

Image Captioning Using Deep Learning Techniques

Project report Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

In

Computer Science and Engineering/Information Technology

By

Gulshan Sankhyan (161384)

Under the supervision of

Dr. Himanshu Jindal

To



Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Wanknaghat, Solan-173234,
Himachal Pradesh

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled "Image Captioning Using Deep Learning Techniques" in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of our own work carried out over a period from August 2019 to May 2020 under the supervision of **Dr. Himanshu Jindal** (Assistant Professor Grade: II).

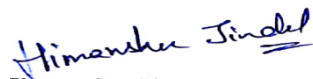
The matter embodied in the report has not been submitted for the award of any other degree or diploma.



(Student Signature)

Gulshan Sankhyan (161384)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



(Supervisor Signature)

Dr. Himanshu Jindal

Assistant Professor (Grade-II)

Department of Computer Science & Engineering and Information Technology

Dated: 28/05/2020

ACKNOWLEDGEMENT

I take upon this opportunity endowed upon me, to thank all those who have been part of this endeavour. I would like to thank my supervisor '**Dr. Himanshu Jindal**' for giving me the right direction to follow and proper guidance regarding the topic. Without their active involvement and the right guidance this would not have been possible. I sincerely thank our Head of Department '**Prof. Dr. Satya Prakash Ghrera**' for giving me the chance as well as the support for the time being. Last but not the least, I heartily appreciate all those people who have helped me directly or indirectly in making this task a success. In this context, I would like to thank all the other staff members, both teaching and non-teaching, which have extended their timely help and eased my task.

CONTENTS

CANDIDATE’S DECLARATION	i
ACKNOWLEDGEMENT	ii
CONTENTS	iii
ABBREVIATIONS	iv
LIST OF FIGURES	v
LIST OF TABLES	vi
1. INTRODUCTION	1
1.1 Introduction	1
1.1.1 Image Processing	1
1.1.2 Computer Vision Image Processing and Machine Learning	2
1.1.3 Computer Vision	6
1.1.4 Convolution Filters and Edge detectors	6
1.1.5 CNN Layers and Feature Visualizations	6
1.1.6 Recurrent and Recursive Neural Networks	9
1.1.7 Long Short-Term Memory Networks	14
1.1.8 Part-of-Speech Tagging	16
1.1.9 Word Embeddings	17
1.2 Problem Statement	18
1.3 Objectives	20
1.4 Methodology	20
2 LITERATURE SURVEY	22
2.1 Literature Survey	22
3 SYSTEM DEVELOPMENT	29
3.1 Pre-requisite Tools	29
3.1.1 Anaconda	29
3.1.2 Spyder	29
3.2 Model Development	30
3.2.1 Image Captioning	30
3. 2.2 CNN-RNN model	30
3.2.3 DataSet	31
3.2.4 Learning Rate	31
3.2.5 Optimizers	32
3.2.6 Underfitting and Overfitting	32

3.2.7 Training Phase	33
4 PERFORMANCE ANALYSIS	34
4.1 Input Images and Output Text	34
5 CONCLUSIONS	42
5.1 Conclusions	42
5.2 Future Scope	43
REFERENCE	44

ABBREVIATIONS

CV	Computer Vision
IP	Image Processing
ML	Machine Learning
DL	Deep Learning
AI	Artificial Intelligence
CNN	Convolutional neural network
RNN	Recurrent neural network
LSTM	Long short-term memory
MS	MicroSoft
COCO	Common Objects in Context
POS	Part-of-Speech
SGD	Stochastic Gradient Descent

LIST OF FIGURES

Fig 1: Head movement ventriculogram signal captured by low noise camera	3
Fig 2: Edge detection in image processing software	4
Fig 3: Venn diagram showing comparison between ML CV IP and SP	5
Fig 4: Standard CNN model	7
Fig 5: Basic RNN model	10
Fig 6: Red arrow shows gradient flow	13
Fig 7: LSTM	16
Fig 8: CNN-RNN model	31
Fig 9: Images (a) (b) (c) (d) and (e) are some random images fed into the model after second epoch	37

LIST OF TABLES

Table 1: Summarizes the input and output of every domain	5
Table 2: Output Table	38

1. INTRODUCTION

1.1 Introduction

An image or digital image is a representation of visual information. An image is a picture that has been made or copied and stored in electronic form. An image can be described by a two-dimensional array arranged in rows and columns. A digital image is composed of a set number of segments all which parts have a value at a particular location. These elements referred to as picture elements image elements and pixels. A pixel is most universally used to denote the components of a digital image.

Types of Images

- Binary Image: These are the types of images that take discrete values (0 or 1). Since only two values are taken hence, they are called binary images. The Black color is denoted by 1 and white color by 0.
- Greyscale: These images are also known as monochrome images since they do not represent any color. They only state the level of brightness for one color. This type of image consists of only 8 bytes. 0 denotes black and 255 represents white and in between are various levels of brightness.
- Coloured: These images contain three bands namely red green and blue. The intensity of all the three bands is 8 bytes. The various intensity levels in each band convey the entire colored image. The size of the colored image is 24 bits.

1.1.1 Image Processing

Image Processing is a technique to perform certain activities on an image to get an improved picture or to isolate some significant information from it. It is a sort of signal processing wherein input is a picture and yield might be picture or qualities/highlights related to that picture. Image processing includes three steps:

- The image is imported using image acquisition tool

- The image is then analysed and then manipulated
- Then the output in which the result may be an altered image, or it may be in the form of the report of image analysis.

The fundamental advantage of Digital Image Processing techniques is its versatility reiterability and the conservation of original data precision. The various Image Processing techniques are:

- Image pre-processing:
- Image enhancement:
- Image segmentation:
- Image restoration:
- Image transformation:
- Feature extraction:

1.1.2 Computer Vision Image Processing and Machine Learning

Computer vision image processing signal processing machine learning – you have heard the terms but what is the difference between them? Each of these fields is based on the input of an image or signal. They process the signal and then give us altered output in return. So, what distinguishes these fields from each other? The boundaries between these domains may seem obvious since their names already imply their goals and methodologies. However, these fields draw heavily from the methodologies of one another which can make the boundaries between them blurry. In this article we will draw the distinction between the fields according to the type of input used and more importantly the methodologies and outputs that characterize each one.

Let us start by defining the input used in each field. Many if not all inputs can be thought of as a type of a signal. We favour the engineering definition of a signal that is a signal is a sequence of discrete measurable observations obtained using a capturing device be it a camera a radar ultrasound a microphone et cetera... The dimensionality of the input signal

gives us the first distinction between the fields. Mono-channel sound waves can be thought of as a one-dimensional signal of amplitude over time whereas a picture is a two-dimensional signal made up of rows and columns of pixels. Recording consecutive images over time produces video which can be thought of as a three-dimensional signal.

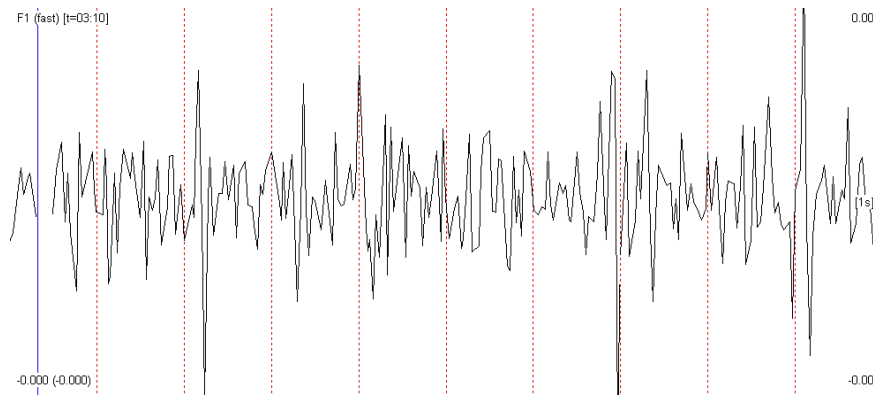


Fig 1: Head movement ventriculogram signal captured by low noise camera

Input of 1 kind will generally be reworked to a different. as an example, ultrasound pictures are recorded victimisation the reflection of sound waves from the article determined reworked to a visible modality. X-ray will be thought of equally to ultrasound solely that hot absorption is reworked into a picture. resonance Imaging (MRI) records the excitation of ions and transforms it into a visible image. during this sense signal process could be understood as image process.

Let us verify the x-ray as a prototypic example. Let us assume we have got non heritable one image from AN apparatus. Image process engineers (or software) would typically have to be compelled to improve the standard of the image before it passes to the physician s show. Hence the input is a picture and the output is a picture. Image process is as its name implies all concerning the process of pictures. each of the input and the output are pictures. strategies often employed in image process are filtering noise removal edge detection color process so forth. software package packages dedicated to image process are as an example Photoshop and limping.



Fig 2: Edge detection in image processing software

In computer vision we tend to would like to receive quantitative and qualitative info from visual information. remarkably like the method of visual reasoning of human vision; we can distinguish between objects classify them type them in step with their size and so forth. laptop vision like image process takes pictures as input. However, it returns another style of output particularly info on size color number et cetera. Image process strategies are controlled for achieving tasks of laptop vision.

Extending on the far side one image in laptop vision we tend to attempt to extract info from video. as an example, we tend to might want to count the number of cats passing by an exact purpose within the street as recorded by a video camera. Or we tend to might want to live the gap go past an athlete throughout the sport and extract different statistics. Therefore, temporal info plays a significant role in computer vision terribly like it is with our own manner of understanding the globe.

With training the classifier learns to apart a diver from a fish. Once the training set is completed the classifier is intended to repeat the same observation because the human expert can create in an exceedingly new scenario. Thus, machine learning is kind of a general framework in terms of input and output. Like humans it can receive any signal as an input and give almost any type of output.

Table 1: Summarizes the input and output of every domain:

Domain	Input	Output
Image processing	Image	Image
Signal processing	Signal	Signal quantitative information e.g. Peak location
Computer vision	Image/video	Image quantitative/qualitative information e.g. size color shape classification etc...
Machine learning	Any feature signal from e.g. image video sound etc..	Signal quantitative/qualitative information image

This can also be presented as a Venn diagram:

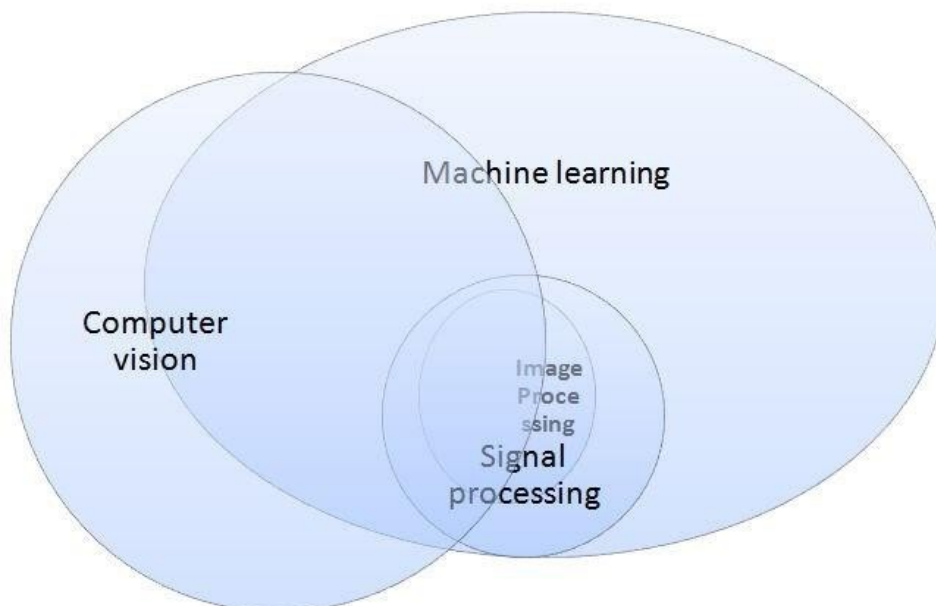


Fig 3: Venn diagram showing comparison between ML CV IP and SP

1.1.3 Computer Vision

Computer Vision is how machines like Smartphones or Robotic Systems visually perceives the world and respond to it. It is modelled after human vision. A robot can gather data through cameras or other sensors and then use that input to identify different objects and safely move through its environment.

1.1.4 Convolution Filters and Edge detectors

- Canny Edge Detector^[3]: Canny Edge Detection is an extremely popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and with each stage as:
 - Noise Reduction^[3]
 - Finding Intensity Gradient of the Image^[3]
 - Non-maximum Suppression^[3]
 - Hysteresis Thresholding^[3]
- Hough Line Transform^[4]: Hough Transform is a popular technique to detect any shape if you can represent that shape in mathematical form. It is capable of detecting the shape even if it is broken or distorted a little bit.

1.1.5 CNN Layers and Feature Visualizations

Image classification is a tedious task for computers. Convolutional neural networks represent one data based approach to this challenge.

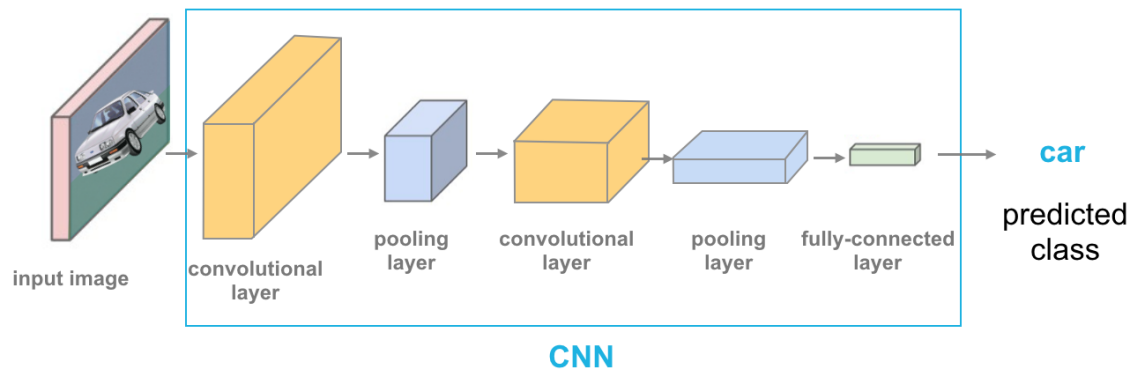


Fig 4: Standard CNN model

In the case of image classification, we want a neural network that scans an image and outputs the correct class for that image.

An image is seen by a neural network (and by computers) as a grid of numerical values. Below you will see a zoomed-in portion of a grayscale image of a car. The image is split into a fine grid and each of the grid cells is called a pixel. For grayscale images each pixel has a value between 0 and 255 where 0 is black and 255 is white; shades of grey are anywhere in between.

To create an image classifier, we need an algorithm that can look at these pixel values and classify this image. We also require a classifier to be able to detect the objects of image under varying light conditions (at night or during a sunny day) and we want the classifier to generalize well so that it can recognize a variety of cars in different environments and in different poses.

Every CNN is made up of multiple layers the three main types of layers are convolutional pooling and fully-connected as pictured below. Typically, you will see CNNs made of many layers especially convolutional and pooling layers. Each of these layers is made of nodes that look at some input data and produce an output so let us go over what each of these layers does!

The convolutional layer can be thought of as the feature extractor of this network it learns to find spatial features in an input image. This layer is produced by applying a series of many different image filters also known as convolutional kernels to an input image. These filters are exceedingly small grids of values that slide over an image pixel-by-pixel and produce a filtered output image that will be about the same size as the input image. Multiple kernels will produce multiple filtered output images.

We use a **maxpooling layer** for a few reasons.

First **dimensionality reduction**; as an input image moves forward through a CNN, we are taking a flat image in x-y space and expanding its depth dimension while decreasing its height and width. The network distills information about the content of an image and squishes it into a representation that will make up a reasonable number of inputs that can be seen by a fully-connected layer. Second maxpooling makes a network **resistant** to small pixel value changes in an input image. Imagine that some of the pixel values in a small patch are a little bit brighter or darker or that an object has moved to the right by a few pixels. For similar images even if a patch has some slightly different pixel values the maximum values extracted in successive pooling layers should be similar. Third by reducing the width and height of image data as it moves forward through the CNN the maxpooling layer **mimics an increase in the field of view** for later layers. For example a 3x3 kernel placed over an original input image will see a 3x3 pixel area at once but that same kernel applied to a pooled version of the original input image (ex. an image reduced in width and height by a factor of 2) will see the same number of pixels but the 3x3 area corresponds to a 2x larger area in the original input image. This allows later convolutional layers to detect features in a larger region of the input image.

At the end of a convolutional neural network is a **fully-connected layer** (sometimes more than one). Fully-connected means that every output that is produced at the end of the last pooling layer is an input to each node in this fully-connected layer.

CNN s is made of several layers: a series of convolutional layers + activation and maxpooling layers and at least one final fully-connected layer that can produce a set of class scores for a given image. The convolutional layers of a CNN act as feature extractors; they extract shape and color patterns from the pixel values of training images. It is important to note that the behaviour of the convolutional layers and the features they learn to extract are defined entirely by the weights that make up the convolutional kernels in the network. A CNN learns to find the best weights during training using a process called backpropagation which looks at any classification errors that a CNN makes during training finds which weights in that CNN are responsible for that error and changes those weights accordingly.

1.1.6 Recurrent and Recursive Neural Networks

The CNN architectures we have discussed before were trained using the current inputs only. We did not consider previous inputs when generating the current output. In other words, our systems did not have any memory elements. RNNs address this considerably basic and important issue by using memory (i.e. past inputs to the network) when producing the current output.

In an NLP problem if you want to predict the next word in a sentence it is important to know the words before it. RNNs are called recurrent because they perform the same task for every element of a sequence with the output being depended on the previous computations. Another way to think about RNNs is that they have a “memory” which captures information about what has been calculated so far.

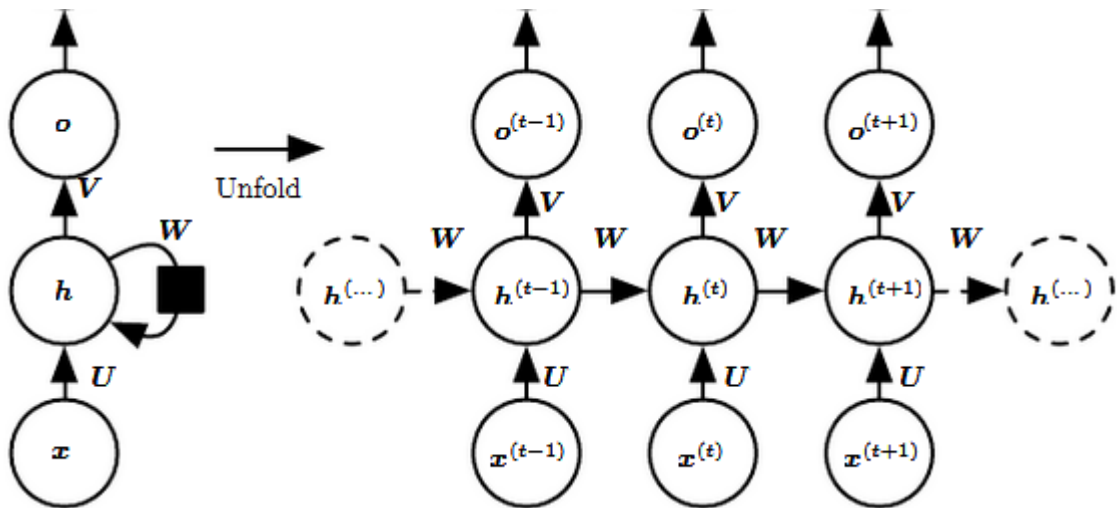


Fig 5: Basic RNN model

The left side of the above diagram shows a notation of an RNN and on the right side an RNN being unrolled (or unfolded) into a full network. By unrolling we mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 3 words the network would be unrolled into a 3-layer neural network one layer for each word.

Input: $x(t)$ is taken as the input to the network at time step t . For example, x_1 could be a one-hot vector corresponding to a word of a sentence.

Hidden state: $h(t)$ represents a hidden state at time t and acts as “memory” of the network. $h(t)$ is calculated based on the current input and the previous time step s hidden state: $h(t) = f(U x(t) + W h(t-1))$. The function f is taken to be a non-linear transformation such as tanh or ReLU.

Weights: The RNN has input to hidden connections parameterized by a weight matrix U hidden-to-hidden recurrent connections parameterized by a weight matrix W and hidden-to-

output connections parameterized by a weight matrix V and all these weights (U V W) are shared across time.

Output: $o(t)$ illustrates the output of the network. In the figure I just put an arrow after $o(t)$ which is also often subjected to non-linearity especially when the network contains further layers downstream.

Forward Pass

The figure does not clarify the choice of activation function for the hidden units. Before we proceed let us make a few assumption: 1) suppose the hyperbolic tangent activation function for hidden layer. 2) We assume that the output is discrete as if the RNN is used to predict words or characters. A natural way to show discrete variables is to take the output o as giving the un-normalized log probabilities of each probable value of the discrete variable. We can then apply the softmax operation as a post-processing step to get a vector \hat{y} of normalized probabilities over the output.

The RNN forward pass can thus be represented by below:

$$\begin{aligned} \mathbf{a}^{(t)} &= \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \\ \mathbf{h}^{(t)} &= \tanh(\mathbf{a}^{(t)}) \\ \mathbf{o}^{(t)} &= \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)} \\ \hat{\mathbf{y}}^{(t)} &= \text{softmax}(\mathbf{o}^{(t)}) \end{aligned}$$

This is an example of a recurrent network that maps an input sequence to an output sequence of equal length. The net loss for a given sequence of x values paired with a sequence of y values would then be just the sum of the losses over all the time steps. We assume that the outputs $o(t)$ are used as the argument to the softmax function to obtain the vector \hat{y} of probabilities over the output. We also assume that the loss L is the negative log-likelihood of the true target $y(t)$ given the input so far.

Backward Pass

The gradient computation involves performing a forward propagation pass moving left to right through the graph shown above followed by a backward propagation pass moving right to left through the graph. The runtime is $O(\tau)$ and cannot be reduced by parallelization because the forward propagation graph is inherently sequential; each time step may be computed only after the previous one. States computed in the forward pass must be stored until they are reused during the backward pass, so the memory cost is also $O(\tau)$. The back-propagation algorithm applied to the unrolled graph with $O(\tau)$ cost is called back-propagation through time (BPTT). Because the parameters are shared by all time steps in the network the gradient at each output depends not only on the calculations of the current time step but also the previous time steps.

Computing Gradients

Given our loss function L we need to calculate the gradients for our three weight matrices U V W and bias terms b c and update them with a learning rate α . Similar to normal back-propagation the gradient gives us a sense of how the loss is changing with respect to each weight parameter. We update the weights W to minimize loss with the following equation:

$$W \leftarrow W - \alpha \frac{\partial L}{\partial W}$$

The same is to be done for the other weight's U V b c as well.

Let us now compute the gradients by BPTT for the RNN equations above. The nodes of our computational graph include the parameters U V W b and c as well as the sequence of nodes indexed by t for $x(t)$ $h(t)$ $o(t)$ and $L(t)$. For each node n we need to compute the gradient $\nabla_n L$ recursively based on the gradient computed at nodes that follow it in the graph.

Gradient with respect to output $o(t)$ is calculated assuming the $o(t)$ are used as the argument to the softmax function to obtain the vector \hat{y} of probabilities over the output. We also assume that the loss is the negative log-likelihood of the true target $y(t)$.

$$(\nabla_{\mathbf{o}^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i=y^{(t)}}$$

Let us now understand how the gradient flows through hidden state $h(t)$. This we can clearly see from the below diagram that at time t hidden state $h(t)$ has gradient flowing from both current output and the next hidden state.

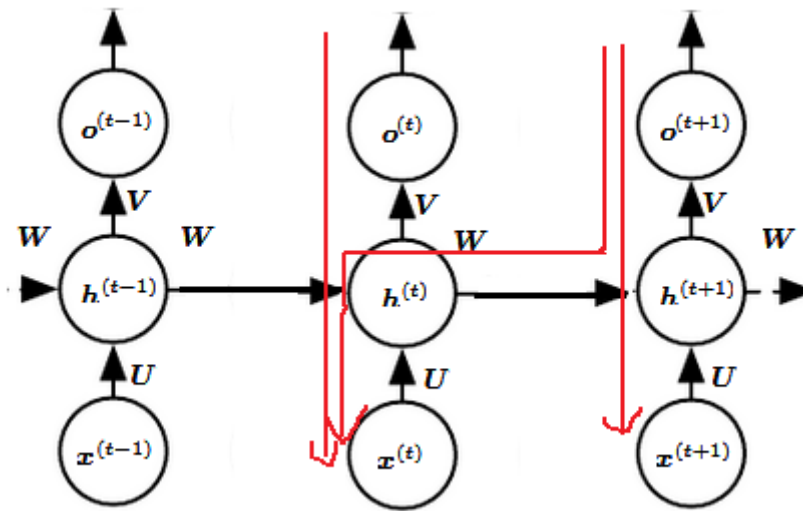


Fig 6: Red arrow shows gradient flow

We work our way backward starting from the end of the sequence. At the final timestep τ $h(\tau)$ only has $o(\tau)$ as a descendant so its gradient is simple:

$$\nabla_{h^{(\tau)}} L = V^\top \nabla_{o^{(\tau)}} L$$

We can then iterate backward in time to back-propagate gradients through time from $t=\tau-1$ down to $t=1$ noting that $h(t)$ (for $t < \tau$) has as descendants both $o(t)$ and $h(t+1)$. Its gradient is thus given by:

$$\begin{aligned}\nabla_{\mathbf{h}^{(t)}} L &= \left(\frac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}} \right)^\top (\nabla_{\mathbf{h}^{(t+1)}} L) + \left(\frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}} \right)^\top (\nabla_{\mathbf{o}^{(t)}} L) \\ &= \mathbf{W}^\top \text{diag} \left(1 - \left(\mathbf{h}^{(t+1)} \right)^2 \right) (\nabla_{\mathbf{h}^{(t+1)}} L) + \mathbf{V}^\top (\nabla_{\mathbf{o}^{(t)}} L)\end{aligned}$$

Once the gradients on the internal nodes of the computational graph are obtained, we can obtain the gradients on the parameter nodes. The gradient calculations using the chain rule for all parameters is:

$$\begin{aligned}\nabla_{\mathbf{c}} L &= \sum_t \left(\frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{c}} \right)^\top \nabla_{\mathbf{o}^{(t)}} L = \sum_t \nabla_{\mathbf{o}^{(t)}} L \\ \nabla_{\mathbf{b}} L &= \sum_t \left(\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{b}^{(t)}} \right)^\top \nabla_{\mathbf{h}^{(t)}} L = \sum_t \text{diag} \left(1 - \left(\mathbf{h}^{(t)} \right)^2 \right) \nabla_{\mathbf{h}^{(t)}} L \\ \nabla_{\mathbf{V}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}} \right) \nabla_{\mathbf{V}^{(t)}} o_i^{(t)} = \sum_t (\nabla_{\mathbf{o}^{(t)}} L) \mathbf{h}^{(t)\top} \\ \nabla_{\mathbf{W}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\mathbf{W}^{(t)}} h_i^{(t)} = \sum_t \text{diag} \left(1 - \left(\mathbf{h}^{(t)} \right)^2 \right) (\nabla_{\mathbf{h}^{(t)}} L) \mathbf{h}^{(t-1)\top} \\ \nabla_{\mathbf{U}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\mathbf{U}^{(t)}} h_i^{(t)} = \sum_t \text{diag} \left(1 - \left(\mathbf{h}^{(t)} \right)^2 \right) (\nabla_{\mathbf{h}^{(t)}} L) \mathbf{x}^{(t)\top}\end{aligned}$$

1.1.7 Long Short-Term Memory Networks

RNNs have a key flaw as capturing relationships that span more than 8 or 10 steps back is practically impossible. This flaw stems from the "vanishing gradient" problem in which the contribution of information decays geometrically over time.

While training our network we use backpropagation. In the backpropagation process we adjust our weight matrices with the use of a gradient. In the process gradients are calculated by continuous multiplications of derivatives. The value of these derivatives may be so small that these continuous multiplications may cause the gradient to practically "vanish".

So far, we have placed no constraints on this update so its knowledge can change chaotically. This chaos means information quickly transforms and vanishes and it is difficult for the model to keep a long-term memory.

LSTM is one option to overcome the Vanishing Gradient problem in RNNs. In LSTM we:

1. **Adding a forgetting mechanism.** If a scene ends for example the model should forget the current scene location the time of day and reset any scene-specific information; however if a character dies in the scene it should continue remembering that he is no longer alive. Thus, we want the model to learn a separate forgetting/remembering mechanism: when new inputs come in it needs to know which beliefs to keep or throw away.
2. **Adding a saving mechanism.** When the model sees a new image, it needs to learn whether any information about the image is worth using and saving. Maybe your mom sent you an article about the Kardashians but who cares?
3. So, when new an input comes in the model first forgets any long-term information it decides it no longer needs. Then it learns which parts of the new input are worth using and saves them into its long-term memory.
4. **Focusing long-term memory into working memory.** Finally, the model needs to learn which parts of its long-term memory are immediately useful. For example Bob's age may be a useful piece of information to keep in the long term (children are more likely to be crawling adults are more likely to be working) but is probably irrelevant if he is not in the current scene. So instead of using the full long-term memory all the time it learns which parts to focus on instead.

Whereas an RNN can overwrite its memory at each time step in a fairly uncontrolled fashion an LSTM transforms its memory in a very precise way: by using specific learning mechanisms for which pieces of information to remember which to update and which to pay attention to. This helps it keep track of information over longer periods of time.

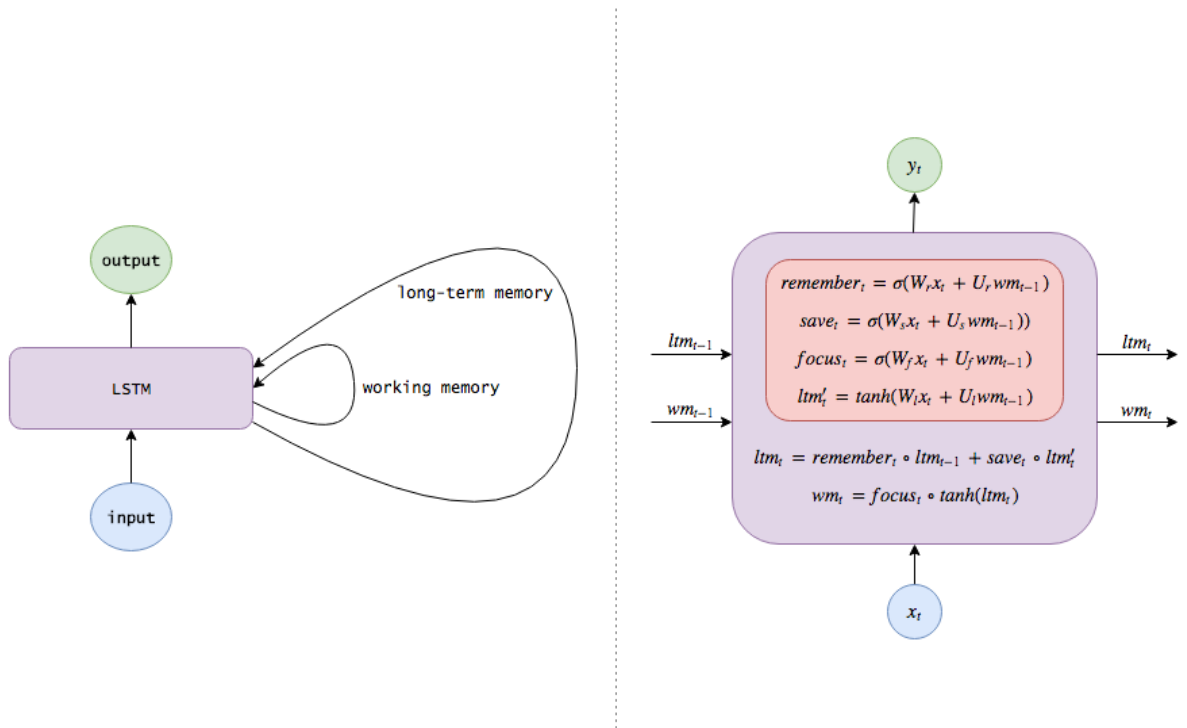


Fig 7: LSTM

1.1.8 Part-of-Speech Tagging

Speech Tagging is the process of determining the category of word from the words in the surrounding context. The Tagger reads text in some language and assigns parts of speech to each word (and other token) such as noun verb adjective etc. although generally computational applications use more fine-grained POS tags like noun-plural .

The reason we use this is because it can be implemented quickly and with high level of accuracy.

1.1.9 Word Embeddings

LSTM takes in an expected input size and hidden layer dimension. That means the input of an LSTM must be of fixed size. But this creates a problem because the normal sentences are rarely of a constant size.

So, we create an Embedding layer that takes in the size of our vocabulary and return a vector of a specified size

1.2 Problem Statement

Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph.

It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. Recently deep learning methods have achieved state-of-the-art results on examples of this problem.

Deep learning methods have demonstrated state-of-the-art results on caption generation problems. What is most impressive about these methods is a single end-to-end model can be defined to predict a caption given a photo instead of requiring sophisticated data preparation or a pipeline of specifically designed models.

The development of the image description system may help the visually impaired people “see” the world in the future. Recently it has drawn increasing attention and become one of the most important topics in computer vision. Early image description generation methods aggregate image information using static object class libraries in the image and modelled using statistical language models. Aker and Gaizauskas use a dependency model to summarize multiple web documents containing information related to image locations and propose a method for automatically tagging geotagged images. Li et al. propose a n-gram method based on network scale collecting candidate phrases and merging them to form sentences describing images from zero. Yang et al. propose a language model trained from the English Gigaword corpus to obtain the estimation of motion in the image and the probability of collocated nouns scenes and prepositions and use these estimates as parameters of the hidden Markov model. The image description is obtained by predicting the most likely nouns verbs scenes and prepositions that make up the sentence. Kulkarni et al. propose using a detector to detect objects in an image classifying each candidate region and processing it by a prepositional relationship function and finally applying a conditional random field (CRF) prediction image tag to generate a natural language description. Object detection is also performed on images. Lin et al. used a 3D visual analysis system to infer objects attributes and relationships in an image and convert them into a series of semantic trees and then learn the grammar to generate text descriptions for these trees.

Some indirect methods have also been proposed for dealing with image description problems such as the query expansion method proposed by Yagcioglu et al. by retrieving similar images from a large dataset and using the distribution described in association with the retrieved images. The expression is used to create an extended query and then the candidate descriptions are reordered by estimating the cosine between the distributed representation and the extended query vector and finally the closest description is taken as a description of the input image. In summary the methods described are brainstorming and have their own characteristics but all have the common disadvantage that they do not make intuitive feature observations on objects or actions in the image nor do they give an end-to-end mature general model to solve this problem. The efficiency and popularization of neural networks have made breakthroughs in the field of image description and saw new hopes until the advent of the era of big data and the outbreak of deep learning methods.

We review the development process of image description methods in recent years and summarize the basic framework and some improved methods. Then we analyse the advantages and shortcomings of existing models and compare their results on public large-scale datasets. Finally, we summarize some open challenges in this task.

This paper is organized as follows. The second part give a brief literature survey. The third part focuses on details of the basic models and methods. The fourth part is performance analysis. The fifth part summarizes the work and proposes the direction and expectations of future work.

1.3 Objectives

- How to prepare photo and text data for training a deep learning model.
- How to design and train a deep learning caption generation model.
- How to evaluate a train caption generation model and use it to caption entirely new photographs.

1.4 Methodology

- Analysis in the unruly statement in the system.
- Gathering the condition to run software in system
- Installation of software like ANACONDA.
- Selection of dataset for the comparison of images
- Reading previous research paper to get data of work being done.
- Photo and Caption Dataset
- Prepare Photo Data
- Prepare Text Data
- Develop Deep Learning Model
- Verification and testing phase
- Analysis of the various difficulties.
- Development in a general layout and diagrams.
- Developing analog per requirement.
- Analysis of algorithm by guide to verify errors.
- Develop Deep Learning Model
- Coding for python.
- Train with Progressive Loading (NEW)

- Evaluate Model.
- Making accuracy verifications
- Generate New Captions

2 LITERATURE SURVEY

2.1 Literature Survey

More in-depth information regarding the topics in this report can be found in the resources below.

- **Show Attend and Tell: Neural Image Caption Generation with Visual Attention**

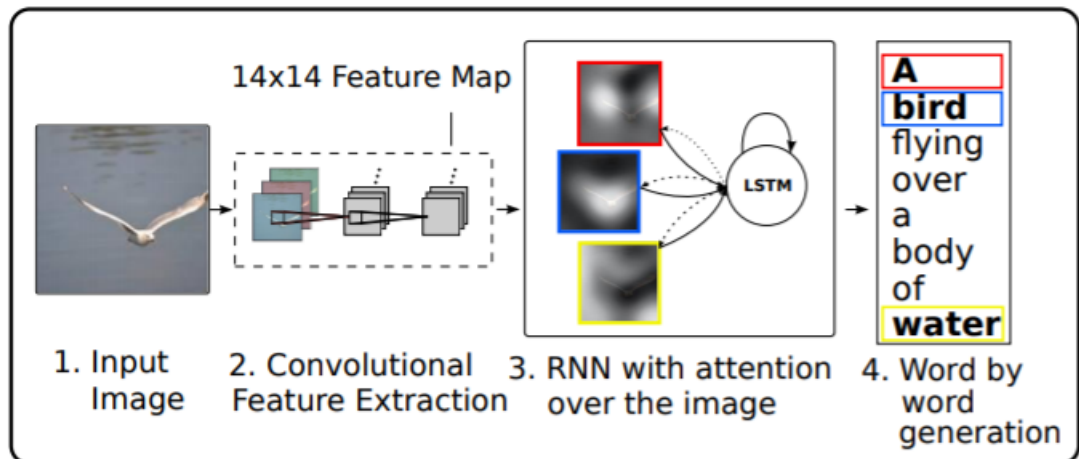
By Kelvin Xu Jimmy Ba Ryan Kiros Kyunghyun Cho Aaron Courville Ruslan Salakhutdinov Richard Zemel Yoshua Bengio

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 and stochastically by maximizing a variational lower bound. We also show through visualization how the model can automatically learn to fix its gaze on salient objects while generating the corresponding words in the output sequence. We validate the use of attention with state-of-the art performance on three benchmark datasets: Flickr8k Flickr30k and MS COCO.

Automatically generating captions of an image is a task awfully 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 as a difficult problem. It is an especially important challenge for machine learning algorithms as it amounts to mimicking the remarkable human ability to compress huge amounts of salient visual information into descriptive language. Despite the challenging nature of this task there has been a recent surge of research interest in attacking the image caption generation problem. Aided by advances in training neural networks (Krizhevsky et al. 2012) and large classification datasets (Russakovsky et al. 2014) recent work has significantly improved the quality of

caption generation using a combination of convolutional neural networks (convnets) to obtain vectorial representation of images and recurrent neural networks to decode those representations into natural language sentences.

Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in section 3.1 & 5.4



One of the most curious facets of the human visual system is the presence of attention (Rensink 2000; Corbetta & Shulman 2002). Rather than compress an entire image into a static representation attention allows for salient features to dynamically come to the forefront as needed. This is especially important when there is a lot of clutter in an image. Using representations (such as those from the top layer of a convnet) that distil information in image down to the most salient objects is one effective solution that has been widely adopted in previous work. Unfortunately, this has one potential drawback of losing information which could be useful for richer more descriptive captions. Using more low-level representation can help preserve this information. However, working with these features necessitates a powerful mechanism to steer the model to information important to the task at hand.

In this paper we describe approaches to caption generation that attempt to incorporate a form of attention with two variants: a “hard” attention mechanism and a “soft” attention mechanism. We also show how one advantage of including attention is the ability to visualize what the model “sees”. Encouraged by recent advances in caption generation and inspired by recent success in employing attention in machine translation

(Bahdanau et al. 2014) and object recognition (Ba et al. 2014; Mnih et al. 2014) we investigate models that can attend to salient part of an image while generating its caption.

- **Boosting Image Captioning with Attributes**

Ting Yao Yingwei Pan Yehao Li Zhaofan Qiu Tao Mei; The IEEE International Conference on Computer Vision (ICCV) 2017 pp. 4894-4902

Automatically describing an image with a natural language has been an emerging challenge in both fields of computer vision and natural language processing. In this paper we present Long Short-Term Memory with Attributes (LSTM-A) - a novel architecture that integrates attributes into the successful Convolutional Neural Networks (CNNs) plus Recurrent Neural Networks (RNNs) image captioning framework by training them in an end-to-end manner. Particularly the learning of attributes is strengthened by integrating inter-attribute correlations into Multiple Instance Learning (MIL). To incorporate attributes into captioning we construct variants of architectures by feeding image representations and attributes into RNNs in different ways to explore the mutual but also fuzzy relationship between them. Extensive experiments are conducted on COCO image captioning dataset and our framework shows clear improvements when compared to state-of-the-art deep models. More remarkably we obtain METEOR/CIDEr-D of 25.5%/100.2% on testing data of widely used and publicly available splits in [10] when extracting image representations by GoogleNet and achieve superior performance on COCO captioning Leaderboard.

- **Image Captioning with Semantic Attention**

By Kelvin Xu Jimmy Ba Ryan Kiros Kyunghyun Cho Aaron Courville Ruslan Salakhutdinov Richard Zemel Yoshua Bengio

Automatically generating a natural language description of an image has attracted interests recently both because of its importance in practical applications and because it connects two major artificial intelligence fields: computer vision and natural language processing. Existing approaches are either top-down which start from a gist of an image and convert it into words or bottom-up which come up with words describing various

aspects of an image and then combine them. In this paper we propose a new algorithm that combines both approaches through a model of semantic attention. Our algorithm learns to selectively attend to semantic concept proposals and fuse them into hidden states and outputs of recurrent neural networks. The selection and fusion form a feedback connecting the top-down and bottom-up computation. We evaluate our algorithm on two public benchmarks: Microsoft COCO and Flickr30K. Experimental results show that our algorithm significantly outperforms the state-of-the-art approaches consistently across different evaluation metrics.

- **Knowing When to Look: Adaptive Attention via a Visual Sentinel for Image Captioning**

By Kelvin Xu Jimmy Ba Ryan Kiros Kyunghyun Cho Aaron Courville Ruslan Salakhutdinov Richard Zemel Yoshua Bengio

Attention-based neural encoder-decoder frameworks have been widely adopted for image captioning. Most methods force visual attention to be active for every generated word. However, the decoder likely requires little to no visual information from the image to predict non-visual words such as "the" and "of". Other words that may seem visual can often be predicted reliably just from the language model e.g. "sign" after "behind a red stop" or "phone" following "talking on a cell". In this paper we propose a novel adaptive attention model with a visual sentinel. At each time step our model decides whether to attend to the image (and if so to which regions) or to the visual sentinel. The model decides whether to attend to the image and where in order to extract meaningful information for sequential word generation. We test our method on the COCO image captioning 2015 challenge dataset and Flickr30K. Our approach sets the new state-of-the-art by a significant margin.

- **Convolutional Image Captioning**

By Kelvin Xu Jimmy Ba Ryan Kiros Kyunghyun Cho Aaron Courville Ruslan Salakhutdinov Richard Zemel Yoshua Bengio

Image captioning is an important task applicable to virtual assistants editing tools image indexing and support of the disabled. In recent years significant progress has been made in image captioning using Recurrent Neural Networks powered by long-short term-memory (LSTM) units. Despite mitigating the vanishing gradient problem and despite their compelling ability to memorize dependencies LSTM units are complex and inherently sequential across time. To address this issue recent work has shown benefits of convolutional networks for machine translation and conditional image generation. Inspired by their success in this paper we develop a convolutional image captioning technique. We demonstrate its efficacy on the challenging MSCOCO dataset and demonstrate performance on par with the LSTM baseline while having a faster training time per number of parameters. We also perform a detailed analysis providing compelling reasons in favour of convolutional language generation approaches.

- **Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge**

By Oriol Vinyals ; Alexander Toshev ; Samy Bengio ; Dumitru Erhan

Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. In this paper 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 datasets show the accuracy of the model and the fluency of the language it learns solely from image descriptions. Our model is often quite accurate which we verify both qualitatively and quantitatively. Finally given the recent surge of interest in this task a

competition was organized in 2015 using the newly released COCO dataset. We describe and analyse the various improvements we applied to our own baseline and show the resulting performance in the competition which we won ex-aequo with a team from Microsoft Research.

- **Show-and-Fool: Crafting Adversarial Examples for Neural Image Captioning**

Hongge Chen^{1*} *Huan Zhang*^{2*†} *Pin-Yu Chen*³ *Jinfeng Yi*⁴ *Cho-Jui Hsieh*²
*1*Massachusetts Institute of Technology Cambridge MA 02139 *2*University of California Davis Davis CA 95616 *3*IBM T.J. Watson Research Center Yorktown Heights NY 10598 *4*Tencent AI Lab Bellevue WA 98004

Modern neural image captioning systems typically adopt the encoder-decoder framework consisting of two principal components: a convolutional neural network (CNN) for image feature extraction and a recurrent neural network (RNN) for caption generation. Inspired by the robustness analysis of CNN-based image classifiers to adversarial perturbations we propose Show-and-Fool a novel algorithm for crafting adversarial examples in neural image captioning. Unlike image classification tasks with a finite set of class labels finding visually similar adversarial examples in an image captioning system is much more challenging since the space of possible captions in a captioning system is almost infinite. In this paper we design three approaches for crafting adversarial examples in image captioning: (i) targeted caption method; (ii) targeted keyword method; and (iii) untargeted method. We formulate the process of finding adversarial perturbations as optimization problems and design novel loss functions for efficient search. Experimental results on the Show-and-Tell model and MSCOCO data set show that Show-and-Fool can successfully craft visually similar adversarial examples with randomly targeted captions and the adversarial examples can be made highly transferable to the Show-Attend-and-Tell model. Consequently the presence of adversarial examples leads to new robustness implications of neural image captioning. To the best of our knowledge this is the first work on crafting effective adversarial examples for image captioning tasks.

3 SYSTEM DEVELOPMENT

3.1 Pre-requisite Tools

3.1.1 Anaconda

It is a free and open source dissemination of the Python and R programming dialects for information science and AI related applications this application is utilized for different programming strategies as a stage for making calculations for the most part it is utilized for AI nowadays by utilizing python language. The framework is managed by conda. It is utilized by more than 6 million clients everywhere throughout the world for nothing of cost for advancement purposes and programming and it incorporates in excess of 250 mainstream bundles which are reasonable for different stages (Operating frameworks for example Windows Linux and MacOS. Highlights of Spyder are:

3.1.2 Spyder

Spyder (likewise called as Pydee) is an open source cross-stage coordinated improvement condition utilized for logical programming in generally Python. Spyder itself comprises of furthermore virtual products. It is was discharged under the permits of international colleges. It is utilized with numerous modules it incorporates support for intelligent apparatuses for information assessments and inserts Python codes for quality affirmation and self-assessment devices for example Pyflakes Pylint and Rope. It is accessible cross-stage through Anaconda on Windows with WinPython and Python (x y) on macOS by MacPorts and on significant Linux conveyances for example Arch Linux Debian Fedora Gentoo Linux openSUSE and Ubuntu.

- editor with language structure pressure and self-assessment for code end
- It supports for complex Python solaces (counting IPython)
- It has bent to find and alter factors from a UI

Accessible modules in sypder are:

- It can do examination for a static code with Pylint
- It can do code profiling

- It likewise has Conda Package Manager
- Hardware requirements(recommended)

3.2 Model Development

3.2.1 Image Captioning

CNN: Used for image classification and object localization.

RNN: Used to generate text based on previous text or we can say they learn from sequential data.

In image Captioning we join these two networks together to create a model that takes in an image as an input and output a sequence of text that describe that image.

3.2.2 CNN-RNN model

We want our model to take an image as an input and output a text description of it. The input image will be processed by a CNN. Then we will connect the output of a CNN to the input of RNN which will allow us to generate descriptive text.

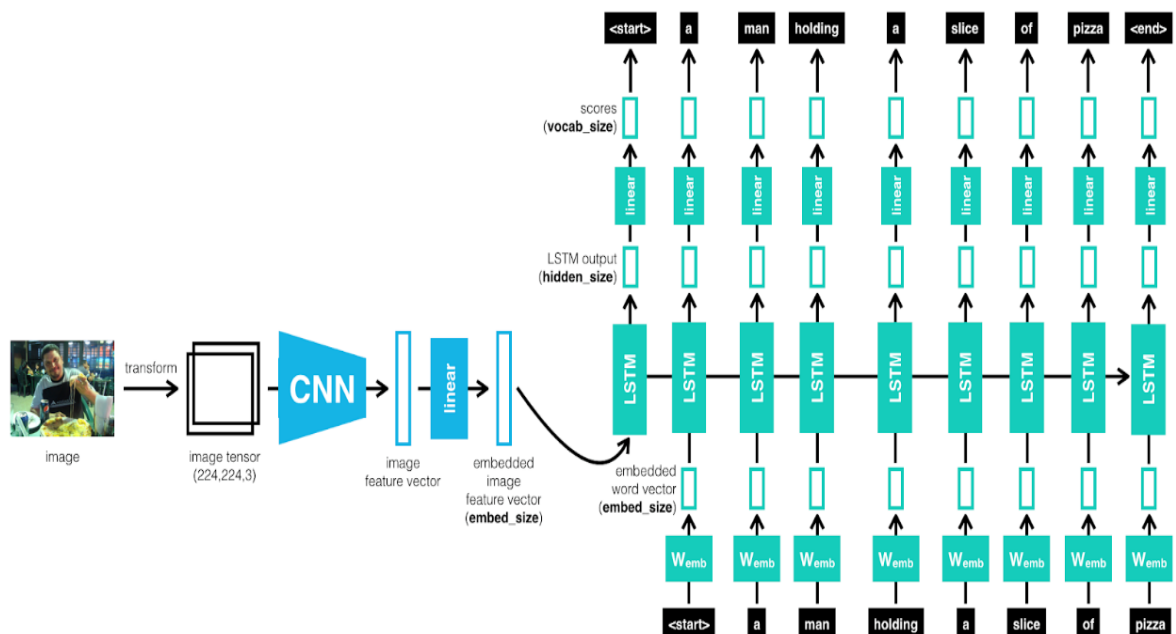


Fig 8: CNN-RNN model

3.2.3 DataSet

We will be using MS COCO dataset^[9] which stands for MicroSoft Common Objects in Context. COCO is a large-scale object detection segmentation and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labelled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250 000 people with keypoints

3.2.4 Learning Rate

Learning rate defines how much parameters should change in each iteration. In other words, it controls how fast or slow we should converge to minimum. On one hand small learning rate can take iterations to converge a large learning rate can overshoot minimum as you can see in the figure above.

3.2.5 Optimizers

Selecting a good optimization algorithm is very essential while training a model. The choice of optimization algorithm can make a difference between getting a good accuracy in hours or days. Famous optimization algorithms:

- **Stochastic Gradient Descent:** Gradient descent starts with calculating gradients (derivatives) for each of the parameter w.r.t cost function. Those gradients give us numerical adjustment we need to make to each parameter to minimize the cost function. This process continues until we hit the local/global minimum. As gradient will be zero at local minimum our gradient descent would report it as minimum value when global minimum is somewhere else.
- **Stochastic gradient descent with momentum:** Here we tend to reduce the oscillations in more sensitive direction and hence make it converge faster.
- **AdaGrad Optimization:** For every parameter we store the sum of squares of all its historical gradients. This sum is later used to scale the learning rate
- **RMSProp Optimization:** Like AdaGrad we will keep the estimate of squared gradient but instead of letting that squared estimate accumulate over training we rather let that estimate decay gradually. To do this we multiply the current estimate of squared gradients with the decay rate.
- **Adam:** This algorithm calculates an exponential moving average of gradients and the squared gradients whereas parameters β_1 and β_2 controls the decay rates of these moving averages. This keeps all the nice features of RMSProp and Gradient descent with momentum.

3.2.6 Underfitting and Overfitting

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model.

Underfitting refers to a model that can neither model the training data nor generalize to new data. Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning.

3.2.7 Training Phase

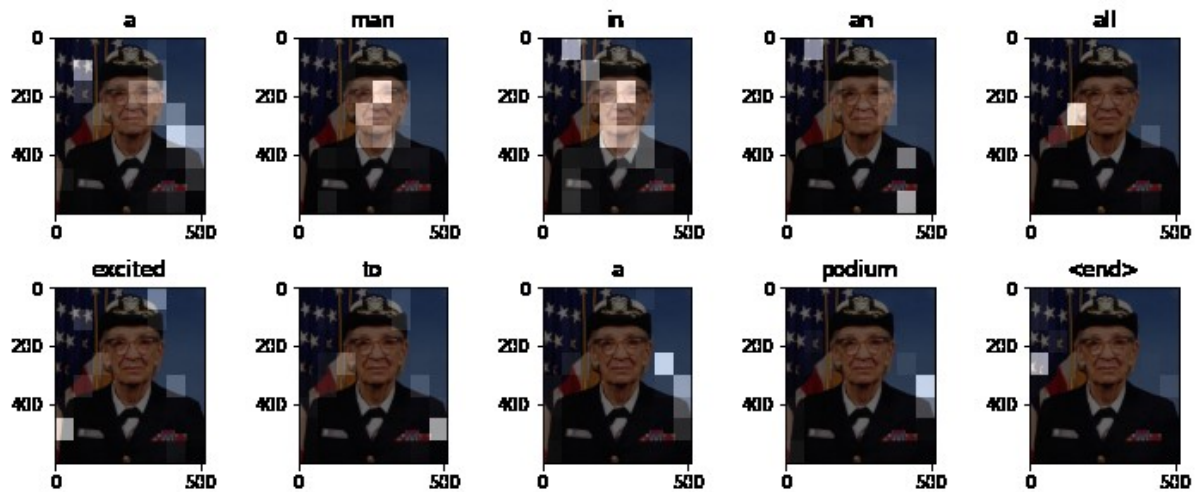
To see how that works let us look at an example. We have a training image and the associated caption. Our goal is to train the model so that it can generate the caption given the image as an input. First, we feed this image to a CNN. We can use a pretrained network like VGG or Resnet. At the end of network, we have a softmax classifier that outputs a vector of class scores but we do not want to classify the image. Instead we want a set of features that represent special content in that image. To get that kind of features we remove the final fully connected layer that classifies the image and look at the earlier layer that distill the spatial information in the image. Now we are using the CNN model as a feature extractor that compresses the huge amount of information contained in the original image into a smaller representation. This CNN is often called an Encoder because it encodes the content of an image into a smaller feature vector. Then we can process this feature vector and use it as the initial input for the following RNN.

The Decoder that is the second half of our network will be made up of LSTM cells which are good at remembering lengthy sequence of words. Each LSTM cell is expecting to see the same shape of input vector at each time step. The very first cell is connected to the output feature vector of the CNN encoder. The input to the RNN for all future time steps will be the individual words of the training caption.

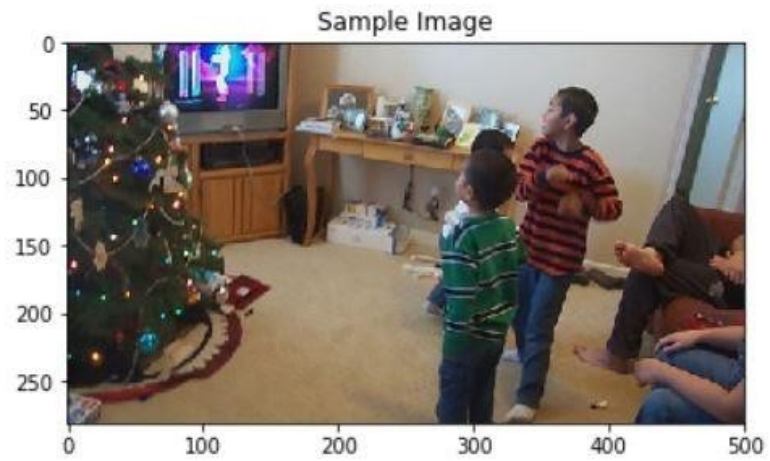
4 PERFORMANCE ANALYSIS

4.1 Input Images and Output Text

We selected some random images and fed it to our trained model. Since we used transfer Learning the CNN part of the model was already quite accurate and we know for an RNN decoder we do not need to add too much layers.



Results after second epoch:



a group of people standing around a table .

(a)



a man riding a wave on top of a surfboard .

(b)



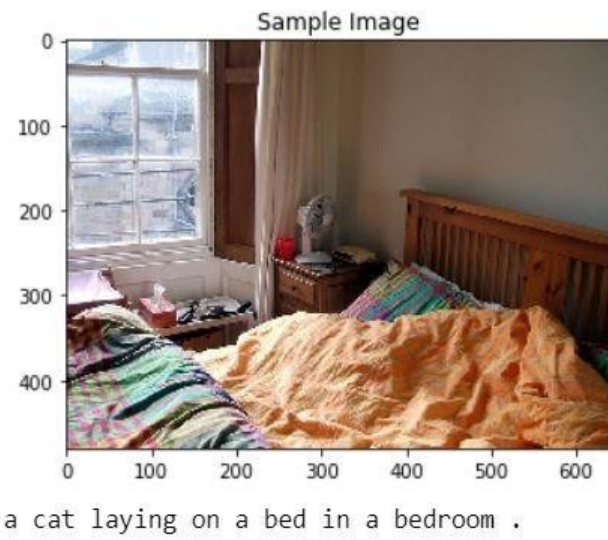
a blender filled with food on a table .

(c)



a cake with a piece of chocolate on a plate .

(d)






(e)

Fig 9: Images (a) (b) (c) (d) and (e) are some random images fed into the model after second epoch

Result obtained after training the model for third and final epoch. Since there were no need to train the model any further and increase the chances of over-fitting we stopped here.

Table 2: Output Table

Image	Caption
	Generated Caption: a person riding a surfboard on a wave
	Generated Caption: a group of people riding motorcycles down a street
	Generated Caption: a young boy brushing his teeth with a toothbrush



Generated Caption: a vase with a flower on a table

5 CONCLUSIONS

5.1 Conclusions

If the training loss is much lower than validation loss you are overfitting. If the training loss and validation loss closer, you are underfitting your model.

Adam optimizer works better than SGD because it converges faster, and it is perfect in this situation.

The main motive of this research work is to develop a deep learning architecture which can take an Image as an Input and produce text as an output. Accuracy and Performance was not the point of focus here because these two factors can be improved with

- Training the model on more diverse data
- Tweaking the hyperparameters to squeeze out more accuracy.

This project just has demonstrated that the model has learned something when you generate captions on the test data. The model was able to generate adequate caption for the Image.

General Outcome

If the training loss is much lower than validation loss you are overfitting. If the training loss and validation loss closer, you are underfitting your model. Best strategy is to try a large network with different dropout values and select the model which has the best validation performance.

5.2 Future Scope

Image Captioning can be used in a variety of different applications. For example:

- Image Caption can be used to describe images to the people who are blind or have low vision and who rely on sound and text to describe a scene
- In web development it is good practice to provide a description to any image that appears on the page so that image can be read or heard as opposed to just seen. This makes web content accessible
- Caption can be used to describe video in real time.

REFERENCE

1. B. Chitradevi and P. Srimanthi, "An Overview on Image Processing Techniques," ISRN Signal Process., vol. 2, no. 11, pp. 6466–6472, 2014
2. rsipvision.com , "DEFINING THE BORDERS WITHIN COMPUTER VISION,". Retrieved from <https://www.rsipvision.com/defining-borders/>
3. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html
4. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html
5. I. goodfello, Y Bengio, and A Courville, "Deep Learning," ISBN 9780262035613 vol. 1, page no. 321-359
6. <http://www.deeplearningbook.org/contents/rnn.html>
7. <http://blog.echen.me/2017/05/30/exploring-lstms/>
8. S. Hochreiter & J. Schmidhuber, "LONG SHORT-TERM MEMORY", Neural Computation 9(8):1735-1780, 1997
9. MS COCO dataset <http://cocodataset.org/#home>
10. Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.
11. Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of HLT-NAACL 2003, pp. 252-259.
12. P. Dillon, D. Lewis and F. Kaspar, Color Imaging System using a Single CCD Area Array, IEEE Transactions on Electron Devices, Vol. ED-25, No. 2 (Feb 1978)102-107.
13. J.J. Gibson (1950), The Perception of the Visual World, Houghton-Mifflin, Boston.
14. E. Hecht and A. Lajac (1974), Optics, Addison-Wesley.
15. R. Haralick and L. Shapiro (1992), Computer and Robot Vision, Volumes I and II, Addison-Wesley.
16. M.D. Levine (1985), Vision in Man and Machine, McGraw-Hill.
17. M. Livingstone, (1988) Art, Illusion and the Visual system, Scientific American, Jan. 1988, 78-85.
18. J. Murray and W. VanRyper (1994), Encyclopedia of Graphics File Formats, O'Reilly and Associates, Inc., 103 Morris St., Suite A, Sebastopol, CA 95472.
19. V. Nalwa (1993), A Guided Tour of Computer Vision, Addison-Wesley.

20. R.J. Schalko (1989), Digital Image Processing and Computer Vision, John Wiley and Sons.

.Project_Report_161384

ORIGINALITY REPORT

12% SIMILARITY INDEX	12% INTERNET SOURCES	0% PUBLICATIONS	0% STUDENT PAPERS
--------------------------------	--------------------------------	---------------------------	-----------------------------

PRIMARY SOURCES

1	towardsdatascience.com	12%
	Internet Source	

Exclude quotes	<input type="checkbox"/> Off	Exclude matches	<input type="checkbox"/> < 800 words
Exclude bibliography	<input type="checkbox"/> Off		

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 16/07/2020

Type of Document (Tick): ~~PhD Thesis~~ ~~M.Tech Dissertation/ Report~~ B.Tech Project Report ~~Paper~~

Name: Gulshan Sankhyan Department: CSE Enrolment No 161384
Contact No. 9459591070 E-mail. g.sankhyan1357@gmail.com
Name of the Supervisor: Dr. Himanshu Jindal
Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): Image Captioning Using Deep Learning Techniques

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 45
- Total No. of Preliminary pages = 8
- Total No. of pages accommodate bibliography/references = 53



(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at¹².....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.



(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none">• All Preliminary		Word Counts	

