Image Captioning Generator

Arash Darakhsh
darakhsh@chalmers

Albin Söderberg
albinso@chalmers

Wai Hong Poon
waiho@chalmers

Abstract

Image captioning is the task of automatically generating a descriptive caption of an image, using a machine learning system that combines both computer vision and natural language processing methodologies. The aim of this study was to investigate the creation of such a model with the dataset Flickr8k providing images and captions for training, validating and testing. An encoder-decoder network was built, composed of the pretrained CNN VGG19 as the encoder combined with a self-designed decoder made up of an LSTM+Classifier block. After training, the model was evaluated using ROUGE/BLEU scores in addition to human overview. The results proved unsatisfactory, not generating well descriptive captions, leaving much room for improvement.

1 Introduction

While robots have so far had a very tough time doing this task, humans can simply define what an image is about. This type of work was thought to be insurmountable for a machine only a few years ago. We can now integrate computer vision and natural language processing techniques to recognize components of an image and have a program explain them in English thanks to the tremendous progress of processing infrastructure for deep learning models.

Since it is an extremely difficult undertaking to be able to automatically describe the content of an image using properly constructed English words could have a significant influence, such as by assisting those who are visually handicapped in understanding the content of images on the internet. In addition to describing the things shown in an image, a description must convey their relationships to one another, their characteristics, and the actions they are taking. Additionally, a language model is required in addition to visual comprehension since the aforementioned semantic information must be conveyed in a language like English.

In the first phase of this study, CNN models for image recognition are used to extract characteristics from photos. For evaluating the results of these three CNN architectures trained, we employed VGG-19 in the CNN encode model. VGG-19 can accommodate up to 19 layers. Compared to VGG-16, which has a different number of CNN layers, it is an improvement. Numerous workloads and datasets have seen better performance with VGG-19, a deep CNN, than with baselines. One of the most popular image recognition architectures right now is VGG. CNN layers are stacked to create it, however because to problems like diminishing gradients, the model's depth is constrained. The VGG-19 can categorize photos into 1000 different object categories after being trained on more than one million photographs. As a result, the model has picked up detailed feature representations for a variety of images. The VGG-19 model is a 19-layer with 16 convational layer and 3 fully-connected CNN that exclusively uses 33 filters with stride and pad of 1, as well as 22 max-pooling layers with stride 2. The VGG-19 model accepts a color image as input, and creates a 3-channel image by allocating 3 color channels in that order. For VGG-19 training and testing, all regional images are scaled to 2242243 and cropped from the 3-channel image. In this approach, the revised VGG-19 model and the 3-channel image show modest variations over time.

The main inspiration of our work comes from

(1).Through this project, we propose to The goal of this project is to create a deep learning model that can recognize items in an image and generate captions for them using the Flickr8k dataset. We aim to make the contributions on a convolutional neural network and LSTM model to unveal information from images.

2 Background

An image caption generator (ICG) can be implemented using several different neural network architectures. In general an ICG consists of an encoder-decoder network. The encoder takes an image I and compresses its feature representation to a fixed sized latent vector z. The encoder takes the latent vector z as input and outputs a target caption t of variable size. The output caption t is made out of a vocabulary V consisting of a set of tokens. The goal of a ICG is to produce a truthful caption to the input image.

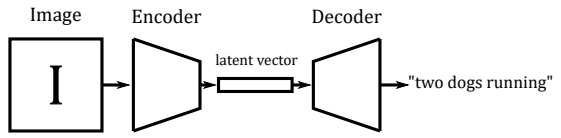

Figure 1: Basic structure of an ICG. The image is passed through the encoder network outputting a latent vector. The latent vector is then fed into the decoder to generate a caption for the image.

2.1 Performance measure

For certain natural language processing (NLP) tasks, the performance measure can be hard or ambiguous to define. In the case of image captioning there are several captions that would fit the image, thus a flexible performance metric is required that doesn't only have one optimal target but several similar ones. Ideally one would have experts rate the captions on a discrete point scale and the performance score for the ICG would be the mean score. This process is however time consuming, subject to bias from the humans and has no reuseability. Thus there is a need for an objective performance metric that can rate captions similar to the target caption with similar score.

The BLEU score is a meassure of how similar two texts are and was developed for machine translation tasks (2). The BLEU score uses a modified N-grams precision to measure the similiary. The standard BLEU score is the geometric mean of all N-gram precisions up to a value N. A BLEU score of 1 means a perfect caption and a score of 0 translates to a caption that isn't similar to the target caption. The BLEU score is very fast and simply to calculate and high has high reuseability, however it has some aspects that doesn't make it perfect. The BLEU algorithm doesnt take meaning into consideration, it's sensitive to the specific tokenizer that is used for the NLP task and doesn't take sentence structure into account (3). While the BLEU score is a precision based performance score, the Rouge score which is divided into 3 main scores, rouge-1, rouge-2 and rouge-L, is a recall based performance score (4).

3 Method

3.1 Encoder

The encoder used for the ICG is the popular pretrained vgg19 model (5). The VGG model has a convolutional part and a classification part. The convolutional part consists of several sequences of convolutional layers, batch-normalization layers and max-pooling layers. The classification layers flattens the output of the convolutional part and passes it through 3 linear layers downsizing the dimension from 25k to 1000 where each of these dimension represent some class. The encoder network used for our ICG uses vgg19's entire convolutional part and the first linear layer in the classification part. These weights are kept frozen. This is followed up by a trainable linear layer that maps the latent vector z of size D_z to the embedding dimension D_{emb} used as input for the decoder network.

3.2 Decoder

The decoder for the ICG is a long short-term memory (LSTM) network. The input to the encoder consists of both the latent vector and embedded tokens, see section 3.3.1 and 3.3.2 for the distinction in the LSTM input. Tokens from the vocabulary V are first embedded before they are fed into the LSTM using a regular embedding layer. Each output from the LSTM is a hidden vector h with size D_h. The hidden vector is then passed through an output embedding layer that maps the hidden vector h to a output vector v, of size $|V|$, containing logits corresponding to a probability distribution for the next sampled token. A categorical sampling is used to sample from the top 3 tokens with

the highest logits. This sort of sampling introduces some stochasticity in the case of mode collapse where the network reaches a local minimum and produces the same output regardless of input.

3.3 Network Modes

The two tasks of training the network and running inference on it have different requirements. To accommodate these differences, it was decided to split the forwarding method into two modes, one for training and one for inference.

3.3.1 Training

During training, teacher forcing is applied in order to improve convergence. Teach enforcing is the method of guiding the network during training, particularly useful for the case of nlp generative systems (6) where the target output is a sentence. Due to the dynamic nature of the generation task, this methodology helps to stabilise performance and lessens the risk of nonsensical output. In our ICG it is used in the decoder, specifically the LSTM network. The encoded latent vector is given as initial input to the LSTM. The output at each step of the LSTM is fed into a linear classifier, with classes corresponding to each word in the vocabulary. Applying softmax on the output gives a probability vector over all classes, on which any generation strategy (such as greedy or beam search) can be used to retrieve a word. During training, only the probability vector is kept, for calculating the loss, while the word output is discarded. Instead of chaining the output word as the next input to the LSTM, the "correct word" from the target caption is used as input to guide the network forward. This is the teacher forcing step. When either a stop token has been generated or a max-token limit has been reached, forwarding is stopped. Here the loss is calculated, and used for backpropagation to adjust the parameters. The loss is defined as the sum of all cross-entropy losses for each output of the LSTM, where the probability vector is compared with the ground truth one-hot-encoding. The PAD token is used as an ignored index in the cross entropy loss computations, which means that when the target is a PAD, i.e. the sentence is completed, all subsequent tokens generated have no effect on the loss.

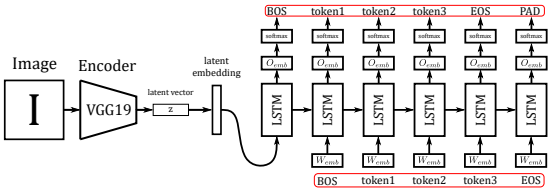

Figure 2

3.3.2 Testing/Inference

Running inference on the model differs in the LSTM throughput. Unlike the training process, the word generation has no target sentence to guide it along. It uses the output at each step of the LSTM, in the form of a generated word, as the input for the next step. This is done until the sentence is complete (stop token generated) or has reached a set maximum length. In order to measure the quality, ROUGE score is used as the objective metric, in addition to human evaluation.

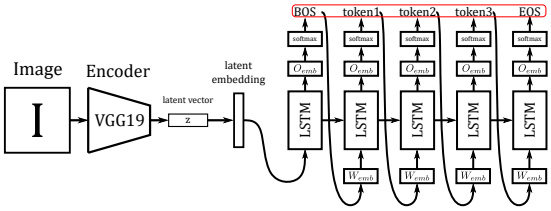

Figure 3: Basic structure of an ICG. The image is passed through the encoder network outputting a latent vector. The latent vector is then fed into the decoder to generate a caption for the image.

3.4 Hyperparameters

Parameter	Value
batch size	100
learning rate α	3e-4
hidden size D_h	256
embedding dimension D_{emb}	512
latent size D_z	4096
dropout rate	0.4
maximum sentence length	20
training epochs	30

These hyperparameters were tweaked during training to achieve better results. A batch size of 100 was deemed appropriate. A smaller batch size of 10 could lead to unstable results, while using a greater amount of images could lead to a model that generalises to the point where learning is halted. The learning rate was set to a common value for the Adam optimizer. The value of the embedding dimension was decided to mimic used mimics the model used by Vinyals et al. (1). Some

experimentation was done with weight decay, but as it seemed to worsen the results, it was discarded in the final model.

3.5 Dataset information

The dataset used is the flickr8k consisting of 8000 images, split up as 6000 train, 1000 validation and 1000 test images. Each image has 5 text labels, with independent image descriptions, consisting of a caption and a lemmatized version of the caption (lemma captions, from here on called lemmas). Only the first of the five available labels are used as the target for training. From the training captions and lemmas respectively a vocabulary is created to interger-encode the tokens. The tokenizer used is Spacy's english tokenizer. In addition, the special symbols "PAD", "UNK", "BOS" and "EOS" are added. The captions and lemmas are represented by an integer tensor with size equal to the longest caption in the dataset, where shorter captions are padded using the integer representation of "PAD".

4 Results

The results obtained during the training process are presented in figure 4. These results were pretty consistent even with larger changes in hyperparameters, in the range of very small networks and very large networks. From the figure we see that the training loss plateaus very early in the training phase. The validation metrics don't change that much either after the initial phase, meaning that the network has issues generalizing.

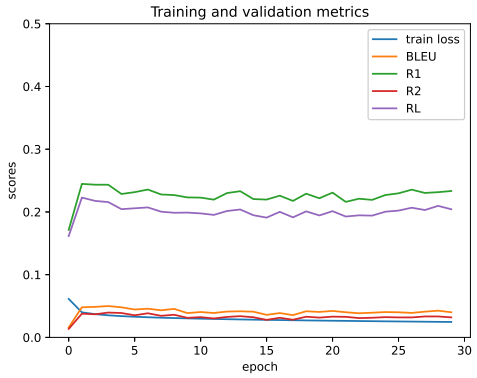

Figure 4: Training and validation metrics for training the network on 30 epochs.

The performance metrics were calculated after the training for the test set and the NLP related scores are presented in table 1. As mentioned earlier in section 2.1, the ideal values for the performance metric should be close to 1. Thus, the test results obtained here are very poor, especially the rouge-1 and BLEU scores that are basically 0. This poor performance is also reflected in the generated captions.

Table 1: Average test scores for 1k test images.

Test scores	Value
BLEU	0.040
Rouge-1	0.241
Rouge-2	0.033
Rouge-L	0.212

4.1 Generated captions

The generated captions range from nonsensical sentences with seemingly no connection to the image to sentences containing some reasonable parts. It did not succeed in the task of being able to generate well descriptive captions. However it usually managed to generate reasonable lemma sentences, and capture the general sentence structure including the BOS and EOS tokens. As the EOS was set as a separate token from general punctuation symbols, it could create the unwanted behavior of a caption being split in several sentences or multi-dots being present, as seen in 5. Below are two examples of images, with their target captions as well as the generated one.

Figure 5: Caption: A group of man wear yellow shawl walk in a line

Prediction: Girl in red and red jacket be jump over some tree . .

Figure 6: Caption: A brown dog be sit on the grass wear a blue sport jacket.

Prediction: Child be play on a sidewalk with a blue and yellow ball in park.

5 Discussion

The results we obtained were not to our expections and the poor performance is reflected in the scores in 1 and the generated captions for the example images. Potential changes and modifications that could potentially help the training process are discussed in section 5.1

5.1 Further work

5.1.1 Pretrained embedding

One possible weak point of our model is the use of randomized embeddings instead of integrating pretrained embeddings into the network. With the way the model is built, the embedding layer is trained alongside the layers of the LSTM and its output classifier. As the embedding is used as input to the LSTM, this in turn affects the output. Updating the parameters of both could have negative consequences for the model, where it fails to both learn good word representations and adjust the LSTM+Classifier layers. Using pretrained embeddings would therefore be a good idea that could potentially lead to significant improvements of the model performance. Either a complete pretrained model could applied, or one could train the embedding separately on the training data.

5.1.2 Special token symbols

The encoding of our lemma captions were surrounded with 'BOS' and 'EOS' special characters. There is an argument to be made that the these tokens are redundant. Since the latent embed vector is the first one could argue that the encoder part could adapt such that giving the latent embed as input, initializes the hidden state such that the network knows the it should generate the first token in the caption instead of the 'BOS' symbol. Similarly the need for 'EOS' is redundant if we have period in our vocabulary since captions of images are very rarely consisting of more than one sentence.

5.1.3 Captions

The idea behind using lemmatized captions instead of regular captions was that there would be less room for error. The lemmatized vocabulary would be smaller, and thereby more efficient. The NLP part of the model would not need to be as advanced either, as the grammar complexity would decrease. It would however be worth investigating the effects of the type of caption used.

The Flickr8k dataset used provided five lemma+caption pairs per image. As a supporting file to the dataset, there was an incomplete grading of the captions, made by human evaluators. It could have been a useful tool for retrieving only the best among the captions, had it been complete, as the quality of the captions influence the ICG results. Out of the five captions, only one was used, as there was no easy way to distinguish their quality. Using a combination of them as a target, could have been an alternative option.

5.1.4 Alternative datasets

To train our model, we utilized the modest dataset Flikr8k of 8000 photos, however the business level model often used bigger datasets of more than 100,000 images for improved accuracy. If we want to improve the model to give more comprehensive and more accurate captions, we need to use a larger dataset like Flickr30k, MSCOCO which has 328,000 images, and Stock3M which is even 26 times larger than the former.

5.1.5 Architecture variants

ResNet-50 and Inceptionv3 will be used in place of VGG-19 as the feature extractor in the deeper layers network that we are building. GRU can also be used in place of VGG-19. After that, we'll compare the various combinations of the models VGG+LSTM, VGG+GRU, Inceptionv3+GRU, and ResNet-50+LSTM. This will aid in our analysis of how the CNN element affects the network as a whole. A mechanism based on attention is another option. Since they can dynamically focus on the many components of the input image while

the output sequences are being created, they are becoming in popularity in deep learning.

6 Conclusion

We have demonstrated a CNN-LSTM system that can automatically analyze an image and produce a plausible plain English description. It is built on a CNN that compresses a picture into a small representation, then a recurrent neural network generates a sentence that corresponds to the compressed image. The model is trained to increase the sentence's likelihood given the image. A metric used in machine translation to assess the quality of generated phrases, ROUGE or BLEU, is utilized in experiments on several datasets to demonstrate the robustness of the model in terms of qualitative findings and quantitative evaluations. These tests show that the effectiveness of techniques will grow with the number of the datasets that are available for image description.

References

[1] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," 2014.

[2] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu," *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 2001.

[3] E. Reiter, "A structured review of the validity of BLEU," *Computational Linguistics*, vol. 44, pp. 393–401, Sept. 2018.

[4] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Annual Meeting of the Association for Computational Linguistics*, 2004.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[6] A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," 2016.