Indian Institute of Technology, Palakkad

IIT PALAKKAD

CS4806 : Theory And Practice Of Data Science Project

Image Description

"*Under the guidance of Dr. Mrinal Kanti Das*"

April 6, 2018

Submitted by :-

Adithya O V
Jayaprakash A
Polu Varshith

Contents

Chapter 1

Introduction

Image descriptions provide textual information about non-text content that appears on your website, allowing it to be presented auditorily, as visual text, or in any other form that is best for the user. Image descriptions are plain text (Generally, text you can copy and paste) descriptions of images, gifs, videos, and other media.

1.1 Aim

This project will involve developing a model that generates suitable captions for images. This will help in analyzing image and converting the textual content to other useful forms.

1.2 Motivation

Why do we need image descriptions ?

1. Screen readers can't interpret images and must rely on text to read out loud the information on the page to people who are blind, as well as others who use them.
2. When content must be enlarged to be seen, text often scales better than images, which can become pixelated and/or can easily take over the entire screen, causing users to have to scroll the image vertically and/or horizontally.
3. Some people with cognitive disabilities can understand text better than they are able to interpret images.
4. People who use voice recognition software need text alternatives for controls that are displayed as images so they know what to speak to activate the control.

This field has large practical and social applications.

1.3 Contribution of team members

All the members of the team have put their maximum efforts. Every one helped each other throughout the project, both in debugging of the code as well conceptual part. All the members have put the same amount of man hours for this project. So, one cannot pinpoint of

less contribution of any particular member of the team.

Every one has helped in writing the each and every part of code and also in writing of report and creating the presentation.

Adithya OV: Preprocessing of images i.e, encoding images into useful form. Providing with ideas and suggesting libraries that can ease our task.

Jayaprakash A: Coding the LSTMs. Creating image models. Suggesting alternatives to any failed approach. Suggesting tricks that could fasten the computation.

Varshith Polu: Generating captions from the model. Literature survey. Clarifying the team about approach of building model from the paper. Motivating others to actively participate in the project.

1.4 Organization of the report

The rest of the thesis is organized in the following manner:

Chapter 2 presents a detailed explanation of the problem we are going to solve and the challanges we faced.

Chapter 3 shows the current state of art attempts at solving this problem. Available datasets and models created.

Finally, in **Chapter 4 and 5**, we provide a brief summary on the approach we made to solve the project.

Chapter 6 describes our problems and difficulties faced in training the model.

Chapter 7 illustrates the immense capacity of our model. This chapter deals with the results obtained after testing and some captions generated to images by our model.

Chapter 2

Problem Statement

2.1 What are we intending to solve

Humans have been captioning images involuntary since decades and now in the age of social media where every image have a caption over various social platforms. Psychologically those things are affected by events and scenarios running in mind or influenced by nearby activities and emotion. Sometimes those are far-far away from real context. Describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing.

2.2 Sub problems involved in this problem

Image to Vector conversion:
we need a vector representation of an image . Converting images into some representation that preserves the content of the image is a hard problem. These features strive to be invariant (they do not change) under certain constraints. Sadly, these techniques require large amounts of data, computation, and infrastructure to use.
Caption (sentence) encoding:

This can be solved by the celebrated recurrent neural networks .

Other problem would be to evaluate loss:

Training require loss but how to calculate loss for any particular caption for an image .we are going to use some kind of standard ranking algorithm which we describe in later stages.

2.3 Application

This has some very good application for instance by helping visually impaired people better understand the content of images.

Chapter 3

Literature Survey

3.1 Various attempts at image captioning

There has been several attempts in this field. Some of them have been described here. In 2014, research scientists on the Google Brain team trained a machine learning system to automatically produce captions that accurately describe images. Further development of that system led to its success in the Microsoft COCO 2015 image captioning challenge, a competition to compare the best algorithms for computing accurate image captions, where it tied for first place. [1]

Picture this: Microsoft Research project can interpret, caption photos. Microsoft researchers are at the forefront of developing technology that can automatically identify the objects in a picture, interpret what is going on and write an accurate caption explaining it. For decades, researchers have been tantalized by the possibility of creating systems that could accurately interpret and caption photos. But until a few years ago, most of the systems being developed just weren't getting it right, said Margaret Mitchell, a researcher in Microsoft Research's natural language processing group who also is working on the technology. [2]

That changed when researchers hit upon the idea of using neural networks, which are computing elements that are modeled loosely after the human brain, to connect vision to language. With that technology, the systems began to get it right more often, and error rates have been decreasing ever since.

3.2 Available datasets

The competitors are all using a dataset of images, called Microsoft COCO, which was developed by researchers from Microsoft and other research institutions. This dataset with the goal of advancing the state-of-the-art in object recognition by placing the question of object recognition in the context of the broader question of scene understanding. This is achieved by gathering images of complex everyday scenes containing common objects in their natural context.

Objects are labeled using per-instance segmentations to aid in precise object localization. This dataset contains photos of 91 objects types that would be easily recognizable by a 4 year old. With a total of 2.5 million labeled instances in 328k images, the creation of

our dataset drew upon extensive crowd worker involvement via novel user interfaces for category detection, instance spotting and instance segmentation.

3.3 Current models

Project **Picture this**'s algorithm is trained to automatically write a caption using several steps. First, it predicts the words that are likely to appear in a caption, using what's called a convolutional neural network to recognize what's in the image. The convolutional neural network is trained with many examples of images and captions, and automatically learns features such as color patches, shapes and other features. That's much like the way the human brain identifies objects. Next, it uses a language model to take that set of words and create coherent possible captions.

Chapter 4

Algorithm

4.1 Image representation

Usually we represent image as an array of values which we interpret as pixel values but this problem requires a different representation of the image. [3]

The idea is to take an image I as input , and is trained to maximize the likelihood p($S \mid I$) of producing a target sequence of words S $= S_1, S_2, \ldots$, where each word S_t comes from a given dictionary, that describes the image adequately. The main inspiration of our work comes from recent advances in machine translation, where the task is to transform a sentence S written in a source language, into its translation T in the target language, by maximizing Pr($T \mid S$). Many famous techniques can be done in a much simpler way using Recurrent Neural Networks (RNNs) and still reach 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 in used as the initial hidden state of a "decoder" RNN that generates the target sentence.

But since we are dealing with images rather then sentences we would want a representation generated like how real images are modeled in animals. Not sequentially the way in which language is modeled. This suggests us to use convolutional neural networks. CNNs can produce a rich representation of the input image by embedding it to a fixed-length vector, such that this representation can be used for a variety of vision tasks.

Hence, it is natural to use a CNN as an image "encoder", by first pre-training it for an image classification task and using the last hidden layer as an input to the RNN decoder that generates sentences.

4.2 Generating Sequence

Recent advances in statistical machine translation have shown that, given a powerful sequence model, it is possible to achieve state-of-the-art results by directly maximizing the probability of the correct translation given an input sentence in an "end-to-end" fashion – both for training and inference. These models make use of a recurrent neural network which encodes the variable length input into a fixed dimensional vector, and uses this representation to "decode" it to the de- sired output sentence. Thus, it is natural to use the same approach where, given an image (instead of an input sentence in the source language), one

applies the same principle of "translating" it into its description.

$$\theta^* = \arg\max_{\theta} \sum_{(I,S)} \log \Pr(S \mid I; \theta) \tag{4.1}$$

where θ are the parameters of our model, I is an image, and S its correct transcription. Since S represents any sentence, its length is unbounded. Thus, we can use chain rule to model the joint probability over $S_1, S_2, \ldots, S_N$, where N is the length of this particular example as

$$\log \Pr(S \mid I) = \sum_{t=0}^{N} \log \Pr(S_t \mid I, S_0, S_1, \ldots, S_{t-1}) \tag{4.2}$$

It is natural to model $\Pr(S_t \mid S_0, S_1, \ldots, S_{t-1})$ with a Recurrent Neural Network (RNN), where the variable number of words we condition upon up to t-1 is expressed by a fixed length hidden state or memory h(t). This memory is updated after seeing a new input x_t by using a non-linear function f :

$$h_{t+1} = f(h_t, x_t) \tag{4.3}$$

f completes the conditional probability. For f we use a Long-Short Term Memory (LSTM) net, which has shown state-of-the art performance on sequence tasks such as translation.

Chapter 5

Implementation Details

5.1 Introduction

This neural system for image captioning is roughly based on the paper "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention" by Xu et al. (ICML2015). The input is an image, and the output is a sentence describing the content of the image. It uses a convolutional neural network to extract visual features from the image, and uses a LSTM recurrent neural network to decode these features into a sentence. A soft attention mechanism is incorporated to improve the quality of the caption. This project is implemented using the Tensorflow library, and allows end-to-end training of both CNN and RNN parts.

5.2 Prerequisites

The following are the various libraries used in the implementation. Let us start with a brief introduction of these libraries.

5.2.1 Tensorflow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

5.2.2 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects etc.

5.2.3 Natural Language Toolkit (NLTK)

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

5.3 Dataset Description

Coco dataset would be used for training the model. These datasets were made available as a part of COCO 2015 Captioning Challenge. This dataset has 12 super categories of images ranging from outdoor, food, indoor, appliance, sports, person, animal to kitchen. Each super category has sub categories and a total of around 84 categories of images.

5.3.1 CNN Architecture

It starts out with 224 * 224 * 3 tensor, which are the dimensions of the image. The layers have the following parameters.
Layer 1: `filters = 64, kernel_size = (7, 7), strides = (2, 2)`

1. Layer 1: The depth of the image increases from 3 to 64
2. Layer 2: The depth of the image increases from 64 to 128 as filters increase to 128
3. Layer 3: The depth of the image increases from 128 to 256
4. Layer 4: The depth of the image increases from 256 to 512

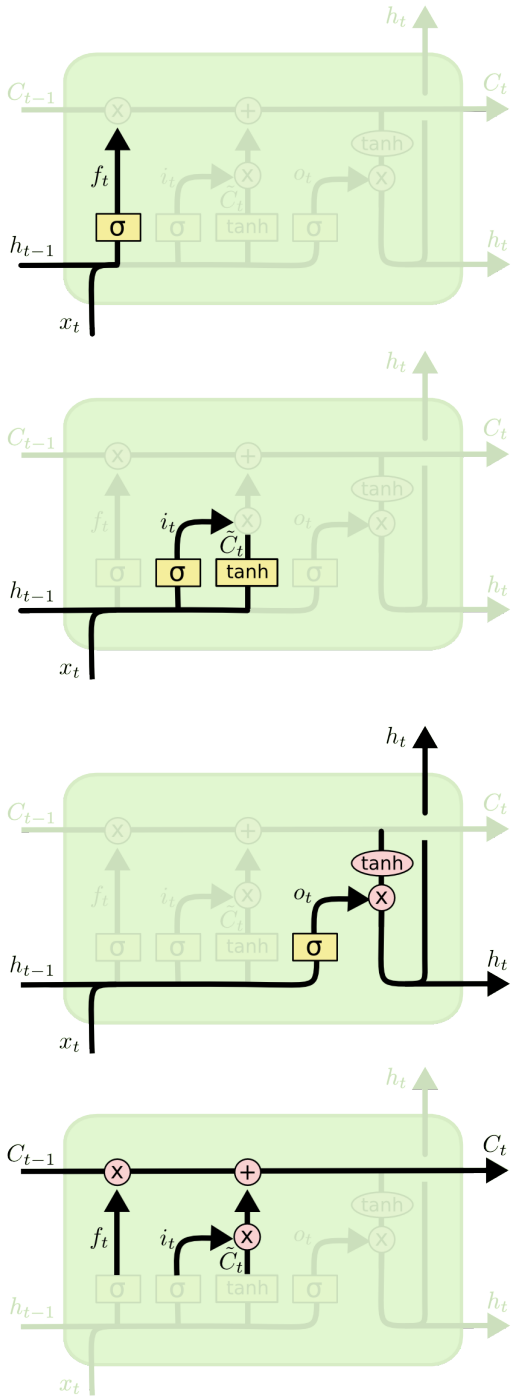
5.4 RNN

5.4.1 Basics

RNN was implemented using Tensorflow. [4]. The images below contain the equations that govern RNN.

5.4.2 RNN Architecture

The basic cells in this recurrent neural network in lstm . we have described about lstm in previous sections.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 5.2: Equations governing the CNN

The architecture that I am following is, we initially have some fully connected layers to

initialize the lstm network

$$\begin{aligned} temp1 &= self.nn.dense(context\_mean, units = 512) \\ temp1 &= self.nn.dropout(temp1) \\ memory &= self.nn.dense(temp1, units = 512) \\ temp2 &= self.nn.dense(context\_mean, units = 512) \\ temp2 &= self.nn.dropout(temp2) \\ output &= self.nn.dense(temp2, units = 512) \end{aligned}$$

We have a layer of lstm cells. Layer 1: 512 cells. Then we have a dropout wrapper layer with parameters:

$$\begin{aligned} input\_keep\_prob &= 1.0 - lstm\_drop\_rate, \\ output\_keep\_prob &= 1.0 - lstm\_drop\_rate, \\ state\_keep\_prob &= 1.0 - lstm\_drop\_rate) \end{aligned}$$

where lstm_drop_rate = 0.3
Our word vector doesn't have 512 dimension so we have to compress it to the dimension of our embedding, for that we use logits.

$$\begin{aligned} temp &= self.nn.dense(expanded\_output, units = 1024) \\ temp &= self.nn.dropout(temp) \\ logits &= self.nn.dense(temp, units = config.vocabulary\_size) \end{aligned}$$

5.4.3 Training

We have a matrix of embed-dings for words . with our lstm model we predict a vector for a word. The word closest to that vector is the word that we are going to give as output in the sequence.

Then we find cross entropy loss of the sequence with the original caption which was given.

Chapter 6

Learnings

6.1 Computational difficulty of earlier model

The training of the model was computationally difficult. The model had 1 lakh images to be trained. Training CNN, encoding images was alone time consuming. In addition, we also had to train RNN. On a typical laptop without GPU it would take around 2-3 weeks which is quite impossible.

The main problems are computational complexity. Also size of training image dataset was a problem too. Just reducing size of dataset was not sufficient as we need images of various types and categories for proper and unbiased training of the model.

This lead us to think off a new approach to problem. We later decided to use a pre-trained model for CNN.

6.2 Model 2

The approach to the problem was same as compared to first one. This model ..... . This model was built using some advanced APIs like keras and h5py and bazel.

6.2.1 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. It supports both convolutional networks and recurrent networks, as well as combinations of the two.

6.2.2 H5py

The h5py package is a Pythonic interface to the HDF5 binary data format. It lets us store huge amounts of numerical data, and easily manipulate that data from NumPy. For example, you can slice into multi-terabyte datasets stored on disk, as if they were real NumPy arrays. Thousands of datasets can be stored in a single file, categorized and tagged however you want. H5py uses straightforward NumPy and Python metaphors, like dictionary and NumPy array syntax.

It is used to save the trained model so that it can be retrieved faster compared to other binary data files like pickle.

6.2.3 Bazel

Bazel only rebuilds what is necessary. With advanced local and distributed caching, optimized dependency analysis and parallel execution, you get fast and incremental builds.

6.2.4 Pretrained CNN model

VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVR (ImageNet) competition in 2014. To this day is it still considered to be an excellent vision model, although it has been somewhat outperformed by more revent advances such as Inception and ResNet.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 6.1: VGG Architectures

6.2.5 Limitations

There are some limitations using a pre-trained model for generating image embeddings which involve the datasets. The pretrained model would have been generated by training over images which are too general and mainly specified for classification. But we need a model which is trained over the images which are similar to the ones used in tasks like ours. but due to computational problems we are limited to use the pre-trained VGG model.

6.3 Model 2

6.3.1 Introduction

This model is very much similar to the prevoius one differing only in the CNN embedding that we use .

6.3.2 Flickr 8k Dataset

The Flickr 8K dataset includes images obtained from the Flickr website. University of Illinois at Urbana, Champaign has the sole link of this dataset. The images do not contain any famous person or place so that the entire image can be learnt based on all the different objects in the image.

Image search is based on associating the query text with the tags of the image that help to identify the image. Identification of images then becomes multi-label classification problem of associating images with individual words or tags. It is a much harder problem to automatically associate images with complete and novel descriptions of images such as captions.

Multiple captions for each image are taken because there is a great amount of variance that is possible in the captions that can be written to describe a single image. This also helps satisfy the dynamic nature of images. There are multiple objects in the image but in a caption usually the main subject and either one or two of the secondary subjects are included in the caption.

6.3.3 Model Architecture

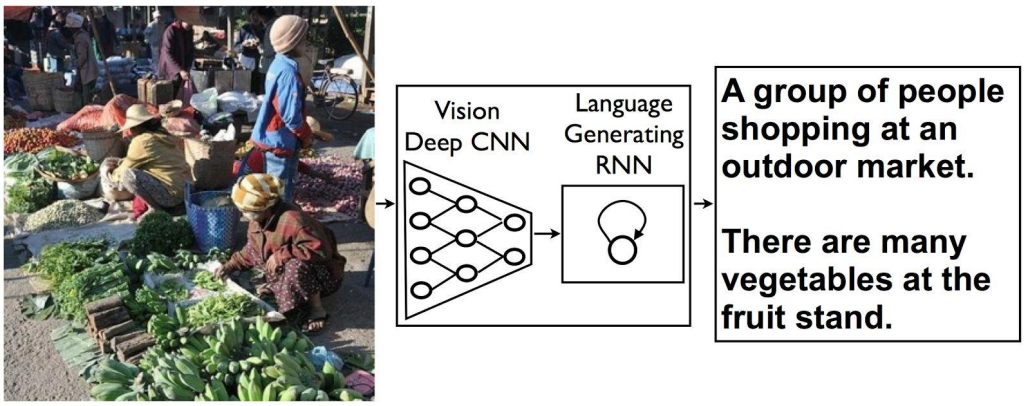


Figure 6.2: Architecture of model

Chapter 7

Experimentation & Results

7.1 Observations

Encoding th images into required format takes a lot of time. Training of model depends on the size of the dataset. The number of samples per epoch determines the accuracy of model. When the model was first run on a very small dataset, the results were quite bad. We tried many tricks to reduce the training time like reducing batch size but keeping samples same. However the results were not good.

7.2 Training Results

For training the model on 8k images with 32 batch size, 50 epochs and 5000 objects per epoch it took around 3 days 18 hrs. Training accuracy metric obtained was 0.84.

7.3 Testing Results

After the training of the model, we tried to test on some images and obtained interesting results. Some of them are shown below. However this model too has some errors. These are shown below

(a) a group of people in a on a bike

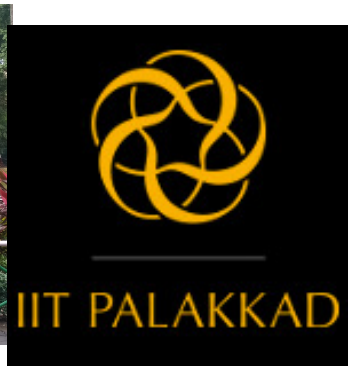
(b) a large elephant in water

(c) a group of people standing around a table

(d) a bottle of water and glass

Figure 7.1: Captions generated by the model

(a) a group of people standing next to a bunch of bananas



(b) a street sign with a sky background

Figure 7.2: Erroneous caption generations by the model

Chapter 8

Conclusion & Future Work

8.1 Conclusion

We are successful in building a model that can tag images. We have created a working model that outputs a caption to a given image that is related on the actions going in the image.

8.2 Future Work

We can make our model even more efficient by training on more number of images. Increasing the number of samples per epoch leading to higher accuracy. Increasing the number of LSTM layers. We can try to resolve issues with all conflicting images. This would result in better model.

This model can further be used in some camera applications such that the blind people can be benefited. They can use the caption to describe the scenario going around them.

References

[1] "Show and tell: image captioning open sourced in tensorflow." https://research.googleblog.com/2016/09/show-and-tell-image-captioning-open.html, 2016. 4

[2] "Picture this : Microsoft research project." https://blogs.microsoft.com/ai/picture-this-microsoft-research-project-can-interpret-caption-photos/, 2015. 4

[3] "A neural image caption generator." https://arxiv.org/pdf/1411.4555.pdf, 2015. 6

[4] "Tensor flow documentation." https://www.tensorflow.org/api_docs/python/tf/nn/conv2d, 2015. 9