Image Captioning using Neural Networks

Rishabh Aryan Das

UFID: 0536-9273

Amit Bhadra

UFID: 6649-4087

***Abstract*—The fundamental problem in artificial intelligence is generating captions for images which is associated with computer vision and natural language processing. In this project we attempted to build a generative model that aims to produce natural sentences to describe an image using latest computer vision and machine translation technology. The model present here attempts to maximize the likelihood of the target sequence for a given input image.**

***Index Terms*—InceptionV3, attention mechanism, recurrent neural nets, CNN, tokenizer**

I. Introduction

It is a quite difficult task to describe an image with proper articulation. This has various implications in several fields the foremost of them being to assist visually impaired people. The field of computer vision is mainly associated with object recognition or image classification. A proper description of an image not only highlights the objects present in it but also explains how the objects are related to each other. The description should be able to explain their attributes properly. Previous attempts at this problem has tried to combine existing solutions of the sub problems in order to generate a caption from the image. In our approach, we have built a single end-to-end architecture involving both CNN and RNN in which an image *I* is provided as input and is trained to produce a target caption $S = S_1, S_2, ...S_n$ by maximizing the likelihood *p(S—I)* such that individual word S_i is received from multiple sets of key-value pairs which illustrates the given image.

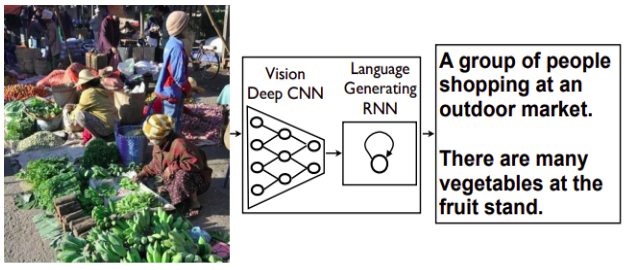

Fig. 1. Architecture of a Neural Network showing a convolutional Neural Network and a Recurrent Neural Network for contextual awareness of an image and its corresponding text.

Recent advancement in machine translation techniques has made it possible to translate a sentence S to T, it's target language, by maximizing $p(T|S)$. Machine translation was a very complicated task which involved translating words individually, aligning them, reordering, etc. But translation can be achieved in much simpler way using Recurrent Neural Networks and reach the state of the art performance. An "encoder" RNN reads the source sentence and transforms it into a rich fixed-length vector representation which in turn is used as the initial hidden state of a "decoder" RNN that produces the target caption.

In this project, we have used a deep convolutional neural network CNN in the encoder. CNN are able to produce rich representation of the input image by embedding it to a fixed-length vector such that this representation can be used for various vision tasks. So we applied a pretrained CNN encoder to classify images and feed the last hidden layer into an RNN which acts as a decoder and generates captions.

II. Dataset

For this project we used to COCO dataset. COCO here stands for **Common Objects in Context** and is designed to represent a vast array of objects that we regularly encounter in everyday life. The COCO dataset contains several millions of labeled data which is available for training supervised models.

Dataset	train	valid	test
MSCOCO	82783	40504	40775

III. Model

In order to produce captions from images we have developed a neural and probabilistic framework. Recent developments in statistical machine learning have shown that, we can generate state-of-the art results using a powerful sequence model, by maximizing the probability of the correct translation on an input sentence. These models are implemented using recurrent neural networks which encodes the variable length input into a fixed dimensional vector and uses this representation to decode it to the desired output sentence. Therefore by using the same technique we can translate an input image into a caption. By using the following formulation we tend to maximize the probability of the correct description of the input image:

$$\theta^* = arg\max_{\theta} \sum_{(I,S)} logp(S|I;\theta) \tag{1}$$

Here θ are the parameters of the model, I is the input image and S is the desired transcription. The length of the S is

unbounded since it can represent any sentence. Using chain rule over $S_0, \ldots, S_N$

$$logp(S|I) = \sum_{t=0}^{N} logp(S_t|I, S_0, \ldots, S_{t-1}) \tag{2}$$

which is represented without the parameter θ for our convenience. Here, N is the size of the example. During training, (S,I) is given as input and the model intends to optimize the log probabilities as mentioned in (2) using SGD. Therefore for convenience we model $p(S_t|I, S_0, \ldots, S_{t-1})$ with an RNN represented by memory h_t. This hidden state gets updated on receiving a new input x_t. They are related by a non-linear function represented as:

$$h_{t+1} = f(h_t, x_t) \tag{3}$$

We have implemented two crucial design choices to make our decoder RNN more concrete: the concept class of f and the method of feeding images and words as inputs x_t. To represent f we use a LSTM network, which is the best implementation for sequencing techniques like translation.
For representation of the images, we use the pre-trained InceptionV3 CNN architecture.

IV. LSTM-based Sentence Generator

We choose f in (3) because of it can work with vanishing and exploding gradients, which is why people move away from RNNs. To solve this issue, a particular form of recurrent nets called LSTM was introduced.
LSTM contains a memory cell c which encodes information each time step of the inputs that are observed up until a time t. The behavior is controlled by "gates"-layers which are applied multiplicatively. If the value returned is 1 then the value is kept or canceled if 0. There are 3 gates that are being used which are the forget gate f which deals with whether we should remove the current value, the input gate i and whether to output the new cell value(output gate o). This is given by:

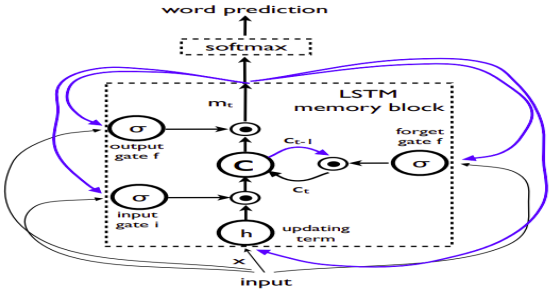

Fig. 2. LSTM: A cell c in the memory block controlled by 3 gates where blue lines represent recurrent connections; ouput m at time t-1 is fed back into the memory at time t. The cell value is fed back through the forget gate and the predicted word at time t-1 is fed back to the memory output m at time t into the Softmax for caption generation.

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \tag{4}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \tag{5}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \tag{6}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot h(W_{cx}x_t + W_{cm}m_{t-1}) \tag{7}$$

$$m_t = o_t \cdot c_t \tag{8}$$

$$p_{t+1} = Softmax(m_t) \tag{9}$$

where $\cdot$ is the product with a gate value and the various W matrices are trained parameters. The multiplicative gates solve the problem of exploding and vanishing gradients which is the benefit of using LSTM. The activation functions used are sigmoid $\sigma(\cdot)$ and hyperbolic tangent $h(\cdot)$. The m_t in last equation is fed into the Softmax which will produce a probability distribution p_t over all words.

V. Attention Mechanism

In natural language processing the attention mechanism brought about with it improvements over the encoder-decoder models. This approach is now heavily used across several fields of AI.
Bahdanau et al(2015) came up with a very novel idea. They suggested that in the context vector along with all the input words we should give relative priorities to each of them.
To create the context vector we put importance on embeddings

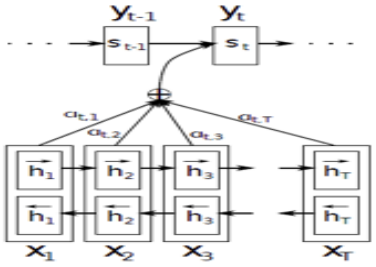

Fig. 3. This generates a sequence of annotations (h1, h2,. . . .., hTx) for each input sentence.

of all the words i in the hidden states. This is implemented simply by taking the weighted sum of the hidden states. The context vector c_i for the output word y_i is produced using the weighted sum of the annotations.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{10}$$

The weights α_{ij} are calculated using softmax given by :

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{11}$$

$$e_{ij} = a(s_{i-1}, h_j) \tag{12}$$

e_{ij} is the output score of a feedforward neural network described by the function a that attempts to capture the alignment between input at j adn output at i

VI. Training

The decoder RNN model is trained to predict each word of the expected caption based on the input image as defined by $p(S_t|I, S_0, \ldots, S_{t-1})$. Thus the LSTM in the decoder can be seen in unrolled form where a copy of the LSTM memory cell is created for the image and each caption word such that all LSTMs share the same parameters and the output m_{t-1} of the decoder at time $t-1$ is fed to the decoder at time t. All recurrent connections are unrolled into their corresponding

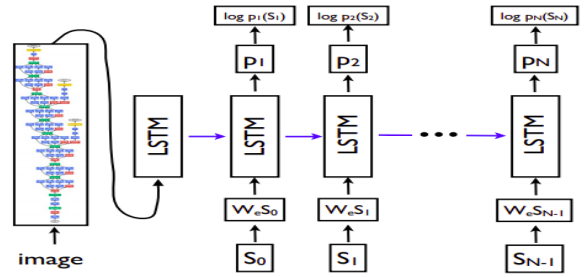

Fig. 4. Here we have a LSTM model along with a CNN image embedder and word embeddings. The blue lines highlights the unrolled connections among the LSTM memory cells. All the LSTM share the same parametes.

feed-forward connections. If we denote the input image by I and the input caption by $S = (S_0, \ldots, S_N)$, the unrolling procdure reads:

$$x_{t-1} = CNN(I) \tag{13}$$

$$x_t = W_e S_t, t \in 0-1 \tag{14}$$

$$p_{t+1} = LSTM(x_t) t \in 0-1 \tag{15}$$

where we represent each word as a one-hot vector S_t of dimension equal to the size of the dictionary. The captions are tokenized by particular "$start$" and "end" phrases which denotes the start and end of the sentences. So emitting the "end" phrase enables the LSTM to understand the end of an entire sentence, with respective mapping of the image and its corresponding text. At $t = -1$, the image I is given as only input one time, to provide the LSTM with the extracted image features. In order to avoid overfitting, we don't provide an image at every time step, which otherwise performed poorly. The loss is the sum of the negative log likelihood of the correct word at each step:

$$L(I, S) = -\sum_{t=1}^{N} log p_t(S_t) \tag{16}$$

The loss is minimized over the entire set of parameters including the LSTM, CNN (top layer of the image embedding) and the word embeddings W_e.

A. Steps to run the code

The primary execution script is **main.py**. It has the following input parameters:

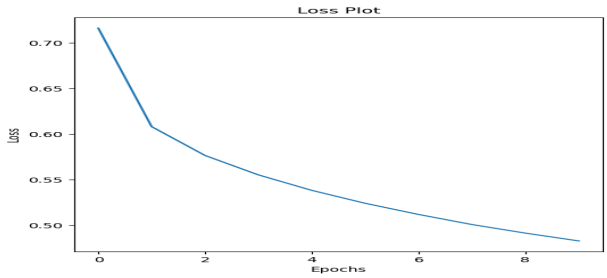

Fig. 5. Loss plot of training the model over 10 epochs

- *–annotations_dir* This is the path to the directory where the annotations will be saved. The default value is **annotations/**
- *–images_dir* This is the path to the directory where the images of the dataset are downloaded and stored. The default value is **train2014/**
- *–cache_inception_features* Set this parameter to true if you wish to store the extracted features of the training image set. It is advised to set this parameter to true if you are running the code for the first time. The default value is **False**. The extracted features are stored in the directory **extracted_v3Features/**
- *–train_examples* Use this parameter to pass an integer value which determines the subset of the whole dataset the network will train on. If you wish to train on the whole dataset then ignore this argument.
- *–test_image* This parameter holds the path to the testing image. The result image with attention plots will be found in the **results/** directory with index number as 108.
- *–rmv_data* The COCO dataset is 13 GB in size. So after extraction of the features by the CNN the main script deletes the dataset to save space. If you wish to retain the downloaded images then set this parameter to False. The default value is **True**

So to train the model first time on 10000 images run the below command:
$python main.py –cache_inception_features True –train_examples 10000
To test the model on a particular image run the below command:
$python main.py –test_image path_to_image
The evaluation results along with their attention plots are stored in the **results** directory.

VII. Evaluation Metrics

It is quite a ambiguous task to determine the quality of an generated caption. Prior works in this area have proposed several evaluation metrics. The BLEU score is the most commonly used metric which is used in image description literature. It is basically a form of precision of word n-grams between generated and reference sentences.
Besides BLEU, there is another metric which is the perplexity of the model for a given transcription. The perplexity is the

geometric mean of the inverse probability for each predicted word. This metric is used to choose models and in hyperparameter tuning. But BLEU score is the preferred choice over this.

In most recent literatures on image description, they have used a metric where they rank the descriptions corresponding to an image which helps in having a standard evaluation metric like precision or recall. But such a ranking based evaluation suffers when the image complexity increases many folds as its corresponding description shoots up exponentially. The demand for such computation renders such a process to be infeasible. This same roadblock in present in speech recognition where we generate a sentence with similar meaning to a given audio clip. Early attempts in this field tried to classify isolated phonemes or words. But with the recent development of generative models have proved to be feasible and works for huge dictionaries.

Out of all the available descriptions which does not mean the ground-truth,confusion involving the evaluation of an image should be based on metrics instead of ranking.

Metric	BLEU-4	METEOR	CIDER
NIC	27.7	23.7	85.5
Random	4.6	9.0	5.1
Nearest Neighbor	9.9	15.7	36.5
Human	21.7	25.2	85.4

TABLE I
SCORES ON THE MSCOCO DEVELOPMENT SET

VIII. RESULTS

The primary challenge that we faced when training our models was that of overfitting. There is a requirement of large amount of data for supervised approaches, but standard datasets have less than 100000 images. The task of generating a descriptive caption is much harder than object classification and data driven approaches have seen promising developments because of large datasets like ImageNet. Hence we assume that even with the results we obtained which are of sufficient quality, our method will gain more popularity in the coming years over human engineered approaches in the next few years. This is due to the fact that sizes of training data will keep on growing.

Several techniques were explored to deal with the problem of overfitting. The most obvious way was to initialize the weights of the CNN component of our system to a pretrained model. We implemented this and it did help quite a lot in terms of generalization. Another set of weights that could be initialized are the word embeddings, W_e. We tried to initialize them from a large new corpus, but there was no significant improvements in performance and hence they were just left uninitialized for simplicity. To avoid overfitting we tested several techniques like dropout and emsembling models. We decided on the size of the model by trading of number of hidden units vs depth of the model. A few BLEU point imporvements were made by means of dropout and ensembling as reported in this paper.

All the set of parameters were trained using stochastic gradient descent with a fixed learning rate. No momentum technique was applied. We initialized the random weights except for the CNN weights which were pretrained. For the size of the LSTM memory block and embeddings we used 512 dimensions. Descriptions were preprocessed with basic tokeninzation techniques keeping all the words that appeared at least 5 times in the training set.

A. Generation Diversity

After training a generative model that outputs $p(S|I)$, we need to determine whether the model generates an ideal caption and if the generated sentence is diverse and of standard quality. Generating captions using beam search decoder instead of the best hypothesis yields better diversity and highlights different features of the same image. The agreement in BLEU score between the top 15 generated sentences is 58 which is similar to that of humans among them. This validates the diversity in outputs of our model. The bold sentences are not present on the training set. If we select the best candidate, the sentence is present in the training set 80% of the times. While analyzing the top 15 generated captions, we see about half of the times a completely novel description,but still with a similar BLEU score, thus proving that they are of standard quality yet they provide adequate diversity.

A man throwing a frisbee in a park.
A man holding a frisbee in his hand.
A man standing in the grass with a frisbee.
A display case filled with lots of donuts.
A display case filled with lots of cakes.
A bakery display case filled with lots of donuts.
A close up of a sandwich on a plate.
A close up of a plate of food with french fries.
A white plate topped with a cut in half sandwich.

Fig. 6. N-BEST EXAMPLES FROM THEMSCOCOTEST SET. BOLD LINESINDICATE A NOVEL SENTENCE NOT PRESENT IN THE TRAINING SET

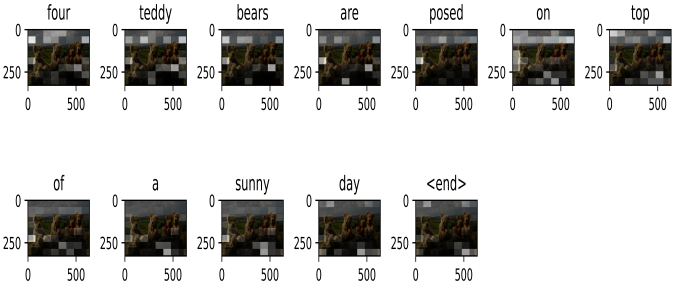

Fig. 7. Attention plot 1

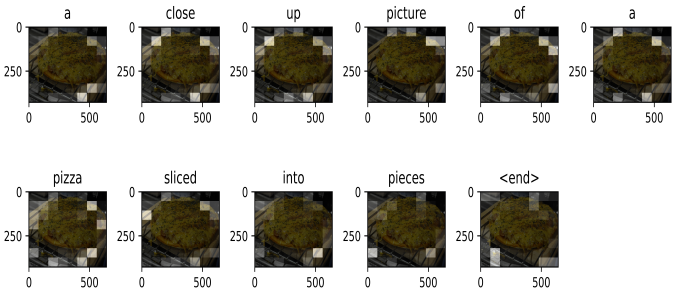

Fig. 8. Attention plot 2

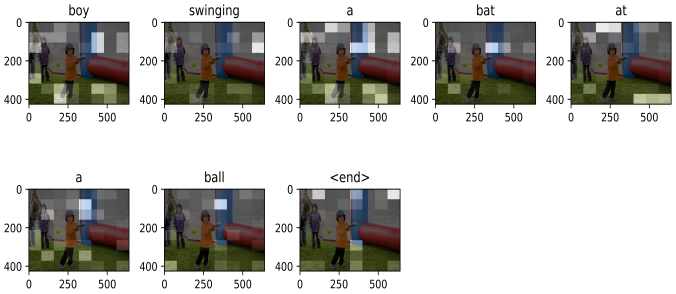

Fig. 9. Attention plot 3

B. Embeddings Analysis

To represent the previous word S_{t-1} as input to the decoding LSTM producing S_t, we used word embedding vectors, which have the advantage of being independent of the size of the dictionary. Furthermore, these words embeddings can be jointly trained with rest of the model. The learned representations have captured some semantic from the statistics of the language.

IX. Conclusion

We have presented an end-to-end neural network system that can automatically view an image and generate a reasonable description in plain English. This is based on a convolutional neural network that encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence. The model is trained to maximize the likelihood of the sentence given the image. The generated sentences are quite reasonable and quantitative evaluations, using either ranking metrics or BLEU, a metric used in machine translation to evaluate the quality of generated sentences. It is clear from these experiments that, as the size of the available datasets for image description increases so will the performance. One drawback of this model is the training time. Since the size of the COCO dataset is around 13 GB it takes a huge amount of time to train this model over the entire dataset.

One way to address this issue is to implement a generative pre-trained Transformer (GPT) which is an autoregressive language model that uses deep learning to produce human like text. This will speed up the training process significantly and improve the quality of the generated captions.

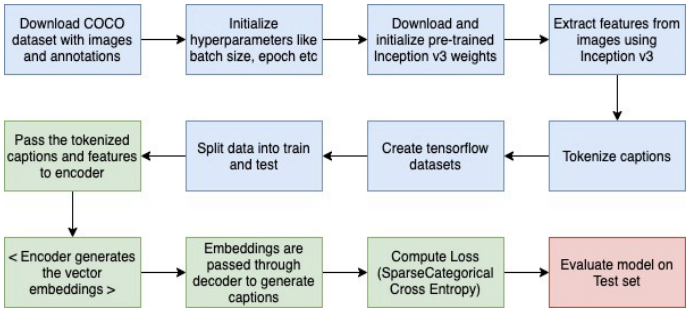

Fig. 10. Flow of the codebase

References

1 Vinyals, O., Toshev, A., Bengio, S., and Erhan, D., "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

2 V, A. L. P. and Tolunay, E. M., "Dank learning: Generating memes using deep neural networks." *CoRR*, 2018.

3 Luong, M.-T., Pham, H., and Manning, C. D., "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

4 Hochreiter, S. and Schmidhuber, J., "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

5 Memegenerator.net.(2018), "Meme generator online," 2018. [Online]. Available: http://memegenerator.net/