Image Captioning of Novel Objects

Shen haowen

April 14, 2023

Abstract

Novel Image Captioning addresses the task of generating descriptions of novel objects which are not present in paired image. Intrinsically, this task can reflect the generalization ability of models in understanding and captioning the semantic meanings of visual concepts and objects unseen in training set, sharing the similarity to one/zero-shot learning. In the project, we first build a basic CNN- RNN model and find it has poor performance in the task of Novel Image Captioning due to lack of paired image. Then, inspired by the NOC model, we modify the original model and conduct some evaluations and visualization.

1 Introduction

1.1 Image captioning

Image captioning is the task of generating a textual description that accurately represents the content of an image. It combines computer vision and natural language processing techniques to enable machines to understand and describe visual information. The process of generating image captions typically involves the following steps: Image Feature Extraction: The input image is analyzed using a pre-trained convolutional neural network (CNN) such as VGG16, ResNet, or Inception. Sequence Generation: A recurrent neural network (RNN), often in the form of a Long Short-Term Memory (LSTM) network or a Gated Recurrent Unit (GRU), is employed to generate a sequence of words based on the extracted image features. The RNN processes the image features step-by-step and generates a sequence of words until a stop token or a maximum length is reached. Finally the models are typically evaluated according to a CIDEr-D, METEOR or SPICE metric.

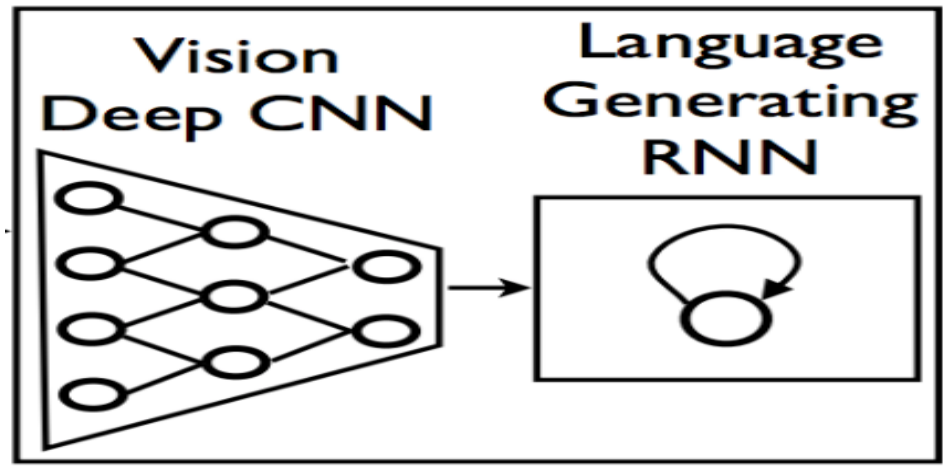

Figure 1: Typical Image captioning structure

1.2 COCO Dataset

The Microsoft Common Objects in COntext (MS COCO) dataset is a large-scale dataset for scene understanding. The dataset is commonly used to train and benchmark object detection, segmentation, and captioning algorithms. The dataset consists of 328K images.

Figure 2: Caption

1.3 Novel Image Captioning

While recent deep neural network models have achieved promising results on the image captioning task, they rely largely on the availability of corpora with paired image and sentence captions to describe objects in context. We would like to address the task of generating descriptions of novel objects which are not present in paired imagesentence datasets. As a result, we follow the novel object captioning split (NOC split) to evaluate our proposed method. Eight objects (bottle, bus, couch, microwave, pizza, racket, suitcase and zebra) are selected as novel objects in NOC split. Correspondingly, all image sentence pairs that include novel objects are removed from the standard training set. In the standard validation dataset, half of the pairs are randomly selected into new validation set, and others are selected into the test set.

2 model

2.1 Sample Visualization

Each figure is described with five sentences in the COCO dataset. We can do a sample visualization of this. I randomly choose a figure in the zebra category. The figure and annotations are as follows:

1. Zebra grazing on dry grass in a fried up field.
2. A mother zebra and her young on a grassy plain.
3. a zebra and a young zebra in a field.
4. A zebra with its offspring grazing in the desert.
5. Two Zebras in a brown field grazing and eating.

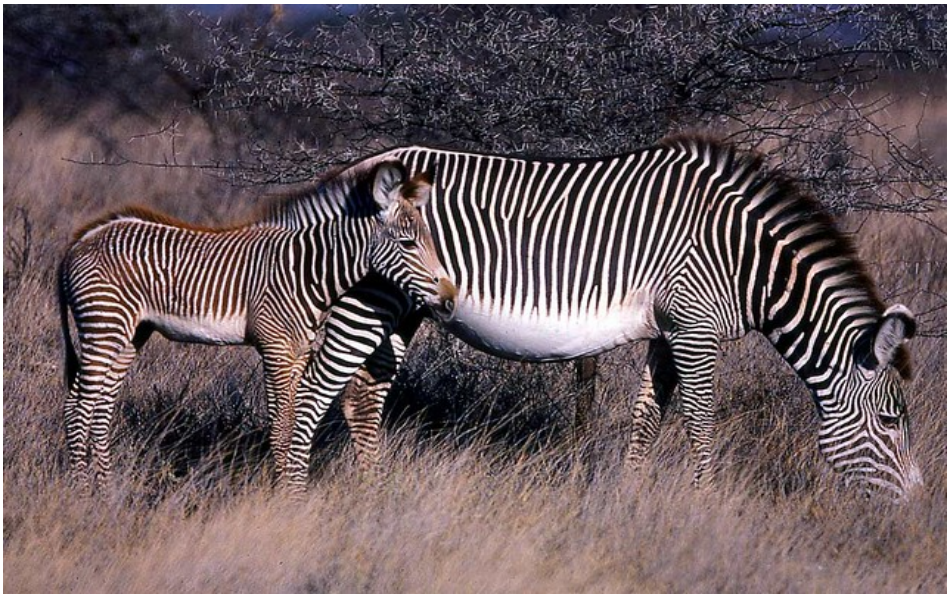

Figure 3: Sample figure of zebra

2.2 Data Pre-Processing

For the image pre-processing part, we conduct image Resize, RandomCrop, RandomHorizontalFlip and Normalize. As for the caption Pre-Processing part, we aim to create a model that can predict the next token of a sentence from previous tokens, so we turn the caption associated with any image into a list of tokenized words before casting it to a PyTorch tensor that can be used to train the network. A dictionary is also needed to embed the list of tokenized words into a list of integers, where every distinct word in the vocabulary has an associated integer value. As a result, a word2idx dictionary is created by looping over the captions in the training dataset. If a token appears no less than vocab_threshold times in the training set, then it is added as a key to the dictionary and assigned a corresponding unique integer.
For more straightforward understanding, let us take the above annotation as an example. Suppose that we plan to embed the annotation:

"Zebra grazing on dry grass in a fried up field."

First the model converts every letter in the caption to lowercase, and then uses nltk.tokenize.word_tokenize function to obtain a list of string-valued tokens like this:

[$< start >$, **'zebra', 'grazing', 'on', 'dry', 'glass', 'in', 'a', 'fried', 'up', 'field', '.',** $< end >$]

Also, $< start >$ and $< end >$ need to be added to the list to allow the model to recognize that it is a complete sentence. In the word2idx dictionary, the integer 0 is always used to mark the start of a caption and 1 for the end. Then, the list of tokens is turned into a list of integers, where every distinct word in the vocabulary has an associated integer value:

[0, 245, 476, 23, 435, 359, 43, 3, 735, 64, 236, 18, 1]

All of the captions in the COCO dataset are pre-processed using this same procedure. You can check the detailed code in the **data_loader.py, vocabulary.py** files.

2.3 Data loader

From [1] we know that using training samples with the same length can prove to be computationally efficient without degrading performance. Training on the same length data can also facilitate efficient batch processing, improve memory allocation and avoid padding for LSTM. As a result, we begin by first sampling a caption length where the probability of a drawn length is proportional to the number of captions with that length in the dataset. Then, we retrieve a batch size of image-caption pairs,

where all captions have the sampled length. Each time the data_loader iterates, a different caption length is sampled, and a different batch of training data is returned.

2.4 Network Architecture

The architecture comprises of a CNN encoder and an RNN decoder. The CNN encoder is a pre-trained ResNet on ImageNet, which is a VGG convolutional neural network with skip connections. Its effectiveness in tasks like image recognition has been demonstrated due to the residual connections that aid in modeling the residual differences before and after convolution through the identity block. Being pre-trained on ImageNet, it is proficient in extracting both low-level and high-level features that are valuable for image-related tasks. Hence, it naturally serves as an image feature encoder for our captioning purpose. Since our objective differs from traditional image classification, we discard the last fully connected layer and introduce a new trainable fully connected layer to better transform the final feature map into an encoding that is more suitable for the RNN decoder.

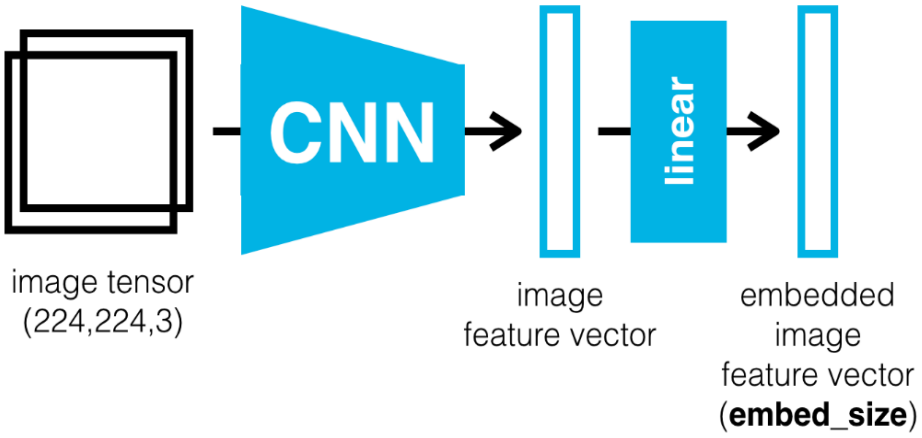

Figure 4: Encoder architecture

The idea of LSTMs generator is inspired by [2]. The LSTM model is trained to predict each word of the sentence after it has seen the image as well as all preceding words as defined by $p(S_t|I, S_0, \ldots, S_{t-1})$. As we can see in figure 5, we denote by S_0 a special start word and by S_N a special stop word which designates the start and end of the sentence. In particular by emitting the stop word the LSTM signals that a complete sentence has been generated. Both the image and the words are mapped to the same space, the image by using a vision CNN, the words by using word embedding W_e. The image I is only input once, at t = 1, to inform the LSTM about the image contents. I apply a fully connected layer on the hidden states at that timestamp to output a softmax probability over the words in our entire vocabulary, where we choose the word with the highest probability as the word generated at that timestamp. This predicted word is then fed back as input for the subsequent step, and the process continues until a caption of maximum length is generated or the network produces the "STOP" token, indicating the end of the sentence. You can check the **model.py, net.ipynb** files for detailed code.

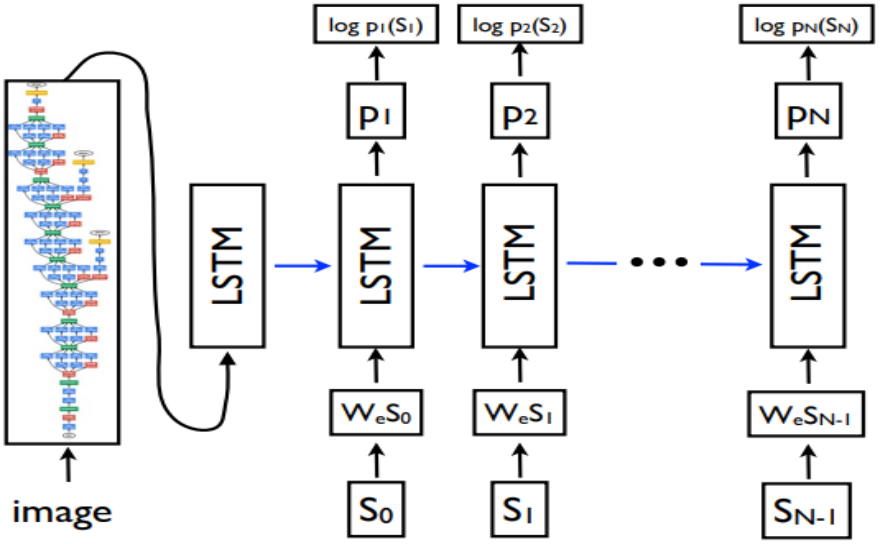

Figure 5: Decoder architecture

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

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

2.5 Experience results

I used a minimum word count threshold of 5, an embedding size of 512, and a hidden size of 512 as well. I trained the network for 3 epochs with the COCO dataset after removing eight cluster. I chose the Adam optimizer to optimize the CrossEntropyLoss and save the final model weights.
However, simply using this basic CNN-RNN network with unpaired training data can't give us ideal results. Without similar training data to finetune the model, it tends to generate captions with similar meaning but wrong subject. Here I visualize some of the results.

Figure 6: cluster = "Zebra"

Figure 7: cluster = "Bus"

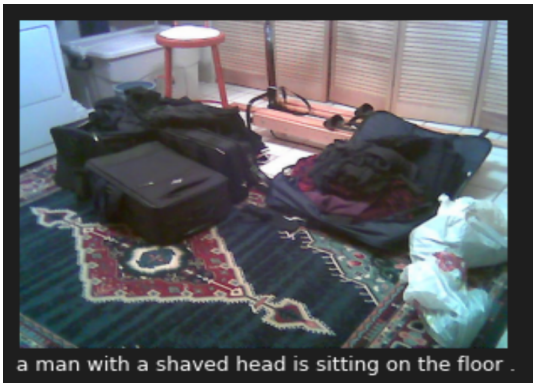

Figure 8: cluster = "Suitcase"

Figure 9: cluster = "Racket"

3 Novel Object Captioner (NOC)

Since it is difficult for traditional models to describe new objects without pairs of images and sentences about these objects, here I modified my original model based on the idea of [3].

3.1 Model introduction

NOC is the later work of Deep Compositional Captioning (DCC) from Subhashini Venugopalan and Lisa Anne Hendricks. The NOC model is based on a network structure similar to our original model. Both of them implement a pre-trained visual recognition model(CNN) and a LSTM-based language model for sentence generation. The novelty points of NOC is that it takes advantage of external data sources and employs joint training strategy in order to generate captions about diverse categories of objects outside the image-caption training data. To be specific, NOC uses ImageNet images with object labels as the unpaired image data source and sentences from unannotated text corpora such as Wikipedia as text data source. These are used to train the visual recognition CNN and language model respectively.

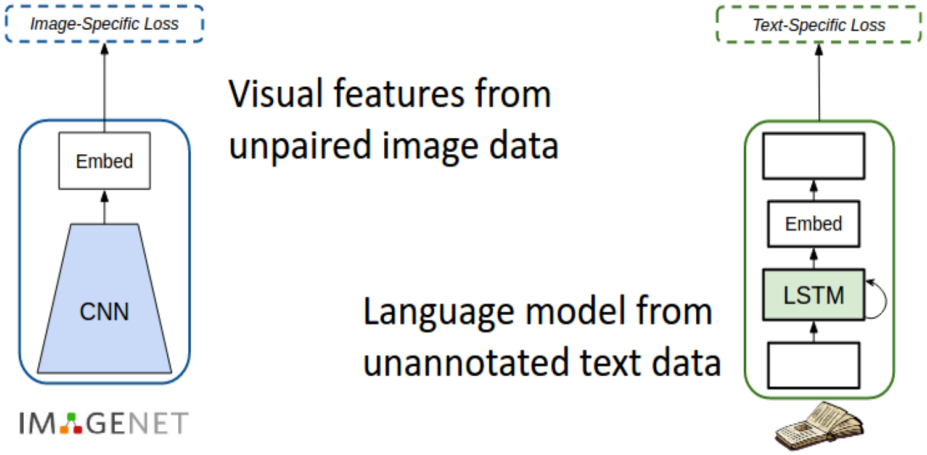

Figure 10: NOC Training effectively on external resources

3.2 Joint Training Strategy

NOC introduces Auxiliary Training Objectives in the model. Typically, image-captioning models incorporate a visual classifier pre-trained on a source domain (e.g. ImageNet dataset) and then tune it to the target domain (the image-caption dataset) just like our original model. However, important

information from the source dataset can be suppressed if similar information is not present when fine-tuning, leading the network to forget (over-write weights) for objects not present in the target domain. This is problematic in which the model relies on the source datasets to learn a large variety of visual concepts not present in the target dataset. As we can see in the figure below, while visual classifiers successfully identify the image as okapi, existing captioners still 'forget' the object and identify it as a horse due to lack of similar training data when fine-tuning. However, with NOC the model 'recovers' its memory and accurately identifies it as okapi without paired information either when fine-tuning.

Figure 11: Traditional captioners tend to forget objects not present in the target domain

To tackle the above problem, the authors employ the Joint Training Strategy in NOC. Specifically, the NOC model has three components: a visual recognition network, a caption model, and a language model. All three components share parameters and are jointly trained. During training, each batch of inputs contains some images with labels, a different set of images and captions, and some plain sentences. These three inputs train the different components of the network. Since the parameters are shared between the three components, the network is jointly trained to recognize objects in images, caption images and generate sentences. This joint training helps the network overcome the problem of forgetting, and enables the model to generate descriptions for many novel object categories.

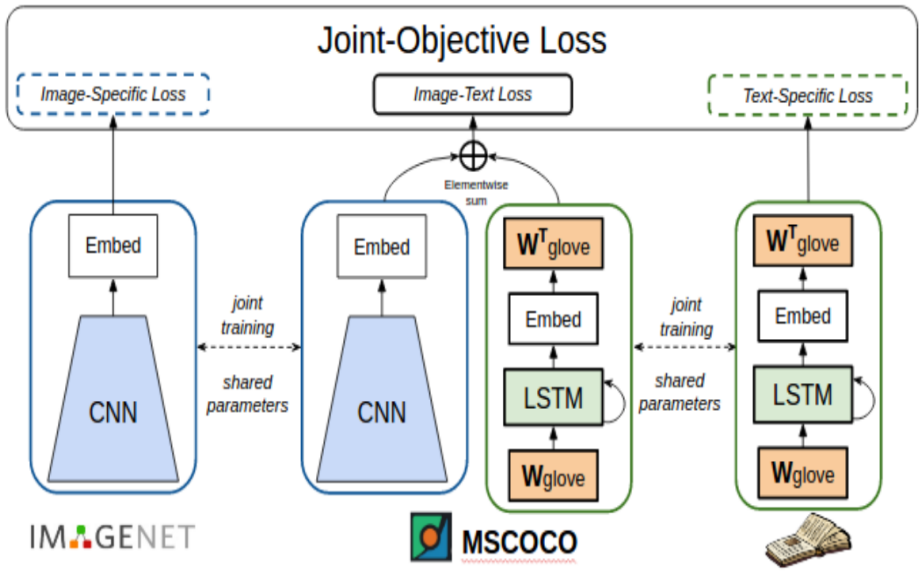

Figure 12: Joint Training Strategy

3.3 Training details

Based on my original model and inspired by the NOC model, I continued to use the pre-trianed ResNet on ImageNet and finetuned the final layer(left part of Figure 12). For the language model, I used GloVe

embeddings pre-trained on external text corpora, including Gigaword and Wikipedia. This implicitly enables the model to capture semantic similarity when describing unseen objects. Also I applied a LM vocabulary which contains a set of 72,700 words that also had GloVe embeddings. Due to the algorithm power limit, instead of training the GloVe embeddings and LM vocabulary from scratch, I used pre-trained versions of them.
To generate descriptions conditioned on image content, finally I combined the predictions of the language and visual recognition networks by summing textual and visual confidences over the vocabulary of words. To be specific, the NOC model's final objective is simultaneously minimizes the three individual complementary objectives:

$$L = L_{\text{IM}} + L_{\text{LM}} + L_{\text{CM}}$$

where $L_{\text{IM}}, L_{\text{LM}}, L_{\text{CM}}$ represents image-specific BCEWithLogitsLoss loss, text-specific CrossEntropyLoss loss and caption-specific CrossEntropyLoss loss respectively. These loss functions, when trained jointly, influence the model to not only produce reasonable image descriptions, but also predict visual concepts as well as generate cohesive text. Detailed code can be found in the **train.py** file.

```
# calculating the text-specidied loss
loss = criterion(outputs, targets.to(device))
# calculate the image-specified loss
image_loss = visual_loss(visual_features, visual_targets)
# calculating the caption-specific loss
language_model_loss = caption_loss(novel_outputs, novel_targets)

# sum up the final loss
final_loss = loss + image_loss + language_model_loss
final_loss.backward()
loss.backward()
optimizer.step()
optimizer.zero_grad()
```

Figure 13: Loss Code of Joint Training

For the training hyperparameters, here I set dimension of word embedding vectors as 300, dimension of lstm hidden states as 512, num_epochs as 25, learning rate as 0.001 with ADAM optimizer, imagenet_batch_size as 24, caption_batch_size as 30.
For the evaluation part, I applied CIDEr-D, METEOR, SPICE and F1 scores. F1-score considers false positives, false negatives, and true positives, indicating whether a generated sentence includes a new object. In addition, metrics like CIDEr-D and SPICE are also used to evaluate the quality of the generated descriptions.

3.4 Experience results

Here I compare our NOC model's F1-score with DCC and NOC in paper.

Method	F_{bottle}	F_{bus}	F_{couch}	$F_{\text{microwave}}$	F_{pizza}	F_{racket}	F_{suitcase}	F_{zebra}	F_{average}
DCC	4.63	29.79	45.87	28.09	64.59	52.24	13.16	79.88	39.78
NOC(paper)	17.78	68.79	25.55	24.72	69.33	55.31	39.86	89.02	48.79
NOC(ours)	13.48	70.82	26.35	21.68	67.73	51.25	27.48	87.88	45.83

Table 1: Comparison on F1-score

CIDEr-D, METEOR, and SPICE are popular evaluation metrics used in the field of Natural Language Processing (NLP) and specifically in the evaluation of image captioning systems. CIDEr-D is an automatic evaluation metric that assesses the quality of image captions by comparing them to multiple reference captions. METEOR evaluates the quality of generated captions by computing a harmonic mean of precision and recall scores. SPICE focuses on capturing the semantic similarity between the generated captions and the reference captions by measuring the semantic triples The evaluation of CIDEr-D, METEOR, SPICE is also presented.

Method	CIDEr-D	METEOR	SPICE
DCC		19.90	
NOC(papaer)		21.32	
NOC(ours)	62.50	19.38	14.81

Table 2: CIDEr-D, METEOR and SPICE evaluation metrics

Some visualization by our NOC model is also conducted and displayed as follows. You can find more in the **sample_image** folder.

Figure 14: cluster = "Zebra"

Figure 15: cluster = "Bus"

Figure 16: cluster = "Suitcase"

Figure 17: cluster = "Racket"

4 Conclusion

In this project, we build a model based on NOC to deal with the task of Novel Image Captioning on the COCO dataset. It reachs the average F1-score of 45.83 and 62.50 CIDEr-D, 19.38 METEOR and 14.81 SPICE. The result is not bad and most of visualization is quite promising.

References

[1] 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 Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul 2015. PMLR.

[2] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society.

[3] Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, Raymond Mooney, Trevor Darrell, and Kate Saenko. Captioning images with diverse objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.