Image Captioning – IDS 576 Project Report

Sruthi Reddy Narapareddy, Ankit Bagusetty, Deepika Kolli

Abstract

Automatically describing the content of an image is one of the challenging problems in Deep Learning where a textual description must be generated. It requires methods from computer vision to understand the content of the image and a language model from natural language processing to convert the understanding of the image into words in the correct structure. For this project we have taken Flickr8k dataset which consists of images along with their captions and we implemented our neural network-based image caption generator in Pytorch. For this project we have five major components, first one is data pre-processing, second is Encoder part which is the implementation of Convolutional Neural Network model that extracts features from images, third is attention mechanism, fourth is Decoder part which is the implementation of LSTM model that translates the features and objects in the image to a natural sentence and finally using greedy Search we generated the captions for the images. We have chosen BLEU metric to evaluate the quality and the accuracy of the captions generated by the model. We were able to implement the components mentioned above and were able to train our network on Google Colaboratory. Our model was able to achieve a BLEU-4 score of 13.5.

1.Introduction

To generate descriptive captions automatically for images is a very interesting topic which is a mixture of image context understanding, feature extraction, and converting the visual representations into a textual description. There are several applications of image captioning in various fields. For example, like medicine, military, education, digital libraries, Social media platforms such as Facebook, Twitter where the task or location can be automatically captioned and can also be used in assisting visually impaired people. For this project, we have implemented Encoder-Decoder framework where we have two major parts, first part is image based model for which we employed transfer learning and implemented pretrained CNN model to extract the features of the image and the second part is the language based model for which we implemented LSTM based Recurrent Neural Network which translates the features and objects in the image to a natural sentence. The feature vector obtained from the CNN model is given as an input to the LSTM network. Model takes an image I as input and is trained to maximize the likelihood $p(W|I)$ of producing a target sequence of words $W = \{W_1, W_2, W_3 \ldots W_t\}$ where each word W_t comes from a given vocabulary. CNNs are better in representing the input image by embedding it to a fixed-length vector. So, we used CNN as an image encoder and then using this as an input to the RNN decoder that generates sentences. For evaluation of our model, we measured the model's performance on the Flickr8K dataset using the BLEU standard metric.

2. Related Work

There is extensive research work happening on image captioning with neural networks and attention mechanisms. Before starting this project, we have read few research papers which tried to accomplish image captioning task. One was Show, attend and tell: Neural image caption generator with visual attention [1].This paper proposed to use a combination of a deep Convolutional Neural Network and a Recurrent Neural Network to achieve this task by adding attention mechanism and interpret which parts of the image are most important for the model to determine the words. Another one is Deep Visual-Semantic Alignments for Generating Image Descriptions by Karpathy and Li generated captions for distinct locations with an image [2]. Few other literatures that we read while working on the project to improve our model architecture can be found in [3][4].

3. Model

In this section we explain about the model which consists of three main parts: encoding, attention and decoding. A pre-trained Convolutional Neural Network architecture as an encoder to extract and encode image features into a higher dimensional vector space. CNN represents images in a form of feature vectors where each feature vector is of fixed dimension D. Second is the LSTM-based Recurrent Neural Network as a decoder to convert encoded features to natural language descriptions. Attention mechanism allows decoder to see features from a focused region of the input image to improve the overall performance and finally use greedy search to generate a caption word by word. Each individual component will be discussed in detail below.

3.1. Convolutional Neural Network (Encoder)

We're using the CNN as a feature extractor that compresses the huge amount of extraction contained in the original image into a smaller representation. This CNN is often called the encoder because it encodes the content of the image into a smaller feature vector. Then we can process this feature vector and use it as an initial input to the following RNN. We considered ResNet-152 model which is pretrained model as our encoder and removed the final fully connected layer of ResNet and added a linear layer along with batch normalization layer.

3.2. Attention Mechanism

We have used Attention mechanism here to pay close attention to each part of the image and generate captions most relevant to the image. Attention gives weightage to the part of the image on which we want to focus more and assigns weights accordingly. Higher weights are given to the sections of the image where we want to focus on the most. There are several steps in this attention process flow.

First, the encoded feature vector which we get from CNN is passed to a Linear function to get different internal states i.e. h1, h2, h3.. hn. For instance, If we want to focus on the bird section of an image and if the bird features are present in h1 and h2 then h1 and h2 will have higher values compared to the rest states.

$h_t = f(x_t)$ f = Linear function , x_t = Feature vector obtained from Encoder CNN (1)

Next step is using the Decoder LSTM hidden states, we get the internal states corresponding to the hidden states of LSTM.

$h_{s'} = f(h_{t-1})$ $\quad h_{t-1} = \text{Previous Hidden state values obtained from Decoder LSTM}$ (2)

Using h_t and $h_{s'}$ we calculate the attention scores i.e. s1,s2,s3, .. sn. The model will learn to relevant encoder states by generating a high score for the states for which attention is to be paid while low score for the states which are to be ignored.

score = f(relu(h_t, $h_{s'}$)) f = Linear function (3)

Upon identifying the scores for images, we use a softmax activation function and calculate the probabilities for these scores i.e. the attention weights e1,e2,e3,..,en.

e = {e1,e2,e3,..,en} and $0 \leq e \leq 1$

$\alpha_{ts} = \frac{\exp(\text{score})}{\sum_{s'}^{S} \exp(\text{score})}$ Soft max Activation function is used (4)

Using these attention scores, we calculate the context vector which will be used by the decoder in order to predict the next word in the sequence.

$C_v = \sum_s \alpha_{ts} h_{s'}$ $\quad C_v = \text{Context Vector}$ (5)

We have used a Deeper attention technique here by using sigmoid gate over the previous hidden state values and calculating new attention weighted which will have even more higher weights for the part of image where we can paying more attention.

$\bar{h}_{s'} = f(h_{t-1})$ $\quad h_{t-1} = \text{Previous Hidden state values obtained from Decoder LSTM}$ (6)

$g_t = \frac{1}{1+ e^{-\bar{h}_{s'}}}$ g = Sigmoid function applied for deeper attention (7)

$\bar{C}_v = \sum_s g_t C_t$ $\quad \bar{C}_v = \text{New Context Vector}$ (8)

LSTM

Training Phase

During the Training phase, we have used Teacher enforcing technique by passing the embedded captions along with the attention weighted values and LSTM hidden state values.

x_t = cat($C_e, \bar{C}_v$) $\quad \bar{C}_e = \text{Captions embedding}$ (9)

Testing Phase

During the Testing phase, we have passed the predicted captions as the input in the next time stamp(t).

$x_t = \text{cat}(P_t, \bar{C}_v)$ $\quad P_t = \text{Predictions from LSTM at time stamp(t)}$ (10)

LSTM Equations

First equation is for Input Gate which tells us that what new information we're going to store in the cell state (that we will see below).Second is for the forget gate which tells the information to throw away from the cell state.Third one is for the output gate which is used to provide the activation to the final output of the LSTM block at time stamp 't'.

$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$ $\quad C^t = \text{cell state(memory) at time stamp(t)}$ (11)

$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$ $\quad \tilde{C}_t = \text{candidate for cell state at time stamp(t)}$ (12)

$h_t = o_t * \tanh(C^t)$ $\quad h_t = \text{hidden state values}$ (13)

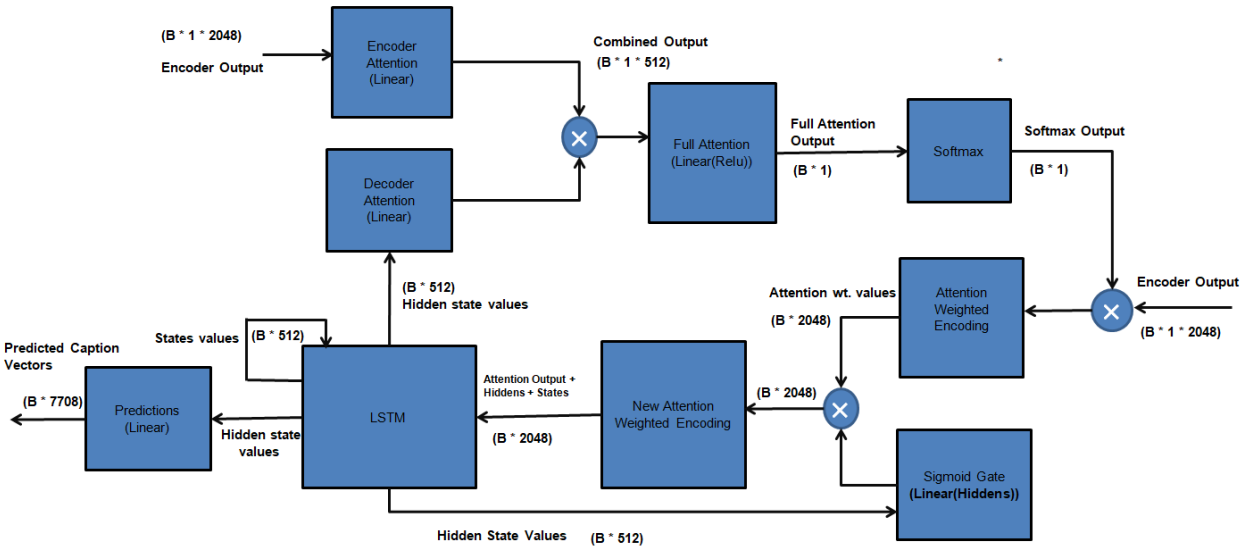
Final Predictions

The LSTM decoder hidden state values are passed to a Linear function in the end to obtain the predicted captions.

$P_t = f(h_t)$ $\quad P_t = \text{Prediction at time stamp(t)}$ (14)

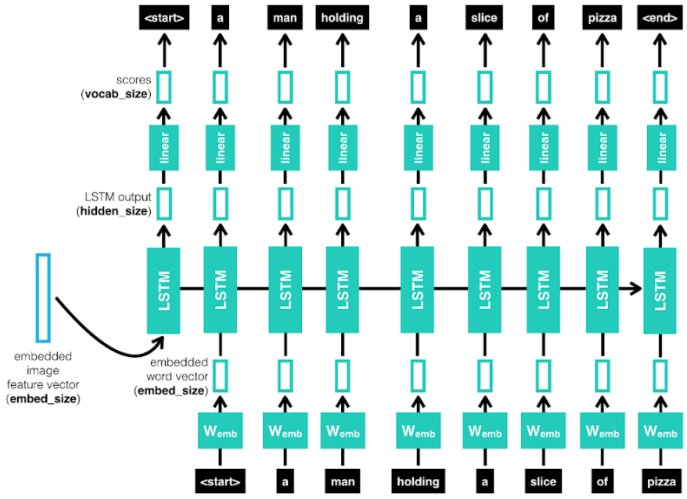
Attention Training Phase Process Flow

During the Training phase, we have passed the Attention weighted encoded values along with the embedded caption and previous hidden state values from Decoder LSTM. During the Testing phase, the predicted caption vector is passed as the next input to LSTM along with attention weighted encoded values and previous hidden state values from Decoder LSTM. Doing this we are making sure the Decoder LSTM generated meaningful captions in sequence.



3.3. LSTM-based Sentence Generator (Decoder)

We used LSTM that produces a caption by generating one word at every time step conditioned on a context vector, the previous hidden state and the previously generated words. At every step of the RNN, the probability distribution of the next word is output. Depending on the situation, a slightly naive approach would be to take the word with the highest probability at each step after extracting the output from the RNN. We have generated captions using greedy search algorithm now and planning to use Beam search algorithm in future work to get better results .



Three gates are being used which control whether to forget the current cell value (forget gate f), if it should read its input (input gate i) and whether to output the new cell value (output gate o). The definition of the gates and cell update and output are as follows[1] :

$i_t = \sigma(W_{ix}x_t + W_{im}m_t{-}1)$

$f_t = \sigma(W_{fx}x_t + W_{fm}m_t{-}1)$

$o_t = \sigma(W_{ox}x_t + W_{om}m_t{-}1)$

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

$m_t = o_t \; o \; c_t$

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

where represents the product with a gate value, and the various W matrices are trained parameters. Such multiplicative gates make it possible to train the LSTM robustly as these gates deal well with exploding and vanishing gradients [10]. The nonlinearities are sigmoid σ(·) and hyperbolic tangent h(·). The last equation mt is what is used to feed to a Softmax, which will produce a probability distribution pt over all words.

4. Experiments

In this section, we first describe the dataset used in our experiments as well as the experimental methodology followed by the detail results discussion. We performed an extensive set of experiments to assess the effectiveness of our model using several metrics and model architectures**.** We report the BLEU-1,2,3,4 scores for the model.

4.1. Datasets

We have used Flickr8k dataset for this project. Initially we planned to use Flickr30K dataset but due to limited storage and computational power we went ahead with Flick8k dataset. The dataset we have used consists of 8,000 images extracted from Flickr website. For each image, it provides five different sentence annotations. Each annotation is a sentence of around 10-20 words. We adopt the standard separation of training, validation and testing set provided by the dataset. There are 6,000 images for training, 1,000 images for validation and 1,000 images for testing.

4.2.Training Details

Dataset consists of images and captions, so we needed to pre-process the data, images to input for the CNN network and the text annotations for RNN network. For the encoder part of the model, CNN network uses Pre trained model, which is ResNet-152, and to use the ResNet model we needed to transform the images into correct format. Pre-trained ResNet-152 model expect that input images are normalized to within range [0, 1] and as 3-channel RGB images of shape (3 × H × W), where H and W should be 224. Therefore, data preprocessing involves loading and resizing image data into (N × 3 × 224 × 224) dimension and normalizing pixel value to be within range [0, 1] with mean value of [0.440, 0.417, 0.384] and std of [0.286, 0.280, 0.290]. We then pass CNN Output which has dimensions of size (2048 * 1 * 1) to RNN. For the decoder part, we first tokenized the words and converted to lower cases and then we had to convert all the words into indexes to get the word embeddings. We constructed a vocabulary of words which consists of frequent words in the training caption data along with which we added few special words like <start>, <end>, <pad>. We have also added the word <unk> where if the word is not present in the vocabulary it is represented as <unk>. In the LSTM part of model, each word in the caption is passed to the model one by one along with the corresponding image. Initially, the image is passed into the model along with the first word and it is mapped to generate the corresponding second word. This process is repeated until <end> is encountered it stops generating sentence and it marks end of the caption. Two input arrays are passed into the model, one for passing features of image and the other one for passing text data in encoded format. The output of the model is the encoded next word of the sequence. Later, when the model is used to generate descriptions for an image, the words generated in the previous sequence are concatenated and fed into the model to generate the next word.

We have trained our model with batch gradient descent using adaptive learning rate algorithms. For the Flickr8k dataset, we found that Adam gave better results. Loss function for model is calculated as, where I represent input image and S represents the generated caption. N is length of generated sentence and Pt is the probability of the occurrence of the word and St represents the predicted word at the time t.

$$L(I,S) = - \Sigma \log p_t(S_t) \; [t=1 \text{ to } N]$$

During the process of training we have tried to minimize this loss function. Since RNN output is to get the likelihood of words occurrence, we used Cross Entropy Loss as our loss function. To avoid troubles with LSTM, we have padded each caption to the maximum length of the caption within a batch. In order to avoid the extra computations, we have used pack padding sequence by passing the actual lengths of the captions in the batch. we used a fixed vocabulary size of 2933. **Hyperparameters** we used are 256 dimensions for the embeddings and 512 hidden dimensions for LSTM for the model without attention and used 512 embed size for the attention model. There are several other hyper parameters we adjusted in our experiments. Learning rate of 0.0001 which gave better results. We also adjusted the batch size in our experiments. The initial batch size we used was 10, the loss didn't converge. Later we experimented with different batch size values and finally batch size of 32 gave us better results.

4.3. Evaluation Metrics

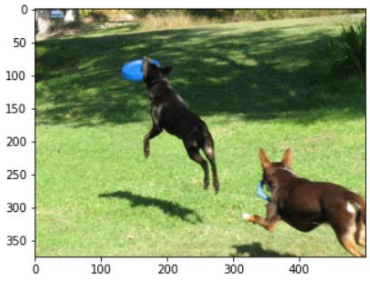
Though we were able to generate the captions, but we were not sure whether the captions generated were correct and are learned as expected and since the data is huge, we cannot possibly show the results for each image. Evaluating the performance of the caption generated is significantly important. Thus, to find the average accuracy on the whole dataset, BLEU-1, BLEU-2, BLEU-3, and BLEU-4 (Bilingual Evaluation Understudy) are metrics we chose to evaluate the quality of the caption generated. Sentences are compared based on modified n-gram precision method for generating BLEU score [5] (modified n-gram precision = maximum number of times n-gram occurs in reference/ total number of n-grams in hypothesis) . For BLEU calculation, each occurrence of an n-gram in ground truth can account for only one occurrence in predicted . Currently, we got a BLEU-4 score achieved with 20 epochs is 11.4 without attention and 13.5 with attention. BLEU scores on test data are mentioned in the below table.

Metrics	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Without Attention	**42.704**	**30.623**	**22.334**	**11.401**
With Attention	**51.456**	**39.984**	**27.392**	**13.514**

4.4.Results

Below are some of the successful captions generated on test data. One thing we observed is that dog images are predicted accurately which may be because in the train data there are many dog images. CNN provides information to the model about what a dog looks like.

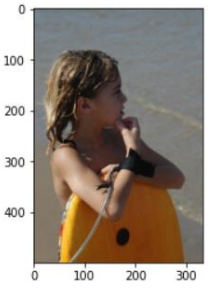
predicted caption: two dogs catching frisbee in grass



predicted caption: a dog is running through the grass

The generated captions were not all correct, some of which miss important information in the image and others have not identified the visual features. Some of the unsuccessful captions generated by the model are shown below:

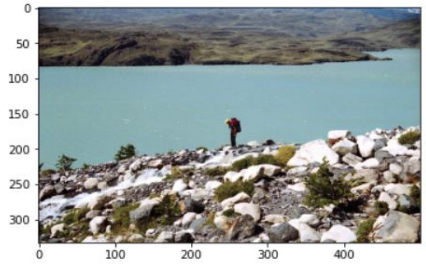
predicted caption: a girl standing on the beach holding <unk>



In the figure on left, RNN failed to generate the word so it might have given unknown as the word since training data must not be having the word boogie board in the vocabulary.

predicted caption: a woman in a blue sweatshirt taking a picture .

predicted caption: a man is sitting on a bench .



Some captions for the images describe the context completely different than the one in the image which might be due to not properly recognizing the image part and few other captions have no proper structure or meaning which might be due to failure in text generation.

5. Conclusion and Future Work

We were able to develop an image captioning model and extended the architecture to use attention mechanism and achieved a BLEU-4 score of 13.5. There can be many improvements in this project, and we would like to continue working on it. Firstly, instead on greedy search we should have implemented beam search which would have given more optimal captions and overall better results. Also, we have used

pre-trained ResNet-152 model by disabling the gradient. We could have fine-tuned it to get better results on this dataset and must have achieved a slightly higher BLEU4 score. Also, we could have tuned the hyperparameters more to get better results. Overall working on this project has helped us develop more interest in understanding the Deep Learning concepts and we would like to explore more on this project in future.

References

[1] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. CoRR, abs/1502.03044, 2015

[2] A. Karpathy and F. Li. Deep visual-semantic alignments for generating image descriptions. CoRR, abs/1412.2306, 2014.

[3] Raimonda Staniut¯ e and Dmitrij Šešok . A Systematic Literature Review on Image Captioning, 2019.

[4] Simao Herdade, Armin Kappeler, Kofi Boakye, Joao Soares. Image Captioning: Transforming Objects into Words, 2019.

[5] Blaine Rister, Dieterich Lawson, Image Captioning with Attention.

[6] Lakshminarasimhan Srinivasan1 , Dinesh Sreekanthan2 , Amutha A.L3. Image Captioning - A Deep Learning Approach, 2018.

[7] Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu - BLEU: a Method for Automatic Evaluation of Machine Translation Kishore ,IBM T. J. Watson Research Center Yorktown Heights, NY 10598, USA