Natural Language Processing Project Report
on

Caption Generation

by

1. Imran Khan
2. Manish Meshram
3. Mandar Morye

under the guidance of

Prof. Suresh Mestry

MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI

Department of Computer Engineering

University of Mumbai

2019 - 2020

Abstract

In this project we are automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. In this project, we present a generative model based on a deep recurrent architecture that combines recent advances in computer vision and machine translation and that can be used to generate natural sentences describing an image. The model is trained to maximize the likelihood of the target description sentence given the training image. Experiments on several data sets show the accuracy of the model and the fluency of the language it learns solely from image descriptions. Our model will often quite accurate.

Contents

List of Figures

List of Algorithms

List of Acronyms

DL = Deep Learning
CNN = Convolutional Neural Network
LSTM = Long Short-Term Memory
NLP = Natural Language Processing
RELU = Rectified Linear Unit

Chapter 1

Introduction

1.1 Introduction Description

Being able to automatically describe the content of an image using properly formed English sentences is a very challenging task, but it could have great impact, for instance by helping visually impaired people better understand the content of images on the web. This task is significantly harder, for example, than the well-studied image classification or object recognition tasks, which have been a main focus in the computer vision community [1]. Indeed, a description must capture not only the objects contained in an image, but it also must express how these objects relate to each other as well as their attributes and the activities they are involved in. Moreover, the above semantic knowledge has to be expressed in a natural language like English, which means that a language model is needed in addition to visual understanding.

Most previous attempts have proposed to stitch together existing solutions of the above sub-problems, in order to go from an image to its description [2], [3]. In contrast, we would like to present in this work a single joint model that takes an image I as input, and is trained to maximize the likelihood p(S—I) of producing a target sequence of words S = S1, S2, . . . where each word St 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 p(T—S). For many years, machine translation was also achieved by a series of separate tasks (translating words individually, aligning words, reordering, etc), but recent work has shown that translation can be done in a much simpler way using Recurrent Neural Networks (RNNs) [4], [5], [6] 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.

Here, we propose to follow this elegant recipe, replacing the encoder RNN by a deep convolution neural network (CNN). Over the last few years it has been convincingly shown that 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 [7]. 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 (see Fig. 1). We call this model the Neural Image Caption, or NIC.

Our contributions are as follows. First, we present an end-to-end system for the problem.

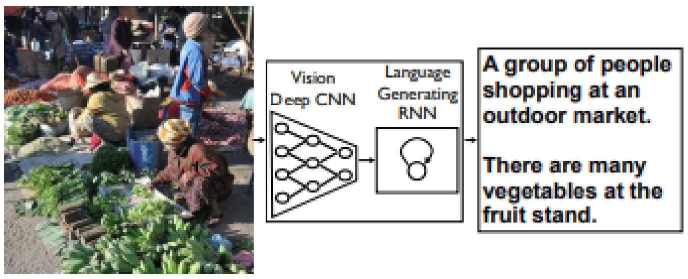

Vision
Deep CNN
Language
Generating
RNN
A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.

Figure 1.1: NIC, our model, is based end-to-end on a neural network consisting of a vision CNN followed by a language generating RNN. It generates complete sentences in natural language from an input image, as shown on the example above.

It is a neural net which is fully trainable using stochastic gradient descent. Second, our model combines state-of-art sub-networks for vision and language models. These can be pre-trained on larger corpora and thus can take advantage of additional data.

1.2 Organization of Report

Describe every chapter (what every chapter contains)

- **Ch.1 Introduction:** Description and Overview about the automatic captioning generation of the image using machine learning.
- **Ch.2 Review of Literature:** Limitation of the existing system such as generating caption using audio and various other systems.
- **Ch.3 Proposed System:** Detail design and components of the system and algorithms like CNN, RNN and LSTM.
- **Ch.4 Implementation Plan:** Sequential representation of the process require to implement image to caption generation.
- **Ch.5 Conclusion:** Implemented caption generating system from image using machine learning.

Chapter 2

Literature Review

The problem of generating natural language descriptions from visual data has long been studied in computer vision, but mainly for video [8], [9]. Traditionally, this has led to complex systems composed of visual primitive recognizers combined with a structured formal language, e.g. And Or Graphs or logic systems, which are further converted to natural language via rule-based systems. Such systems are heavily hand-designed, relatively brittle and have been demonstrated only on limited domains, e.g. traffic scenes or sports.

2.1 Survey Existing system

The problem of still image captioning in natural language has recently enjoyed increased interest. Recent advances in object recognition and detection as well as attribute recognition has been used to drive natural language generation systems, though these are limited in their expressivity. Farhadi et al. [2] use detections to infer a triplet of scene elements which is converted to text using templates. Similarly, Li et al. [10] start off with detections and piece together a final description using phrases containing detected objects and relationships. A more complex graph of detections beyond triplets is used by Kulkani et al.[3], but with template-based text generation. More powerful language models based on language parsing have been used as well [11], [12]. The above approaches have been able to describe images "in the wild", but they are heavily hand-designed and rigid when it comes to text generation.

A large body of work has addressed the problem of ranking descriptions for a given image [8], [9], [10]. Such approaches are based on the idea of co-embedding of images and text in the same vector space. For an image query, descriptions are retrieved which lie close to the image in the embedding space. Most closely, neural networks are used to co-embed images and sentences together [2] or even image crops and subsentences [3] but do not attempt to generate novel descriptions. In general, the above approaches cannot describe previously unseen compositions of objects, even though the individual objects might have been observed in the training data. Moreover, they avoid addressing the problem of evaluating how good a generated description is. More recently neural net based recognizer are used to detect a larger set of words and in conjunction with a language model sentences are generated [23].

In this work we combine deep convolutional nets for image classification [4] with recurrent networks for sequence modeling [5], to create a single network that generates descriptions of images. The RNN is trained in the context of this single "end-to-end" network. The model is inspired by recent successes of sequence generation in machine translation [4], [5], [6], with the difference that instead of starting with a sentence, we provide an image processed by a convolutional net.

In the summer of 2015 a few approaches were introduced which follow the above general paradigm. The closest works are by Kiro setal. [6] who use a neural net, but a feedforward one, to predict the next word given the image and previous words. A recent work by Mao et al. [7], [8] uses a recurrent NN for the same prediction task. This is very similar to the present proposal but there are a number of important differences: we use a more powerful RNN model, and provide the visual input to the RNN model directly, which makes it possible for the RNN to keep track of the objects that have been explained by the text. As a result of these seemingly insignificant differences, our system achieves substantially better results on the established benchmarks. Further, Kiros et al. [9] propose to construct a joint multimodal embedding space by using a powerful computer vision model and an LSTM that encodes text. In contrast to our approach, they use two separate pathways (one for images, one for text) to define a joint embedding, and, even though they can generate text, their approach is highly tuned for ranking. A recurrent network is being used by Donahue et al. [3] who address in addition activity recognition and video description.

In addition, some approaches try to model in a more explicit fashion the visual anchoring of sentence parts claiming a performance benefit. Xu et al. [1] explore attention mechanisms over image regions where while emitting words the system can focus on image parts. An explicit word to region alignment is utilized during training by Karpathy et al. [2]. Finally, Chen et al. [3] build a visual representation for sentence parts while generating the description. Further analysis of the above approaches were reported by Devlin et al. [4].

2.2 Limitation Existing system or research gap

Sometimes highly noisy image can be difficult to detect object. Images contains too many objects in images are hard to link logically which become chaotic. Irregularity in images like very low contrast or high brightness which require image filter for processing the image is out scope of the project.

2.3 Problem Statement and objectives

The aim of image caption is generate the context of the image, the image contains various objects and their properties, which should be logically linked and generate informative text about image using machine learning algorithm.

2.3.1 Objectives

The proposed system can be deployed as a computer application and API service. Following objectives can be fulfilled with the help of the image caption generator:

- Take a image form user as input for the context generation or caption generation of image.
- The input image should further processed by detecting various objects and their properties of the objects.

- The objects and properties of the objects should be linked logically or to find relations between the objects.
- Using the logical relationship between objects a informative text should be generated using Language Generator Neural Network.

2.4 Scope

Image captioning is a popular research area of Artificial Intelligence (AI) that deals with image understanding and a language description for that image. Image understanding needs to detect and recognize objects. It also needs to understand scene type or location, object properties and their interactions. Generating well-formed sentences requires both syntactic and semantic understanding of the language.

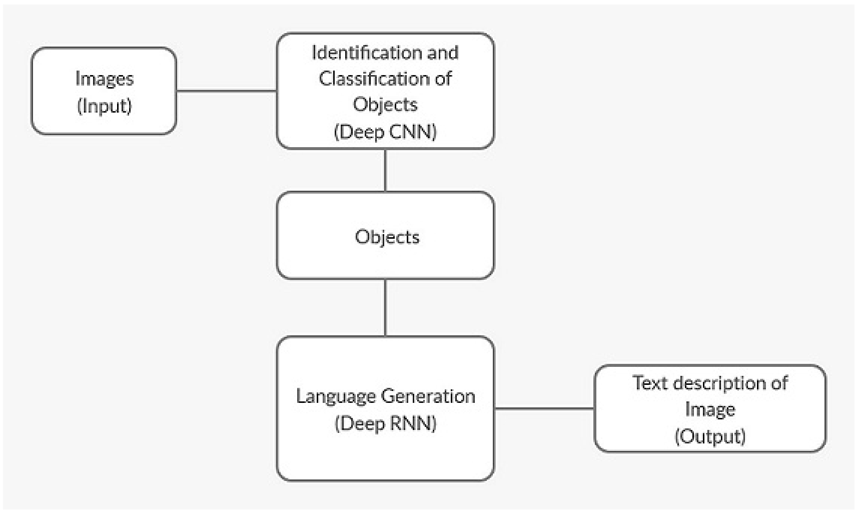


Figure 2.1: Architecture of System

Chapter 3

Proposed System

The project main aim of project is interpret the image, process the image and generate small descriptive text. In this system, we are identifying the objects using Convolution Neural Network. The identified objects are logically linked using Recurrent Neural Networks and Informative Text is generated.

3.1 Analysis/Framework/Algorithm

3.1.1 CNN

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.

3.1.2 RNN

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.

3.1.3 LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections that make it a ”general purpose computer” (that is, it can compute anything that a Turing machine can).

3.2 Details of Hardware and Software

3.2.1 Software Requirements

- OS
- Python 3.5+

- Python IDE
- Tensorflow

3.2.2 Hardware Requirements

- Processor
- RAM
- ROM
- GPU

3.3 Design Details

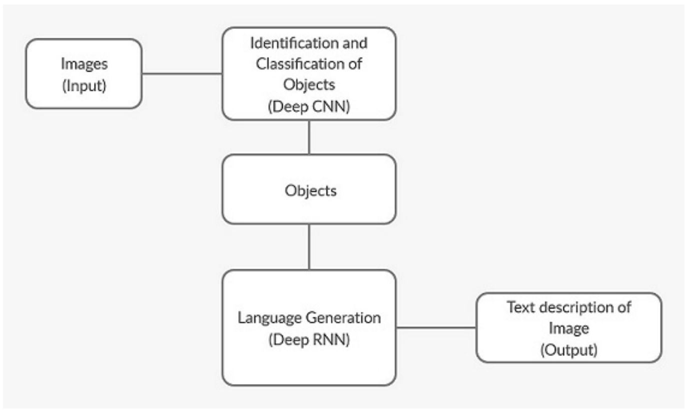


Figure 3.1: The Proposed System/System Architecture

- Object Detection. In the past few years, significant progress have been done in object detection. These advances are driven by the success of region proposal methods
- This part is designed to extract the information of objects spatial locations which in turn reflect their spatial relationships.

3.3.1 Detailed Design

Faster CNN is composed of two modules. The first module is a deep fully convolution network that propose regions, the second module is the Fast CNN detector [5] which uses the proposed regions. To generate

This part is designed to extract the information of objects spatial locations which in turn reflect their spatial relationships. Junqi etal. [4] also considered the locations of different localized regions. However, they just added the boxes central's with x location, y location, width, height and area ratio with respect to the entire image's geometry to the end of the

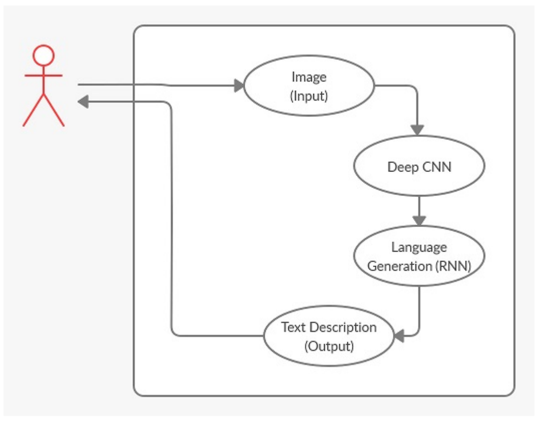


Figure 3.2: use case

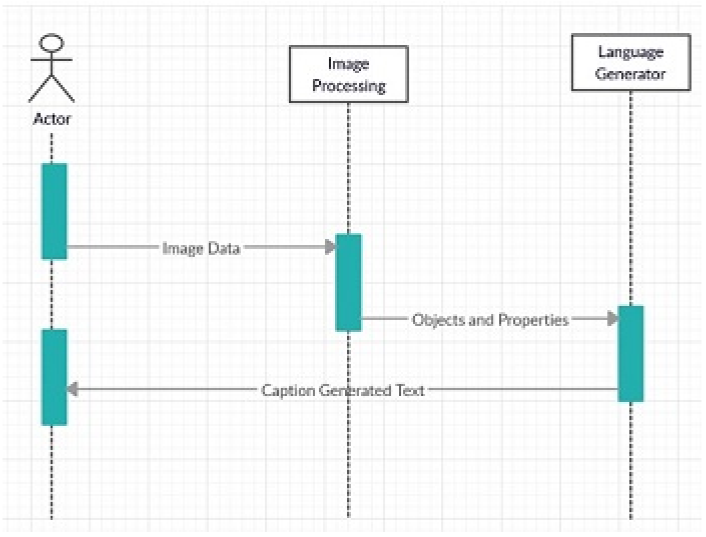


Figure 3.3: Sequence Diagram

vector of each localized regions. In this paper, the implementation of extracting information of each object location is completely different from [14].

In this paper, we describe a decoding part based on an LSTM network with attention mechanism. Attention mechanism was first used in neural machine translation area by [15]. Following the same mechanism, the authors of [10] introduced it into image processing domain whereas, [11] was the first to apply it in image captioning task. The key idea of attention mechanism is that when a sentence is used to describe an image, not every word in the sentence is "translated" from the whole image but actually it just has relation to a few subregions of an image. It can be viewed as a form of alignment from words of the sentence to subregions of the image.

3.4 Methodology/Procedures

In this project, we propose a neural and probabilistic framework to generate descriptions from images. Recent advance 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 thevariable length input into a fixed dimensional vector, and uses this representation to "decode" it to the desired 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.

Thus, we propose to directly maximize the probability of the correct description given the image by using the following formulation:

$$\theta* = \text{arg max}\ _{\theta} \sum_{I}^{S} \log \text{p(S} \mid \text{I}; \theta)$$

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, it is common to apply the chain rule to model the joint probability over S0, . . . , SN , where N is the length of this particular example as

$$\sum_{t=0}^{N} \log \text{p(S} \mid \text{I) = X log p(St} \mid \text{I, S0, . . . , St1)}$$

where we dropped the dependency on θ for convenience. At training time, (S, I) is a training example pair, and we optimize the sum of the log probabilities as described in (2) over the whole training set using stochastic gradient descent (further training details are given in Section 4). It is natural to model p(St | I, S0, . . . , St1) with a Recurrent Neural Network (RNN), where the variable number of words we condition upon up to t1 is expressed by a fixed length hidden state or memory ht. This memory is updated after seeing a new input xt by using a non-linear function f:

$$\text{ht+1 = f(ht, xt) .}$$

To make the above RNN more concrete two crucial design choices are to be made: what is the exact form of f and how are the images and words fed as inputs xt. 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. This model is outlined in the next section. For the representation of images, we use a Convolutional Neural Network (CNN). They have been widely used and studied for image tasks, and are currently state-of-the art for object recognition and detection. Our particular choice of CNN uses the recent approach of batch normalization and yields the current best performance on the ILSVRC 2014 classification competition [4]. Furthermore, they have been shown to generalize to other tasks such as scene classification by means of transfer learning [5]. The words are represented with an embedding model [6].

3.4.1 Procedures

Prepare Photo Data

Convolutional Neural Network(CNN)

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

LSTM-based Sentence Generator

The choice of f in (3) is governed by its ability to deal with vanishing and exploding gradients [5], the most common challenge in designing and training RNNs. To address this challenge, a particular form of recurrent nets, called LSTM, was introduced [5] and applied with great success to translation [4], [6] and sequence generation [7].

The core of the LSTM model is a memory cell c encoding knowledge at every time step of what inputs have been observed up to this step (see Figure 2) . The behavior of the cell is controlled by "gates" – layers which are applied multiplicatively and thus can either keep a value from the gated layer if the gate is 1 or zero this value if the gate is 0. In particular, 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:

$$\begin{aligned}
i_t &= \sigma(W_{ix}x_t + W_{im}m_{t-1}) \\
f_t &= \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \\
o_t &= \sigma(W_{ox}x_t + W_{om}m_{t-1}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \\
m_t &= o_t \odot c_t \\
p_{t+1} &= \text{Softmax}(m_t)
\end{aligned}$$

where $\odot$ 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 [5]. The nonlinearities are sigmoid (·) 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. 3.1.1 Training

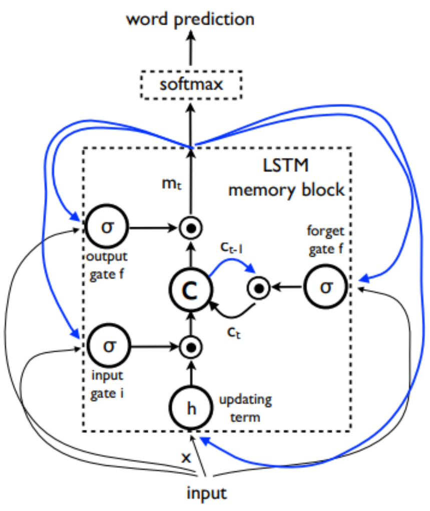


Figure 3.4: LSTM: the memory block contains a cell c which is controlled by three gates. In blue we show the recurrent connections – the output m at time t 1 is fed back to the memory at time t via the three gates; the cell value is fed back via the forget gate; the predicted word at time t 1 is fed back in addition to the memory output m at time t into the Softmax for word prediction.

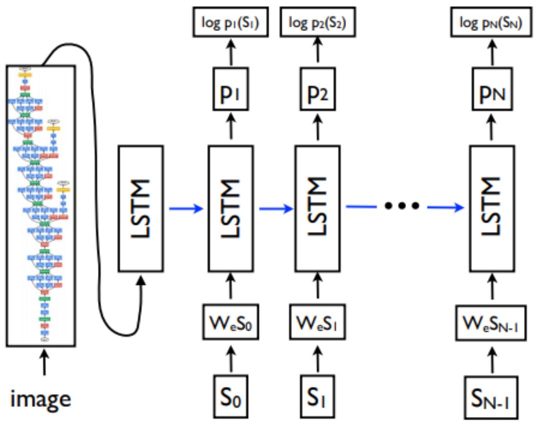


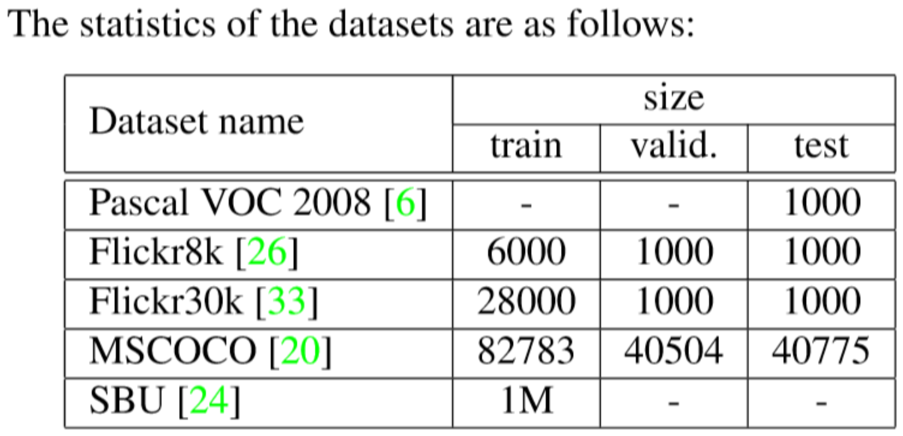
Figure 3.5: LSTM model combined with a CNN image embedder (as defined in [4]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in above fig. All LSTMs share the same parameters.

Chapter 4

Results and Discussions

4.1 Dataset

We downloaded the data from the above dataset link and imported the data which contains an image of 224*224 pixels and five captions to each image.



The statistics of the datasets are as follows:

Dataset name	size		
	train	valid.	test
Pascal VOC 2008 [6]	-	-	1000
Flickr8k [26]	6000	1000	1000
Flickr30k [33]	28000	1000	1000
MSCOCO [20]	82783	40504	40775
SBU [24]	1M	-	-

Figure 4.1: Dataset

- train images
  http://msvocds.blob.core.windows.net/coco2014/train2014.zip
- validation images
  http://msvocds.blob.core.windows.net/coco2014/val2014.zip
- captions for both train and validation
  http://msvocds.blob.core.windows.net/annotations-1-0-3/captions$_{t}rain - val2014.zip$

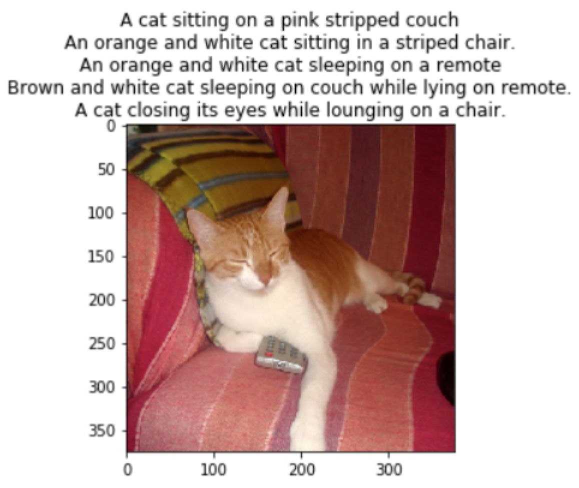

A cat sitting on a pink stripped couch
An orange and white cat sitting in a striped chair.
An orange and white cat sleeping on a remote
Brown and white cat sleeping on couch while lying on remote.
A cat closing its eyes while lounging on a chair.
0
50
100
150
200
250
300
350
0
100
200
300

Figure 4.2: labelled data with 5 caption

4.2 Preprocessing the data

4.2.1 Prepare Photo Data

We will use a pre-trained model to interpret the content of the photos. There are many models to choose from. In this case, we will use the Oxford Visual Geometry Group, or VGG, model that won the ImageNet competition in 2014.

Keras provides this pre-trained model directly. Note, the first time you use this model, Keras will download the model weights from the Internet, which are about 500 Megabytes. This may take a few minutes depending on your internet connection.

We could use this model as part of a broader image caption model. The problem is, it is a large model and running each photo through the network every time we want to test a new language model configuration (downstream) is redundant.

Instead, we can pre-compute the “photo features” using the pre-trained model and save them to file. We can then load these features later and feed them into our model as the interpretation of a given photo in the dataset. It is no different to running the photo through the full VGG model; it is just we will have done it once in advance.

This is an optimization that will make training our models faster and consume less memory.

We can load the VGG model in Keras using the VGG class. We will remove the last layer from the loaded model, as this is the model used to predict a classification for a photo. We are not interested in classifying images, but we are interested in the internal representation of the photo right before a classification is made. These are the “features” that the model has extracted from the photo.

Keras also provides tools for reshaping the loaded photo into the preferred size for the model (e.g. 3 channel 224 x 224 pixel image).

Below is a function named extract$_features()that, givenadirectoryname, willloadeachphoto, prepare$
$dimensional4, 096elementvector.$

4.2.2 Training Details

Many of the challenges that we faced when training our model shad to do with overfitting. Indeed,purely supervised approaches require large amounts of data, but the datasets that are of high quality have less than 100000 images. The task of assigning a description is strictly harder than object classification and data driven approaches have only recently become dominant thanks to datasets as large as ImageNet (with ten times more data than the datasets we described in this paper, with the exception of SBU). As a result, we believe that, even with the results we obtained which are quite good, the advantage of our method versus most current human-engineered approaches will only increase in the next few years as training set sizes will grow.

Nonetheless, we explored several techniques to deal with overfitting. The most obvious way to not overfit is to initialize the weights of the CNN component of our system to a pretrained model (e.g., on ImageNet). We did this in all the experiments, and it did help quite a lot in terms of generalization. Another set of weights that could be sensibly initialized are We, the word embeddings. We tried initializing them from a large news corpus [22], but no significant gains were observed, and we decided to just leave them uninitialized for simplicity. Lastly, we did some model level overfitting-avoiding techniques. We tried dropout [34] and ensembling models, as well as exploring the size (i.e., capacity) of the model by trading off number of hidden units versus depth. Dropout and ensembling gave a few BLEU points imp of high quality have less than 100000 images. The task of assigning a description is strictly harder than object classification and data driven approaches have only recently become dominant thanks to datasets as large as Image Net (with ten times more data than the datasets we described in this paper, with the exception of SBU). As a result, we believe that, even with the results we obtained which are quite good, the advantage of our method versus most current human-engineered approaches will only increase in the next few years as training set sizes will grow.

We trained all sets of weights using stochastic gradient descent with fixed learning rate and no momentum.

All weights were randomly initialized except for the CNN weights, which we left unchanged because changing them had a negative impact. We used 512 dimensions for the embeddings and the size of the LSTM memory.

Descriptions were preprocessed with basic tokenization, keeping all words that appeared at least 5 times in the training set.

4.2.3 Prepare Text Data

The dataset contains multiple descriptions for each photograph and the text of the descriptions requires some minimal cleaning.

First, we will load the file containing all of the descriptions.

Each photo has a unique identifier. This identifier is used on the photo filename and in the text file of descriptions.

Next, we will step through the list of photo descriptions. Below defines a function loaddescriptions() that, given the loaded document text, will return a dictionary of photo identifiers to descriptions. Each photo identifier maps to a list of one or more textual descriptions.

Next, we need to clean the description text. The descriptions are already tokenized and easy to work with. We will clean the text in the following ways in order to reduce the size of the vocabulary of words we will need to work with:

- Convert all words to lowercase.
- Remove all punctuation.
- Remove all words that are one character or less in length (e.g. ‘a’).
- Remove all words with numbers in them.
- Addition of special tags like start tag, end tag and tag for some missing word in caption.

Once cleaned, we can summarize the size of the vocabulary.

Ideally, we want a vocabulary that is both expressive and as small as possible. A smaller vocabulary will result in a smaller model that will train faster.

For reference, we can transform the clean descriptions into a set and print its size to get an idea of the size of our dataset vocabulary.

Finally, we can save the dictionary of image identifiers and descriptions to a new file named descriptions.pkl, with one image identifier and description per line.

4.3 Develop Deep Learning Model

4.3.1 Loading Data into model

First, we must load the prepared photo and text data so that we can use it to fit the model. We are going to train the data on all of the photos and captions in the training dataset. While training, we are going to monitor the performance of the model on the development dataset and use that performance to decide when to save models to file.

Now, we can load the photos and descriptions using the pre-defined set of train or development identifiers.

The model we will develop will generate a caption given a photo, and the caption will be generated one word at a time. The sequence of previously generated words will be provided as input. Therefore, we will need a ‘first word’ to kick-off the generation process and a ‘last word‘ to signal the end of the caption. We will use the strings ‘startseq‘ and ‘endseq‘ for this purpose. These tokens are added to the loaded descriptions as they are loaded. It is important to do this now before we encode the text so that the tokens are also encoded correctly.

The description text will need to be encoded to numbers before it can be presented to the model as in input or compared to the model’s predictions. The first step in encoding the data is to create a consistent mapping from words to unique integer values. Keras provides the Tokenizer class that can learn this mapping from the loaded description data.

Each description will be split into words. The model will be provided one word and the photo and generate the next word. Then the first two words of the description will be provided to the model as input with the image to generate the next word. This is how the model will be trained.

Later, when the model is used to generate descriptions, the generated words will be concatenated and recursively provided as input to generate a caption for an image.

The input text is encoded as integers, which will be fed to a word embedding layer. The photo features will be fed directly to another part of the model. The model will output a prediction, which will be a probability distribution over all words in the vocabulary.

4.3.2 Defining the Model

Images are nothing but input (X) to our model. As you may already know that any input to a model must be given in the form of a vector. We need to convert every image into a fixed sized vector which can then be fed as input to the neural network. For this purpose, we opt for transfer learning by using the InceptionV3 model (Convolutional Neural Network) created by Google Research. This model was trained on Imagenet dataset to perform image classification on 1000 different classes of images. However, our purpose here is not to classify the image but just get fixed-length informative vector for each image. This process is called automatic feature engineering.

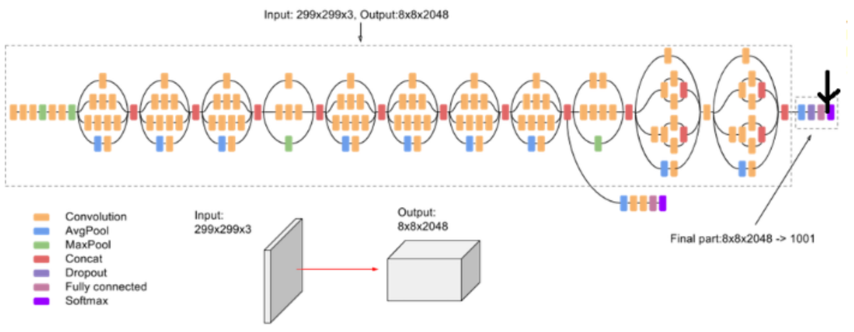


Figure 4.3: InceptionV3 Model

4.3.3 Fitting the Model

Now that we know how to define the model, we can fit it on the training dataset. The model learns fast and quickly overfits the training dataset. For this reason, we will monitor the skill of the trained model on the holdout development dataset. When the skill of the model on the development dataset improves at the end of an epoch, we will save the whole model to file. At the end of the run, we can then use the saved model with the best skill on the training dataset as our final model.

4.3.4 Generate New Captions

Now that we know how to develop and evaluate a caption generation model, how can we use it? Almost everything we need to generate captions for entirely new photographs is in the model file. We also need the Tokenizer for encoding generated words for the model while generating a sequence, and the maximum length of input sequences, used when we defined the model. We can hard code the maximum sequence length. With the encoding of text, we can create the tokenizer and save it to a file so that we can load it quickly whenever we need it without needing the entire dataset. An alternative would be to use our own vocabulary file and mapping to integers function during training. We can create the Tokenizer as before and save it as a pickle file tokenizer.pkl. The complete example is listed below.

4.4 Results

Since our model is data driven and trained end-to-end, and given the abundance of datasets, we wanted to answer questions such as “how dataset size affects generalization”, “what kinds of transfer learning it would be able to achieve”, and “how it would deal with weakly labeled examples”. As a result, we performed experiments on five different datasets, which enabled us to understand our model in depth.

We can now load the tokenizer whenever we need it without having to load the entire training dataset of annotations.

Now, let’s generate a description for a new photograph. Below is a new photograph that I chose randomly from dataset.

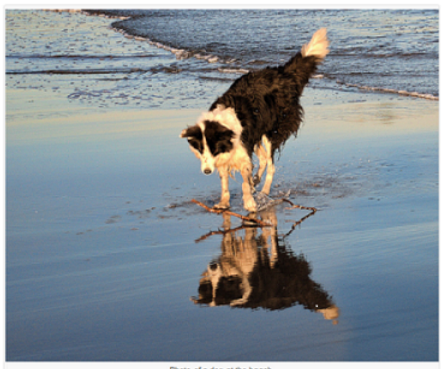
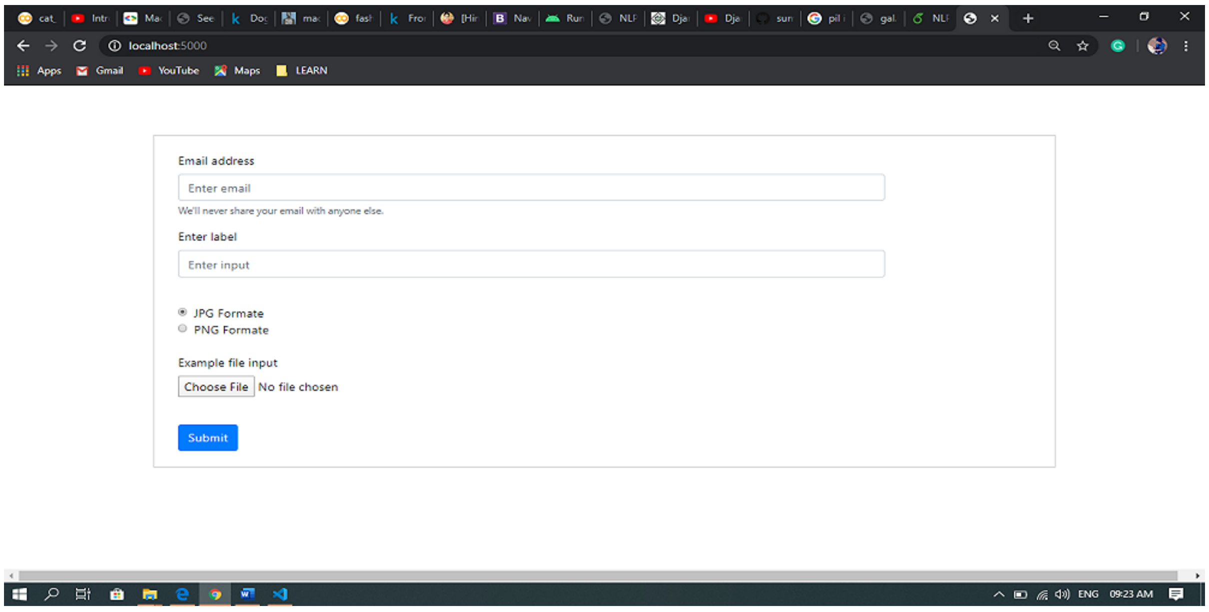


Figure 4.4: A random example from the dataset with generated caption



Email address

Enter email

We'll never share your email with anyone else.

Enter label

Enter input

● JPG Formate
○ PNG Formate

Example file input

Choose File No file chosen

Submit

Figure 4.5: Input form for uploading an image

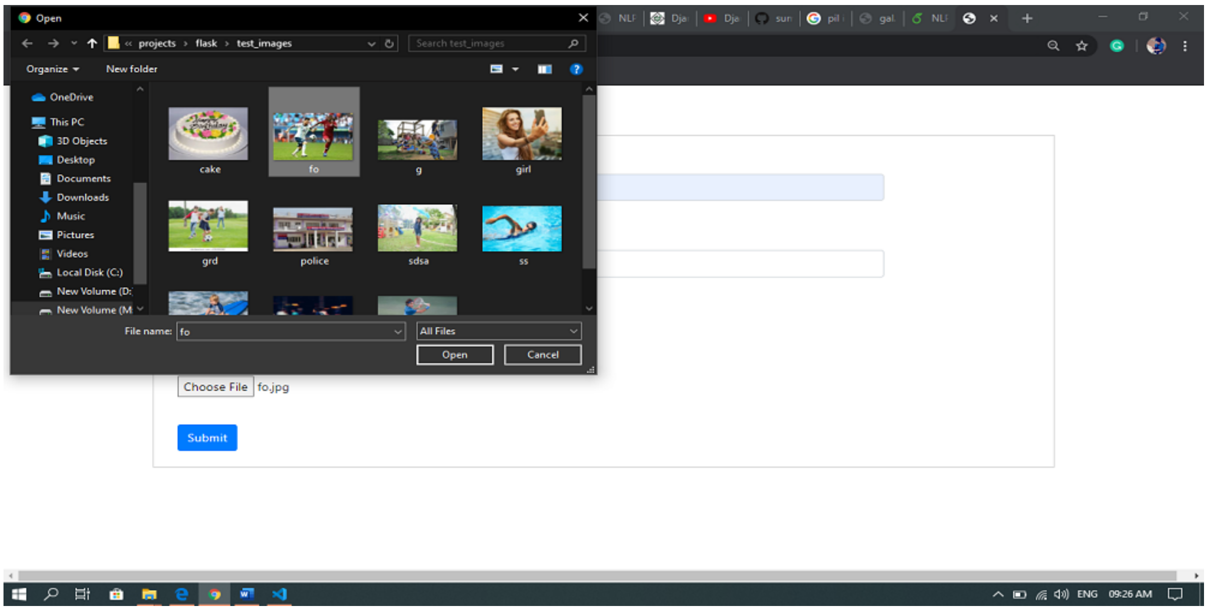


Figure 4.6: Uploading an Image

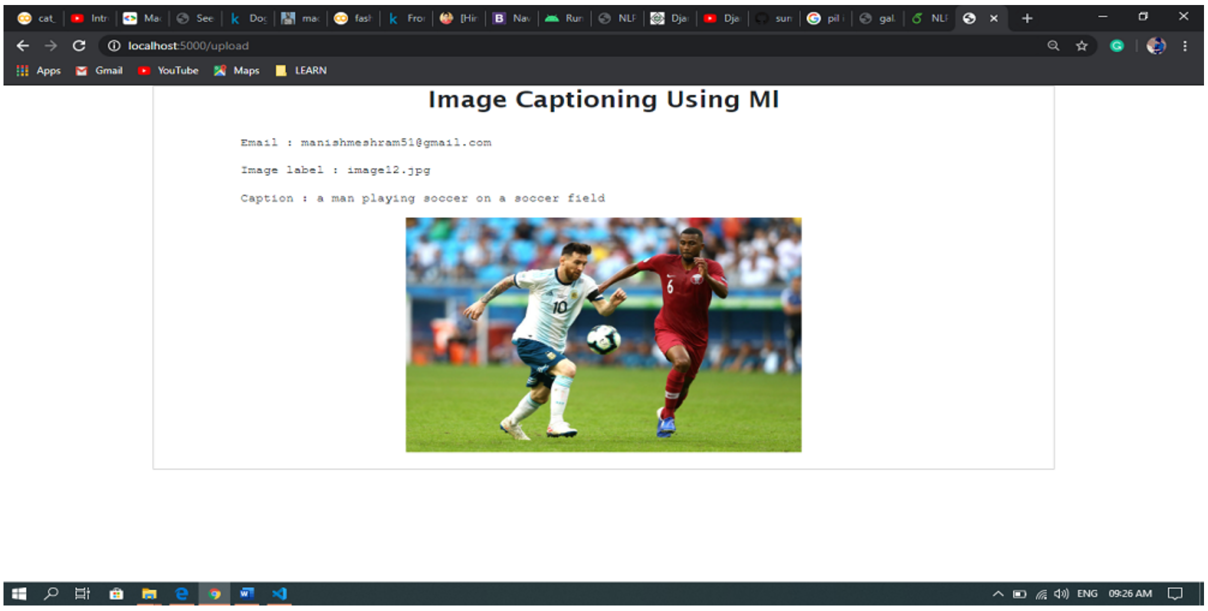


Image Captioning Using Ml

```
Email : manishmeshram51@gmail.com
Image label : image12.jpg
Caption : a man playing soccer on a soccer field
```

Figure 4.7: Caption Generation on selected Image

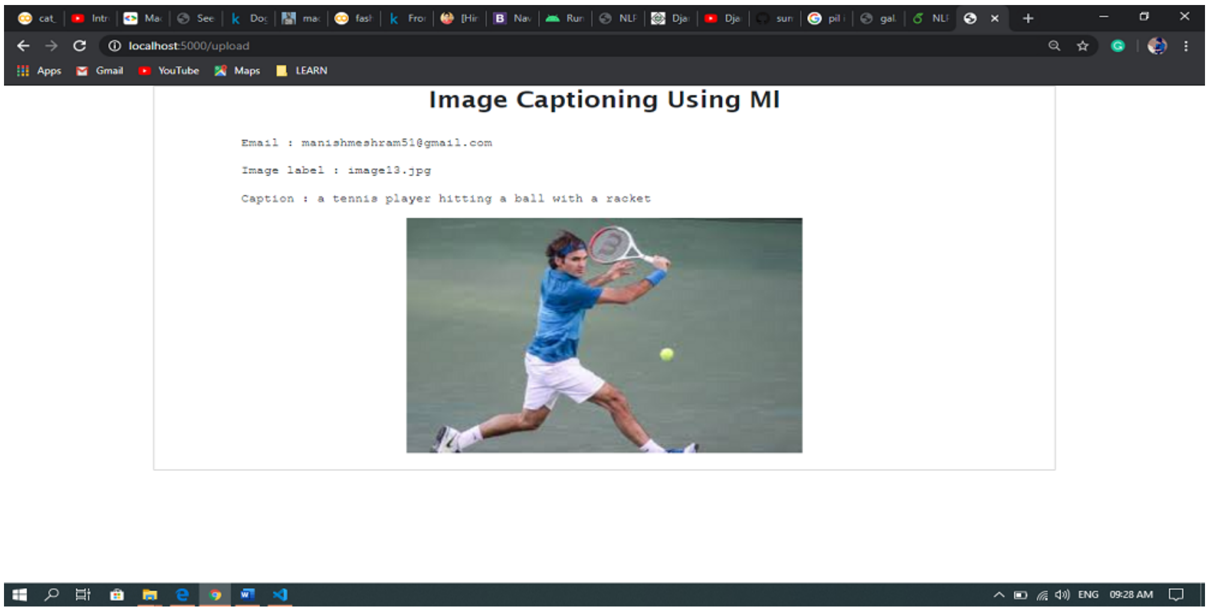


Image Captioning Using Ml

Email : manishmeshram51@gmail.com

Image label : image13.jpg

Caption : a tennis player hitting a ball with a racket

Figure 4.8: Caption Generation on another Image

Chapter 5

Conclusions

The development of proposed system will helpful to various application and further academic research and implementation. Hence, In this project we will be generating automatic captions for the image using CNN and RNN for objects detection and then text generating by logically linking these objects. The CNN or object detector not only extract the object but also the properties of the objects. Thus, we can generate informative text about the image.