Image Captioning Model

INDEX

1. Abstract 3
2. Introduction 3
3. Theory: 3

3.1 Encoder-Decoder architecture. 3

3.2 Attention 3

3.3 Transfer Learning. 3

4. **BLOCK DIAGRAM** **5**

5. Dataset and Training 7

6. Output 17

7. Conclusion 18

Abstract :

Inspired by recent work in machine translation and object detection, we introduce an attention based model that automatically learns to describe the content of images.We describe how we can train this model in a deterministic manner using standard backpropagation techniques.. We validate the use of attention with state-of-the-art performance on Flickr30k dataset.

Introduction :

This project is an implementation of Show , Attend and Tell research paper. Automatically generating captions of an image is a task very close to the heart of scene understanding — one of the primary goals of computer vision. Not only must caption generation models be powerful enough to solve the computer vision challenges of determining which objects are in an image, but they must also be capable of capturing and expressing their relationships in a natural language. For this reason, caption generation has long been viewed asa difficult problem.

Theory:

Encoder-Decoder architecture.

Typically, a model that generates sequences will use an Encoder to encode the input into a fixed form and a Decoder to decode it, word by word, into a sequence.

Attention

The use of Attention networks is widespread in deep learning, and with good reason. This is a way for a model to choose only those parts of the encoding that it thinks is relevant to the task at hand. The same mechanism is employed here can be used in any model where the Encoder's output has multiple points in space or time. In image captioning, we consider some pixels more important than others.

Transfer Learning.

This is when you borrow from an existing model by using parts of it in a new model. This is almost always better than training a new model from scratch (i.e., knowing nothing). As you will see, you can always fine-tune this second-hand knowledge to the specific task at hand. Using pretrained word embeddings is a dumb but valid example. For our image captioning problem, we will use a pretrained Encoder, and then fine-tune it as needed.

We have used pretrained resnet 101 as encoder available in pytorch.

The Decoder's job is to **look at the encoded image and generate a caption word by word**.

Since it's generating a sequence, it would need to be a Recurrent Neural Network (RNN). We use an LSTM.

The Attention network **computes these weights**.

Intuitively, how would you estimate the importance of a certain part of an image? You would need to be aware of the sequence you have generated *so far*, so you can look at the image and decide what needs describing next.

The given research paper mentions two methods of attention , namely

Soft Attention

Hard Attention.

We have used soft attention following Luong model.(reference research paper : https://arxiv.org/pdf/1508.04025.pdf)

As mentioned in this research paper , there are 3 attention based models ,

Here, score is referred as a *content-based* function for which we consider three different alternatives:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

We have used general attention .In the above formulae , ht refers hidden state,

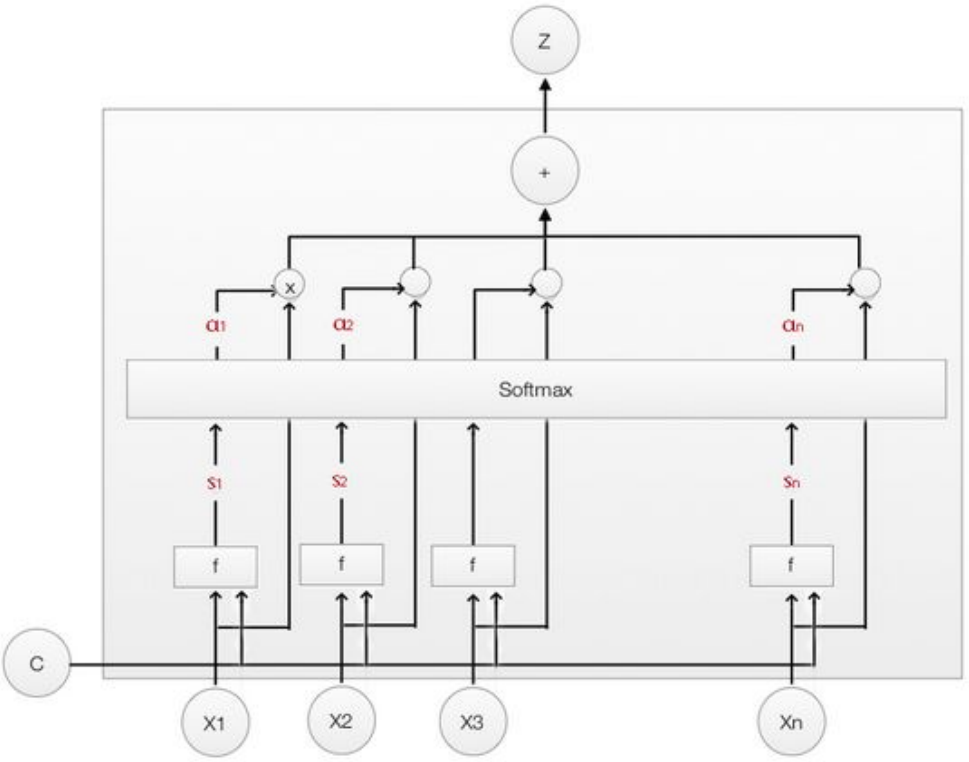
hs refers image encoded context vector obtained from CNN.

$$\begin{aligned} \boldsymbol{a}_t(s) &= \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \end{aligned}$$

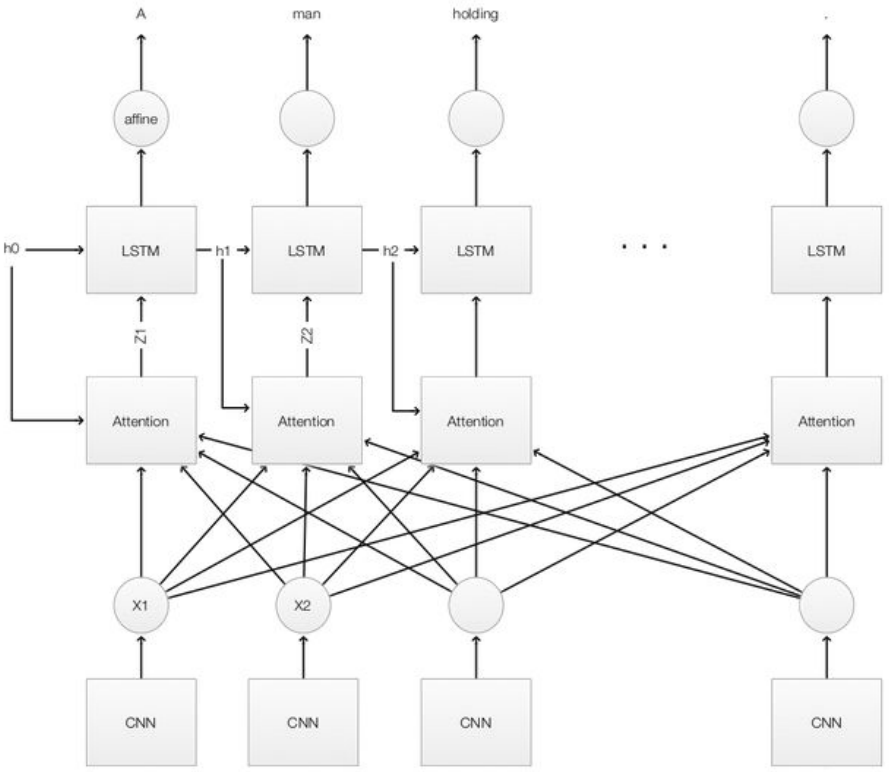
So , once attention weights are calculated , we simply multiply these weights with encoded context vector , to obtain attention_weighted_encoding, this goes as input to the LSTM , along with word embedding .

Following are the block diagrams for better understanding of attention and LSTM architecture.

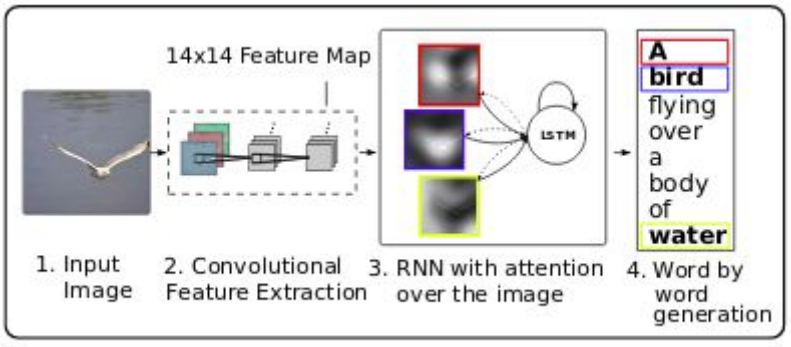
BLOCK DIAGRAM :

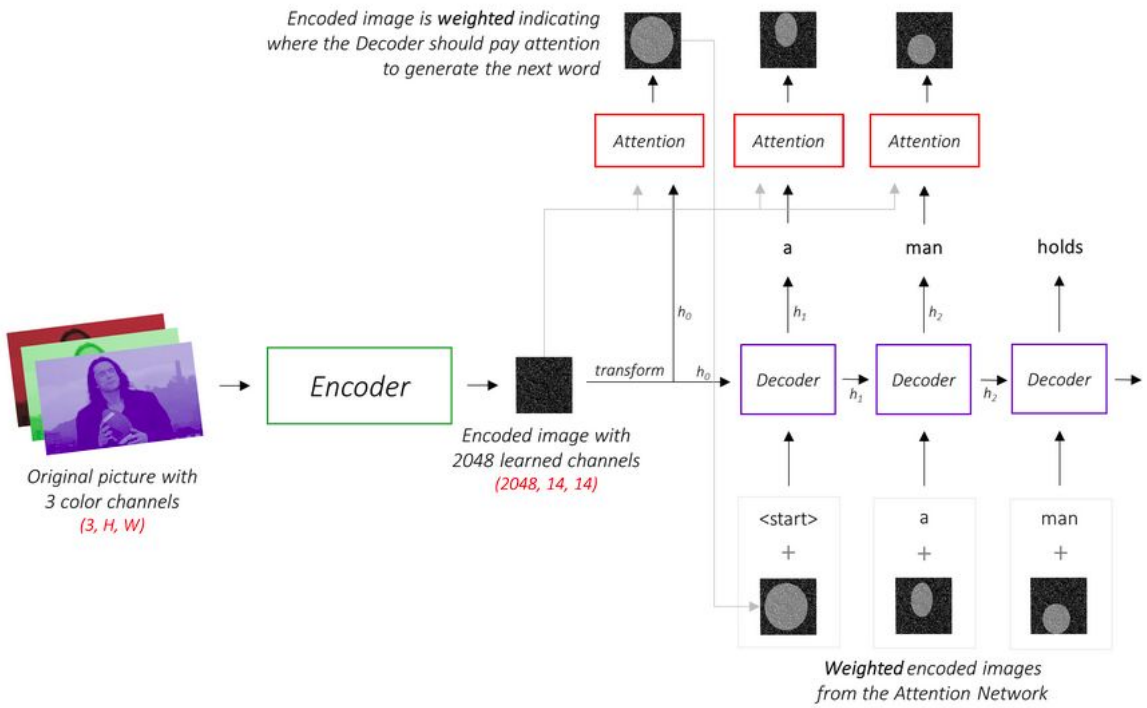


The following is the complete flow of the LSTM model using attentions.



Complete block diagram:





In training the deterministic version of our model form of doubly stochastic regularization is used as mentioned in paper.This can be interpreted as encouraging the model to pay equal attention to every part of the image over the course of generation. In addition, the soft attention model predicts a gating scalar β from previous hidden state h_{t-1} at each time step t, such that,$\varphi(\{a_i\},\{\alpha_i\}) = \beta \sum_i \alpha_i a_i$, where$\beta_t = \sigma(f_\beta(h_{t-1}))$. We notice our attention weights put more emphasis on the objects in the images by including the scalar β.

The loss function used along with regularization constant lambda is,

Concretely, the model is trained end-to-end by minimizing the following penalized negative log-likelihood:

$$L_d = -\log(P(\mathbf{y}|\mathbf{x})) + \lambda \sum_{i}^{L}(1 - \sum_{t}^{C}\alpha_{ti})^2 \qquad (14)$$

Dataset and Training :

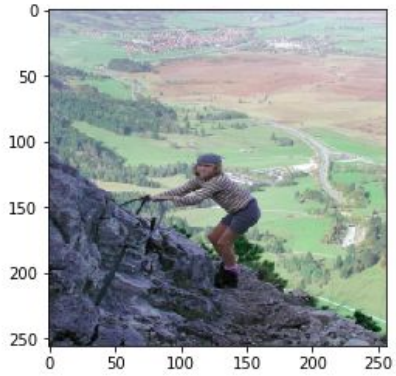
We have used Flickr30k dataset , with train-test-validation splits given by Andrej Karpathy.
We trained model using gpu in pytorch in KAGGLE . Every 6 hrs our code used to generate checkpoint file which contained all weights,parameters learnt.

We have used BLEU score as metric for measuring accuracy .
Because of lack of proper resources , we have trained model for 9000 images only from dataset.We did check the bleu score on validation set , and we

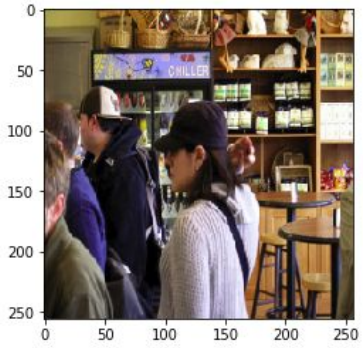
observed that it reached a peak of 9 , and later started degrading (probably due to overfitting).

Output :

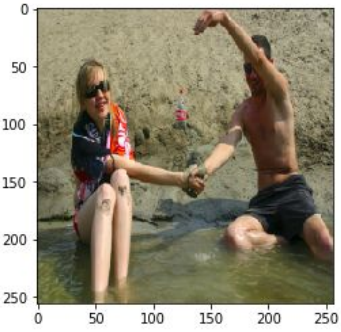
Some of outputs on test images,



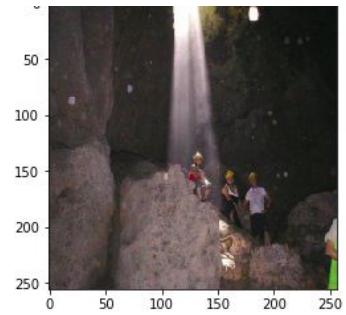
```
['a', 'man', 'is', 'standing', 'on', 'top', 'of', 'a', 'rock', 'at', 'the', 'view']
```



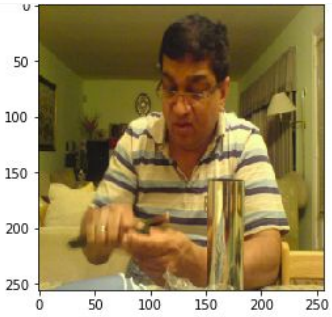
```
['a', 'group', 'of', 'people', 'are', 'standing', 'in', 'line', 'at', 'a', 'cafe', 'table']
```



```
['a', 'young', 'girl', 'in', 'a', 'red', 'sequined', 'on', 'water', 'looking', 'at', 'the', 'beach']
```




```
['two', 'people', 'climbing', 'a', 'snowy', 'mountain']
```




```
['a', 'man', 'in', 'a', 'white', 'shirt', 'and', 'glasses', 'is', 'cutting', 'a', 'hamburger', 'in', 'half']
```

Conclusion:

Thus we implemented an attention based image captioning model that automatically learns to describe the content of images.