IMAGE CAPTION GENERATOR

Author: Akash Yadav

Reg.: 200968184

Problem Statement

Training computers to be able to generate descriptive captions for images automatically is currently a scorching topic in Computer Vision and Machine Learning. This task combines image scene understanding, feature extraction, and translation of visual representations into natural languages. This project shows great promise, such as building assistive technologies for visually impaired people. There are a series of relevant research papers attempting to accomplish this task in the last decades. Still, they face various problems, such as grammar problems, cognitive absurdity, and content irrelevance. However, with the unparalleled advancement in Neural Networks, some groups started exploring Convolutional Neural networks and recurrent neural networks to accomplish this task and observed promising results [2]. The most recent and most popular ones include

Show and Tell: A Neural Image Caption Generator and

Show, attend and tell: Neural image caption generator with visual attention.

While both papers propose to use a combination of a deep Convolutional Neural Network and a Recurrent Neural Network to achieve this task, the second paper is built upon the first one by adding an attention mechanism.

Meta Data

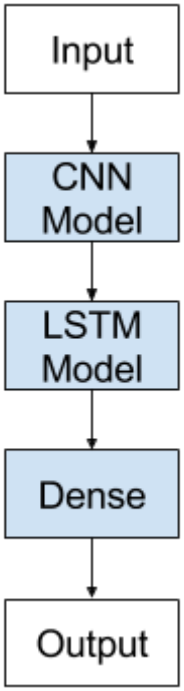
Flickr8k Dataset: Contains a total of 8092 images in JPEG format with different shapes and sizes. Of which 6000 are used for training, 1000 for test and 1000 for development.

Flickr8k text: Contains text files describing train set and test set. Flickr8k.token.txt contains five captions for each image, i.e., 40460 captions.

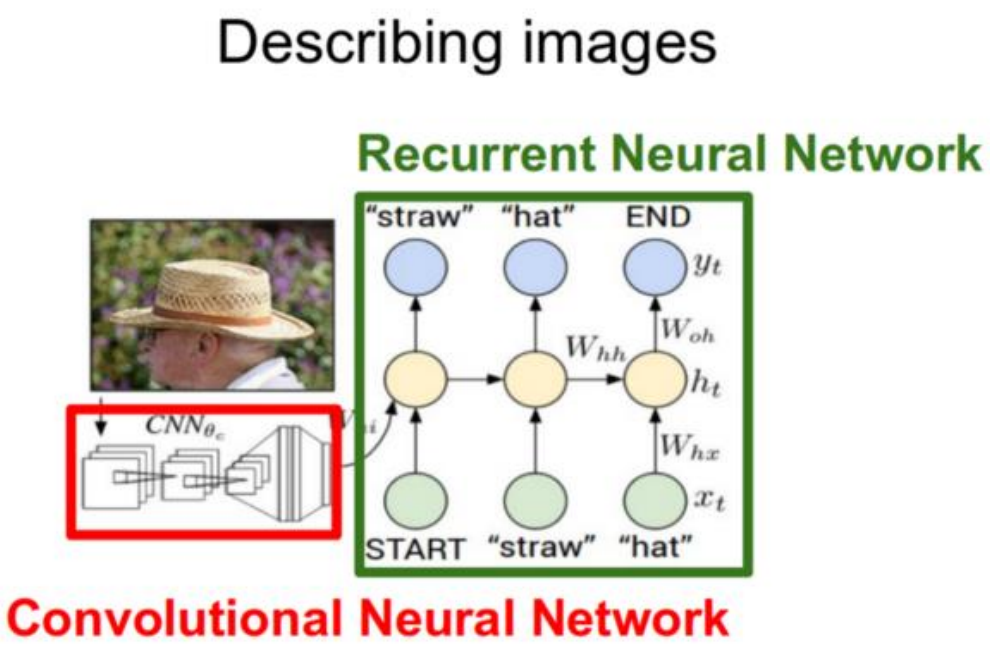
PROJECT OBJECTIVES

I aim to perform image-to-sentence generation, also known as "Image Captioning," which will bridge the gap between vision and natural language. We can utilize NLP technologies to understand the world in images. The current dataset that I am working on is Flickr8K. To achieve the goal, I will be studying and applying existing pre-trained CNN Models, i.e., VGG16, ResNet50, and Inception V3. Further, I'll be attaching the RNN using the LSTM language model to it. In the end, I'll be evaluating the results and comparing each one of them.

The general architecture of the CNN-LSTM Model is as follows:



CNN-LSTMs are generally used when their inputs have spatial structure, such as the 2D structure or pixels in an image or the 1D structure of words in a sentence, paragraph, or document, and have a temporal structure in their input, such as the order of images in a video or words in the text or require the generation of output with a temporal structure such as words in a textual description.



A pre-trained CNN extracts the features from our input image. The feature vector is linearly transformed to have the same dimension as the input dimension of the LSTM network. This network is trained as a language model on our feature vector.

For training our LSTM model, we predefine our label and target text. For example, if the caption is “An old man is wearing a hat.”, our label and target would be as follows –

Label — [<start> ,An, old, man, is, wearing, a , hat .]

Target — [An old man is wearing a hat .,<End>]

This is done so that our model understands the start and end of our labelled sequence.

Evaluation Metrics

BLEU (**B**i**L**ingual **E**valuation **U**nderstudy) is a metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high-quality reference translations. A value of 0 means that the machine-translated output has no overlap with the reference translation (low quality). In comparison, a value of 1 means there is perfect overlap with the reference translations (high quality).

It has been shown that BLEU scores correlate well with the human judgment of translation quality. Note that even human translators do not achieve a perfect score of 1.0.

Interpretation

Trying to compare BLEU scores across different corpora and languages is strongly discouraged. Even comparing BLEU scores for the same corpus but with different numbers of reference translations can be highly misleading.

However, as a rough guideline, the following interpretation of BLEU scores (expressed as percentages rather than decimals) might be helpful.

BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations

The mathematical details

Mathematically, the BLEU score is defined as:

$$\text{BLEU} = \underbrace{\min\left(1, \exp\left(1 - \frac{\text{reference-length}}{\text{output-length}}\right)\right)}_{\text{brevity penalty}} \underbrace{\left(\prod_{i=1}^{4} precision_i\right)^{1/4}}_{\text{n-gram overlap}}$$

$$precision_i = \frac{\sum_{\text{snt} \in \text{Cand-Corpus}} \sum_{i \in \text{snt}} \min(m^i_{cand}, m^i_{ref})}{w^i_t = \sum_{\text{snt'} \in \text{Cand-Corpus}} \sum_{i' \in \text{snt'}} m^{i'}_{cand}}$$

where

- $\mathbf{m^i_{cand}}$ is the count of i-gram in candidate matching the reference translation
- $\mathbf{m^i_{ref}}$ is the count of i-gram in the reference translation
- $\mathbf{w^i_t}$ is the total number of i-grams in candidate translation

The formula consists of two parts: the brevity penalty and the n-gram overlap.

- **Brevity Penalty**
  The brevity penalty penalizes generated translations that are too short compared to the closest reference length with an exponential decay. The brevity penalty compensates for the fact that the BLEU score has no recall term.
- **N-Gram Overlap**
  The n-gram overlap counts how many unigrams, bigrams, trigrams, and four-grams (i=1,...,4) match their n-gram counterpart in the reference translations. This term acts as a precision metric. Unigrams account for *adequacy* while longer n-grams account for *fluency* of the translation. To avoid overcounting, the n-gram counts are clipped to the maximal n-gram count occurring in the reference(m^n_{ref}).

MODELS IMPLEMENTED

ResNet50 was used as an image encoding to encode the images, which were input into the model. Keras embedding layer was used to generate word embeddings on the captions encoded earlier. The embeddings were then passed into the LSTM, after which the image and text features were combined and sent to a decoder network to generate the next word.

Convolutional Neural Network

In deep learning, a **convolutional neural network** (**CNN**, or **ConvNet**) is a class of artificial neural network (**ANN**), most applied to analyze visual imagery.

PROS:

1. Learning accurate patterns and insights from the provided data. (Depends on how well structured, clean or feature-engineered data is)
2. One can tune the network to achieve better and more accurate results.
3. Can provide better outcomes than other machine learning algorithms if tuned better and feeded a good amount of data.

CONS:

1. CNN do not encode the position and orientation of the object.
2. Lack of ability to be spatially invariant to the input data.
3. Lots of training data is required and is computationally expensive.

RNN

Recurrent neural networks are very famous deep learning networks that are applied to sequence data: time series forecasting, speech recognition, sentiment classification, machine

translation, Named Entity Recognition, etc.
The use of feedforward neural networks on sequence data raises two major problems:

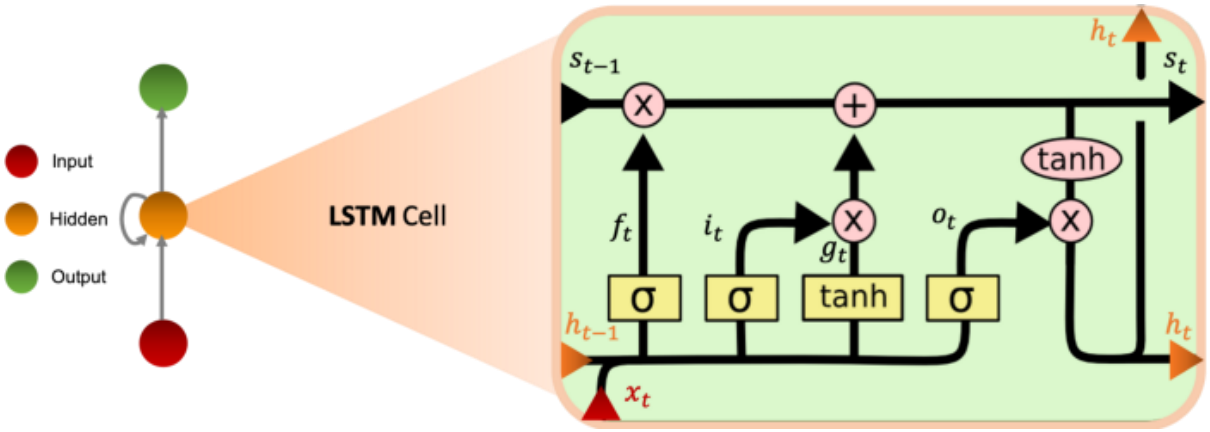
- Input & outputs can have different lengths in different examples
- MLPs do not share features learned across different positions of the data sample

LSTM

LSTMs (Long Short Term Memory) were also introduced to overcome the problem of short memory, they have 4 times more memory than Vanilla RNNs. This model uses the notion of gates and has three :

- Input gate *i*: controls the flow of incoming information.
- Forget gate *f*: Controls the amount of information from the previous memory state.
- Output gate *o*: controls the flow of outgoing information

- The graph below shows the operation of the LSTM cell:



- When the input and output doors are closed, activation is blocked in the memory cell.
- **Equations**
  We define the equations in the LSTM cell as follows:
  - **Input gate**: $i_t = \phi(W^{ix}x_t + W^{ih}h_{t-1})$
  - **Forget gate**: $f_t = \phi(W^{fx}x_t + W^{fh}h_{t-1})$
  - **Output gate**: $o_t = \phi(W^{ox}x_t + W^{oh}h_{t-1})$
  - **Input node**: $g_t = tanh(W^{gx}x_t + W^{gh}h_{t-1})$
  - **Hidden node**: $s_t = s_{t-1}.f_t + g_t.i_t$
  - **Memomy state**: $h_t = tanh(s_t).o_t$
- **Pros & Cons**
  We can summarize the advantages and disadvantages of LSTM cells in 4 main points:

  - **Advantages**

+They are able to model long-term sequence dependencies.
+They are more robust to the problem of short memory than 'Vanilla' RNNs since the definition of the internal memory is changed from:

$$h_t = f(Ux_t + Wh_{t-1}) \rightarrow h_t = tanh(s_{t-1}.f_t + g_t.i_t).o_t$$

- **Disadvantages**

+They increase the computing complexity compared to the RNN with the introduction of more parameters to learn.
+The memory required is higher than the one of 'Vanilla' RNNs due to the presence of several memory cells.

CONCLUSION

The caption for the given input image is produced using deep learning model. Initially the images were pre-processed and the text in order to train a deep learning model. And designed and trained a deep learning image caption generation model. Then, evaluate the train caption generation model, which produced captions for new images given as input apart from the loaded images.

The model was able to attain a validation accuracy of 80% during training and was evaluated using BLEU score. Two types of search, namely Greedy Search and Beam search were used, out of which Beam Search turned out to improve the model(Average BLEU score of 0.6218 with k=3) while Greedy Search has an average BLEU score of 0.5901.

It can be noted that the accuracy of the model is not very high which can be attributed to the smaller dataset containing only 8000 images. For a more accurate prediction, the model should be trained on larger datasets like Flick30K(which contains 30,000 images) or MSCOCO(which contains 180,000 images). The model also cannot predict the words which are not present in the vocabulary.