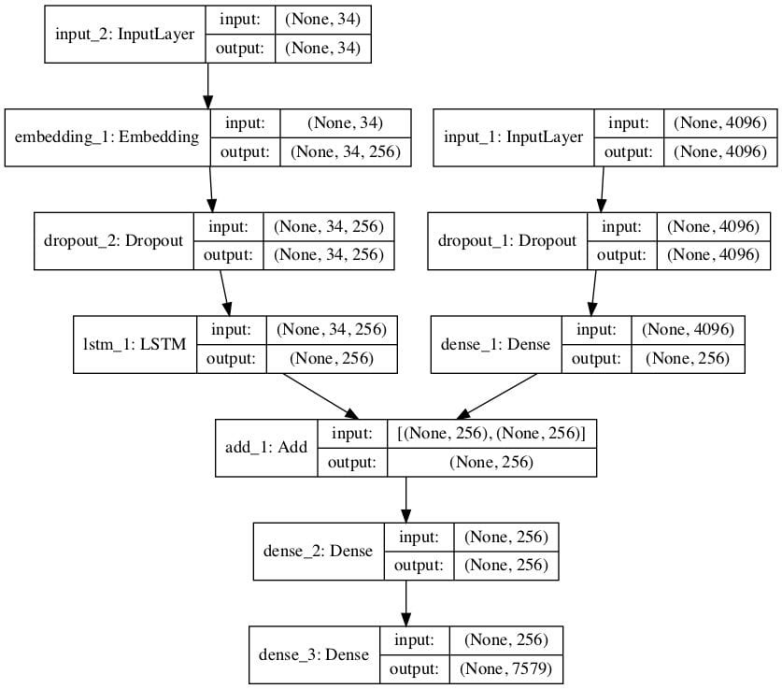

Figure 8: Image-Caption model

5 Experimental result

Our experiment looks into the performance of some alternative options across the different feature extractors, different optimizers, dropout s during the training, and with/without manually tuning vocabulary.

5.1 feature extractor

Our team first evaluates the performance of two feature extractors between VGG 16 and VGG 19 of same dor 0.5 and applied the same optimizer Adam. We can tell The VGG 19 yields a slight advantage over VGG 16 over 15 epochs, however there are no obvious differences between two CNN.

	BLEU-4	BLEU-3	BLEU-2	BLEU-1
vgg16	0.068573	0.157678	0.243836	0.534382
vgg19	0.070203	0.165435	0.257391	0.498788

further experiment should explore more different CNN like inception v3 and more to see the feature extractors' influence on the image caption generation.

5.2 dropout rate

Our team evaluates the performance of different dropout rates, we choose 0.1 0.3 and 0.5 dor. We keep other configurations the same like VGG 16 and same optimizer.

vgg19 Drop Rate	BLEU-4	BLEU-3	BLEU-2	BLEU-1
0.1	0.080047	0.182463	0.272523	0.528840
0.3	0.075623	0.178729	0.268927	0.522455
0.5	0.073722	0.172398	0.260732	0.519279

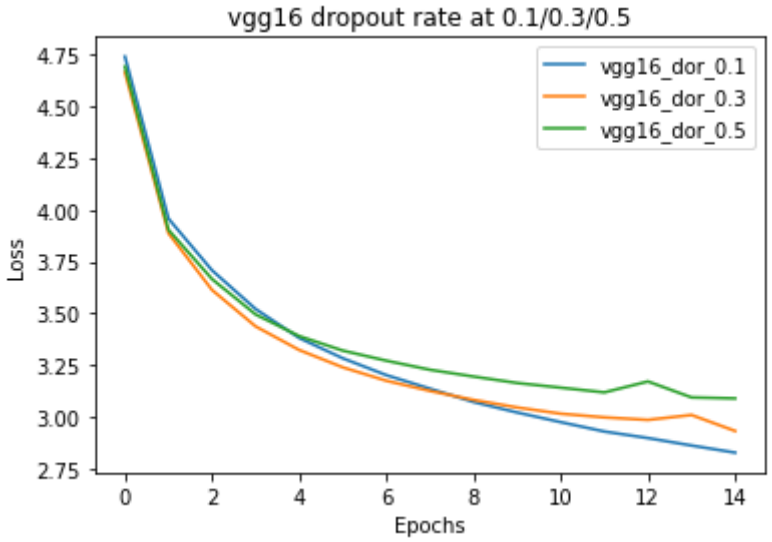

Figure 9: loss vs epoch for dor

As we can tell the smaller the dor the lower the loss can achieve, how ever the BLEU score does not show a big difference among three different dor. Since the training process is time consuming, we will stop at the epoch 15 for convenience. Further experiment will be testing the converge epoch number for different dor.

5.3 optimizer

Our team then evaluate the performance between various optimizers on VGG 16 dor 0.1 shown as figure 10

As we can tell during the first 15 epochs, the Adam and RMEprop perform much better than the rest optimizer, however due to reason that the training job takes too long, we have to stop at the epoch 15, further experiment will be testing the final loss on the converge epoch of each optimizer.

5.4 manually tuning

Our team then explore the effect of the manually tuning, here we try to get rid of the one length words and modify the original word into all lower case words, the BLEU score are shown below:

	BLEU-4	BLEU-3	BLEU-2	BLEU-1
vgg16 original	0.068573	0.157678	0.243836	0.534382
vgg16 tuning	0.072212	0.178729	0.264511	0.553669
vgg19 original	0.070203	0.165435	0.257391	0.498788
vgg19 tuning	0.080047	0.182463	0.272523	0.528840

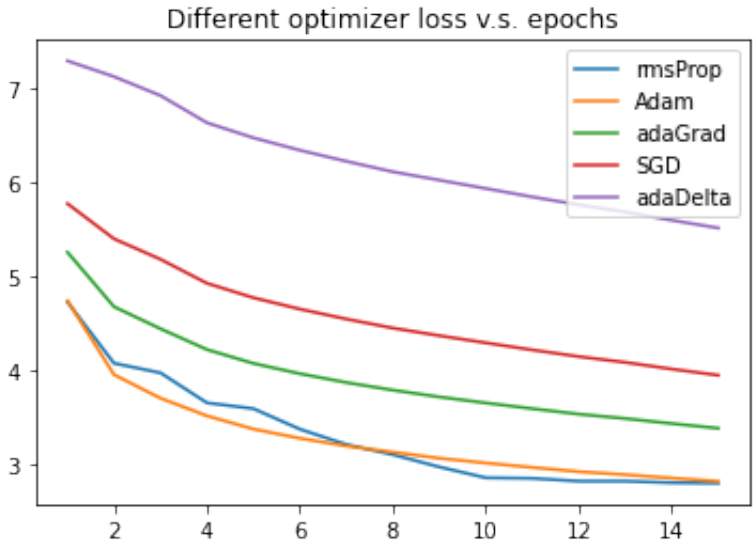

Figure 10: loss vs epoch for optimizer

Manually tuning does influence the probability distribution of some words in the vocabulary, our team only tunes the vocabulary in a very simple fashion and the result BLEU score doesn't show much improvement over the original experiment. further test should be looking at a better way to implement the tuning technique.

6 Conclusion

Our experiments explored different options of the model and evaluate the performance by BLEU score. Turns out the feature extractors between VGG 16 and 19 does not yields obvious influence on the performance; the lower drop out rate will trends to converge fast, but will have a better performance in the limited epochs. Among different optimizers, the Adam and the RMSprop will have better performance over the rest of the optimizers both in the loss curve and the BLEU score. finally the manually tuning may have positive effect on the performance if apply reasonable vocabulary cleaning policy by observing the input data and focus on getting rid of the potential noise.

References

[1] Hodosh, Micah, Peter Young, and Julia Hockenmaier. "Framing image description as a ranking task: Data, models and evaluation metrics." Journal of Artificial Intelligence Research 47 (2013): 853-899.

[2] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[3] Tanti, Marc, Albert Gatt, and Kenneth P. Camilleri. "Where to put the image in an image caption generator." Natural Language Engineering 24.3 (2018): 467-489.

[4] Tanti, Marc, Albert Gatt, and Kenneth P. Camilleri. "What is the role of recurrent neural networks (rnns) in an image caption generator?." arXiv preprint arXiv:1708.02043 (2017).

[5] Hossain, MD Zakir, et al. "A comprehensive survey of deep learning for image captioning." ACM Computing Surveys (CsUR) 51.6 (2019): 1-36.