

Image Captioning Using Deep Learning Techniques

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

Bachelor of Technology

In

Computer Science and Engineering/Information Technology

By

Naveen Sharma (161372)

Under the supervision of

Mr. Vikas Kumar

To




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

CANDIDATE'S DECLARATION

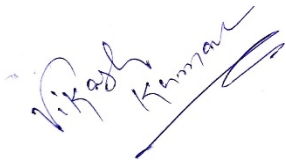
We hereby declare that the work presented in this report entitled “**Image Captioning Using Deep Learning Techniques**” in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of our own work carried out over a period from February 2020 to May 2020 under the supervision of **Mr. Vikas Kumar** (Senior Technical Lead and Manager, Thales Group).

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

(Student Signature) 

Naveen Sharma (161372)

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



(Supervisor Signature)

Mr. Vikas Kumar

(Senior Technical Lead and Manger)

ABSTRACT

In today's digital world an image is defined as digital image. There are various techniques that can be applied on images, among them image dehazing is one of the important technique. This research first of all studied the various processes that can be applied on the images namely: Compression, Watermarking, Contrast Equalization, De-hazing and Enhancement. Image dehazing is taken up for consideration that uses single dehazing approaches like Independent Component Analysis, Dark Channel Prior, and contrast specific; among fast dehazing techniques like Tan's, Fattal and Dark Channel Prior methods. The second step in the research moved to the case study of DCP and fusion techniques along with the advantage and disadvantages of each and every technique of fusion. In this thesis, a strategy has been proposed using Dark Channel Prior and Fusion. DCP is a technique to remove haze from an image using dark channel prior. The dark channel prior is a type of statistics to generate outdoor haze free images. It emphasizes on the key observation that outdoor images have some pixels with very low intensity. This implies that pixels are very dark and difficult to see with human eye. Technique helps to estimate the thickness of the haze and generate a good quality haze free image. This technique produces high quality depth map that gives the good estimation of transmission. The drawback of DCP is that when objects are similar to atmospheric light and no shadow is formed then this technique generates radiance with lower intensity. After performing DCP, different fusion techniques have been applied.

ACKNOWLEDGEMENT

I take upon this opportunity endowed upon me, to thank all those who have been part of this endeavor. I would like to thank my supervisor 'Mr. Vikas Kumar', Senior Technical Lead and Manager for giving me the right direction to follow and proper guidance regarding the topic. Their active involvement and the right guidance made the project a possibility. Last but not the least, I heartily appreciate all those people who have been involved directly or indirectly in making this project 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	1
ABSTRACT	2
ABBREVIATIONS	5
LIST OF FIGURES	7
LIST OF Tables	8
1. Introduction	9
1.1 Introduction	9
2. Introduction to Computer Vision	15
2.1 What is Computer Vision?	15
2.2 Convolution Filters and Edge detectors	Error! Bookmark not defined.
2.3 CNN Layers and Feature Visualizations[5]	Error! Bookmark not defined.
2.4 Recurrent and Recursive Neural Networks	17
2.5 Long Short Term Memory Networks[8]	24
2.6 Part-of-Speech Tagging	26
2.7 Word Embeddings	Error! Bookmark not defined.
3. Image Captioning (Model design)	26
3.1 Image Captioning	28
3.2 CNN-RNN model	Error! Bookmark not defined.
3.3 DataSet	Error! Bookmark not defined.
3.4 Learning Rate	Error! Bookmark not defined.
3.5 Optimizers	Error! Bookmark not defined.
3.6 Underfitting and Overfitting	30
3.7 Training Phase	Error! Bookmark not defined.
4. Performance Analysis	Error! Bookmark not defined.
4.1 Input Images and Output Text	32
4.2 Observations	34
Results	35
Future Scope	36

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

Figure Description	Page No.
Fig 1: Head movement vestibulogram signal captured by low noise camera	11
Fig 2: Edge detection in image processing software	12
Fig 3: Venn diagram showing comparison between ML, CV, IP and SP	14
Fig 4: Standard CNN model	16
Fig 5: Basic RNN model	19
Fig 6: Red arrow shows gradient flow	23
Fig 7: LSTM	26
Fig 8: CNN-RNN model	28
Fig 9: Images (a), (b), (c), (d), (e) and (f) are some random images fed into the model	23-34

LIST OF Tables

Table Description	Page No
Table 1: Summarizes the comparison between CV, ML and IP	14

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 of which parts have a particular value at a particular location. These elements referred to as picture elements, image elements, and pixels. A pixel is most broadly used to denote the components of a digital image.

1.1.1 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.

1.1.2 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:

1. The image is imported using image acquisition tool
2. The image is then analyzed and then manipulated
3. 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 main advantage of Digital Image Processing techniques is its repeatability, versatility and the preservation of original data precision. The various Image Processing techniques are:

- **Image preprocessing**
- **Image enhancement**
- **Image segmentation**
- **Image restoration**
- **Image transformation**
- **Feature extraction**

1.1.3 Computer Vision, Image Processing and Machine Learning

Computer vision, image processing, signal processing, machine learning – we've heard the terms but what's 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. Here we'll 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's 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 favor 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.

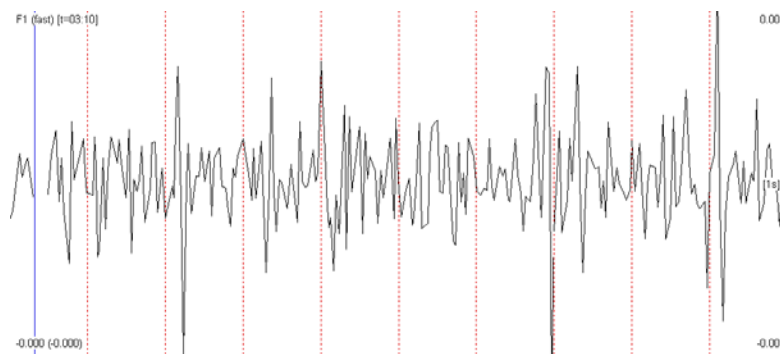


Fig 1: Head movement vestibular gram signal captured by low noise camera

Input of 1 kind will generally be reworked to a different. as an example, ultrasound pictures are recorded victimization 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 really as image process.

Let's verify the x-ray as a prototypic example. Let's assume we've 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 also 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 very similar to the method of visual reasoning of human

vision; we are able to 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 amount 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, very similar to it's 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.

The following table 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

Table 1

This can also be presented in a Venn diagram:

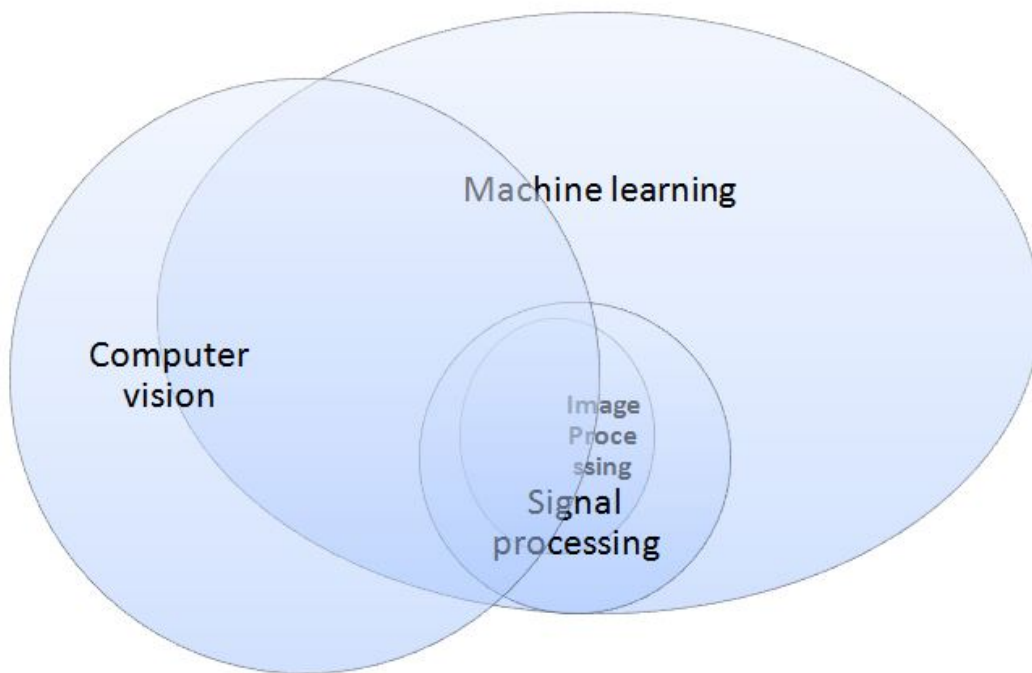


Fig 3: Venn diagram showing comparison between ML, CV, IP and SP

2. Introduction to Computer Vision

2.1 What is Computer Vision?

Computer Vision is how machines like Smartphones or Robotic Systems visually perceives the world and respond to it. It is modeled 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.

2.2 Convolution Filters and Edge detectors

- Canny Edge Detector: Canny Edge Detection is a very popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-step algorithm and we will go through each step:
 - Noise Reduction
 - Finding Intensity Gradient of the Image
 - Non-maximum Suppression
 - Hysteresis Thresholding
- Hough Line Transform: 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.

2.3 CNN Layers and Feature Visualizations

Image classification is a challenging task for computers. Convolutional neural networks represent single data-driven approach to this task. It will be about image representation and the layers that make up CNN.

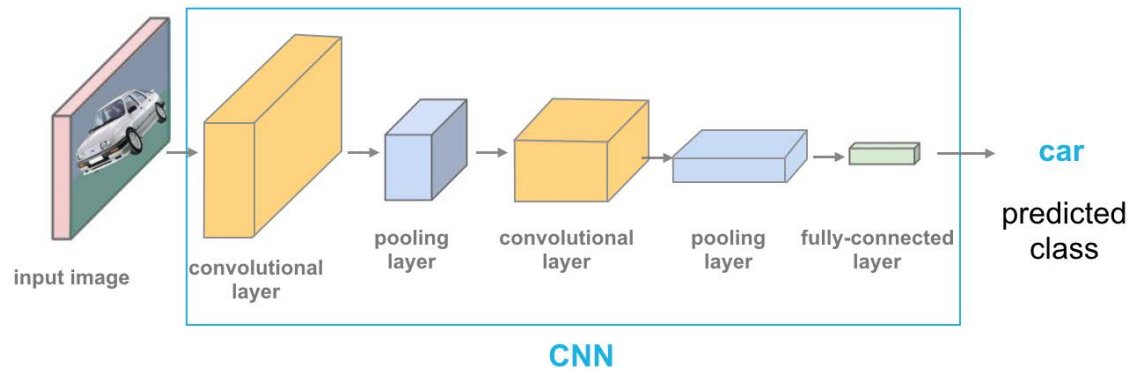


Fig 4: Standard CNN model

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

The input is seen by a neural network (and by computers) as a grid consisting of numerical values. Below, you'll see a zoomed-in portion of a grayscale image of a car. The image is disintegrated 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 gray lie in between.

To create an image classifier, we need an algorithm that check these pixel values and classify this image as a car. We also want a classifier to be able to detect this car under varying light conditions (at night or on 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 angles.

Every CNN is made up of many layers, the three main types of layers are convolutional, pooling and fully-connected, as pictured below. Typically, CNN's made of many layers, especially convolutional and pooling layers. Each of them is made of nodes that look at some input data and produce an output, so let's go over what each of these layers do.

The convolutional layer can be seen as the feature extractor of this network, it teaches itself to find spatial features in an input image. This layer is produced by applying a series of many image filters, also known as convolutional kernels, to an input image. These filters are very 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 various reasons.

Firstly, **dimensionality reduction**: as an input image moves forward through a CNN, we are taking a generally 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. Secondly, maxpooling makes a network resistant to small pixel value changes in an input image. Imagine that some of the pixel values in a fairly 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 same. Thirdly, 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 instance, 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 permits later convolutional layers to detect features in a larger region of the input image.

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

CNN's are made of different 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 set images. It's important to note that the behavior 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 cause for that error, and changes those weights accordingly.

2.4 Recurrent and Recursive Neural Networks

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

In a NLP problem, if you want to predict the next word in a sentence it is vital to know the words before it. RNNs are known as recurrent because they perform the same task for each element of a sequence, with the output being depended on the previous combinations. Another

way to think about RNNs is that they have a “memory” which stores information about what has been calculated so far.

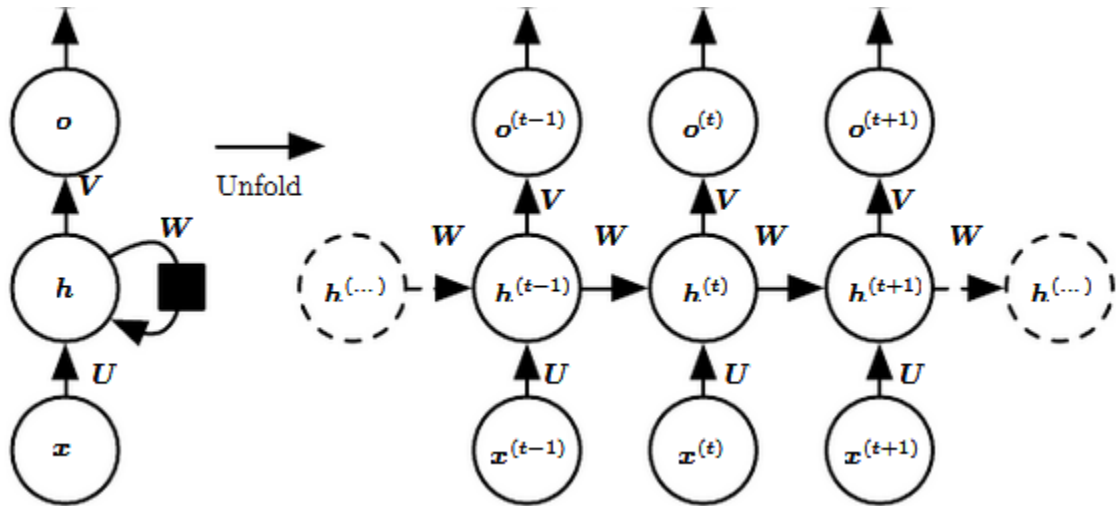


Fig 5: Basic RNN model^[6]

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 instance, if the sequence we consider 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 , 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 distributed 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 assumptions: 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 weights 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 $y(t)$.

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

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.

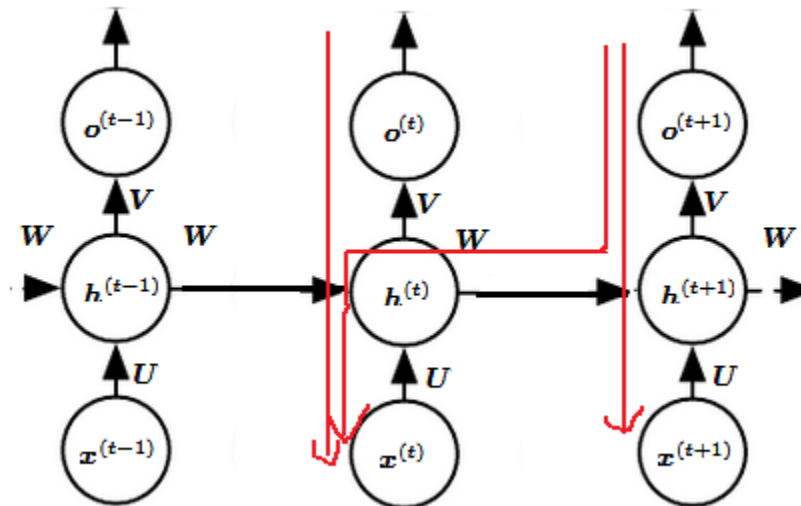


Fig 6: Red arrow shows gradient flow

We work our way backward, starting from the end of the sequence. At the last time step τ , $\mathbf{h}(\tau)$ only has $\mathbf{o}(\tau)$ as a descendant, so its gradient is simple:

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

We will then iterate backward in time to back-propagate gradients through time, from $t = \tau - 1$ down to $t = 1$, noting that $\mathbf{h}(t)$ (for $t < \tau$) has as descendants both $\mathbf{o}(t)$ and $\mathbf{h}(t+1)$. 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 extracted, we can obtain the gradients on the parameter nodes. The gradient calculations using the chain rule for all parameters is:

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

2.5 Long Short Term Memory Networks

RNNs have a key flaw, as capturing relationships that span more than 8 or 10 steps back is practically not feasible. This problem stems from the "vanishing gradient" problem in which the contribution of information decays geometrically over time.

While training our network we make use of backpropagation. In the backpropagation process we adjust the weight matrices with the use of a gradient. In the process, gradients are evaluated by continuous multiplications of derivatives. The value of these derivatives can be so small, that these continuous multiplications may result 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 disappears, and it's 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 instance, the model need not remember 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's dead. Therefore, 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 discard.

2. **Adding a saving mechanism.** When the model gets 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 sports, but who cares?

3. So when new input comes in, the model first forgets any long-term information it decides it no longer requires. Then it learns which parts of the latest 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 instantly useful. For example, Penny'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 not useful if he's not in the current scene. So instead of using the full long-term memory every time, it learns which parts to focus on instead.

Whereas an RNN can overwrite its memory at every time step in an uncontrolled fashion, an LSTM transforms its memory in a very accurate way: by using particular learning mechanisms for which pieces of information to remember and which to update, and which to pay attention to. This assist it in keeping track of information over longer periods of time.

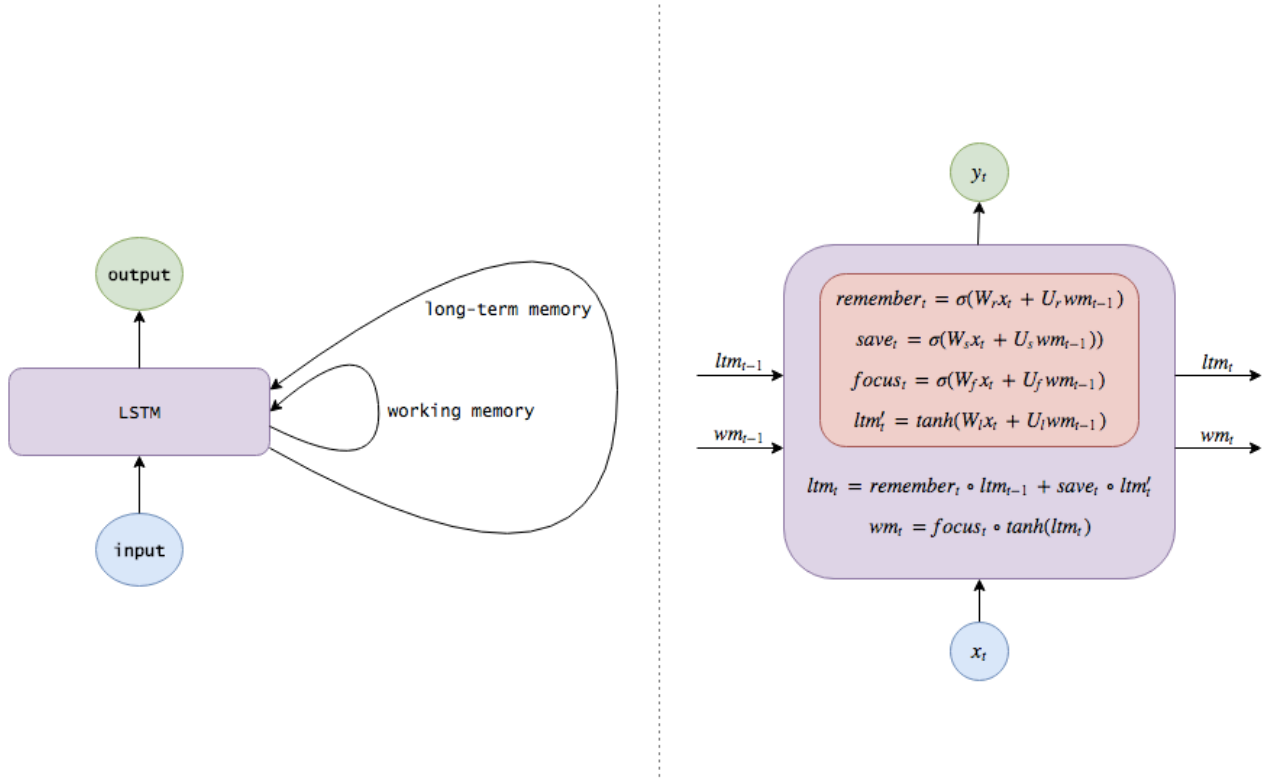


Fig 7: LSTM^[7]

2.6 Part-of-Speech Tagging^{[10][11]}

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

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

2.7 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.

3. Image Captioning (Model design)

3.1 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 CNN-RNN model

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

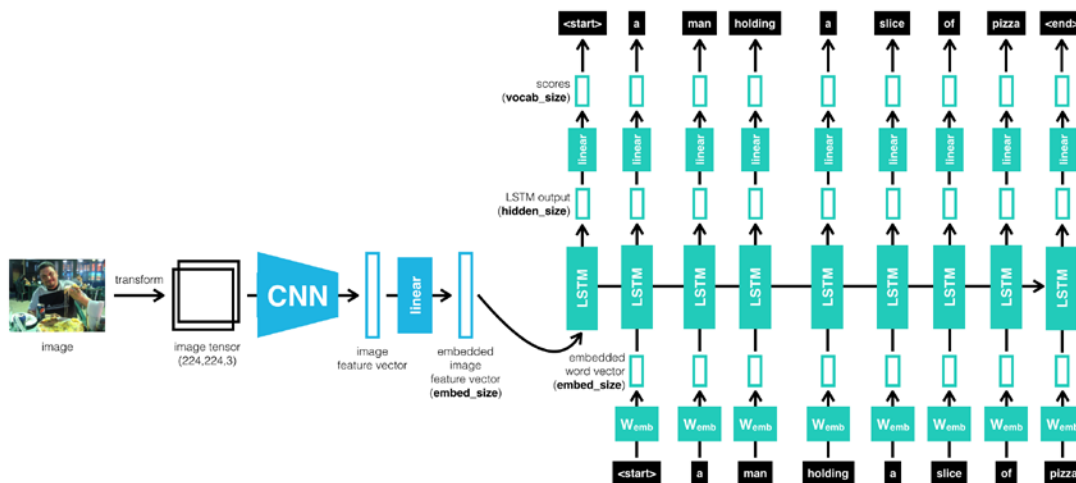


Fig 8: CNN-RNN model

3.3 DataSet

We will be using MS COCO dataset[9] which stands for MicroSoft Common Objects in Context.

COCO is a large-scale object detection, captioning and segmentation dataset. COCO has several features:

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

3.4 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.

3.5 Optimizers

Selecting a good optimization algorithm is very essential while training a model. The choice of optimization algorithm can actually bring 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 gives us numerical adjustment we need to make to each parameter so as to minimize the cost function. This process continues until we hit the local/global minimum. When gradient will be zero at local minimum our gradient descent will 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 utilized later 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.

3.6 Underfitting and Overfitting

Over fitting occurs when a model learns the detail and noise in the training dataset to the extent that it negatively impacts the results of the model on new data. This means that the noise or random fluctuations in the training data is considered and learned as concepts by the model.

Under fitting occurs 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 particular examples not seen by the model when it was learning.

3.7 Training Phase

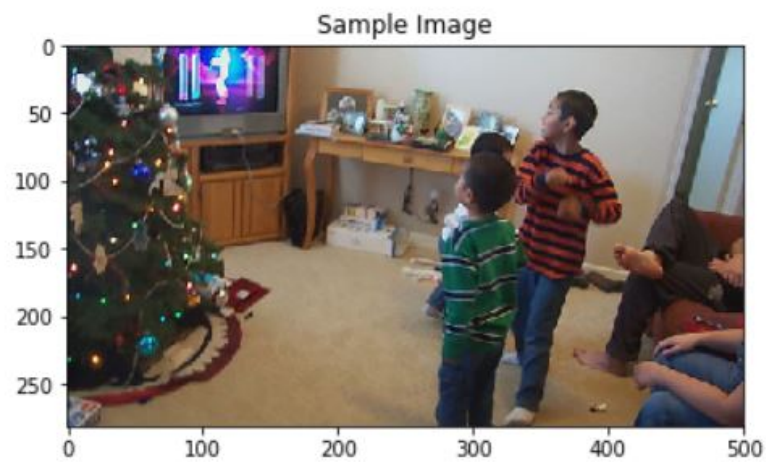
To see how that works let us look at a particular example. We have a training image and the associated caption. Our goal is to train the model so that it can generate the caption given the image as an input. First we feed this image to a CNN. We can use a retrained network like VGG or Resnet. At the end of network we have a softmax classifier that outputs a vector of class scores but we don't 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 classify 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 encode 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 quit accurate and we know for an RNN decoder we don't need to add too much layers.



a group of people standing around a table .

(a)



a man riding a wave on top of a surfboard .

(b)



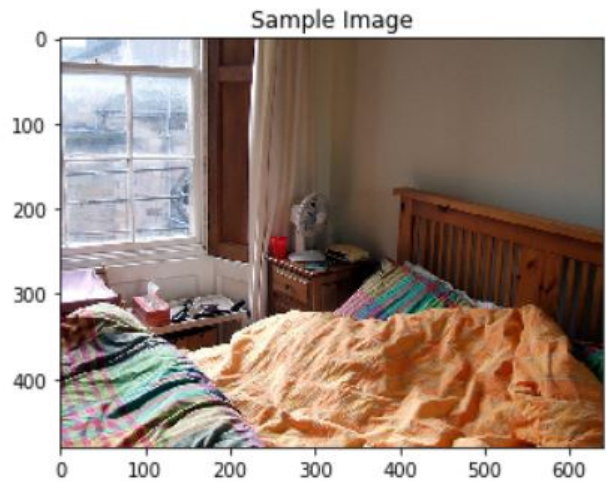
a blender filled with food on a table .

(c)



a cake with a piece of chocolate on a plate .

(d)



a cat laying on a bed in a bedroom .

(e)

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

4.2 Observations

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's perfect in this situation.

Results

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 have to demonstrate 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.

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.

References

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. 2. J.J. Gibson (1950), The Perception of the Visual World, Houghton-Mifflin, Boston.
14. 3. E. Hecht and A. Lajac (1974), Optics, Addison-Wesley.
15. 4. R. Haralick and L. Shapiro (1992), Computer and Robot Vision, Volumes I and II, Addison-Wesley.
16. 5. M.D. Levine (1985), Vision in Man and Machine, McGraw-Hill.
17. 6. M. Livingstone, (1988) Art, Illusion and the Visual system, Scientific American, Jan. 1988, 78-85.
18. 7. 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. 8. V. Nalwa (1993), A Guided Tour of Computer Vision, Addison-Wesley.
20. 9. R.J. Schalko (1989), Digital Image Processing and Computer Vision, John Wiley and Sons.

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

Date:24/11/2020.....**PLAGIARISM VERIFICATION REPORT**

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

Name: Naveen Sharma Department: CSE Enrolment No 161372

Contact No. _____

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:

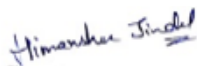
- Total No. of Pages =36
- Total No. of Preliminary pages =1
- Total No. of pages accommodate bibliography/references =1



(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at 17..... (%). 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)	
Report Generated on	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File)

through the supervisor at plagcheck.juit@gmail.com