A Project Report

on

Automated Image Caption Generator using deep learning

Submitted in partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

in

Information Technology

by:

Abhinav Sharma (1709713005)

Hansin Malhotra (1709713044)

Jaskaran Singh Luthra (1709713051)

Under the supervision of

Mr. Prem Prakash Agarwal

(Assistant Professor)

Galgotias College of Engineering and Technology,
Greater Noida 201306
Uttar Pradesh, India

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University,
Lucknow
July 2021

GALGOTIAS COLLEGE OF ENGINEERING & TECHNOLOGY
GREATER NOIDA - 201306, UTTAR PRADESH, INDIA.

CERTIFICATE

This is to certify that the project report entitled "**AUTOMATED IMAGE CAPTIONING GENERATOR USING DEEP LEARNING**" submitted by **Abhinav Sharma(1709713005), Hansin Malhotra (1709713044), Jaskaran Singh Luthra (1709713051)** to the **Galgotias College of Engineering and Technology** , Uttar Pradesh in partial fulfillment for the award of Degree of Bachelor of Technology in Information Technology is a bonafide record of the project work carried out by them under my supervision during the year 2020-2021.

Mr. Prem Prakash Agarwal
Assistant Professor
Deptt. of IT

Dr. Sanjeev Kumar Singh
HOD IT

ACKNOWLEDGEMENT

I would like to take this opportunity to show gratitude towards Mr Prem Prakash Agarwal (Assistant Professor IT Department) who helped in bringing a project to success. He has been a motivator & a source of inspiration to carry out the necessary proceedings for the seminar to be completed successfully.

Finally, I would like to take the opportunity to thank the organization GCET which helped me to acquire proper knowledge and success in the B.Tech (IT*)*

Abhinav Sharma: 1709713005

Hansin Malhotra: 1709713044

Jaskaran Singh Luthra: 1709713051

Abstract

The paper aims at generating automated captions by learning the contents of the image. At present images are annotated with human intervention and it becomes a nearly impossible task for huge commercial databases. The image database is given as input to a deep neural network (Convolutional Neural Network (CNN)) encoder for generating "thought vector" which extracts the features and nuances out of our image and RNN (Recurrent Neural Network) decoder is used to translate the features and objects given by our image to obtain a sequential, meaningful description of the image. In this paper, we systematically analyze different deep neural network-based image caption generation approaches and pre-trained models to conclude on the most efficient model with finetuning. The analyzed models contain both with and without 'attention' concept to optimize the caption generating ability of the model. All the models are trained on the same dataset for concrete comparison.

KEYWORDS*: Deep Neural Network, Image, automated Caption, Description, Long Short Term Memory (LSTM), Deep Learning, CNN, RNN, feature extraction, attention, Visual to text, image and video captioning, content 29 understanding, text description, summarization*.

CONTENTS

List of Tables

List of Figure

ABBREVIATIONS

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
ANN	Artificial Neural Network
NLTK	Natural Language Tool Kit
DTR	Dependency Tree Relations
SC-NLM	Structure-Content Neural Language Model
m-RNN	multimodal Recurrent Neural Network
GANs	Generative Adversarial Networks
RPN	Region Proposal Network
MIL	Multiple Instance Learning
MERT	Minimum Error Rate Training
DMSM	Deep Multimodal Similarity Model
ME	Maximum Entropy

CHAPTER 1

INTRODUCTION

Caption generation is an interesting artificial intelligence problem where a descriptive sentence is generated for a given image.

Generating a description of an image is called image captioning. Image captioning requires recognizing the important objects, their attributes and their relationships in an image. It also needs to generate syntactically and semantically correct sentences. Deep learning-based techniques are capable of handling the complexities and challenges of image captioning.

It involves the dual techniques 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. Image captioning has various applications such as recommendations in editing applications, usage in virtual assistants, image indexing, for visually impaired persons, social media, and several other natural language processing applications. Recently, deep learning methods have achieved state-of-the-art results on examples of this problem. It has been demonstrated that deep learning models can achieve optimum results in the field of caption generation problems. Instead of requiring complex data preparation or a pipeline of specifically designed models, a single end-to-end model can be defined to predict a caption, given a photo.

Aim

This project aims to develop software that can generate descriptive captions for images using neural networks language models. This project can act as a vision for visually impaired people, as it can identify nearby objects through the camera and give output in audio form. The application can provide a highly interactive platform for specially-abled people.

Project Scope

The goal is to design an web application that covers all the functions of image description and provides an interface to the user.

By using Deep Learning techniques the project performed:

- Image Captioning: Recognising Different types of objects in an image and creating a meaningful sentence that describes that image to visually impaired persons.
- Text to speech conversion: Forgiving the result in audio form.

CHAPTER 2

Literature Review

S.No	Research Paper	Observation	Limitations	Author	Year
1	Image Captioning: Transforming Objects into Words	The proposed approach shows how to identify the objects and their position in the image.	It is not capable of making sentences.	Simao Herdade, Armin Kappeler, Kofi Boakye, Joao Soares	2019
2.	Visual to text: Survey of image and video captioning	Visual to text technique highly depends on the training data in terms of both quality and quantity.	Long-standing gap semantic gap between computable low-level feature and semantic information that they encode.	Sheng Li, Zhiqiang Tao	2019
3.	Camera2Caption: A real-time image caption	By making use of Inception architecture and by simplifying overall flow design, this model works well in real-time image captioning	This model doesn't use visual attention which makes the inference process slower.	Aayush Yadav	2018
4.	Show and Tell: A Neural Image Caption Generator	It is an end to end neural network system that can automatically view an image and generate a reasonable description in plain English.	This model might generate diverse sentences but it might not able to generate a quality sentence.	Alexander Toshev, Oriol Vinyals, Samy Bengio, Dumitru Erhan	2015

Table-1 Literature Review

CHAPTER 3

Feasibility Study

Product

Image caption generator is a task that involves computer vision and natural language processing concepts to recognize the context of an image and describe them in a natural language like English

Technical Feasibility

Image caption generator is a task that involves computer vision and natural language processing concepts to recognize the context of an image and describe them in a natural language like English

In this Python project, we will be implementing the caption generator using CNN (Convolutional Neural Networks) and LSTM (Long short term memory). The image features will be extracted from Xception which is a CNN model trained on the imagenet dataset and then we feed the features into the LSTM model which will be responsible for generating the image captions. Hence we can say that our project is technically feasible

Socially Feasible

The image Captioning Generator would be compatible with Windows OS, Mac OS and even Android. We will also be Implementing this on IOS.

Economically Feasible

The project Image Captioning Generator is more cost-effective than other similar projects in the market and more user friendly.

CHAPTER 4

RELATED WORK

Several methods and algorithms have been developed to enhance the proficiency of the Image Captioning model and some of them are as follows:

4.1. Sheng Li, Zhiqiang Tao, Kang Li, and Yun Fu et al, in this paper Visual to Text: Survey of Image and Video Captioning classify existing methods by their mechanism to link visual information (including both images and videos)and text descriptions, and emphasize the latest advances on deep learning-based approaches. The visual to text techniques aim at accurately describing visual contents using natural language descriptions. One well-known challenge is the long-standing semantic gap between computable low-level features and semantic information that they encode. This paper gave a comprehensive survey of relevant work on this topic.

Future Work: a. To use most advanced supervised machine learning techniques on large-scale visual-text aligned data sets.
b. How to build large scale image and video datasets with accurate and diverse text descriptions in an effective manner.

4.2 Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan et al, in this paper Show and Tell: Image Caption Generator had presented NIC, an end-to-end neural network system that can automatically view an image and generate a reasonable description in plain English. NIC is based on a convolution neural network that encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence.

4.3. Shuang Liu, Liang Bai, Yanli Hu and Haoran Wang et al**,** in this paper Image Captioning based on Deep Neural Networks, had shown that how the text description of the image can improve the content-based image retrieval efficiency, the expanding application scope of visual understanding in the fields of medicine, security, military and other fields, which has a broad application prospect.

4.4. Pranay Mathur, Aman Gill, Aayush Yadav, Anurag Mishra and Nand Kumar Bansode et al, in the paper Camera2Caption: A Real-Time Image Caption Generation proposed a method by making use of Inception architecture and by simplifying the overall flow design, They had optimized model to perform well in real-time (on mobile devices) and manage to obtain high-quality captions.

4.5. Christopher Elamri, Teun de Planque et al, proposed in the paper Automated Neural Image Caption Generator for Visually Impaired People that Static images can only provide blind people with information about one specific instant of time, while video caption generation could potentially provide blind people with continuous real-time information. LSTMs could be used in combination with CNNs to translate videos to English description.

4.6. Varsha Kesavan, Vaidehi Muley, Megha Kolhekar et al, in the paper Deep Learning-based Automatic Image Caption Generation showed a convolution neural network that encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence. The model is trained to maximize the likelihood of the sentence given the image.

4.7. Chetan Amritkar, Vaishali Jabade et al, showed in the paper Image Caption Generation using Deep Learning Technique that the use of larger datasets increases the performance of the model. The larger dataset will increase accuracy as well as reduce losses. Also, it shows that how unsupervised data for both images as well as text can be used for improving the image caption generation approaches.

4.8. Simao Herdade, Armin Kappeler, Kofi Boakye, Joao Soares et al, in this paper Image Captioning: Transforming Objects into word presented the Object Relation Transformer, a modification of the conventional Transformer, specifically adapted to the task of image captioning. The proposed Transformer encodes 2D position and size relationships between detected objects in images, building upon the bottom-up and top-down image captioning approach. They also presented qualitative examples of how incorporating this information can yield captioning results demonstrating better spatial awareness. At present, this model only takes into account geometric information in the encoder phase.

CHAPTER 5

METHODOLOGY AND ARCHITECTURE

We use a pre-trained Convolutional Neural Network architecture as an encoder to extract and encode image features into a higher dimensional vector space. A CNN is a feed-forward type of Artificial Neural Network which, unlike fully connected layers, has only a subset of nodes in the previous layer connected to describe the image.

To leverage this potential to natural language, a usual method is to extract sequential information and convert them into language. For this task, an LSTM-based Recurrent Neural Network is used as a decoder to convert encoded features to natural language descriptions. In most recent image captioning works, they extract feature map from top layers of CNN, pass them to some form of RNN and then use a softmax to get the score of the words at every step.

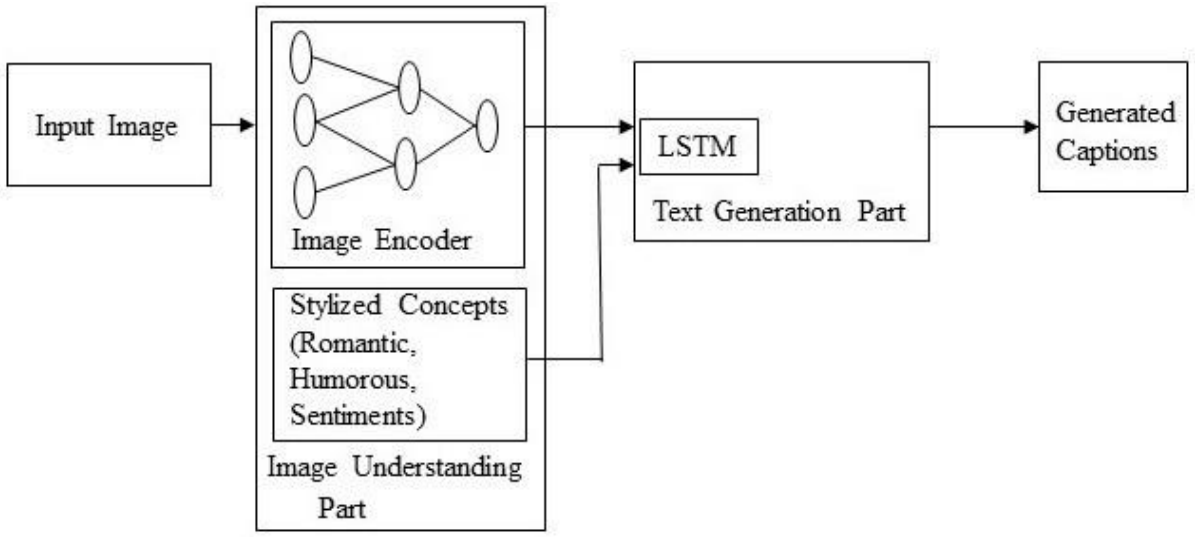

Fig.1 Image captioning model

For the image-caption relevancy task, recurrent neural networks help accumulate the semantics of a sentence. Strings of sentences are parsed into words, each of which has a GloVe vector representation that can be found in a lookup table.

These word vectors are fed into a recurrent neural network sequentially, which captures the notion of semantic meaning over the entire sequence of words via its hidden state. We treat the hidden state after the recurrent net has seen the last word in the sentence as the sentence embedding.

5.1 SYSTEM ARCHITECTURE

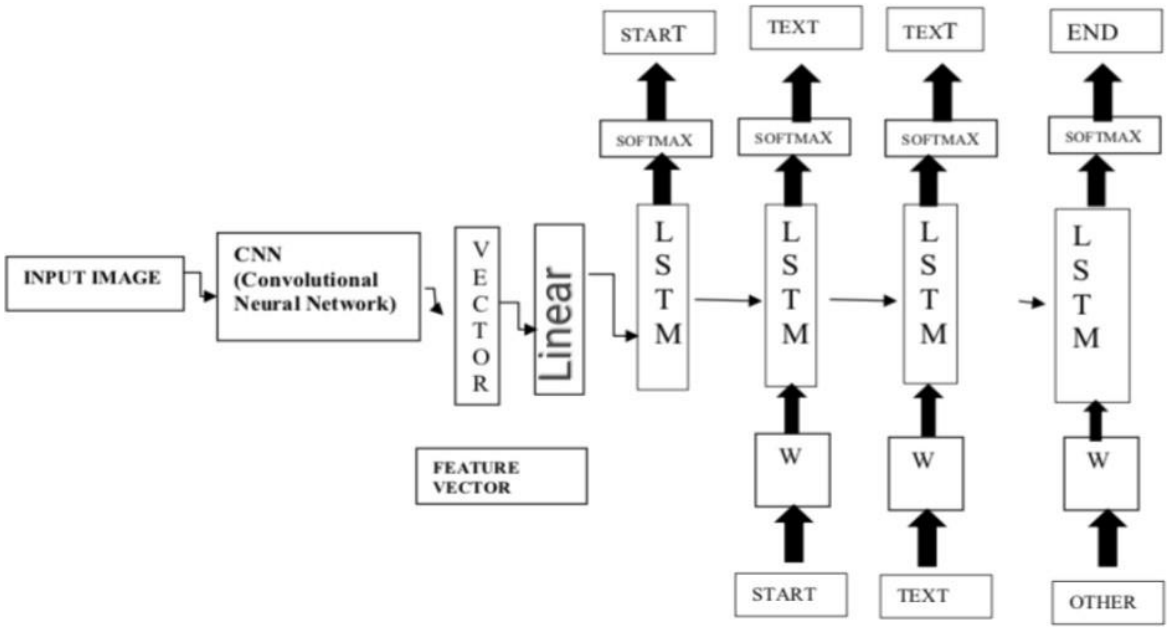

Figure 2. Proposed Model for Image Caption Generator

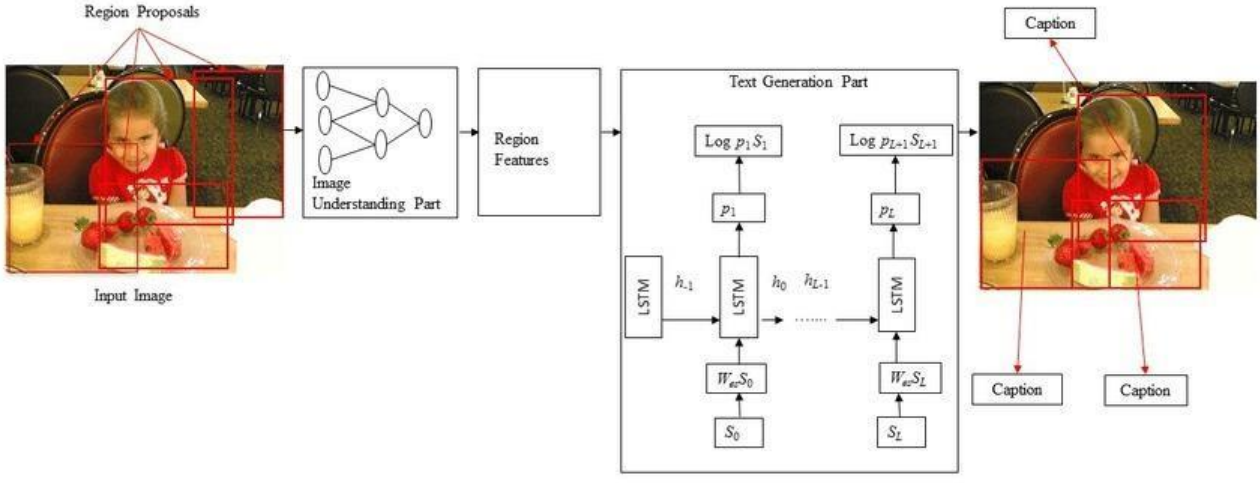

Figure 3. Image Caption Generator with bounding boxes on image

5.2 Planning of work

1. At first, we studied the image classification and natural language generation algorithms for example CNN and RNN.
2. Then, we will link the algorithms to create a hybrid approach for creating the Image Caption Generation model.
3. After that, we will test the hybrid approach on distinct test cases and measure its accuracy.
4. The generated hybrid algorithm will be able to generate captions with multiple objects in one frame.
5. Caption generated in the form of text is further converted into audio for helping a visually impaired person to understand the content of the image.
6. Finally, we will obtain Real-time Simulation Results

5.3 Process for Creating the Image Caption Generator Model

So, to make our image caption generator model, we will be merging these architectures. It is also called a **CNN-RNN model**.

1. First, we import all the necessary packages.
2. Getting and performing data cleaning.
3. Extracting the feature vector from all images
4. Loading dataset for Training the model.
5. Tokenizing the vocabulary.
6. Create a Data generator.
7. Defining the CNN-RNN model.
8. Training the model
9. Testing the model

CHAPTER: 6

Dependencies

6.1. Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result as fast as possible is key to doing good research.*

Keras is:

- **Simple** -- but not simplistic. Keras reduces developer *cognitive load* to free you to focus on the parts of the problem that really matter.
- **Flexible** -- Keras adopts the principle of *progressive disclosure of complexity*: simple workflows should be quick and easy, while arbitrarily advanced workflows should be *possible* via a clear path that builds upon what you've already learned.
- **Powerful** -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

6.2. Tensorflow

TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

6.3. NLTK

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.
Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

6.4. Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,

6.5 ResNet50

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 Floating points operations. It is a widely used ResNet model and we have explored **ResNet50** architecture in depth.

CHAPTER: 7

Convolutional Neural Network

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

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

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks make them prone to overfitting data. Typical ways of regularization, or preventing overfitting, include: penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.)

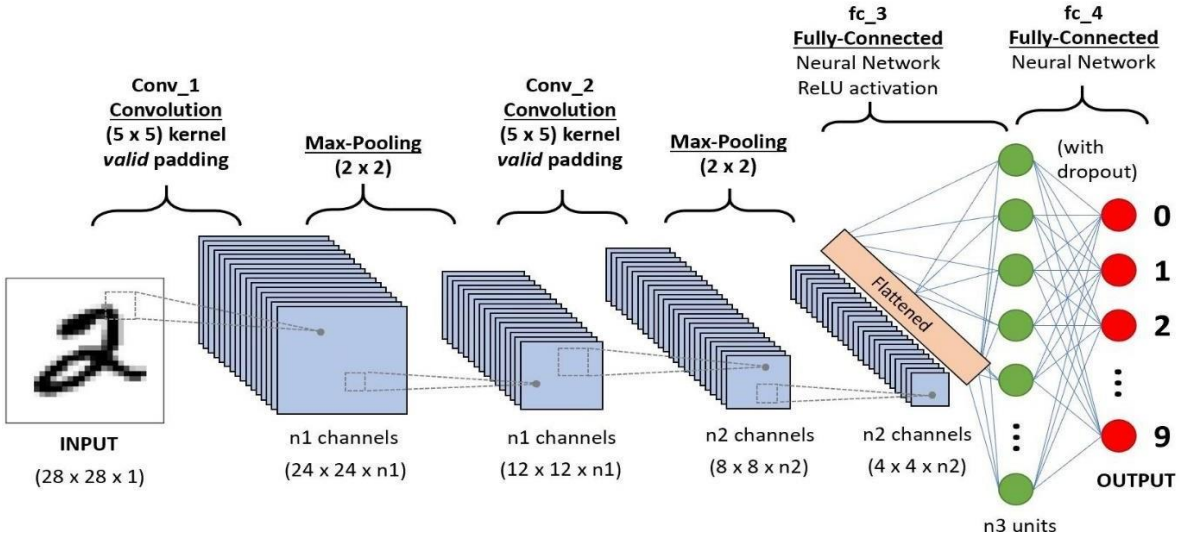

Fig 4. CNN model

Model: "resnet50"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	conv1_pad[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_1_conv[0][0]
conv2_block1_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_1_bn[0][0]
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block1_1_relu[0][0]
conv2_block1_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_2_conv[0][0]
conv2_block1_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_2_bn[0][0]
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16640	pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block1_2_relu[0][0]
conv2_block1_0_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block1_0_conv[0][0]
conv2_block1_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block1_3_conv[0][0]
conv2_block1_add (Add)	(None, 56, 56, 256)	0	conv2_block1_0_bn[0][0] conv2_block1_3_bn[0][0]
conv2_block1_out (Activation)	(None, 56, 56, 256)	0	conv2_block1_add[0][0]
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16448	conv2_block1_out[0][0]
conv2_block2_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block2_1_conv[0][0]
conv2_block2_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block2_1_bn[0][0]
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block2_1_relu[0][0]
conv2_block2_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block2_2_conv[0][0]
conv2_block2_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block2_2_bn[0][0]
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block2_2_relu[0][0]
conv2_block2_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block2_3_conv[0][0]

conv5_block1_0_conv (Conv2D)	(None, 7, 7, 2048)	2099200	conv4_block6_out[0][0]
conv5_block1_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block1_2_relu[0][0]
conv5_block1_0_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block1_0_conv[0][0]
conv5_block1_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block1_3_conv[0][0]
conv5_block1_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_0_bn[0][0] conv5_block1_3_bn[0][0]
conv5_block1_out (Activation)	(None, 7, 7, 2048)	0	conv5_block1_add[0][0]
conv5_block2_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	conv5_block1_out[0][0]
conv5_block2_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block2_1_conv[0][0]
conv5_block2_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block2_1_bn[0][0]
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block2_1_relu[0][0]
conv5_block2_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block2_2_conv[0][0]
conv5_block2_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block2_2_bn[0][0]
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block2_2_relu[0][0]
conv5_block2_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block2_3_conv[0][0]
conv5_block2_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_out[0][0] conv5_block2_3_bn[0][0]
conv5_block2_out (Activation)	(None, 7, 7, 2048)	0	conv5_block2_add[0][0]
conv5_block3_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	conv5_block2_out[0][0]
conv5_block3_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_1_conv[0][0]
conv5_block3_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_1_bn[0][0]
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block3_1_relu[0][0]
conv5_block3_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_2_conv[0][0]
conv5_block3_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_2_bn[0][0]
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block3_2_relu[0][0]
conv5_block3_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block3_3_conv[0][0]
conv5_block3_add (Add)	(None, 7, 7, 2048)	0	conv5_block2_out[0][0] conv5_block3_3_bn[0][0]
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	conv5_block3_add[0][0]
avg_pool (GlobalAveragePooling2	(None, 2048)	0	conv5_block3_out[0][0]
predictions (Dense)	(None, 1000)	2049000	avg_pool[0][0]

Total params: 25,636,712
Trainable params: 25,583,592
Non-trainable params: 53,120

Table-2 ResNet 50 model Architecture

CHAPTER: 8

Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.

This is a behavior required in complex problem domains like machine translation, speech recognition, and more.LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field.

Long Short Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network is also known as RNN is used for persistent memory.

Let's say while watching a video you remember the previous scene or while reading a book you know what happened in the earlier chapter. Similarly RNNs work, they remember the previous information and use it for processing the current input. The shortcoming of RNN is, they can not remember Long term dependencies due to vanishing gradient. LSTMs are explicitly designed to avoid long-term dependency problems.

The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.

These three parts of an LSTM cell are known as gates. The first part is called **Forget gate, the** second part is known as **the Input gate** and the last one is **the Output gate**.

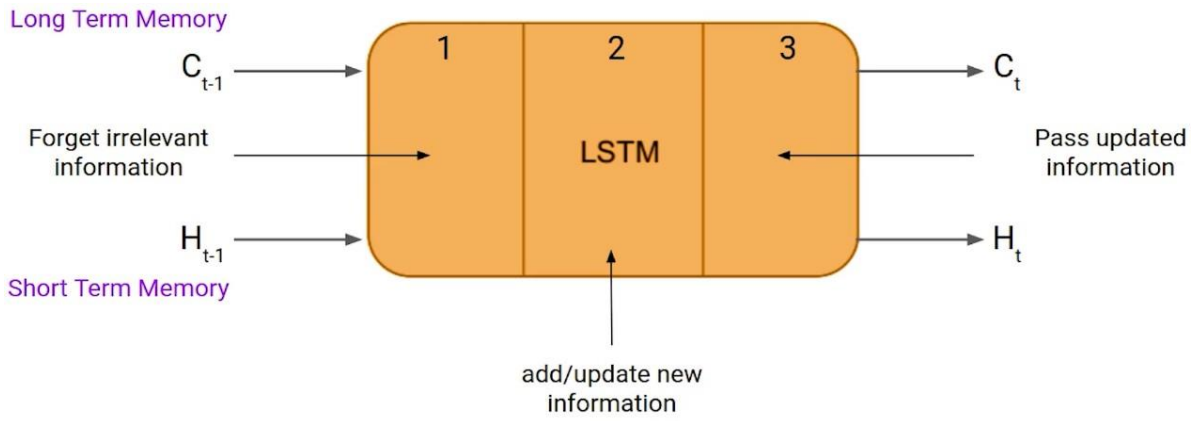

Fig 5. LSTM model

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 35)]	0	
input_1 (InputLayer)	[(None, 2048)]	0	
embedding (Embedding)	(None, 35, 50)	92400	input_2[0][0]
dropout (Dropout)	(None, 2048)	0	input_1[0][0]
dropout_1 (Dropout)	(None, 35, 50)	0	embedding[0][0]
dense (Dense)	(None, 256)	524544	dropout[0][0]
lstm (LSTM)	(None, 256)	314368	dropout_1[0][0]
add (Add)	(None, 256)	0	dense[0][0] lstm[0][0]
dense_1 (Dense)	(None, 256)	65792	add[0][0]
dense_2 (Dense)	(None, 1848)	474936	dense_1[0][0]

Total params: 1,472,840
Trainable params: 1,472,840
Non-trainable params: 0

Table -3 RNN Architecture

CHAPTER : 9

DEEP LEARNING BASED IMAGE CAPTIONING METHODS

We draw an overall taxonomy in Figure 1 for deep learning-based image captioning methods. We discuss their similarities and dissimilarities by grouping them into visual space vs. multimodal space, dense captioning vs. captions for the whole scene, Supervised learning vs. Other deep learning, Encoder-Decoder architecture vs. Compositional architecture, and one 'Others' group that contains Attention-Based, Semantic Concept-Based, Stylized captions, and Novel Object-Based captioning. We also create a category named LSTM vs. Others.

A brief overview of the deep learning-based image captioning methods is shown in Table 1. Table 1 contains the name of the image captioning methods, the type of deep neural networks used to encode image information, and the language models used in describing the information. In the final column, we give a category label to each captioning technique based on the taxonomy in Figure 6.

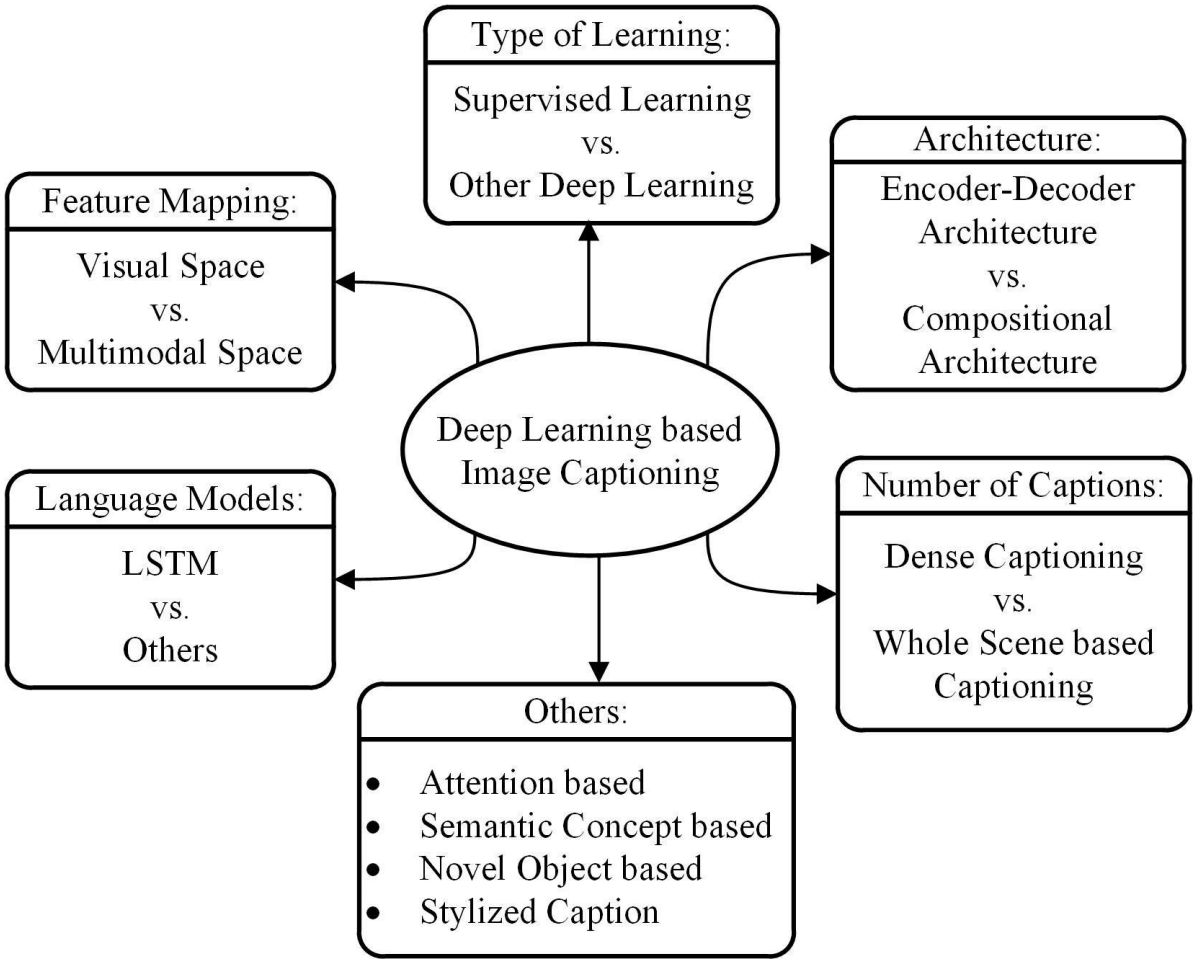

Fig. 6. An overall taxonomy of deep learning-based image captioning

9.1 Visual Space vs. Multimodal Space

Deep learning-based image captioning methods can generate captions from both visual space and multimodal space. Understandably image captioning datasets have the corresponding captions as text. In the visual space-based methods, the image features and the corresponding captions are independently passed to the language decoder. In contrast, in a multimodal space case, a shared multimodal space is learned from the images and the corresponding caption-text. This multimodal representation is then passed to the language decoder.

9.1.1 *Visual Space.* Bulk of the image captioning methods use visual space for generating captions. These methods are discussed in Section 3.2 to Section 3.5.

9.1.2 *Multimodal Space.* The architecture of a typical multimodal space-based method contains a language Encoder part, a vision part, a multimodal space part, and a language decoder part. A general diagram of multimodal space-based image captioning methods is shown in Figure 2. The vision part uses a deep convolutional neural network as a feature extractor to extract the image features. The language encoder part extracts the word features and learns a dense feature embedding for each word. It then forwards the semantic temporal context to the recurrent layers. The multimodal space part maps the image features into a common space with the word features.

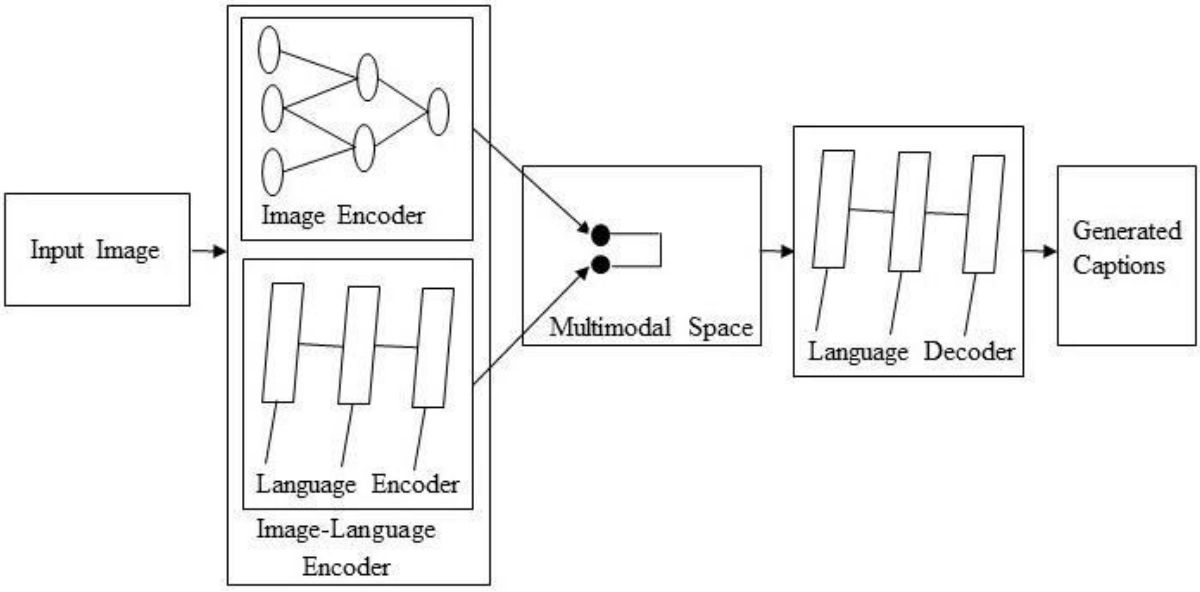

Fig. 7. A block diagram of multimodal space-based image captioning.

The resulting map is then passed to the language decoder which generates captions by decoding the map.

The methods in this category follow the following steps:

(1) Deep neural networks and multimodal neural language model are used to learn

both image and text jointly in a multimodal space.

(2)The language generation part generates captions using the information from Step 1 .

An initial work in this area proposed by Kiros et al.. The method applies a CNN for extracting image features in generating image captions. It uses a multimodal space that represents both image and text jointly for multimodal representation learning and image caption generation.

It also introduces the multimodal neural language models such as Modality-Biased Log-Bilinear Model (MLBL-B) and the Factored 3-way Log-Bilinear Model (MLBL-F) of followed by AlexNet . Unlike most previous approaches, this method does not rely on any additional templates, structures, or constraints. Instead it depends on the high level image features and word representations learned from deep neural networks and multimodal neural language models respectively. The neural language models have limitations to handle a large amount of data and are inefficient to work with long term memory .

Kiros et al.extended their work in to learn a joint image sentence embedding where LSTM is used for sentence encoding and a new neural language model called the structure-content neural language model (SC-NLM) is used for image captions generations. The SC-NLM has an advantage over existing methods in that it can extricate the structure of the sentence to its content produced by the encoder. It also helps them to achieve significant improvements in generating realistic image captions over the approach proposed by

Karpathy et al. proposed a deep, multimodal model, embedding of image and natural language data for the task of bidirectional images and sentences retrieval. The previous multimodal-based methods use a common, embedding space that directly maps images and sentences. However, this method works at a finer level and embeds fragments of images and fragments of sentences. This method breaks down the images into a number of objects and sentences into a dependency tree relations (DTR) and reasons about their latent, inter-modal alignment. It shows that the method achieves significant improvements in the retrieval task compared to other previous methods.

The phrase “black and white dog” can be formed into two relations (CONJ, black, white) and (AMOD, white, dog). Again, for many dependency relations we do not find any clear mapping in the image (For example: “each other” cannot be mapped to any object).

Mao et al.proposed a multimodal Recurrent Neural Network (m-RNN) method for generating novel image captions. This method has two sub-networks: a deep recurrent neural network for sentences and a deep convolutional network for images. These two sub-networks interact with each other in a multimodal layer to form the whole m-RNN model. Both image and fragments of sentences are given as input in this method. It calculates the probabilty distribution to generate the next word of captions. There are five more layers in this model: Two-word embedding layers, a recurrent layer, a multimodal layer and a SoftMax layer. Kiros et al. proposed a method that is built on a Log-Bilinear model and used AlexNet to extract visual features. This multimodal recurrent neural network method is closely related to the method of Kiros et al. . Kiros et al. use a fixed length context (i.e. five words), whereas in this method, the temporal context is stored in a recurrent architecture, which allows an arbitrary context length. The two word embedding layers use one hot vector to generate a dense word representation. It encodes both the syntactic and semantic meaning of the words. The semantically relevant words can be found by calculating the Euclidean distance between two dense word vectors in embedding layers. Most sentence-image multimodal methods use pre-computed word embedding vectors to initialize their model. In contrast, this method randomly

initializes word embedding layers and learn them from the training data. This helps them to generate better image captions than the previous methods. Many image captioning methods are built on recurrent neural networks at the contemporary times. They use a recurrent layer for storing visual information. However, (m-RNN) use both image representations and sentence fragments to generate captions.

9.2 Supervised Learning vs. Other Deep Learning

In supervised learning, training data come with desired output called *label*. Unsupervised learning, on the other hand, deals with unlabeled data. Generative Adversarial Networks (GANs) are a type of unsupervised learning techniques. Reinforcement learning is another type of machine learning approach where the aims of an agent are to discover data and/or labels through exploration and a reward signal. A number of image captioning methods use reinforcement learning and GAN based approaches. These methods sit in the category of “Other Deep Learning".

9.2.1 Supervised Learning-Based Image Captioning. Supervised learning-based networks have successfully been used for many years in image classification , object detection , and attribute learning. This progress makes researchers interested in using them in automatic image captioning. In this paper, we have identified a large number of supervised learning-based image captioning methods. We classify them into different categories: (i)

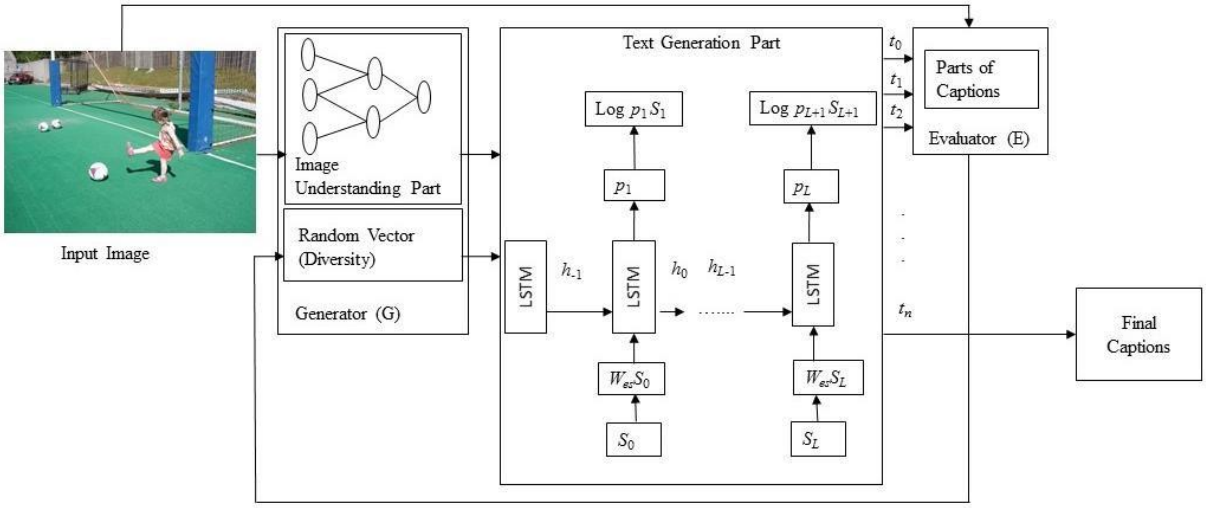

Fig. 8. A block diagram of other deep learning-based captioning.

Encoder-Decoder Architecture, (ii) Compositional Architecture, (iii) Attention-based, (iv) Semantic concept-based, (v) Stylized captions, (vi) Novel object-based, and (vii) Dense image captioning.

9.2.2 Other Deep Learning-Based Image Captioning.

In our day to day life, data are increasing with unlabled data because it is often impractical to accurately annotate data. Therefore, recently, re- searchers are focusing more on reinforcement learning and unsupervised learning-based techniques for image captioning.

A reinforcement learning agent chooses an action, receives reward values, and moves to a new state. The agent attempts to select the action with the expectation of having a maximum long-term reward. It needs continuous state and action information, to provide the guarantees of a value function. Traditional reinforcement learning approaches face a number of limitations such as the lack of guarantees of a value function and uncertain state-action information. Policy gradient methods are a type of reinforcement learning that can choose a specific policy for a specific action using gradient descent and optimization techniques. The policy can incorporate domain knowledge for the action that guarantees convergence. Thus, policy gradient methods require fewer parameters than value-function based approaches.

Existing deep learning-based image captioning methods use variants of image encoders to extract image features. The features are then fed into the neural network-based language decoders to generate captions. The methods have two main issues: (i) They are trained using maximum likeli- hood estimation and back-propagation approaches. In this case, the next word is predicted given the image and all the previously generated ground-truth words. Therefore, the generated captions look-like ground-truth captions. This phenomenon is called exposure bias problem.

(ii) Evaluation metrics at test time are non-differentiable. Ideally sequence models for image cap- tioning should be trained to avoid exposure bias and directly optimise metrics for the test time. In actor-critic-based reinforcement learning algorithm, critic can be used in estimating the expected future reward to train the actor (captioning policy network). Reinforcement learning-based image captioning methods sample the next token from the model based on the rewards they receive in each state. Policy gradient methods in reinforcement learning can optimize the gradient in order to predict the cumulative long-term rewards. Therefore, it can solve the non-differentiable problem of evaluation metrics.

The methods in this category follow the following steps:

(1) A CNN and RNN based combined network generates captions.
(2) Another CNN-RNN based network evaluates the captions and send feedback to the first network to generate high quality captions.

A block diagram of a typical method of this category is shown in Figure 3.

Ren et al. 2017 introduced a novel reinforcement learning based image captioning method. The architecture of this method has two networks that jointly compute the next best word at each time step. The “policy network” works as local guidance and helps to predict next word based on the current state. The “value network”’ works as global guidance and evaluates the reward value considering all the possible extensions of the current state. This mechanism is able to adjust the networks in predicting the correct words. Therefore, it can generate good captions similar to

ground truth captions at the end. It uses an actor-critic reinforcement learning model to train the whole network. Visual semantic embedding is used to compute the actual reward value in predicting the correct word. It also helps to measure the similarity between images and sentences that can evaluate the correctness of generated captions.

Rennie et al. proposed another reinforcement learning based image captioning method. The method utilizes the test-time inference algorithm to normalize the reward rather than estimating the reward signal and normalization in training time. It shows that this test-time decoding is highly effective for generating quality image captions.

Zhang et al. proposed an actor-critic reinforcement learning-based image captioning method. The method can directly optimize non-differentiable problems of the existing evaluation metrics. The architecture of the actor-critic method consists of a policy network (actor) and a value network (critic). The actor treats the job as sequential decision problem and can predict the next token of the sequence. In each state of the sequence, the network will receive a task-specific reward (in this case, it is evaluation metrics score). The job of the critic is to predict the reward. If it can predict the expected reward, the actor will continue to sample outputs according to its probability distribution.

GAN based methods can learn deep features from unlabeled data. They achieve this repre- sentations applying a competitive process between a pair of networks: the Generator and the Discriminator. GANs have already been used successfully in a variety of applications, including image captioning, image to image translation , text to image synthesis , and text generation .

There are two issues with GAN. First, GAN can work well in generating natural images from real images because GANs are proposed for real-valued data. However, text processing is based on discrete numbers. Therefore, such operations are non-differentiable, making it difficult to apply back-propagation directly. Policy gradients apply a parametric function to allow gradients to be back-propagated. Second, the evaluator faces problems in vanishing gradients and error propagation for sequence generation. It needs a probable future reward value for every partial description. Monte Carlo rollouts is used to compute this future reward value.

GAN based image captioning methods can generate a diverse set of image captions in contrast to conventional deep convolutional network and deep recurrent network based model. Dai et al.
also proposed a GAN based image captioning method. However, they do not consider multiple captions for a single image. Shetty et al. introduced a new GAN based image captioning method. This method can generate multiple captions for a single image and showed impressive improvements in generating diverse captions. GANs have limitations in backpropagating the discrete data. Gumbel sampler is used to overcome the discrete data problem. The two main parts of this adversarial network are the generator and the discriminator. During training, generator learns the loss value provided by the discriminator instead of learning it from explicit sources

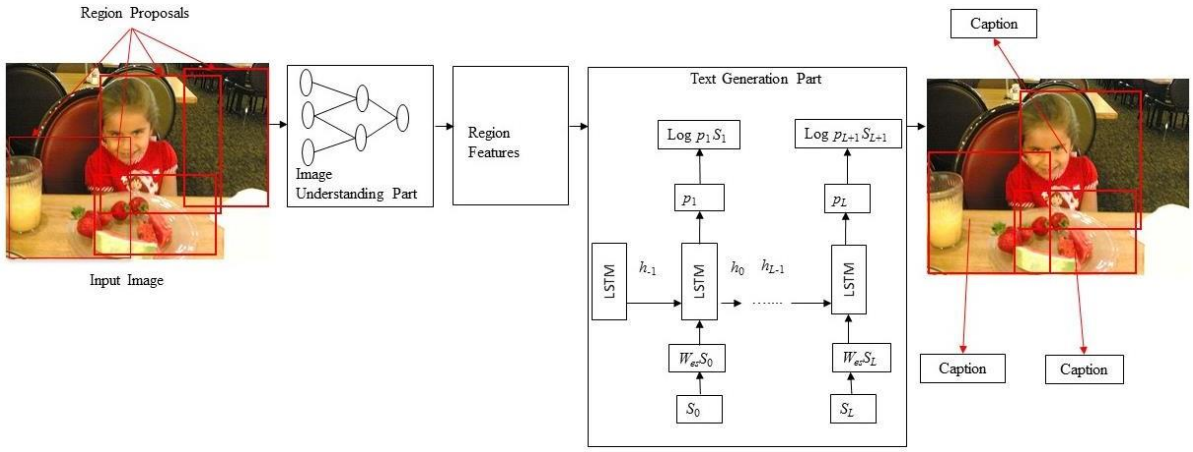

Fig. 9. A block diagram of dense captioning.

Discriminator has true data distribution and can discriminate between generator-generated samples and true data samples. This allows the network to learn diverse data distribution. Moreover, the network classifies the generated caption sets either real or fake. Thus, it can generate captions similar to human generated one.

9.3 Dense Captioning vs. Captions for the whole scene

In dense captioning, captions are generated for each region of the scene. Other methods generate captions for the whole scene.

Dense Captioning. The previous image captioning methods can generate only one caption for the whole image. They use different regions of the image to obtain information of various objects. However, these methods do not generate region wise captions.

Johnson et al. proposed an image captioning method called DenseCap. This method localizes all the salient regions of an image and then it generates descriptions for those regions.

A typical method of this category has the following steps:

(1)Region proposals are generated for the different regions of the given image. (2)CNN is used to obtain the region-based image features.

(3)The outputs of Step 2 are used by a language model to generate captions for every region.

A block diagram of a typical dense captioning method is given in Figure 4.

Dense captioning proposes a fully convolutional localization network architecture, which is composed of a convolutional network, a dense localization layer, and an LSTM language model. The dense localization layer processes an image with a single, efficient forward pass, which implicitly predicts a set of region of interest in the image. Thereby, it requires no external region proposals unlike to Fast R-CNN or a full network (i.e., RPN (Region Proposal Network)) of Faster R-CNN. The working principle of the localization layer is related to the work of Faster R-CNN . However, Johnson et al. use a differential, spatial soft attention mechanism and

bilinear interpolation instead of ROI pooling mechanism. This modification helps the method to backpropagate through the network and smoothly select the active regions. It uses Visual Genome dataset for the experiments in generating region level image captions.

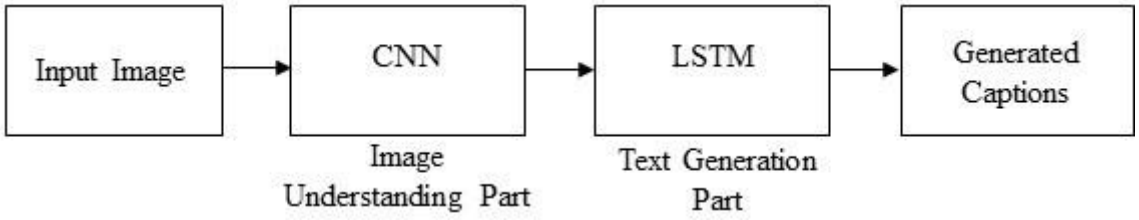

Fig. 10. A block diagram of simple Encoder-Decoder architecture-based image captioning.

challenges in dense captioning. As regions are dense, one object may have multiple overlapping regions of interest. Moreover, it is very difficult to recognize each target region for all the visual concepts. Yang et al. proposed another dense captioning method. This method can tackle these challenges.

First, it addresses an inference mechanism that jointly depends on the visual features of the region and the predicted captions for that region. This allows the model to find an appropriate position of the bounding box. Second, they apply a context fusion that can combine context features with the visual features of respective regions to provide arichsemantic description.

Captions for the whole scene. Encoder-Decoder architecture, Compositional architecture, attention-based, semantic concept-based, stylized captions, Novel object-based image captioning, and other deep learning networks-based image captioning methods generate single or multiple captions for the whole scene.

9.4 Encoder-Decoder Architecture vs. Compositional Architecture

Some methods use just simple vanilla encoder and decoder to generate captions. However, other methods use multiple networks for it.

Encoder-Decoder Architecture-Based Image captioning. The neural network-based image captioning methods work as just simple end to end manner. These methods are very similar to the encoder-decoder framework-based neural machine translation. In this network, global image features are extracted from the hidden activations of CNN and then fed them into an LSTM to generate a sequence of words.

A typical method of this category has the following general steps:

(1) A vanilla CNN is used to obtain the scene type, to detect the objects and their relationships.
(2) The output of Step 1 is used by a language model to convert them into words, combined phrases that produce an image captions.

A simple block diagram of this category is given in Figure 5.

Vinyals et al. proposed a method called Neural Image Caption Generator (NIC). The method uses a CNN for image representations and an LSTM for generating image captions. This special CNN uses a novel method for batch normalization and the output of the last hidden layer of CNN is used as an input to the LSTM decoder. This LSTM is capable of keeping track of the objects that already have been described using text. NIC is trained based on maximum likelihood estimation. In generating image captions, image information is included to the initial state of an LSTM. The next words are generated based on the current time step and the previous hidden state. This process continues until it gets the end token of the sentence. Since image information is fed only at the beginning of the process, it may face vanishing gradient problems. The role of the words generated at the beginning is also becoming weaker and weaker. Therefore, LSTM is still facing challenges in generating long length sentences . Therefore, Jia et al. proposed an extension of LSTM called guided LSTM (gLSTM). This gLSTM can generate long sentences. In this architecture, it adds

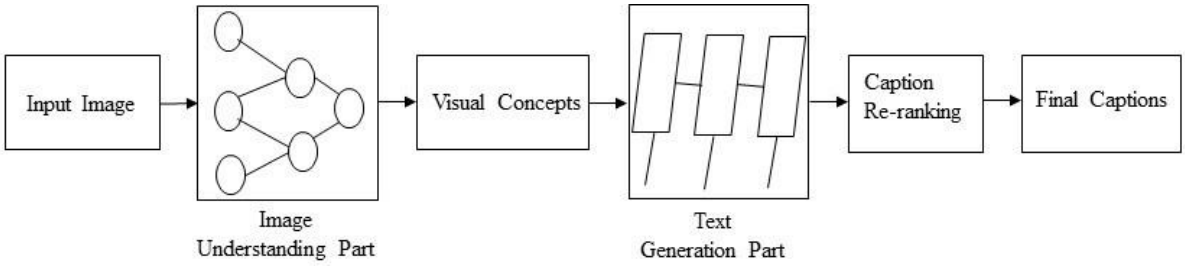

Fig. 11. A block diagram of a compositional network-based captioning.

global semantic information to each gate and cell state of LSTM. It also considers different length normalization strategies to control the length of captions. Semantic information is extracted in different ways. First, it uses a cross-modal retrieval task for retrieving image captions and then semantic information is extracted from these captions. The semantic based information can also be extracted using a multimodal embedding space.

Mao et al. proposed a special type of text generation method for images. This method can generate a description for an specific object or region that is called referring expression. Using this expression it can then infer the object or region which is being described. Therefore, generated description or expression is quite unambiguous. In order to address the referring expression, this method uses a new dataset called ReferIt dataset [68] based on popular MS COCO dataset.

Previous CNN-RNN based image captioning methods use LSTM that are unidirectional and relatively shallow in depth. In unidirectional language generation techniques, the next word is predicted based on visual context and all the previous textual contexts. Unidirectional LSTM cannot generate contextually well formed captions. Moreover, recent object detection and classification methods show that deep, hierarchical methods are better at learning than shallower ones. Wang et al.

proposed a deep bidirectional LSTM-based method for image captioning. This method is capable of generating contextually and semantically rich image captions. The proposed architecture consists of a CNN and two separate LSTM networks. It can utilize both past and future context information to learn long term visual language interactions.

Compositional Architecture-Based Image captioning. Compositional architecture-based methods composed of several independent functional building blocks: First, a CNN is used to extract the semantic concepts from the image. Then a language model is used to generate a set of candidate captions. In generating the final caption, these candidate captions are re-ranked using a deep multimodal similarity model.

A typical method of this category maintains the following steps:

(1) Image features are obtained using a CNN.
(2) Visual concepts (e.g. attributes) are obtained from visual features.
(3) Multiple captions are generated by a language model using the information of Step 1 and Step 2.
(4) The generated captions are re-ranked using a deep multimodal similarity model to select high quality image captions.

A common block diagram of compositional network-based image captioning methods is given in Figure 6.

Fang et al. introduced generation-based image captioning. It uses visual detectors, a language model, and a multimodal similarity model to train the model on an image captioning dataset. Image captions can contain nouns, verbs, and adjectives. A vocabulary is formed using 1000 most common words from the training captions. The system works with the image sub-regions rather that the full image. Convolutional neural networks (both AlexNet and VGG16Net) are used for extracting features for the sub-regions of an image. The features of sub-regions are mapped with the words of the vocabulary that likely to be contained in the image captions. Multiple instance learning (MIL) is used to train the model for learning discriminative visual signatures of each word. A maximum entropy (ME) language model is used for generating image captions from these words. Generated captions are ranked by a linear weighting of sentence features. Minimum Error rate training (MERT) is used to learn these weights. Similarity between image and sentence can be easily measured using a common vector representation. Image and sentence fragments are mapped with the common vector representation by a deep multimodal similarity model (DMSM). It achieves a significant improvement in choosing high quality image captions.

Until now a significant number of methods have achieved satisfactory progress in generating image captions. The methods use training and testing samples from the same domain. Therefore, there is no certainty that these methods can perform well in open-domain images. Moreover, they are only good at recognizing generic visual content. There are certain key entities such as celebrities and landmarks that are out of their scope. The generated captions of these methods are evaluated on automatic metrics such as BLEU , METEOR , and CIDEr These evaluation metrics have already shown good results on these methods. However, in terms of performance there exists a large gap between the evaluation of the metrics and human judgement of evaluation . If it is considered real life entity information, the performance could be weaker. However, Tran et al. introduced a different image captioning method. This method is capable of generating image captions even for open domain images. It can

detect a diverse set of visual concepts and generate captions for celebrities and landmarks. It uses an external knowledge base Freebase in recognizing a broad range of entities such as celebrities and landmarks. A series of human judgments are applied for evaluating the performances of generated captions. In experiments, it uses three datasets: MS COCO, Adobe-MIT FiveK, and images from Instagram. The images of MS COCO dataset were collected from the same domain but the images of other datasets were chosen from an open domain. The method achieves notable performances especially on the challenging Instagram dataset.

Ma et al. proposed another compositional network-based image captioning method. This method uses structural words <object, attribute, activity, scene> to generate semantically mean- ingful descriptions. It also uses a multi-task method similar to multiple instance learning method
, and multi-layer optimization method to generate structural words. An LSTM encoder- decoder-based machine translation method is then used to translate the structural words into image captions.

Wang et al. proposed a parallel-fusion RNN-LSTM architecture for image caption generation. The architecture of the method divides the hidden units of RNN and LSTM into a number of same- size parts. The parts work in parallel with corresponding ratios to generate image captions.

CHAPTER 10

Expected Outcome

Image captioning has made significant advances in recent years. Recent work based on deep learning techniques has resulted in a breakthrough in the accuracy of image captioning. The text description of the image can improve the content-based image retrieval efficiency, the expanding application scope of visual understanding in the fields of medicine, security, military and other fields, which has a broad application prospect. At the same time, the theoretical framework and research methods of image captioning can promote the development of the theory and application of image annotation and visual question answering (VQA), cross-media retrieval, video captioning and video dialogue, which has important academic and practical application value.

Word	Neighbours
Car	van, cab, suv, vehicule, jeep
Boy	toddler, gentleman, daughter, son
Street	road, streets, highway, freeway
Horse	pony, donkey, pig, goat, mule
computer	computers, pc, chip, compute

Table-4 Expected Outcome

Fig 12. Results

We have presented AICG, an end-to-end neural network system that can automatically view an image and generate a reasonable description in plain English. AICG is based on a convolution neural network that encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence. The model is trained to maximize the likelihood of the sentence given the image. Experiments on several datasets show the robustness of AICG in terms of qualitative results (the generated sentences are very reasonable) and quantitative evaluations, using either ranking metrics or BLEU, a metric used in machine translation to evaluate the quality of generated sentences. It is clear from these experiments that, as the size of the available datasets for image description increases.

CHAPTER 11

Conclusion

We have presented a deep learning model that automatically generates image captions intending to help visually impaired people better understand their environments. Our described model is based on a CNN that encodes an image into a compact representation, followed by an RNN that generates corresponding sentences based on the learned image features. We showed that this model achieves comparable to state-of-the-art performance and that the generated captions are highly de- scriptive of the objects and scenes depicted on the images. Because of the high quality of the people. In addition, future work could focus on translating videos directly to sentences instead of generating captions of images. Static images can only provide blind people with information about one specific instant of time, while video caption generation could potentially provide blind people with continuous real-time information. LSTMs could be used in combination with CNNs to translate videos to English descriptions.

References

[1] **A. Aker and R. Gaizauskas.** Generating image descriptions using dependency relational patterns. In ACL, 2010.

A. **Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei**. ImageNet Large Scale Visual Recognition Challenge, 2014.

[2] **Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010.** Every picture tells a story: Generating sentences from images. In European conference on computer vision. Springer, 15–29.

[3] **Alireza Fathi, Yin Li, and James M Rehg. 2012**. Learning to recognize daily actionsusing gaze. In European Conference on Computer Vision. Springer, 314–327.

[4] **Alireza Fathi, Yin Li, and James M Rehg. 2012.** Learning to recognize daily actionsusing gaze. In European Conference on Computer Vision. Springer, 314–327.

[5] **Bo Dai, Dahua Lin, Raquel Urtasun, and Sanja Fidler. 2017**. Towards Diverse and Natural Image Descriptions via a Conditional GAN. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 2989–2998.

[6] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier. Collecting image annotations using amazon's mechanical turk. In NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, pages139– 147, 2010.

[7] **Chuang Gan, Tianbao Yang, and Boqing Gong. 2016.** Learning attributes equals multi-source domain generalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 87–97.

[8] **Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. 2017.** Stylenet: Generating attractive visual captions with styles. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3137–3146.

[9] **D. Bahdanau, K. Cho, and Y. Bengio.** Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, 2014.

[10] **D. S. Whitehead, L. Huang, H. and S.-F. Chang,** "Entity-aware Image Caption Generation," in Empirical Methods in Natural Language Processing, Brussels, 2018.

[11] **Dave Golland, Percy Liang, and Dan Klein. 2010**. A game-theoretic approach to generating spatial descriptions. In Proceedings of the 2010 conference on empirical methods in natural language processing. Association for Computational

[12] **Desmond Elliott and Frank Keller. 2013**. Image description using visual dependency representations. In Proceedings of the 2013 Conference on Empirical Methods in NaturalLanguage Processing. 1292–1302.

[13] **Etienne Denoual and Yves Lepage. 2005**. BLEU in characters: towards automatic MT evaluation in languages without word delimiters. In Companion Volume to the Proceedingsof the Second International Joint Conference on Natural Language Processing. 81–86.

[14] **G. Ding, M. Chen, S. Zhao, H. Chen, J. Han and Q. Liu**, "Neural Image Caption Generation with Weighted Training and Reference," Cognitive Computation, 08 August 2018.

[15] **Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng,**

Piotr DollAąr, Jianfeng Gao, Xiaodong He, Margaret Mitchell, and John C Platt. 2015. From captions to visual concepts and back. In Proceedings of the IEEE conference on computervision and pattern recognition. 1473–1482.
[16] **Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014**. Generative adversarial nets. InAdvances in neural information processing systems. 2672–2680.
In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 1141–1150.
[17]**J. Chen, W. Dong and M. Li,** "Image Caption Generator Based On Deep Neural Networks," March 2018.

[18] **J. Mao, W. Xu, Y. Yang, J. Wang, and A. Yuille.** Explain images with multimodal recurrent neural networks. In arXiv:1410.1090, 2014.
[19] **Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015**. Language models for image captioning: The quirks andwhat works. arXiv preprint arXiv:1505.01809 (2015).
[20] **Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015.** Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEEconference on computer vision and pattern recognition. 2625–2634.
[21] **Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016**. Aconvolutional encoder model for neural machine translation. arXiv preprint arXiv:1611.02344.
[22] **Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin.2017**. Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122.

[23] **K. Papineni, S. Roukos, T. Ward, and W. J. Zhu.** BLEU: A method for automatic evaluation of machine translation. In ACL, 2002.
[24] **Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015.** DRAW: A recurrent neural network for image generation. In Proceedings ofMachine Learning Research. 1462–1471.

[25] **Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, Yoshua Benjio,** "Show, attend and tell: neural image caption generation with visual attention", ICML'15 Proceedings of the 32nd International Conference on Machine Learning – Volume 37, Pages 2048-2057.

[26] **Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu,** "BLEU: a Method for Automatic Evaluation of Machine Translation", Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 311-318.
Linguistics, 410–419.

[27] **M. Mitchell, X. Han, J. Dodge, A. Mensch, A. Goyal, A. C. Berg, K. Yamaguchi, T. L. Berg, K. Stratos, and H. D. III. Midge:** Generating image descriptions from computer vision detections. In EACL, 2012.
[28] **Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D Manning. 2006.** Generating typed dependency parses from phrase structure parses. In Proceedings of LREC,Vol. 6. Genoa Italy, 449–454.
[29] **Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. 2006**. Theiapr tc-12 benchmark: A new evaluation resource for visual information

systems. In International workshop ontoImage, Vol. 5. 10.
[30] **Navneet Dalal and Bill Triggs. 2005.** Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1. IEEE, 886–893.
[31] **Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. 2013**. Learning distributions over logical forms for referring expression generation. In Proceedings of the 2013 conferenceon empirical methods in natural language processing.1914–1925.

[32] **O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy**,

[33] **Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan,** "Show and Tell: A Neural Image Caption Generator", published 2015 IEEE Conference on Computer Vision andPattern Recognition (CVPR)

[34] **P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, andY.LeCun**. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229, 2013.
[35]**R. Staniute and D. Sesok,** "A Systematic Literature Review on Image Captioning," Applied Sciences, vol. 9, no. 10, 16 March 2019.
[36] **Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014.** Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of theIEEE conference on computer vision and pattern recognition.

[37] **Ross Girshick. 2015**. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision. 1440–1448.
[38] **Ross Girshick. 2015.** Fast r-cnn. In Proceedings of the IEEE international conference oncomputer vision. 1440–1448.
[39] **S. Bai and S. An**, "A Survey on Automatic Image Caption Generation," Neurocomputing, 13 April 2018.

[40] **S. Yan, F. wu, J. Smith and W. Lu,** "Image Captioning via a Hierarchical Attention Mechanism and Policy Gradient Optimization," LATEX CLASS FILES, vol. 14, 11 January 2019.
[41] **Simonyan, Karen, and Andrew Zisserman.** "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
Springer, 529–545.

[42] **Sun Chengjian, Songhao Zhu, Zhe Shi,** "Image Annotation Via Deep Neural Network", Published in: 2015 14th IAPR International Conference on Machine Vision Applications (MVA).

[43] **T. Mikolov, K. Chen, G. Corrado, and J. Dean.** Efficient estimation of word representations in vector space. In ICLR, 2013.

[44] **V. Ordonez, G. Kulkarni, and T. L. Berg.** Im2text: Describing images using 1 million captioned photographs. In NIPS, 2011.

[45] **Venkatesh N. Murthy, Subhransu Maji, R Manmatha,** "Automatic Image Annotation using Deep learning representations", ICMR '15 Proceedings of the 5th ACM on International Conference on Multimedia Retrieva.

[46] **William Fedus, Ian Goodfellow, and Andrew M Dai. 2018**. Maskgan: Better textgeneration via filling in the _. arXiv preprint arXiv:1801.07736.

[47] **William Fedus, Ian Goodfellow, and Andrew M Dai. 2018**. Maskgan: Better textgeneration via filling in the _. arXiv preprint arXiv:1801.07736.

[48] **Yao, Benjamin Z., et al.** ”I2t: Image parsing to text description.” Proceedings of the IEEE 98.8 (2010): 1485-1508.

[49] **Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. 2014.** Improving imagesentence embeddings using large weakly annotated photocollections. In European Conference on Computer Vision.

[50] **Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017**. Semantic compositional networks for visual captioning.

Source Code

```
# -*- coding: utf-8 -*-

"""Image Captioning.ipynb


Automatically generated by Colaboratory.


Original file is located at


https://colab.research.google.com/drive/1
SLmYKFhP7qWc5xBprko-bAB0krIDJJgY


# Image Caption Generator
Image caption generator is a model which
generates caption based on the features
present in the input image.


## Introduction
The basic working of the project is that
the features are extracted from the
images using pre-trained ResNet50 model
and then fed to the LSTM model along with
the captions to train. The trained model
is then capable of generating captions
for any images that are fed to it.
```

```
## Dataset

The dataset used here is the **[FLICKR
8K](https://forms.illinois.edu/sec/171339
8)** which consists of around 8091 images
along with 5 captions for each images.
If you have a powerful system with more
than 16 GB RAM and a graphic card with
more than 4 GB of memory, you can try to
take **[FLICKR
30K](http://web.engr.illinois.edu/~bplumm
e2/Flickr30kEntities/)** which has around
30,000 images with captions.


## Dependencies

* Keras
* Tensorflow GPU
* Pre-trained ResNet50 weights
* NLTK
* Matplotlib


## Steps to follow:


### Importing the required libraries
```

```
"""




import numpy as np
import matplotlib.pyplot as plt
from keras.models import Model,
load_model
from keras.preprocessing import image
import time
import pickle
from keras.preprocessing.sequence import
pad_sequences
from tensorflow.keras.utils import
to_categorical
from keras.layers import *


"""## Importing the image dataset and its
respective captions"""


import os
os.getcwd()


# Commented out IPython magic to ensure
```

```
Python compatibility.

from google.colab import drive

drive.mount('/gdrive')

# %cd /gdrive


path = '/gdrive/My Drive/Image Caption

Generator'


# Reading Caption wali file


with

open(path+"/train_val_data/Flickr8k.token

.txt") as f:

    captions = f.read()


print(captions[:1000])


captions = captions.split("\n")[:-1]


# dict key : 'img_name' , value  : list

of all captions

descriptions = {}


for ele in captions:
```

```
    i_to_c = ele.split('\t')
    img_name = i_to_c[0].split(".")[0]
    cap = i_to_c[1]

    if descriptions.get(img_name) is
None:
        descriptions[img_name] = []

    descriptions[img_name].append(cap)




len(descriptions)

descriptions['1000268201_693b08cb0e']



## Data Cleaning

## Cleaning the captions for further
processing
The caption dataset contains
punctuations, singular words and
numerical values that need to be cleaned
```

```
before it is fed to the model because
uncleaned dataset will not create good
captionsfor the images
"""

"""
1. lower case
2. remove punctuations
3. remove words length less than 2
"""
import re

def clean_text(sample):
    sample = sample.lower()

    sample = re.sub("[^a-z]+", " ",
sample)

    sample = sample.split()

    sample = [s for s in sample if
len(s)>1] # list comprehension

    sample = " ".join(sample)
```

```
    return sample


clean_text("My hobby is cricket, father's
I play badmintion....#### 1234 dog
walking..!")


"""# Modifiying all captions - cleaned
captions"""


for key, desc_list in
descriptions.items():
    for i in range(len(desc_list)):
        desc_list[i] =
clean_text(desc_list[i])


descriptions['1000268201_693b08cb0e']


# Commented out IPython magic to ensure
Python compatibility.
os.getcwd()
# %cd /gdrive/MyDrive/Image-Captioning-
CB-main/


f = open('descriptions.txt', 'w')
```

```
f.write(str(descriptions))
f.close()

# finding unique vocabulary

vocabulary = set()

for key in descriptions.keys():
    [ vocabulary.update(i.split()) for i
in descriptions[key]]

print("vocabulary size : " ,
len(vocabulary))

# All words in description dictionary
all_vocab = []

for key in descriptions.keys():
    [ all_vocab.append(i) for des in
descriptions[key] for i in des.split()]

print("total words appearing : " ,
len(all_vocab))
```

```
from collections import Counter

counter = Counter(all_vocab)

dic_ = dict(counter)

sorted_dic = sorted(dic_.items(), key =
lambda x: x[1], reverse=True)

threshold_value = 10

d = [x for x in sorted_dic if
x[1]>threshold_value]

len(d)

all_vocab = [x[0] for x in d]

len(all_vocab)

"""# Load Training and Testing Data
```

```
# Read Train File
"""

with
open(path+'/train_val_data/Flickr_8k.trai
nImages.txt') as f:
    train = f.read()

train = [e[:-4] for e in
train.split('\n')[:-1]]

"""# Read test File"""

with
open(path+'/train_val_data/Flickr_8k.devI
mages.txt') as f:
    test = f.read()

test = [e[:-4] for e in
test.split('\n')[:-1]]



"""## Adding start and end sequence
```

```
tokens for each captions
Start and end sequence has to be added to
the tokens so that it is easier to
identify the captions for the images as
each of them are of different length
"""

train_descriptions = {}

for t in train:
    train_descriptions[t] = []
    for cap in descriptions[t]:
        cap_to_append = "startseq " + cap
+ " endseq"

train_descriptions[t].append(cap_to_appen
d)

len(train_descriptions)

"""# Data Preprocessing - Images"""

from keras.applications.resnet50 import
```

```
ResNet50, preprocess_input


model = ResNet50(weights = 'imagenet',
input_shape = (224,224,3))


model.summary()


model_new = Model(inputs = model.input,
outputs =  model.layers[-2].output)


def preprocess_image(img):
    img = image.load_img(img,
target_size=(224,224))
    img = image.img_to_array(img)
    img = preprocess_input(img)
    img = np.expand_dims(img, axis = 0)


    return img


def encode_image(img):
    img = preprocess_image(img)
    fea_vec = model_new.predict(img)
    fea_vec =
fea_vec.reshape(fea_vec.shape[1], )
```

```
    return fea_vec

images =
path+"/train_val_data/Flickr8k_Dataset/"

start = time.time()

encoding_train = {}

for ix, img in enumerate(train):

    img = images+train[ix]+".jpg"

    p = encode_image(img)

    encoding_train[ img[len(images):] ] =
p


    if ix%100 == 0:
        print("Encoding image :" +
str(ix))

#print("Time taken in sec - " +
```

```
time.time() - start)

time.time() - start

start = time.time()

encoding_test = {}

for ix, img in enumerate(test):

    img = images+test[ix]+".jpg"

    p = encode_image(img)

    encoding_test[ img[len(images):] ] =
p



    if ix%100 == 0:
        print("Encoding image :" +
str(ix))

#print("Time taken in sec - " +
time.time() - start)
```

```
"""# saving features to disk"""


with open("./encoded_train_images.pkl",
'wb') as f:
    pickle.dump(encoding_train, f )


"""# loading features from disk"""


with open("./encoded_train_images.pkl",
'rb') as f:
    encoding_train = pickle.load(f)




with open("./encoded_test_images.pkl",
'rb') as f:
    encoding_test = pickle.load(f)


len(encoding_train)


len(encoding_test)
```

```
"""# Data Preprocessing - Captions"""


word_to_idx = {}

idx_to_word = {}


ix = 1


for e in all_vocab:

    word_to_idx[e] = ix

    idx_to_word[ix] = e


    ix +=1


word_to_idx['startseq'] = 1846

word_to_idx['endseq'] = 1847


idx_to_word[1846] = 'startseq'

idx_to_word[1847] = 'endseq'


vocab_size = len(idx_to_word) + 1

print(vocab_size)


all_caption_len = []
```

```
for key in train_descriptions.keys():
    for cap in train_descriptions[key]:

all_caption_len.append(len(cap.split()))

"""## Finding the max length of the
caption"""

max_len = max(all_caption_len)
print(max_len)



"""# Generator Function"""

def data_generator(train_descriptions,
encoding_train, word_to_idx, max_len,
num_photos_per_batch ):
    X1, X2, y = [], [], []

    n=0

    while True:
```

```
        for key, desc_list in
train_descriptions.items():
            n+=1

            photo =
encoding_train[key+'.jpg']

            for desc in desc_list:

                seq = [word_to_idx[word]
for word in desc.split() if word in
word_to_idx]

                for i in range(1,
len(seq)):
                    in_seq = seq[0:i]
                    out_seq = seq[i]

                    in_seq =
pad_sequences( [in_seq], maxlen=max_len,
value= 0, padding='post')[0]

                    out_seq =
```

```
to_categorical([out_seq],
num_classes=vocab_size)[0]


                    X1.append(photo)
                    X2.append(in_seq)
                    y.append(out_seq)


            if n == num_photos_per_batch:
                yield [[np.array(X1),
np.array(X2)] , np.array(y) ]
                X1, X2, y = [], [], []
                n =0


for i in
data_generator(train_descriptions,
encoding_train, word_to_idx, max_len, 3):
    X, y = i
    print(X[0].shape)
    print(X[1].shape)
    print(y.shape)
    break
```

```
"""# Word Embedding"""

embeddings = {}

with
open(path+'/train_val_data/glove.6B.50d.t
xt', 'r', encoding='utf-8') as f:
    for line in f:
        values = line.split()
        word = values[0]
        coeffs = np.array(values[1:],
dtype="float32")

        embeddings[word] = coeffs

def getOutputEmbeddings():

    emb_dim = 50
    embedding_matrix_output =
np.zeros((vocab_size, emb_dim ))

    for word, idx in word_to_idx.items():

        emb_vec = embeddings.get(word)
```

```
        if emb_vec is not None:
            embedding_matrix_output[idx]
= emb_vec

    return embedding_matrix_output

embedding_output = getOutputEmbeddings()

embedding_output.shape



"""# Model Architecture"""

#  image feature extractor model

input_img_fea = Input(shape=(2048,))
inp_img1 = Dropout(0.3)(input_img_fea)
inp_img2 = Dense(256,
activation='relu')(inp_img1)

# partial caption sequence model
```

```
input_cap = Input(shape=(max_len,))
inp_cap1 = Embedding(input_dim=
vocab_size, output_dim=50,
mask_zero=True)(input_cap)
inp_cap2 = Dropout(0.3)(inp_cap1)
inp_cap3 = LSTM(256)(inp_cap2)

decoder1 = add([inp_img2, inp_cap3])

decoder2 = Dense(256,
activation='relu')(decoder1)
output = Dense(vocab_size,
activation='softmax')(decoder2)



model = Model(inputs = [input_img_fea,
input_cap]  , outputs =  output )

model.summary()

model.layers[2].set_weights([embedding_ou
tput])
model.layers[2].trainable = False
```

```
model.compile(loss="categorical_crossentr
opy", optimizer='adam')

"""# Train Model"""

epochs = 10
number_photos_per_batch = 3
steps =
len(train_descriptions)//number_photos_pe
r_batch

for i in range(epochs):
    generator =
data_generator(train_descriptions,
encoding_train, word_to_idx, max_len,
number_photos_per_batch)
    model.fit_generator(generator,
epochs=1, steps_per_epoch=steps,
verbose=1)

model.save("best_model.h5")

model =
load_model("model_weights/model_9.h5")
```

```
model

"""# Predictions"""

def predict(photo_enc):
    in_text = "startseq"

    for i in range(max_len):
        sequence = [word_to_idx[word] for
word in in_text.split() if word in
word_to_idx]
        sequence =
pad_sequences([sequence], maxlen=max_len,
padding='post')

        y_pred =
model.predict([photo_enc, sequence])
        y_pred = np.argmax(y_pred)
        word = idx_to_word[y_pred]

        in_text += " "+word
```

```
        if word == 'endseq':
            break


    final_caption = in_text.split()
    final_caption = final_caption[1:-1]
    final_caption = "
".join(final_caption)
    return final_caption

rn = np.random.randint(0,1000)
img_id = list(encoding_test.keys())[rn]

photo_enc =
encoding_test[img_id].reshape((1,2048))
pred = predict(photo_enc)
print(pred)

path = images + img_id
img = plt.imread(path)
plt.imshow(img)
plt.show()
```