Caption Generation For Wikipedia Images

Maojie Tang
MCS
Rice University
mt84@rice.edu

Ziren Wang
MCSE
Rice University
zw71@rice.edu

Xijuan Liu
MCSE
Rice University
xl106@rice.edu

1 Introduction

People all rely on online images for knowledge sharing, learning, and understanding. On the other hand, as one of the largest encyclopedia in the world, Wikipedia is missing visual content and metadata to pair with their images. Captions and "alt text" increase accessibility and enable better search. The majority of images on Wikipedia articles don't have any written context connected to the image. Therefore, we are interested in building an image caption model that automatically retrieves the text closest to an image. Specifically, we'll train our model to associate given images with article titles or complex captions, in English.

Current solutions rely on simple methods based on translations or page interlinks, which have limited coverage. Even the most advanced computer vision image captioning isn't suitable for images with complex semantics. In this project, we construct an captioner for Wikipedia based on encoder-decoder approach, and use the MSCOCO'14 dataset for transfer learning.

2 Literature Review

The existing image captioning methods can be categorized into 3 main groups: template-based image captioning, retrieval-based image captioning, and novel caption generation. Template-based approaches utilize several fixed templates where there are a number of blank slots to needed to complete. After the algorithms detect objects, attributes and actions, corresponding language components will be generated and filled into those blanks. For example, Farhadi et al. [2] use a briplet of scene elements to fill the template slots for generating image captions. Li et al. [8] extract the phrases related to detected objects, attributes and their relationships for this purpose A conditional random field is adopted by Kulkarni et al. [7] to infer the contents before filling in the gaps. The advantage for using template-based approaches is that the grammatical correctness of captions is guaranteed. But this also imposes constraints on the outputs so that when dealing with more diverse images the captioning accuracy can decrease.

The second type of captioning methods is retrieval-based. It relies on the similarity of images in which retrieval-based methods find the visually similar images(neighbors) with captions from the training dataset. Then captions for new images are then selected from this captions pool by some criterion. Hodosh et al. [6] propose to frame sentence-based image annotation as the task of ranking a given pool of captions. Their analysis shows that metrics that consider the ranked list of results for each query image or sentence are significantly more robust than metrics that are based on a single response per query. Gong et al. [3] introduced a new algorithm called Stacked Auxiliary Embedding that can successfully transfer knowledge from millions of weakly annotated images to improve the accuracy of retrieval-based image description. These methods produce general and syntactically correct captions. However, they cannot generate image-specific and semantically correct captions.

In this project, our relying methods are from the third category: Novel Caption Generation, which combines the information from both visual and multimodal space. A general approach of this category is to analyze the visual contents of the image first and then generate image captions using a language

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

model. Compared to previous two types of approaches, these methods can generate new captions for each image that are semantically more accurate. There are methods that use attention mechanisms, semantic concepts, and different styles in image descriptions. Some methods can also generate descriptions for unseen objects. For example, Park and kim [1] propose a novel captioning model named Context Sequence Memory Network (CSMN) which uses ResNet to encode image and chose LSTM as its language model. Gu et al. [4] introduce a language CNN model which is suitable for statistical language modeling tasks and shows competitive performance in image captioning, In contrast to other models which predict next word based on one previous word and hidden state, our language CNN is fed with all the previous words and can model the long-range dependencies in history words, which are critical for image captioning.

In conventional encoder-decoder-based approaches for image captioning, a CNN is used as an encoder to extract the visual features from the input image, and an RNN is used as a decoder to convert this representation word by word into a natural language description of the image. One significant disadvantages of this configuration is that analyzing image over time is not feasible while models generate the descriptions for the image. This mainly because they generate captions considering the scene as a whole, rather than focusing on specific components within the image. In light of this limitation, attention-based mechanisms are becoming increasingly popular. The main difference between the attention-based methods and other methods is that they can concentrate on the salient parts of the image and generate the corresponding words at the same time. Xu et al. [10] were the first to introduce an attention-based image captioning method. This method applies two different techniques: stochastic hard attention and deterministic soft attention to generate attention. It can describes the salient contents of an image automatically. We will use this method as our project backbone.

3 Data Exploration

3.1 Dataset Choices

The first dataset we use is MSCOCO'14, which is a large-scale object detection, segmentation, and captioning dataset. There are more than 180,000 images and captions. We split into 170,000 images for training, and 1,000 for testing. The word map generated from COCO dataset contains 9490 words.

From COCO Dataset:
Herd of birds sitting on a tree

From WiKi Dataset:
A giant "flamingo" statue by Alexander Calder in front of the Federal Center

Figure 1: Sample Images from Two Datasets

The Wikipedia dataset contains the url links to images on Wikipedia as well as the captions to images. The dataset we used for training contains 2168.54MB of data but only a small amount of images has associated captions. Thus, it seems like a semi-supervised learning problem. There are 110,302 captions in total. The data contains multiple languages, but we only use captions in English. The number of English samples is 13387; among all we use 12,300 images for training, and the number

of testing samples is 1000. The Figure 1 shows example images of two datasets. By combining the word map from coco dataset and wiki dataset, we obtain a word map of 29237 words in the end.

3.2 Inputs to model

Images Because we're using a pretrained Encoder, we need to convert the images to the format that the pretrained Encoder is used to. Meanwhile, to ensure that each image has the same shape, we resize all MSCOCO images to 256x256 for uniformity. Consequently, images fed to the model are Float tensors of dimension [N, 3, 256, 256] (because PyTorch follows the NCHW convention), with the aforementioned mean and standard deviation normalized.

Captions They are both the target and the inputs of the Decoder, as each word is used to generate the next word. To generate the first word, however, we need a zeroth word, <start> to begin the caption generations. As for the last word, the model is supposed to predict <end> as the indication of stop of inference process. One should notice padding is needed to pre-process the input captions to the same length with whereas tokens, since they are then turned into fixed size tensor for later computations. Meanwhile, we create a word indices set for mapping each word in the corpus, including the <start>,<end>, and <pad> tokens, to specific numerical values. PyTorch, like other libraries, requires words to be encoded as indices in order to look up embeddings or identify their position in the predicted word scores. As a result, captions fed to the model must be an Int tensor with dimensions N, L, and L being the padded length.

Caption Lengths This information should be tracked as well, because all of captions are padded into same length sequence. The value of caption length is the number of tokens plus 2 (for the start and end tokens). As a result, the caption lengths fed to the model must be an N-dimensional Int tensor.

4 Model Architecture

4.1 Encoder-Decoder architecture

In this section, we describe our model design for this project: Encoder-Decoder framework with Attention mechanism.

4.1.1 Encoder

The first piece of our model is encoder. It takes a single three-channel input image and convert it into several smaller output images with "learned" feature maps. These smaller encoded images are condensed representations of the original image's embedding information. Our choice for encoder is convolutional neural networks (CNNs), and it can extract more efficient and useful features than manual feature-extraction. We firstly use ResNet-101 [5] as our CNN choice, then we can do more experiments with different CNN to compare the differences between various choices.

As shown in Figure 2, the encoder gradually produces smaller and smaller representations of the original image, each of which is more "learned" and has a greater number of channels. Our ResNet-101 encoder produces a final tensor with a size of 14x14 and 2048 channels, i.e. a [2048, 14, 14] size tensor.

4.1.2 Decoder

The function of the Decoder is to examine the encoded images, and then generate corresponding caption word by word. We use Recurrent Neural Network (RNN) to generate a sequence, specifically, we'll use an LSTM for Decoder.

In a typical context, the encoded image is simply weighted averaged across all pixels. We may then send this to the Decoder as its first hidden state, with or without a linear transformation, starting to generate the caption. The following word is generated using each predicted word.

4.1.3 Attention

The idea to deploy Attention mechanism is to enable Decoder to look at different areas of the image when generating caption sequence at different locations. For example, the Decoder would know to

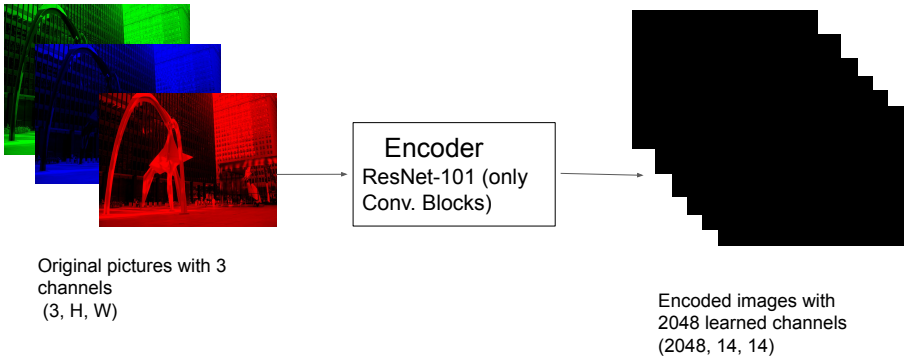

Figure 2: Encoder Module

focus on the 'Federal Center' while creating the word statue, which is in front of the 'Federal Center'. Thus, different weights are learned for each region to arise the model to pay different attention to various regions.

These attentions(weights) are calculated by the Attention network. This idea is trying to answer the question: how to measure the importance of a specific portion of an image? Intuitively, one needs to be aware of the sequence that has been created so far in order to examine the image and determine what needs to be described next. For instance, when mentioning there is a giraffe in the forest, it is easy to know "that animal" means the giraffe for the next time.

This is exactly what the Attention mechanism does: it evaluates the sequence that has been generated so far and focuses on the next component of the image to be described. The contextual information among both images and languages has been taken into considerations through the usage of Attention network.

There are two kinds of attention methods: The first one is the soft attention, where weights alpha was computed then used to weighted average all of features across different pixels. Soft attention is a differentiable, deterministic operation and that's why it is the common choice that researchers implement. By contrast, for hard attention only a few pixels from populations are sampled from an alpha-defined distribution. It's worth noting if any probabilistic sampling is non-deterministic or stochastic, since a given input will not always yield the same result. Gradient descent, on the other hand, assumes that the network is deterministic (and thus differentiable), so the sampling is reworked to eliminate the randomness.

In our model, we implement soft Attention, where the weights of the pixels add up to 1. If there are P pixels in our encoded image, then at each time stamp, we get

$$\sum_{p}^{P} \alpha_{p,t} = 1 \tag{1}$$

We use a weighted average over all pixels instead of a simple average, with the weights of the critical pixels being higher. To construct the next word, this weighted representation of the image can be concatenated with the previously generated word at each step. The following figure 3 demonstrated how works in our Decoder Module.

Together with previous pieces, our complete model configuration is shown below 4.

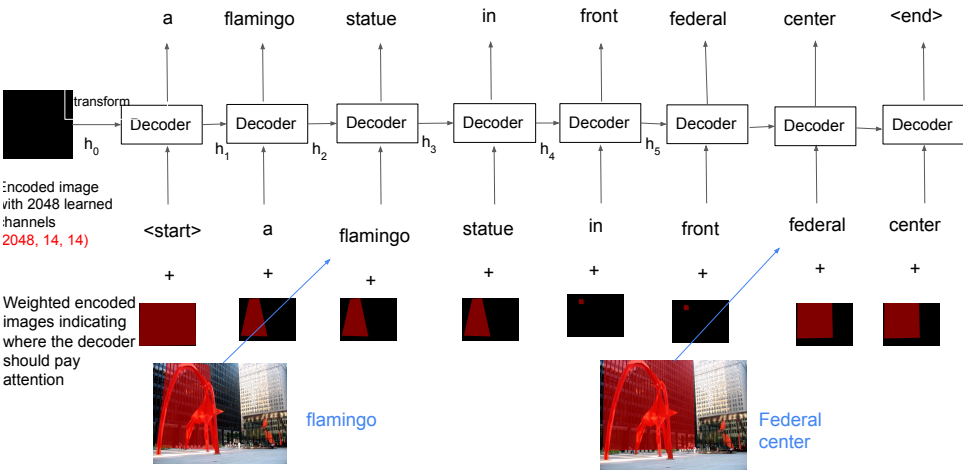

Figure 3: Decoder with Attention

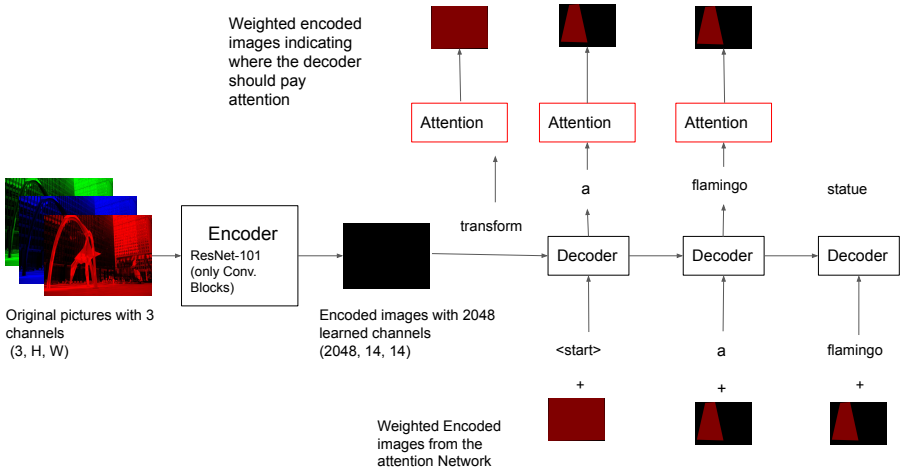

Figure 4: Complete Configuration of Our Model

4.2 Transfer learning

Considering the size of our dataset, training the model completely is infeasible, and such non-convergence will deteriorate the final performance of model. In practice, there is no need to train CNNs from scratch. Researchers have been developing models for years that are extremely good at classifying images into one of a thousand categories. It stands to reason that these models are excellent at capturing the essence of an image.Therefore, we only need to modify some structure parameters in the pre-trained model and fine-tune it in align with our dataset.

In this project, we decide to implement this idea, namely Transfer Learning, to save our computational consumption. The first module that train a model on a large dataset to learn some basic features is called "pretrain" and the second module that fits for our own data is "fine-tune". Combining these two methods together, we achieve a better model performance even on a small amount of dataset.

4.3 Beam search

For each word in the vocabulary, we use a linear layer to convert the Decoder's output into a score. The simple solution (greedy) is to pick the highest-scoring word and use it to predict the next word. However, because the rest of the sequence is dependent on the first word you select, this is risky and may not be the best option. Everything that follows is sub-optimal if that first choice isn't the best. It's not just the first word; each subsequent word has ramifications. It would be ideal if we could wait until we've completed all of the decoding and then pick the sequence with the highest overall score from a pool of candidates, which is called Beam Search [10]. In this project, we implement this idea as well. The working pipeline is as follows:
(1) Consider the top k candidates in the first decode step.
(2) For each of these k first words, generate k second words.
(3) Select the top k [first word, second word] combinations based on additive scores.
(4) Choose the top k [first word, second word, third word] combinations for each of these k second words and k third words.
(5) Repeat this for each decode step.
(6) Choose the sequence with the best overall score after k sequences have finished.

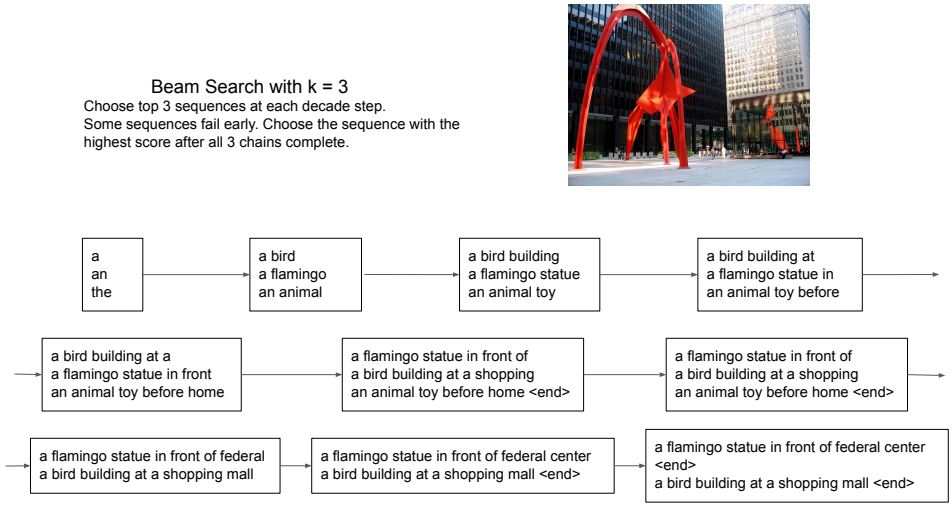

Figure 5: Beam Search Example

Figure 5 shows an example of Beam Search. In this instance, some sequences (highlighted) may fail early because they don't make it to the top k in the next step. Once k sequences (underlined) generate the <end> token, the sequence with the highest score will be chosen.

5 Training and Evaluation

5.1 Transfer Learning with COCO Dataset

To train the model, we need to define loss function, evaluation metrics and optimizer. Considering the output of Decoder is a probability distribution for each word, we use the Cross-Entropy as our loss function. Furthermore, we use Adam optimizer with an initial learning rate of 4e-4, because it run faster than other gradient descent methods.

Also, we need to define a metric to evaluate our model. Same as other classical image caption models, we choose automated BiLingual Evaluation Understudy (BLEU)[9] evaluation metric. This evaluates a generated caption against reference captions. For each generated caption, we use all captions available for that image as the reference captions. Based on previous academic experience[10], if the BLEU score does not increase in 10 epochs, the early stop strategy will be implemented to terminate training process.

In light of the limited images/captions, we firstly introduce COCO dataset for transfer learning. Unfortunately, it is extremely time-consuming to train our model on COCO dataset: it takes more than one and a half hour for the training of one single epoch. Eventually, we obtain a pretrained weights for encoder and decoder with 18 epochs training on COCO dataset after nearly one day's training.

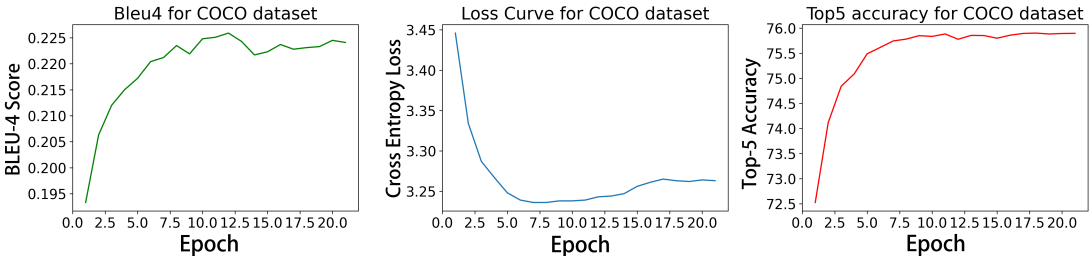

Figure 6: Analysis of COCO dataset

We train the model 50 epochs and the BLEU-4 score peaked at 0.226 at the 11th epoch. In different experiments, this pre-trained model converge consistently. We find there is a trend of over-fitting after the 11-th epoch, and at epoch 21 the early stop condition (BLEU-4 score does not improve within 10 epochs) has been activated.

5.2 Train with Wiki Dataset

We traverse all Wiki captions and add those new words to COCO word map because many of the words in the Wiki dataset have not yet recorded in the word map of COCO dataset. We keep the other hyper-parameter the same as previous model but we introduce a weight decay method in case learning rate is too fast. If BLEU-4 score does not get improved within 8 epochs, we penalty previous learning rate to 0.8 times the original. In addition, considering Convolutional neural network (Encoder) has learned some basic features from COCO dataset, we only train the last layer and freeze the rests to speed up training process. The usage of transfer learning is effective, since the model converges very

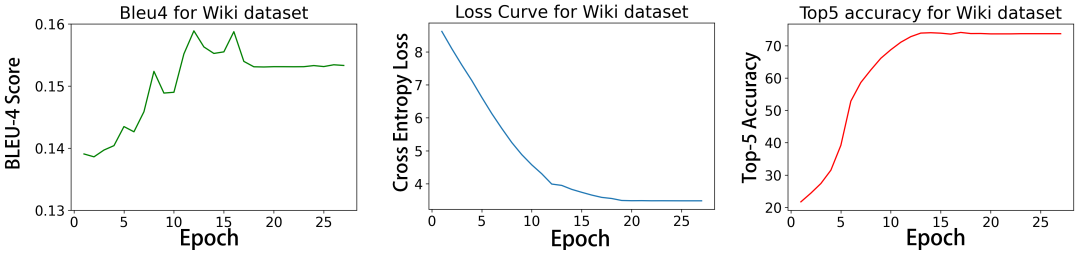

Figure 7: Analysis of Wikipedia dataset

fast. The number of epochs for training is 50 in total but its early-stop happened at epoch 27. Finally, the best BLEU-4 score is 0.159, top-5 accuracy is 75.89% and the minimal loss is 3.485 respectively.

6 Testing and Conclusions

6.1 Experiments with Wiki model

We use Wiki model to test different images and captions downloaded from Wikipedia to check the model's performance.

(1) Caption:"Tour bus being used in France"
Wiki model can capture there is a decker bus in this image but it dismiss the information of tour since there is no patch can illustrate "tour" to Wiki model.

(2) Caption:"Closeup of the head of a northern giraffe"
Wiki model in this example can perfectly capture the key information, namely the giraffe. Also,

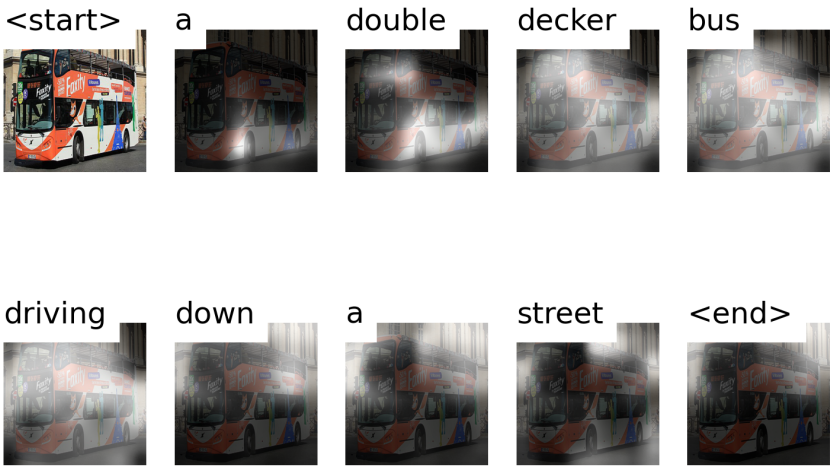

Figure 8: Experiment:Tour bus being used in France

Figure 9: Experiment:Closeup of the head of a northern giraffe

the attention tech shows Wiki model can pay attention to the object (giraffe) and background (field) according to next word.

(3) Caption:"Two singles players playing a tennis match at the Australian Open"
In this example, our Wiki model doesn't perform well. It mistook the color of the clothes perhaps because the background in that patch is blue. Also, It ignores there is a tennis competition rather than solely playing tennis.

6.2 Conclusion

We would like to conclude this report from the perspectives of both project and model performance.

Project The first conclusion from this project is that playing transfer learning in Natural Language Process project should be extremely careful. When we finished training COCO dataset and got a good pre-trained model, we then realized the word map between COCO dataset and Wiki dataset are absolutely different. That's why we have to add words from Wiki dataset to COCO's word map.

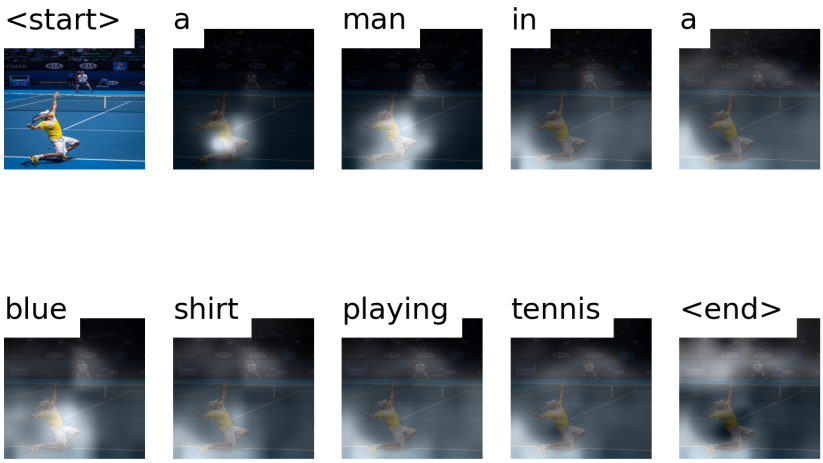

Figure 10: Experiment:Two singles players playing a tennis match at the Australian Open

Model Performance Firstly, we find out that the model is biased because the majority of the words created by the Wiki model come from the COCO word map. The key reason for this is that the COCO dataset contains many repeated or similar images and captions, allowing our model to precisely learn the association between word and image. However, the vast majority of data in the Wiki dataset is independent, implying that there are no identical photos or captions in the Wiki dataset, and that the weights used to obtain these minority words cannot be updated in a timely manner. As a result, words from the COCO word map are preferred over words from the Wiki word map.

Secondly, the dataset with imbalanced word frequency leads to the skewness problem. It is difficult for the Wiki model to converge on a skewed dataset, implying that the symbol <end> has a low likelihood of being generated. As a result, our model may need tens of thousands of words to explain an image that is illogical. To address this problem, we employ fine-tune, which entails freezing the entire model and only training the final layer of Encoder and Decoder. The evidence demonstrates that it can work, but the generated model can be biased.

Lastly, as the experiment shows that the words generated by this model are frequently right, the Wiki model indeed have learned the relation between images and associated captions. In addition, the attention mechanism can help the model focus on different parts of an image, resulting in a long and detailed sequence of words.

6.3 Future Works

For Encoder, we may try different kinds of backbones. Due to its large computational consumption. We may use other backbones like efficientnetV0 to improve performance or mobile net to deploy Wiki model on mobile devices.

For Decoder, since we construct embedding model and train this model from scratch, the embedding model may not acquire enough knowledge due to limited training samples. Thus, we may use pre-trained embedding model from other open sources like Transformer Library.

At last, it is impossible to explain an image with one sentence in many scenarios. Therefore, we may add a new symbol to represent ',', and get a new image caption model to illustrate one image using several sentences.

References

[1] Cesc Chunseong Park, Byeongchang Kim, and Gunhee Kim. Attend to you: Personalized image captioning with context sequence memory networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 895–903, 2017.

[2] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In *European conference on computer vision*, pages 15–29. Springer, 2010.

[3] Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In *European conference on computer vision*, pages 529–545. Springer, 2014.

[4] Jiuxiang Gu, Gang Wang, Jianfei Cai, and Tsuhan Chen. An empirical study of language cnn for image captioning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1222–1231, 2017.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[7] Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. Babytalk: Understanding and generating simple image descriptions. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2891–2903, 2013.

[8] Siming Li, Girish Kulkarni, Tamara Berg, Alexander Berg, and Yejin Choi. Composing simple image descriptions using web-scale n-grams. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 220–228, 2011.

[9] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[10] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.