IMAGE PARAGRAPH GENERATION USING DEEP LEARNING

MAJOR PROJECT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
SIGNAL PROCESSING AND DIGITAL DESIGN

Submitted by:

Avanish Tiwari

2K20/SPD/03

Under the supervision of

Prof. Dinesh Kumar

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I **Avanish Tiwari,** student of M.Tech (Signal Processing and Digital Design), hereby declare that the project Dissertation titled "**Image Paragraph Generation Using Deep Learning**" which is submitted by me to the Department of Electronics and Communication Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi **Avanish Tiwari**

Date:

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Report titled "**Image Paragraph Generation Using Deep Learning**" which is submitted by **Avanish Tiwari**, **2K20/SPD/03** of Electronics and Communication Department, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi
Date:

Prof. Dinesh Kumar
SUPERVISOR

ACKNOWLEDGEMENT

A successful project can never be prepared by the efforts of the person to whom the project is assigned, but it also demands the help and guardianship of people who helped in completion of the project. I would like to thank all those people who have helped me in this research and inspired me during my study.

With profound sense of gratitude, I thank Prof. Dinesh Kumar, my Research Supervisor, for his encouragement, support, patience and his guidance in this thesis work. His kind comments and guidance let me complete this study in an improved manner.

I take immense delight in extending my acknowledgement to my family and friends who have been supporting me morally to keep going with this study and inspired me to come up with the expected results throughout this research work.

Avanish Tiwari

ABSTRACT

Recently, a neural network based approach to automatic generation of image descriptions has become popular. Originally introduced as *neural image captioning*, it refers to a family of models where several neural network components are connected end-to-end to infer the most likely caption given an input image. Neural image captioning models usually comprise a Convolutional Neural Network (CNN) based image encoder and a Recurrent Neural Network (RNN) language model for generating image captions based on the output of the CNN.

Generating long image captions – commonly referred to as *paragraph captions* – is more challenging than producing shorter, sentence-length captions. When generating paragraph captions, the model has more degrees of freedom, due to a larger total number of combinations of possible sentences that can be produced. In this thesis, we describe a combination of two approaches to improve paragraph captioning: using a hierarchical RNN model that adds a top-level RNN to keep track of the sentence context, and using richer visual features obtained from dense captioning networks. In addition to the standard MS-COCO Captions dataset used for image captioning, we also utilize the Stanford-Paragraph dataset specifically designed for paragraph captioning.

This thesis describes experiments performed on three variants of RNNs for generating paragraph captions. The *flat* model uses a non-hierarchical RNN, the *hierarchical* model implements a two level, hierarchical RNN, and the *hierarchical-coherent* model improves the *hierarchical* model by optimizing the coherence between sentences.

CONTENTS

List Of Figures

List of Tables

List of Symbols and Abbreviations

ML :	Machine Learning
CNN:	Convolutional Neural Networks
RNN:	Recurrent Neural Networks
LSTM:	Long Short Term Memory

CHAPTER 1

INTRODUCTION

The amount of data that is available in digital form has grown exponentially in recent years. Image Digitization and metadata associated with it are also a constituent of it. Large volumes of digital photographs are available readily thanks to the growing popularity of internet platforms such as social media sites, online news sites, and digital libraries. There is a rise in demand for automated solutions that help with the management, navigation, and search of these huge datasets as the amount of data available grows. Developments in machine learning and more specifically deep learning, have led to breakthroughs in detection and object recognition [1]. Automatically describing the contents of an image, in particular, is gaining immense popularity.

For humans, just a glance is well enough to describe an image [2]. The process that goes into it can be broken down into 1) Visual Space Perception. 2) Conversion of visual information into language space.

3) Generation of a description of the scene in a human-understandable form. In other words, this involves the translation of information from the visual domain to the textual domain. People, in general, are well versed in both these domains [3]. For computers, understanding about images is very trivial. This huge gap between human understanding of images and low-level features extracted by computers is a big challenge to tackle in the automatic generation of image captions [4,5].

Recently, Image captioning has seen a neural network-based renaissance. The objective of image captioning is to describe objects, actions, and other details present in an image in natural language. Most of the research in the captioning domain has stressed on single-sentence captions, but the amount of information with this kind of captions is very limited. A single sentence can only describe the image with minimal details. Utilizing neural networks to create brief picture captions, studies have moved to lengthier, multi-sentence image captions with 5 to 6 lines, each approximately having a length of 10-12 words. An elongated, paragraph explanation can be really useful in operations such as image retrieval, video transcription, and many operations in need of systems for automatically reasoning about pictures.

There are multiple methods available for image paragraph generation, they are: -

Long-Term Recurrent Convolutional Network: the input can either be an image or a sequence of images obtained from a video frame. The input is provided to a CNN which forms a vector representation of the image after recognizing activities in the image. This vector representation is then given to a LSTM model where a word is generated and a caption is obtained [4].

Visual Paragraph Generation: gives a coherent and paragraph describing the image. Attention detection is used to detect semantic regions in an image. Sentence generation is then done one by one and a paragraph is generated [14].

Recurrent Neural Network (RNN): It is a customised neural network designed to analyse data sequences with time stamp indexes ranging from 1 to t. RNNs are better suited to tasks involving progressive inputs, like speech and linguistic. With NLP, in order to predict a word, one needs to have idea about the word preceding this word [6].

Gated Recurrent Unit (GRU): Cho et al. presented a novel breakthrough termed Gated Recurrent Unit. It has gating units, similar to LSTM units, which alter the flow of information movement inside unit without requiring a discrete memory cell. The update and reset gates are calculated by GRU to govern the flow of information via each hidden unit. The update gate is calculated by means of the present input and the prior time step's concealed state. This gate specifies how much of each new and old memory segment must be included into the final memory. The reset gate is computed in the same way, but with a changed set of weights. It regulates the interaction seen between old memory and the new memory's incoming input data.

1.1 Machine Learning (ML)

A branch of computer science that enables computers to do things without being expressly programmed to do so. It gives computers a capability that makes them more similar to humans: ability to learn. The data is fed to a generic algorithm which then shapes logic on the basis of information specified.

ML as a field branched from Artificial Intelligence (AI). The objective of AI was to develop highly capable and intelligent machines. However, it was unable to be programmed for complex and constantly evolving challenges. The approach then moved towards making machines learn from themselves.

ML is the most extensively utilised method for predicting or categorising data in order to help humans make crucial decisions. In order to learn from prior experiences and analyse historical data, ML algorithms are made to develop understanding from

examples. Building models alone isn't enough. In order for the model to deliver reliable results, it must be appropriately optimised and tuned. The hyperparameters are fine-tuned using optimization techniques to produce the best possible result. As it trains over and over, it can recognise patterns for making accurate conclusions. When a new input is provided to ML model, it uses previously learnt patterns on new data to produce forthcoming forecasts. Models may be optimised using a variety of standardised ways, depending on the ultimate precision. As a consequence, the ML model learns from fresh data and improves its performance.

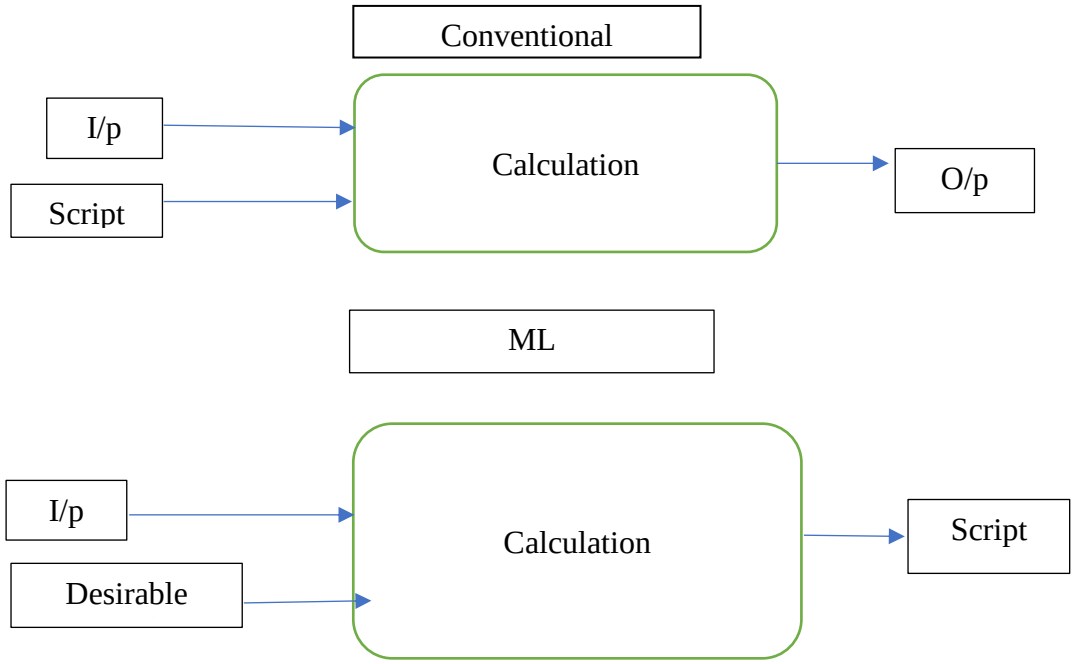


Fig 1.1 ML vs Conventional Programming.

1.1.1 Supervised Learning

Most widely used ML technique is supervised learning. It is the process of generating a mechanism from sample input-output pairs that translates an input to an output. It creates a function using annotated data for training, usually consists of a collection of training instances. Every sample in supervised learning is made up of an input object (usually a vector) and a target output (also called the supervisory signal). A supervised learning algorithm examines the training data and generates an extrapolated equation which can be applied to fresh cases. Training a kid with provided data is akin to supervised learning. We may send these example-label pairs to something like a training algorithm one by one if the data is in the form of instances with labels, causing the system to predict whether or not the response is accurate. The algorithm will

ultimately adapt to guess the true extent of the sample-to-label association. Once completely trained, the supervised learning algorithm will be able to analyze a fresh, never-before-seen sample and predict an appropriate label for it.

Supervised learning is used in the majority of practical machine learning. When you have an input variable (x) and an output variable (Y), supervised learning occurs when you use an algorithm to acquire the mapping function from the input to the output.

The aim is to assess the mapping function to the extent in which the output variables (Y) can be predicted for incoming input data (x). Because the process of learning is overseen by a teacher, the process of an algorithm learning from the training dataset is referred to as supervised learning. Task-oriented supervised learning is a common description. This is laser-focused on a single project, providing the system with more and more examples until it is able to complete that task properly.

1. **Regression:**

A target prediction value is calculated using independent factors. It's largely used in forecasting and figuring out how variables are related. Continuous values can be estimated or predicted using regression.

2. **Classification:**

The technique of labeling the outcome is referred to as "classification.". It's a classification problem if the data is discrete or categorical..

1.1.2 Unsupervised Learning

Unsupervised learning is a sort of ML in which the model is not supervised by anyone. Model discovers information byitself. This normally operates with unlabeled data, searching for formerly overlooked trends in such a dataset with no pre-existing labels and no human oversight. In contrast to supervised learning, which often depends on user data, unsupervised learning, also known as self-organization, allows for the modelling of probability densities across inputs. Without utilising known or labelled outcomes as a reference, unsupervised machine learning approaches discover patterns from a dataset. It is the process of teaching a computer to operate on unmarked or unclassified data without supervision by using unlabeled or unclassified data. The machine's goal is to classify unordered data based on similarities, trends, and discrepancies without any prior data training. There is no teacher present, unlike supervised learning, which implies that computer will not be instructed. Machine, therefore, can extract very few hidden structures in unlabelled data. For example, if we give the machine a few photographs of bus and car to categorise, the computer will

categorise them based on their similarities, patterns, and differences because it has no prior knowledge of the attributes of buses and cars. Unsupervised learning algorithms, as opposed to supervised learning algorithms, allow you to complete more complex computing jobs. But at the other hand, unsupervised learning might be more unpredictable than other natural learning approaches. For unsupervised learning issues, there are two types of algorithms.:

- Clustering: A clustering problem is one in which you wish to uncover the data's intrinsic groupings, such as classifying clients based on their purchasing habits.
- Association: An association rule learning problem is one in which you wish to find rules that describe huge chunks of your data, such as persons who buy X also buy Y.

1.1.3 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning technique that allows an agent to learn by trial and error in an interactive environment using feedback from its own actions and experiences. Machines primarily learn from previous experiences and strive to provide the best feasible answer to a problem. It is the process of teaching machines to make a series of judgments. Though both supervised and reinforcement learning involve mapping between input and output, reinforcement learning uses rewards and punishments as signals for positive and negative behaviour, unlike supervised learning, which provides the agent with a right set of behaviours for executing a task. The most effective technique to hint a machine's creativity is to use reinforcement learning.

1.2 Natural Language Processing

Natural language processing (NLP) refers to a computer program's ability to comprehend spoken human language. NLP is an artificial intelligence component (AI). Because computers usually require people to "talk" to them in a highly structured, precise and unambiguous programming language or by a limited set of well enunciated voice instructions, developing NLP applications is difficult. Human speech, on the other hand, isn't always precise; it's frequently ambiguous, and its grammatical structure is influenced by a variety of factors, including slang, regional dialects, and social context.

1.3 Motivation of the Work

We must first comprehend the significance of this issue in real-world circumstances. Consider a few scenarios in which a solution to this challenge could be extremely beneficial.

- Self-driving automobiles – Automatic driving is one of the most difficult problems, and captioning the area around the car can help the self-driving system.
- Aid to the blind — We can develop a product for the blind that will lead them on the roads without the need for anybody else's assistance. This can be accomplished by turning the scene to text and then the text to voice. Both are now well-known Deep Learning applications.
- CCTV cameras are ubiquitous today, but if we can generate useful captions in addition to watching the world, we can trigger alarms as soon as criminal conduct is detected.
- This is likely to help minimize crime and/or accidents. Automatic captioning can help make Google Image Search as excellent as Google Search by converting every image into a caption first, and then searching based on the caption.

1.4 Problem Statement

The project's main goal is to use deep learning and natural language processing techniques to generate textual descriptions in the form of 4-5 sentences for each input image.

1.4.1 Organization

This thesis studies three variants of Encoder-Decoder (explained in Section 2.1) neural network architectures for producing paragraph-length picture descriptions. In Chapter 2, we'll go through the fundamentals of picture captioning. In Chapter 3, we'll look at how expanding the fundamental picture captioning model by adding another level of hierarchy might result in longer image descriptions. In Chapter 4, we'll talk about the experiments Further, Chapter 5 talks about the conclusions and future scope.

Sections 2.3.3 and 3.2.1 show how to construct the fundamental building pieces needed to produce picture descriptions that are either concise captions or large paragraphs. When discussing the process of creating short, single-sentence captions, we will use the term "image captioning," and when discussing the task of creating lengthier, multi-sentence captions, we will use the term "paragraph captioning." Furthermore, models with a single level of the RNN [42, pp. 367–415] hierarchy will be referred to as "flat," whereas models with several levels of RNN hierarchy will be referred to as "hierarchical." It's difficult to judge machine-generated captions.With longer captions, the problem becomes even worse. Several measures for assessing the quality of generated captions have been presented. Human evaluation can also be employed to evaluate caption quality. For automatic caption evaluation, we will use the popular BLEU [20], Meteor [32], and CIDEr [14] metrics in our experiments. In Section 4.2, we'll look at a variety of human evaluation standards.

The PyTorch neural network programming framework was used in our experiments, which were run in Python 3.5. We'll explore into paragraph captioning using a hierarchical RNN with two recurrent network layers in Sections 4.4.2 and 4.4.3, where the first RNN provides phrase context and the second utilizes the context to learn individual tokens, which can be words or punctuation marks. Figure 1.1 depicts the relationship between the various models studied in Chapter 4 experiments. To produce paragraph captions, we train three alternative models: flat, hierarchical, and hierarchical-coherent hierarchies. The flat model is a simple image captioning model with only one RNN layer decoder. In the hierarchical model, the flat decoder receives an extra RNN level. Finally, the hierarchical-coherent model creates a "coherence" link between the two layers of the RNN hierarchy. To pre-train our models, we employ either the MS-COCO Captions [10] or the Visual Genome Regions [31] datasets.

CHAPTER 2

Image Captioning

Image Captioning can be described as automated production of brief natural language narration of the contents of image [39]. For this objective, neural networks have recently gained popularity [5, 6, 31]. Our picture captioning model generated a few captions, as seen in Figure 2.1.

The basic model for image captioning is explained in this chapter. This thesis calls it the flat-model in order to distinguish it from hierarchical models that will be discussed later. This is how the current chapter is structured: The fundamentals of the ED based framework and the speech model brought to use to perform image captioning are explained in Section 2.1, the input characteristics are described in Section 2.2, and the decoder component is described in Section 2.3. In Section 2.4 entire pipeline for image captioning is defined. Section 2.5, which concludes the chapter, looks at several newly presented enhancements to the baseline captioning paradigm.

2.1 Encoder-Decoder Model

The Encoder-Decoder architecture is the central piece of neural image captioning models [40]. The architectural approach is inspired by machine translation's neural sequence-to-sequence models [41].

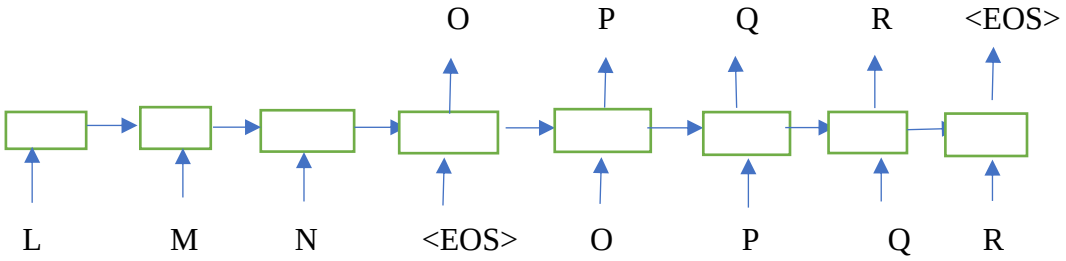


Fig 2.1: A straightforward chain model is shown. "LMN" is the input sequence, while "OPQR" is the output sequence [63]**.**

Captioning can be conceived of as the translation of an image into text. The application of methods from text-to-text translation is enabled by the analogy with machine translation. Figure 2.1 depicts a model which accepts "LMN" sequence as input and memorizes to predict "OPQR" sequence on the basis of input sequence.

Usually, two different subnetworks are used to implement chain models. One such model is shown in Figure 2.2, which consists of two independent RNNs, one of which encodes the input while the other one predicts the output on the basis of input .

The **encoder(E)**, as its output gives context k after processing the input which can either be a sentence in base language or an image

the **decoder(D)**, on the other hand, is a different network that creates the intended output and takes context k as its input.

The advantage of a neural ED architecture is that it is made up of distinct components that can be trained end-to-end by making use of backpropagation. As shown in Figure 2.2, at each sampling instant, the information can be made accessible to the decoder or supplied to decoder only on first timestep. The image theme, at each time sample is given to the top-level RNN.

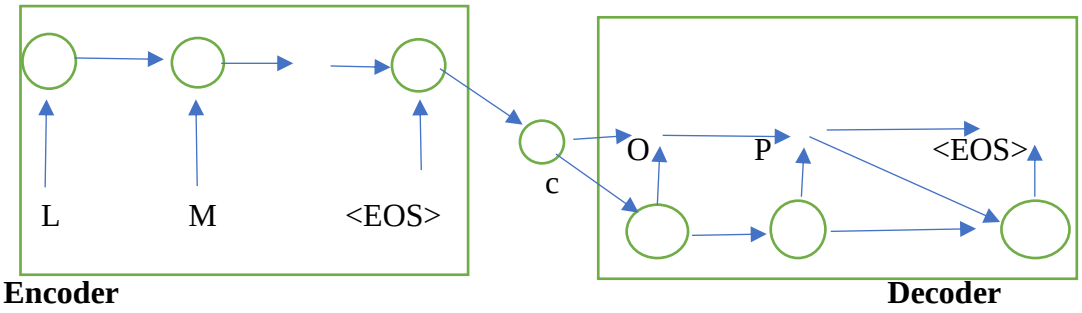


Figure 2.2: Encoder-Decoder design.

The ED architecture was used to generate image captions by Vinyals et al. [6]. The model suggested served as benchmark for neural ED image captioning.[42] uses a CNN+RNN setup where CNN generates a vector representation by encoding image picture into a vector representation and RNN, aided by the language model converts vector representation into a description that humans can understand.

2.1.1 Speech Model for Image Captioning

Vocabulary can be defined as a *B*-sized ordered and pre-defined set of viable tokens that model can use to generate text. Each vocabulary token v_j is saved in the associated index position *j*. Token can either be a punctuation character or a unit word. For the task of image captioning, use of "word" can use either a punctuation letter or an individual word. In its most basic form, a language model establishes allocation of probability for each and every token v_j in the dictionary in this way:

$$\sum_{j=1}^{B} p(v_j) = 1 \tag{2.1}$$

Speech models are helpful to construct token patterns after we have option of conditioning the probability distribution of next token v_t. on the flow of prior tokens $v_1, \ldots, v_{t-1}$. Chain rule can be used to estimate joint probability for a N- sized token sequence:

$$p_t(v_t|v_{t-1},\ldots,v_1) = p_{t-1}(v_{t-1}|v_{t-2},\ldots,v_1)\, p_{t-2}(v_{t-2}|v_{t-3},\ldots,v_1)\ldots p_1(v_1), \tag{2.2}$$

$$p(v_t, \ldots . v_1) = \prod_{t=1}^{N} p_t(v_t|v_{t-1}, \ldots, v_1) \tag{2.3}$$

While training, model is expected to allocate a greater probability to sequence of words that have greater probability of showing up in human descriptions, such as:
When training language models, the goal is for the model to

P(four boys playing cricket) > *p*(boys four playing cricket).

A conditional language model almost resembles a regular language model, the only difference being that in case of a conditional model, each and every token of the pattern has same conditioning circumstances is defined in the same way as a regular language model, with the exception that all tokens in the sequence have the same conditioning circumstance k:

$$p(v_t, \ldots . v_1|k) = \prod_{t=1}^{N} p_t(v_t|k, v_{t-1}, \ldots, v_1) \tag{2.4}$$

Let ($v_1, \ldots v_N$) be a token sequence depicting a N-length caption, non-vocabulary terms denoted by a distinctive token <UNK> . Every vocabulary token has a one-hot representation $\boldsymbol{r_t}$, which is B-sized vector all elements of which are set at value zero except for the one having each and every element places as zero leaving only that index position that corresponds with vocabulary token v_t. As a result, caption is coded in the form of N one-hot vectors with pattern r=($\boldsymbol{r_1}, \ldots, \boldsymbol{r_N}$)

Image captioning aims to develop understanding about certain conditional language model such that the probability for next token is dependent on both , input image Q and tokens generated earlier within the sequence.

$$p(\boldsymbol{r}|\boldsymbol{Q}) = p(\boldsymbol{r_N}, \ldots, \boldsymbol{r_1}|\boldsymbol{Q}) = \prod_{t=1}^{N} p_t(r_t|\mathrm{Q}, r_{t-1}, \ldots, r_1) \quad (2.5)$$

In order to increase the trainable parameters Θ for the models, the following function was proposed with Θ^* being the best value that can be attained by [6]

$$\odot^* = \arg max_{\odot} \sum_{(Q,r)} \log p(r|Q; \odot) \quad (2.6)$$

In logarithmic probability space, conditional language model is depicted by Equation 2.7. Q denotes the input image and true caption is r. The chain rule can be used to describe the conditional probability, in which the likelihood of the token $\boldsymbol{r_t}$ being generated on location t in caption r, which contains N number of tokens relies upon input image and also on sequence of tokens that are in positions 1, ..., t1. As a result, for all tokens in the chain r, in order to represent joint probability, equation can be defined as:

$$\odot^* = \log p(r|Q; \odot) = \sum_{t=1}^{N} \log p(r_t|Q, r_{t-1}, \ldots., r_1; \odot) \quad (2.7)$$

This model employs a method known as teacher forcing [42], that entails providing the model the with ground truth word $\boldsymbol{r_t}$ at each training sampling time. Teacher forcing is not possible at the inference stage. Model therefore, is fed with its output tokens of its own s_t at each sampling time instance

$$\log p(r|Q; \odot) = \sum_{t=1}^{N} \log p(r_t|Q, s_{t-1}, \ldots., s_1; \odot) \quad (2.8)$$

where $s_{t-1}, \ldots, s_1$ are the best guesses model was able to make at each time step before t.

2.2 Encoder: Image Features Extractor

The image must be translated into an adequate representation before the language model can be conditioned on it. An image feature is such type of representation. Feature extraction can be defined as the process involving selectively picking of certain object features from the image. is the technique of extracting picture features. Depending upon purpose, we wish to encode different properties of the image

into features. Low-level features represent the image's visual qualities such as colour, texture, local contrast/edges, and forms (lines, points, circles). It's possible that a neural network-based solution isn't even required to extract such properties. The high-level features, also known as semantic features [43] encode visual attributes in the form of objects and the relationship between different objects.

CNN that have previously been trained previously on any image classification task have emerged as a feasible approach to extract semantic features. These extracted features can then be used for captioning task. These kinds of networks commonly are pre-trained on the ImageNet [44] image classification challenge, in which the trained model must select one class that is most probable out of 1000 choices available for each input image. Transfer learning [45] is a process that allows neural networks trained for a particular task to be used for some other activity. Transfer learnings opens up the possibility of utilizing ImageNet trained CNNs for supplying input characteristics to models trained for tasks little complex than simple classification. Initial layers of networks learn general, basic features which can be used for a variety of tasks. The latter layers however lose generality and become task specific.[46]. When transfer learning is used, the output obtained from a layer sufficiently deep is chosen for subsequent usage. ResNet [2], GoogLeNet (also known as Inception network) [49], VGG16 and VGG19 [48], and AlexNet [47], are only a few of the image classification networks that have lately developed.

ResNet has lately acquired interest as a model for extracting visual features for the task of captioning [26, 50]. The ResNet architecture overcomes disappearing gradients and explosive gradients [51, 18].

With increase in network layers, gradients may vanish. If each gradient's absolute value is a value that nears about to 0, the result of backpropagation will for the initial network layers result in a gradient approaching zero. Gradients of this type are said to be fading since they no longer effect learning. In the best-case situation, tiny gradients indicate that network will converge after a long time. The worst would be the condition if learning stalls before it converges.

The initial objective of ResNet design is to solve the vanishing gradient drawback. The ResNet model is made up of several residual blocks, one of which is depicted in Figure 2.3.

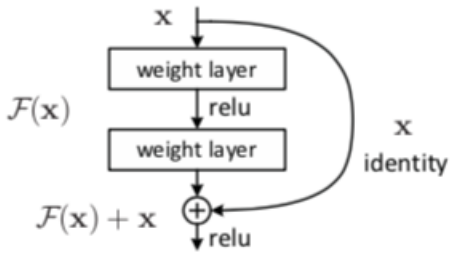


Fig. 2.3: ResNet architecture residual block [2].

Each residual block is made up of a sub-network together with a "shortcut link" that serves as an identity function ensuring a connection of the present layer with the one before it. Two in-between layers learn a mapping F(x) as shown in figure, and the alternative link attaches x to the intermediate layer output. This helps the network in learning at the minimum the identity function between higher and lower levels by ensuring that gradients flow to lower layers while avoiding in-between layers.

In a scenario where the value of gradient is very large in comparison to 1.0, spreading gradients over several layers leads to initial layers with explosive gradients approaching infinity, preventing updating of weights. The residual blocks, according to the ResNet architecture's developers, aids in minimizing the effects of exploding gradients [2].

In all, there are several ResNet variations available, having similarity with one another differing only in number of layers. In this work, tests have been performed with ResNet-152 architecture because of recent work in picture captioning [50].

The whole network diagram for a 34 layer miniature version of ResNet is shown in Figure 2.5. This smaller network's architecture is based on the same concepts as the 152-layer version we're employing. The ResNet network is exhibited next to a 34-layer convolutional net and a VGG19 network with no alternative links so as to compare.

By deleting the last classification layer, which lists the possibility for the image in question to be a part of one out of the 1000 classes available, it becomes possible to get hold of the image features from ResNet-152 that are required for image captioning. Endmost layer generates an average-pooled vector with D_{ResNet} = 2048 dimensions. It serves as the basis for our picture feature vector. The recovered feature vector post operation of matrix multiplication get transferred to the input dimension of the decoder. More information on the methodology implemented to use ResNet-152 network for picture feature extraction is provided in the following section.

2.2.1 Extracting Features for ResNet Model

Each image is preprocessed before ResNet-152 features are extracted. We start by resizing every image and making it into 256 x 256 pixels size . For each 3-dimensional image pixel x RGB channels are then adjusted using the per-channel standard deviations σ^* and averages $\boldsymbol{\mu}^*$

The normalisation settings about for $\boldsymbol{\mu}^*$ and σ^* are calculated from ImageNet [44] data and are published in the PyTorch documentation. This step is necessary since the ResNet-152 network was pre-trained on pictures that were all flattened in the same way.

Once the normalised pixel values for the 256 x 256 image taken as input have been collected, a total of five 224 x 224 dimensions image portions are cropped, and four out of all these images are aligned at ends of the original image. Moreover, every crop is overturned horizontally, yielding ten cropped photos in total. For executing this procedure in one step, there is one built-in function provided by PyTorch by the name of ten-crop.

Following the creation of the ten-crop images, each one is run with the help of a pre-trained ResNet-152 model. For each of the input crops, we maintain the output of the second last layer of the pre-trained model, which has a size of F_{Res} = 2048. There are numerous options for combining ten-crop image outputs into a single image. We try out two different strategies: computing the 10 produced vectors' element-wise mean (advanced pooling) and obtaining the element-wise maximum (maximum pooling). Before training, the pooled output features are pre-computed and saved onto disc to ensure that they can be recovered if need arises and avoiding time-consuming calculation.

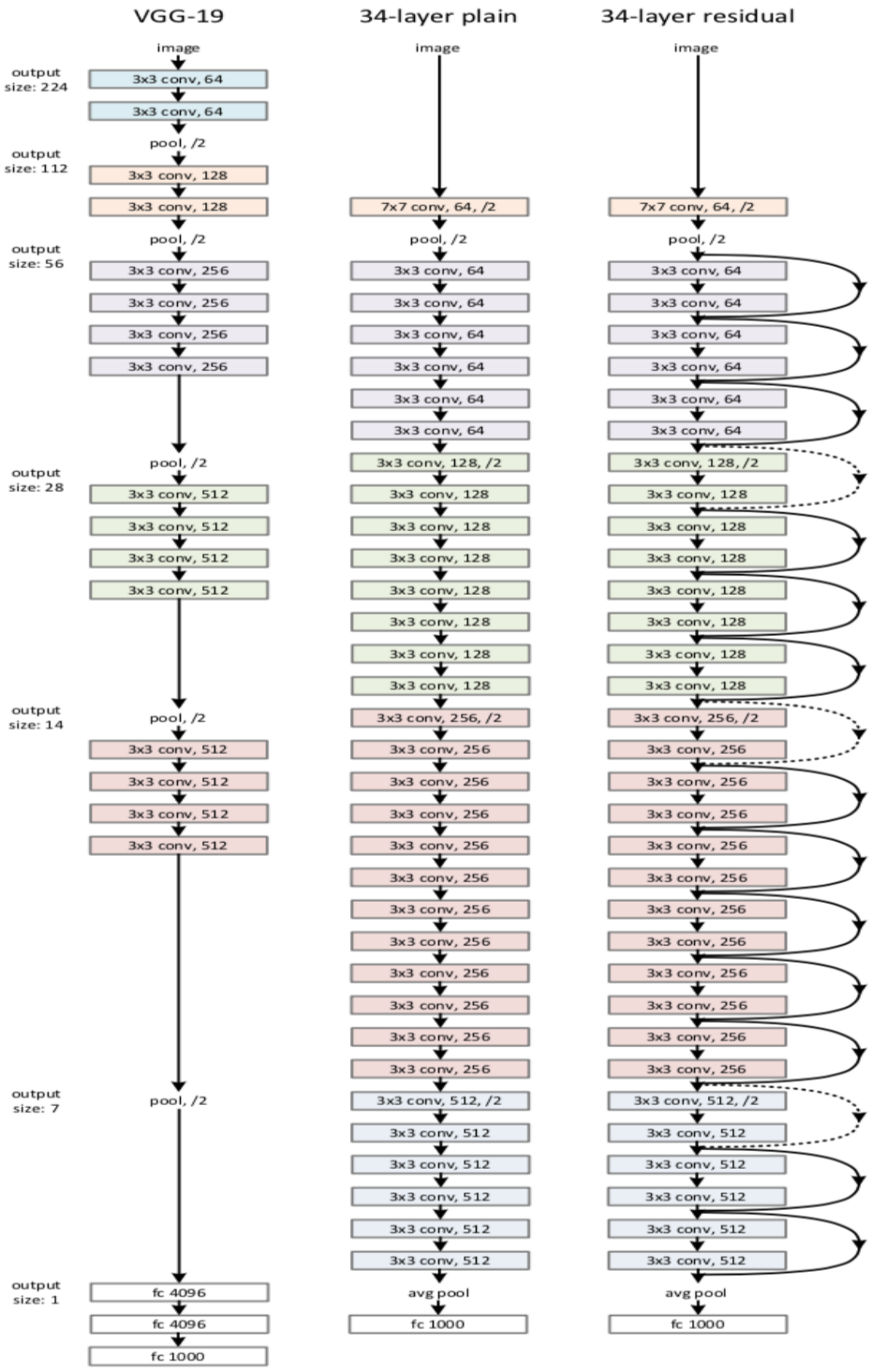


Fig. 2.4: VGG19 (left), ResNet-34 (right), and a 34-layer network without shortcut interconnections (centre) [2].

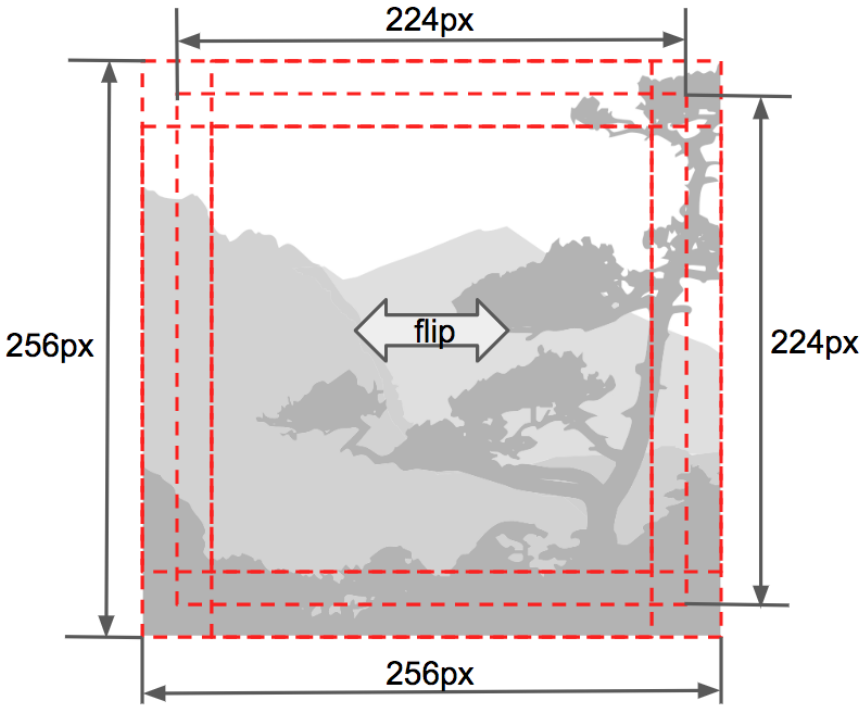


Fig. 2.5: Cropping five 224 x 224 sections from a 256X256 photo and overturning horizontally each chopped section creates a ten-crop image:

$$x_{normalized} = \frac{x - \mu^*}{\sigma^*} \tag{2.9}$$

Table 2.1: Outcomes of the MS COCO captioning challenge employing ResNet-152 ten-crop and random crop properties.

Evaluation Metrics	Single Random Crop	Ten Crop
CIDEr	85.18	87.17
METEOR	23.93	24.26

The pooled features also have F_{Res} dimension as the features produced by single input image in absence of ten-crop yet deliver higher experimental results on the image captioning job. Table 2.1 depicts exactly the same. The model beats standard dataset enhancement technique involving randomly cropping every image at every repetition when employing ten-crop features, as judged by CIDEr [14] and Meteor [32] scores 6 (see Section 4.2). Each run over the data is sped up by the pre-calculated ten-crop characteristics, allowing you additional scope for testing. Meanwhile, random

cropping, necessitates running entire ResNet-152 CNN for every picture, greatly slowing the learning.

2.3 Decoder: Generating Captions

Decoder's picture attributes are transferred onto a G-dimensional vector that contains important information about the input images and can be regarded as a badly condensed representation of the image data. [46]. The decoder's goal is to learn how to rely on this representation of a picture and offer a description that perfectly represents the actual image's characteristics.

2.3.1 Common Depiction for Words and Pictures

Learning a common representation [37] of images and sentences, as well as the usage of a recurrent network-based language model [52], make it possible to condition the language model on the E-dimensional image context vector c. The shared representation allows for the same latent space to be used to encode image attributes and individual words in the caption. Recurrent network-based language models, on the other hand, enable us to simulate the probability distributions of specific word sequences given an input image.

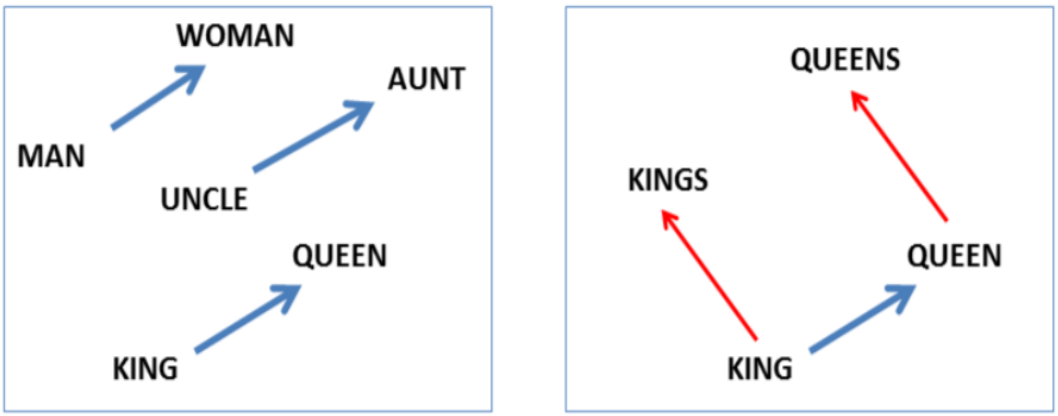


Fig. 2.6: Interactions are projected in lower dimensions [53].

[54] suggested utilising a distributed, continuous representation to encode each word in a vocabulary rather than the usual "bag-of-words" method, in which each word is stored as a discrete 0 or 1 indication at a specific place in a one-hot vector of size V. In bag-of-words, each word in a caption would be a lengthy and sparse vector comprising primarily zeros, given a vocabulary of size V = 10000 words. On the other

hand, if this sparse representation could be translated into continuous space, neural network models might be trained using these changed inputs.

Such a representation is provided by word embeddings. Every word in vocabulary is mapped to an P-dimensional, continuous embedding, where P<< V is the encoder output size and E is the size of the embedding. Instead of mapping each word to a discrete index, we now have a continuous "neighbourhood" of words in an E-dimensional embedding space, where similar words are frequently neighbours. This is due to the way embeddings are built; for example, if two words appear in similar contexts in the training data, they are more likely to occupy neighbouring embedding space places [55]. When projected to lower dimensions, relations between related concepts are frequently represented as vectors in embedding space [53], as shown in Figure 2.7. Many such relations can theoretically be contained at once inside each individual embedding due to the embedding's still large dimensionality [53, 37].

We still lack a shared representation between words and visuals at this time. The concept of compositional semantics [56], which states that each embedding vector can represent either a word or a full phrase, underpins shared representation between individual words and images. Compositional distributed representations of whole sentences can be combined with image features to create a multi-modal [37] representation space in which the compositional image caption representation maps to the same vector space as image features.

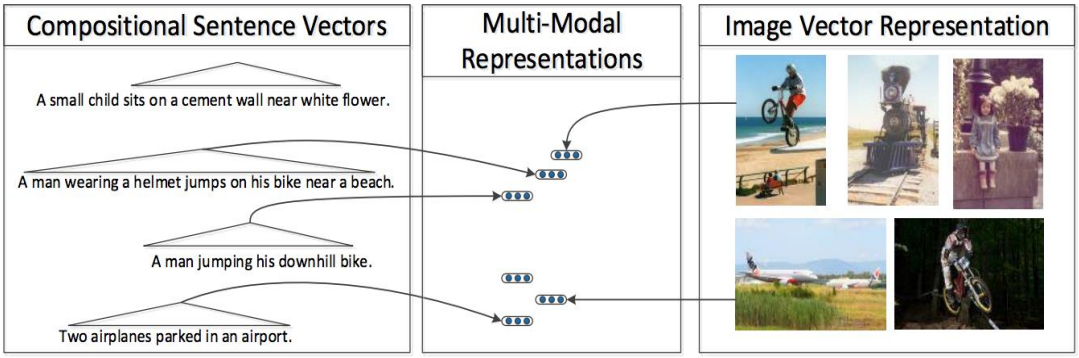


Fig. 2.7: Common vector representation of pictures and sentences**.** [62].

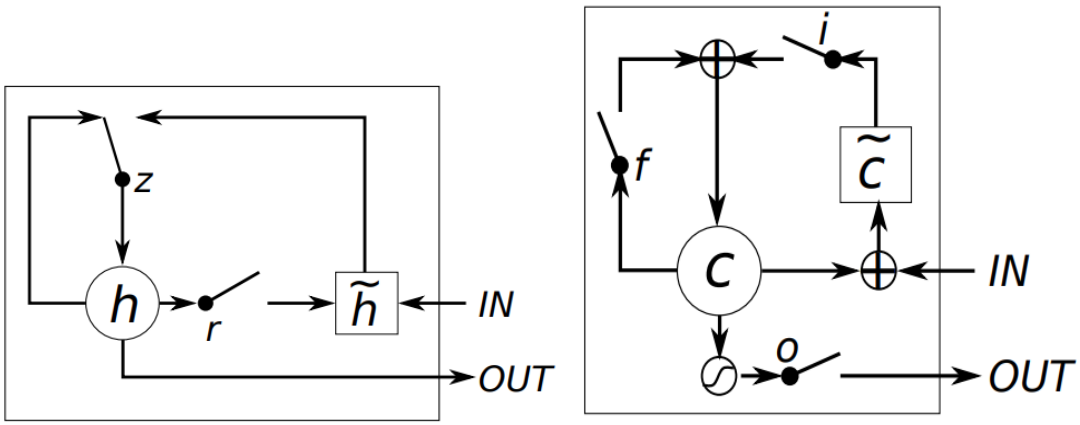
The capability to obtain a common depiction [5] for multi-modal input is at the heart of both picture captioning in this chapter and hierarchical paragraph captioning in Chapter 3, where RNNs at various levels receive diverse modalities as input. Figure 2.8 illustrates image feature vectors and text vectors linked to a common embedding space, illustrating two different input modalities. When training captioning models, it

is feasible to employ pre-trained word embeddings, and numerous reusable embeddings exist, such as Word2Vec [57], Glove [58], and, more recently, ELMO [7] and Bert [8]. However, it is also possible to train the embeddings with the rest of the model, as we did in the experiments described in this thesis.

2.3.2 Recurrent Neural Language Models

The recurrent neural language model [52] is an important component in image captioning. The inputs to the model are context vector and all previously given words and it is expected to predict the word that is most likely to occur next. Recurrent neural networks are a versatile as well as strong family of algorithms for producing both discrete and continuous weighted patterns [59]. The RNN utilises a predefined identifier as an input and then builds the very next identifier it generates on the prior outputs when it comes to creating text. In order to generalise to previously unknown inputs and create unique, semantically accurate sequences, the RNN has to understand to fill in the missing in between training sample it observes.

RNNs, similar to CNN, can experience the problem of vanishing gradients wherein the network does not remember inputs it received some time back in the past. To be effective in producing prolonged sequences, RNN must be able to selectively "remember" and "forget" relatively long history based on its present relevance. Long Short-Term Memory (LSTM) [60] or Gated Recurrent Units (GRU) [40] are commonly used to create modern RNNs. By adopting a sequence of gates, both types of recurrent cells provide apparatuses aimed at selecting maintaining lengthier and shorter contexts. Each gate is sigmoid σ (·) activated single layer neural network.



(a) GRU Cell **(b)** LSTM Cell

Fig. 2.8: Basic block diagram for RNN units [61].

The current element, at each time step t, in the input sequence is denoted by z_t, and the name of the gate is subscripted by t representing the gate's result.
As illustrated in Figure 2.9, the total number of gates in GRU and LSTM cells differs, so does the way they output and store their internal state .
There are two gates in a GRU unit: update and reset :
• Reset gate – specifies how much of the RNN cell $\boldsymbol{e_{t-1}}$ previous hidden state vector should be used for the new hidden state proposal vector $\mathbf{f_t}$

$$q_t = \sigma(v_{kp}\, z_t + v_{kh}\, e_{t-1}) \tag{2.10}$$

where $\boldsymbol{v_{kp}}$ and $\boldsymbol{v_{kh}}$ are learnable weight matrices, and $\boldsymbol{e_{t-1}}$ is the hidden state output from time step t-1. The following is how the planned novel concealed state is computed:

$$f_t = \tanh(v_z \boldsymbol{x_t} + v_e(q_t \odot \boldsymbol{e_{t-1}})) \tag{2.11}$$

where "Θ" is element-wise multiplication.
• Update gate – z serves a twin purpose, acting as a complement to both z and 1-z. Figure 2.9 shows a small "switch" next to gate z that works like a control, displaying the mingling proportion amongst the existing concealed state $\boldsymbol{e_{t-1}}$ and the planned new concealed state $\mathbf{f_t}$.

An LSTM cell differs from a GRU in that it includes three gates: *input, output, and forget*, as well as a separate cell state $\boldsymbol{d_t}$.

• *input gate* – i works in tandem with the forget gate *f*. They are, however, implemented individually, unlike in GRU.

• *Forget gate* – *f* regulates the weighting for the previous value of cell state $\boldsymbol{d_{t-1}}$ and complements the input gate.

• *Output gate* – o is used for calculating the LSTM cell's final output:

2.3.3 Decoder Implementation

We use a single RNN component with two hidden layers to create a flat decoder, as illustrated in Figure 2.10. GRU or LSTM recurrent cells are used in our investigations. By treating each input paragraph as a separate sequence, the flat decoder may generate single "sentence-length" captions as well as paragraph-length captions.

We wrap the ground truth caption with <START> and <END> tokens before we start training. Then, in the constructed ground truth caption, we embed each input token k_t (encoded as a one-hot vector) to a vector z_t of size P. The RNN's initial input, z_0, is provided as the encoder's visual embedding output. The rest of the model parameters are trained alongside the word embeddings.

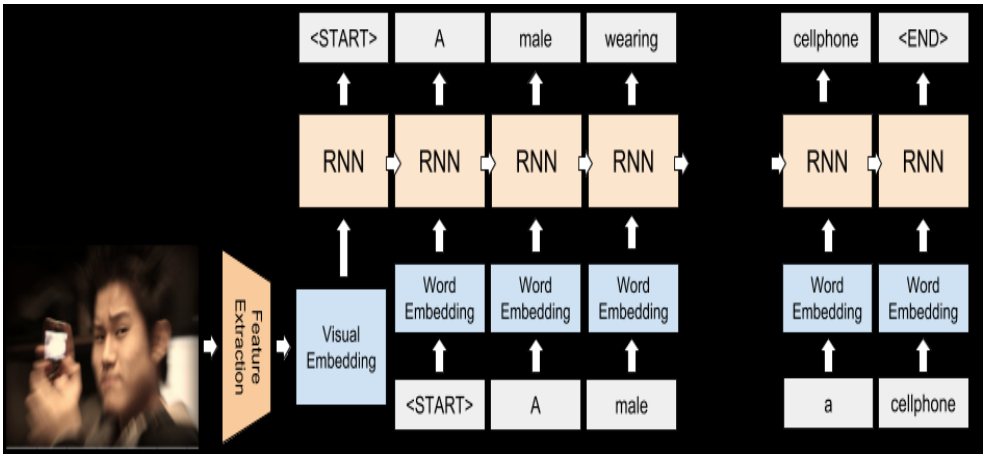


Fig. 2.9: Flat decoder.

In training, we employ mini-batch gradient descent, which divides each pass over the dataset into 128 (image, caption) pairs. PyTorch includes the PaddedSequence object, which allows training RNNs with mini-batches containing varied length input sequences, to support mini-batch training when each caption may be of various length. Each mini-batch must be sorted in decreasing order depending on the caption length in order to use the PaddedSequence class.

2.4 Proposed Extensions to Image Captioning Model

Model suggested in the previous section can be expanded. Discussed below are some of the newly introduced building elements that have the potential to improve image captioning performance:

• The attention mechanism [38, 31] attempts compensating for the problem that the presence of just one context vector be not large enough for encompassing whole

information about the input image. For each step of output, decoder is allowed to extract information out of important sections rather than relying on just one context vector. [26, 50].

• RL [15, 62] enables neural network-based models that make use of neural networks to master the method of optimizing for the process of training of discrete and non-differentiable variables.

• GAN [63] models consist mainly 2 major components:

Generator network which trains with input data and tries to come up with a output as close as possible to the real world output.

Discriminator network which leans continuously and tries to distinguish generator produced data from real world data. Although GAN based methods have been able to generate a diverse range of captions, they have not been able to stay competitive when evaluated on scoring metrics.[27, 64].

• VaE is yet another procedure utilizing GANs. The VaE formulation attempts to mimic the training data's probability distribution by learning a normal distribution of latent data representation (characterized by variance and mean). It is an alternative generative methodology used in image captioning [65, 66]. Models that use attention learning [26] and the methods that make use of reinforcement learning [9] have demonstrated most promising outcomes so far.

CHAPTER 3
Paragraph Image Captioning

Paragraph image captioning, as a field of research as well as the dataset that is used for testing that goes with it, has only lately surfaced [4]. The generated image descriptions in standard image captioning are brief. In the MS COCO Captions dataset [34]. The typical paragraph caption is 67 words long and is divided across numerous sentences.

Table 3.1: Evaluation metrics scores for different baseline models.

Evaluation Metrics	Image Captioning [3]	Paragraph Captioning [9]	Paragraph captioning – human baseline [30]
METEOR	27.6	18.6	19.22
CIDEr	117.9	20.9	28.55

It, sometimes can become impossible to describe an image with intense detailing with just a single sentence. Larger, multi-sentence descriptions are frequently required. While longer captions might capture more detail in an image, they are also more difficult to learn because the number of places where the machine can make a mistake increases as the n length of caption increases. Furthermore, looking at the conventional metrics is insufficient to compare image and paragraph captioning outcomes. The baselines are different, as shown in Table 3.1, and the scores that has been obtained by human performed stanza captions are much less over the best picture results by captioning.

By comparing to image captioning, paragraph by captioning has gotten less attention. However, the concepts that emerge from picture captioning by the work of research are frequently applicable to paragraph captioning. Concluding that there is a significant amount of "cross pollination" between them is important. The fundamental prerequisites to extend a language by modelling mentioned in Section 2.1.1 from generating basic each sentence captioning to paragraphs comprising several phrases are discussed here.

The criteria for the two jobs are identical, but paragraph captioning adds the requirement for modelling language in the sentence as well as word levels. This implies

that the input characteristics must include data that may be utilised to produce sentences that describe various parts of the image.

The fundamental baseline architecture has been mentioned below. It was recently developed to satisfy the needs of stanza size language modelling for briefing images. The project includes a pre-trained encoder for a dense captioning task in which each image is captioned with several per-region annotations. In addition to the regions-based encoder, models the paragraphs on sentence and word levels individually using two corresponding RNNs. Other working projects for paragraph captioning that have recently been published [3, 4, 11] all use a hierarchical decoder and follow this Encoder-Decoder design.

Only about 19, 000 picture caption pairings are currently available in the Stanford-Paragraph dataset [4]. In relation to the MS-COCO Captions dataset [34], which offers five types captions per image, the dataset only has one example paragraph per image. When just using the Stanford-Paragraph dataset for training, the study of paragraph captioning project may suffer from over fitting to the training data. Transfer learning, as was discussed in Section 2.2, is one possible answer to this problem. Krause et al. [4] suggest employing two transfer learning sources: a pre-matured image encoder and pre-engaged weights for the RNN component that generates sentences. The encoder and the language model in their study both were pre-nurtured using the Visual Genome Regions dataset [25], which we discuss in Section 4.1.

The hierarchical decoder will be described in Section 3.2, and the end-to-end picture of the model. The hierarchical-coherent model will be introduced, which extends the hierarchical baseline by attempting to impose coherence between individual phrases inside a paragraph. Section 3.5 of the chapter describes some further paragraph captioning vigilance.

3.1 Encoder: Dense Captioning Network

Acquiring Visual features via a dense captioning network established by Johnson et al. [1], also known as DenseCap, and input to the hierarchical language model can also be used [4].

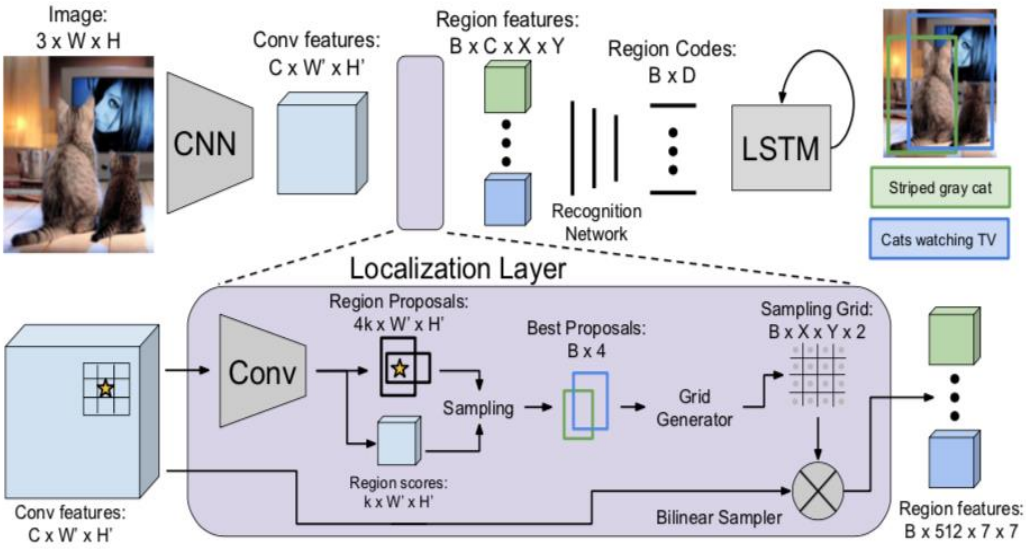


Fig 3.1: Architecture of DenseCap [25].

The dense captioning network is a hybrid model that aggregates the works of region suggestion as so of picture captioning. Visual Genome Regions [25] was the dataset used to train the DenseCap network. Before getting into the specifics of how DenseCap characteristics are retrieved, let's go over how these features are calculated. To generate a series of B tuples, the neural network is trained. Each series comprises of a bounding box and a caption defining the area of an image enclosed by the box, in the task of dense captioning.

The fully obtained result Encoder-Decoder design mentioned in Section 2.1 is similarly followed by the dense captioning network. Unlike image captioning, this is a "one-to-many" interaction, with all images those are responsible for producing B image-region based contexts. Then each context is utilised as an input to a different instance of the image captioning task. The dense captioning network is designed from all the components up to and including the main network. The context vectors are the area codes depicted in Figure 3.1, and the decoder is the LSTM-based RNN.

As shown in Figure 3.2 a collective work of bounding boxes corresponding to different portions of the stimulus image and natural language descriptions of each and every such region. The loss function is made up of five criteria: L1 fail on area placements in the localization layer, binary logistic fail on anticipated (0, 1) confidences for regions in the recognition network. The cross-entropy on the language model output is used.

Dense-Cap can be declared to instantly build larger image details by aggregating the released region descriptions, though it is not technological for a paragraph captioning

model. This method may result statements that are coherent when used as separate captions for image sections, although they can also lose their coherence when combined to form a paragraph. As a result, DenseCap cannot be trusted to generate convincing paragraphs.

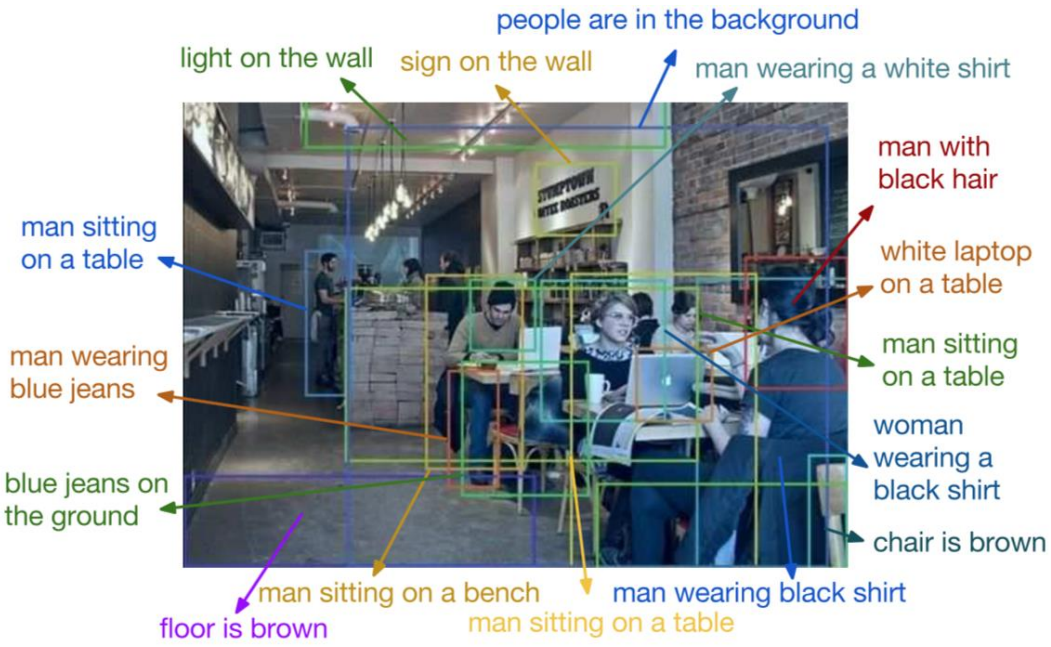


Fig 3.2. A DenseCap network's output as an example [25].

3.1.1 DenseCap Features Extraction

The method for extracting DenseCap features depicted in Figure 3.3 resembles to the ten-crop feature extraction discussed in Section 2.2.1. Unlike the ten-crop features, which are produced from the identical area for all input photos, DenseCap features are converted to region content. The model was trained on a total of 77, 398 images, with an average of 50 region captions per image. As reported in [3, 4], we save B = 50 region characteristics for each input picture, then do additional pooling by selecting the element-wise greatest over all B extracted features, based on the final per-image feature-vector of size DenseCap. In addition to maximum pooling, we tested with average-pooled picture features, which were calculated by taking an element-wise mean of B region vectors.

The DenseCap-dimensional characteristic vector is translated into an E-dimensional context vector, which is passed to the hierarchical decoder discussed in the next section.

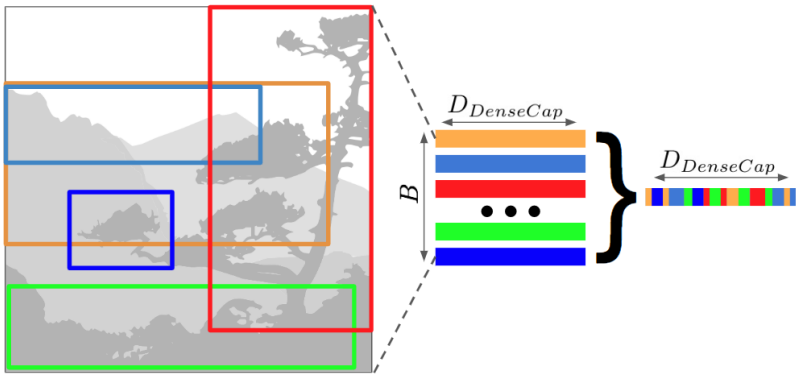


Fig 3.3. Using element-wise maximum, DenseCap features were retrieved from picture regions.

3.2 Decoder: Hierarchical RNN

Captions for paragraphs can be thought of as a collection of single words and sentences. Using a hierarchical language model, Li et al. [30] and Lin et al. [35] propose modelling word-level and paragraph dependencies independently. As a consequence, one RNN models phrases at the term or "token" level, while another model paragraphs at the level of the sentence.

The work [17] advocated using hierarchical models on sequential inputs to help model longer-term connections between structural components that are sequential themselves. In part of text input, these units range from longer-term to shorter-term contexts and include chapters, paragraphs, phrases, and words. RNNs' ability to selectively represent longer and shorter-term contexts has increased since the advent of LSTM. As a result, one can wonder whether explicit hierarchy at the network design level benefits a modern RNN-based language model. In Chapter 4, we shall attempt to answer this question.

Yu et al. [36] were the first to mention the use of a hierarchical word embedding in the highly associated task of video captioning. In their model, they adopt a two-level hierarchy, with term and sentence-level RNNs created with GRUs. The phrase context is set to zero during training, and the picture characteristics are used to start the word-level RNN. The syllable embedding of all words in the first sentence is passed to the sentence-level RNN, which constructs a new context for the following verse after reading the first sentence. As previously said [4] has converted the de-facto paragraph captioning baseline, thus we will go over their formulation in further depth. The

hierarchical RNN employed for sentence captioning is defined as having two layers of hierarchy, each represented by a separate RNN:

• **SentenceRNN** – For each sentence, this is an upper RNN that produces a topic vector Ti. At each time step, Sentence RNN takes the same image characteristics given by the encoder as input and returns a different subject vector for the following sentence.

• **WordRNN** – A final output-level RNN that uses the topic vector ***Ti*** as its initial input to produce individual words in each phrase. This component is identical to the decoder used in the flat model in our tests.

The source picture I is said to be "grounded" [37] for each subtitle in the flat model. In the methodological structure, the grounding is similarly hierarchical. Phrase topics Ti provide a medium level of grounding, with each subject vector performing the same function as the background vector c in Section 2.4, with the exception that each paragraph in a paragraph now has its own generative model.

The "Regions-Hierarchical" model's decoder learning method utilizes that we are provided a paragraph containing sentences, each of which includes Ni words, with I becoming the sentence index. To initialise the decoder, WordRNN weights from a language model pre-trained on the massive Visual Genome Regions description dataset [25] are employed. Then, for M time-steps, Sentence RNN is run with the same input at each time-step. SentenceRNN generates a H = 512 hidden vector, which is then fed to two sub-networks: a halting detector and a topic generator.

3.2.1 Hierarchical Decoder Implementation

Figure 2.10 presented a sub-unit of the hierarchical decoder in Figure 3.4. The hierarchical decoder limits the highest number of permitted sentences per paragraph to MMAX = 6 to ensure that the inferential process can be completed. Each original sentence in the paragraph is generated using the same WordRNN, with the same weights. During learning, each paragraph is degraded into sentences. Input to the decoder is the same kind of mapping from picture features to embedding size E that was utilised in the flat model.

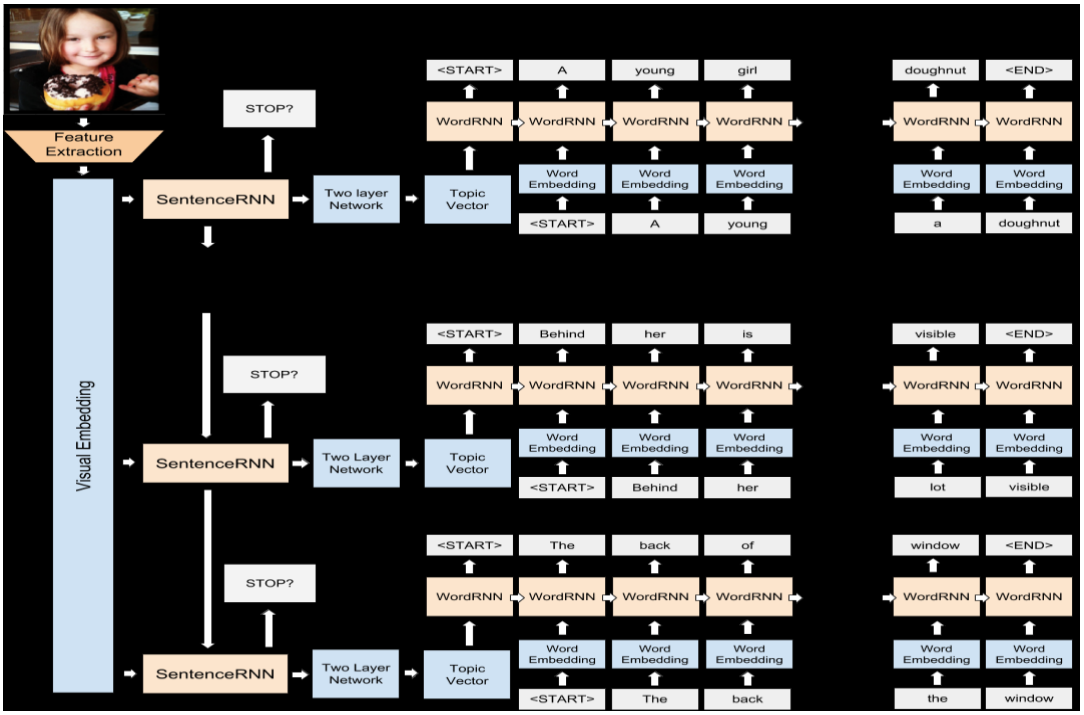


Fig 3.4: Hierarchical decoder.

Image features are not directly employed to construct an image caption in the hierarchical model. Instead, the Sentence RNN is executed for *MMAX* time-steps first. Each timestep's Sentence RNN outputs are used for two purposes:

1. The halting classifier, which is built as a couple of logistic units, determines the probability that the current phrase is the paragraph's last one. We build this unit by providing a pair of values, rather than the commonly utilised technique of using an artificial neural layer with a single value for binary classification.

   The reason for this implementation is that we noticed during the trials that utilising a two-unit neural network produces somewhat better outcomes in standard metrics than using a single output. It's possible that a network with two-unit output develops a somewhat superior stopping classifier representation. At the same hand, it's probable that this will make the model more prone to overfitting and slow convergence.
2. To generate E-dimensional topic vectors ***Ti***, the Sentence RNN outputs are input into a two-layer fully connected network. Between the layers, we apply ReLU non-linearity [22, 45]. Each of the ***Ti*** topic vectors is sent to the WordRNN, which creates each sentence.

PyTorch's underlying hierarchical paradigm is quite simple to implement. The 130-piece small size that we used to learn the flat model is employed again. When overfitting, the most difficult part is ensuring that the Packed Sequence object, which contains all sentences at location I in the mini-batch, is appropriately sorted. For each mini-batch, the sentence data must be arranged in M max ways. We must keep a separate sorting indicator for each phrase in the paragraph, sorting and unsorting them as needed, to guarantee that the sorting order is correct for each phrase at each Sentence RNN time-step. A large portion of the task was spent to ensuring that the functions listed above work as expected.

3.3 Paragraph Captioning Pipeline

When the DenseCap-based coder is integrated with the hierarchical decoder network, the entire "Regions-Hierarchical" structure is presented in Figure 3.5. The output of the dense messaging encoder is compiled into a single vector, which is then utilised as a backdrop for the hierarchical RNN that generates the sentences.

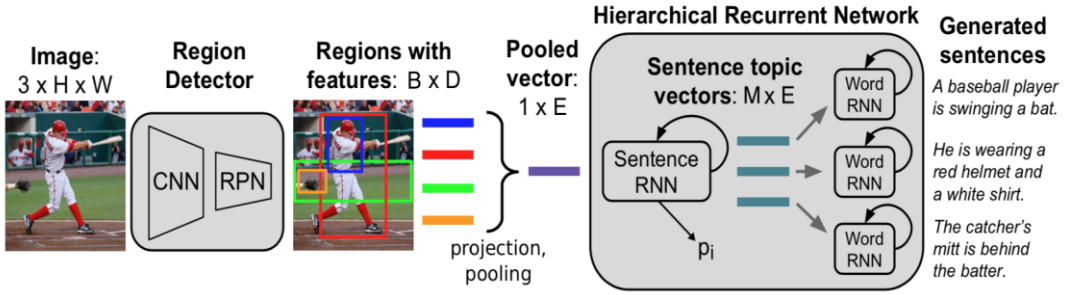


Fig 3.5. The encoder-decoder design for graph captioning was adopted after [30]

Let's define each training input as a pair (***I***,***SP***), where *I* is an image and ***SP*** is the image's paragraph caption. M sentences are in paragraph ***SP***. Ni words appear in sentence *i*. Also, in the *i*th sentence, make ***Sij*** the *j*th word. Two probability distributions are taught to the model:

• p_i – the likelihood of the current sentence being the last sentence in the paragraph over STOP, CONTINUE, so that

$$pi(STOP) + pi(CONTINUE) = 1 \quad (3.1)$$

• p_{ij} – a chance of a given word appearing in place *j* of the *i*th sentence based on the full vocabulary.

The loss proposed by Krause et al. [4] and used by others [3, 11] is divided into two halves, one for each decoder hierarchy level. The algorithm L has two terms: a weighted phrase loss sent and a weighted word loss lword, where lsent is the optimal solution for the Sentence RNN and word is the target value for the Word RNN:

$$\mathcal{L}(I, S_P) = \lambda\text{Sent} \sum_{i=1}^{M} l_{sent}(p_i, \mathbb{I}[i = M]) + \lambda\text{Word} \sum_{i=1}^{M} \sum_{j=1}^{N_i} l_{word}(p_{ij}, S_{ij}) \quad (3.2)$$

3.4 Hierarchical-Coherent Project

The Stanford-Paragraph dataset's modest amount of training data might not even comprise many of the language structures necessary for the produced paragraph-caption to seem like it might have been written by a person. The hierarchical paradigm may also have trouble creating paragraphs that move effortlessly from one sentence to the next [3, 11]. The captions have a few features that make it look natural such as how varied they are – do they all maintain similar pattern or are they seeming to be unrestricted, how spontaneous they are – how much likely it is for a human to come up with comparable description. The following section goes through a possible solution to improve the seeming eminence of captions in greater depth and the subsequent segment would go over a few additional options briefly.

For each paragraph, the hierarchical approach represents numerous levels of structure. The baseline model, on the other hand, does not explicitly relate the preceding and subsequent sentences. SentenceRNN's topic vector from the baseline conceptual framework is immediately employed to create the next sentence. Because they are created before the WordRNN is performed, these subject vectors do not rely on the WordRNN's output.

The hierarchical-coherent implementation is built upon [3], which builds on the previous "Regions-Hierarchical" model to make paragraph titles more human-like. A high-level summary of the model is shown in Figure 3.6. Similar to Krause et al. [4,] SentenceRNN is used in their model to construct sentence topic vectors Ti for each phrase in the paragraph.

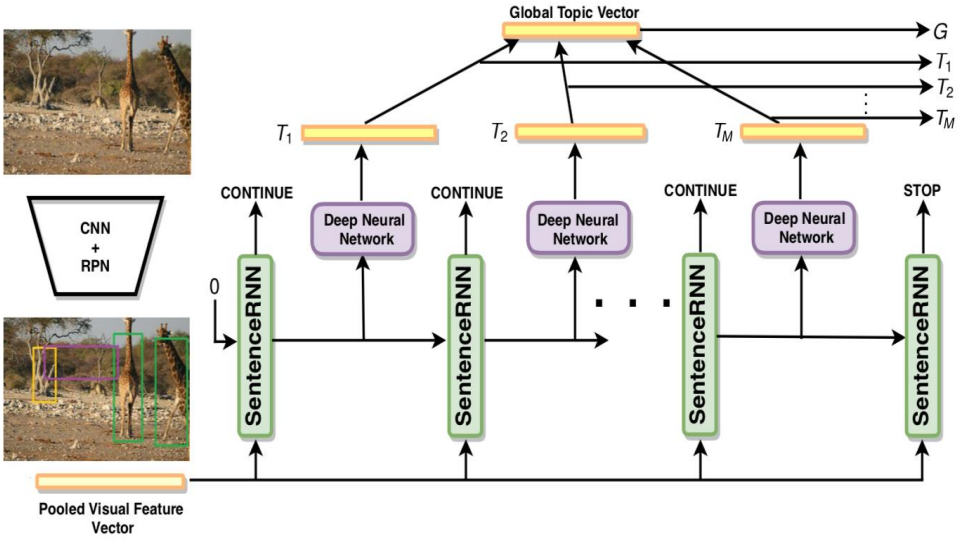


Fig 3.6. Adapted from [3], with names of parameters names altered for uniformity.

However, unlike previous work, after all of the phrase subjects have been formed, they are further analyzed to give the sentences coherence. It's important to note that the released "Diverse-Coherent" model has an optional parameter estimation autoencoder element that aims to create diversity to paragraph descriptions, but this capability was not included in our hierarchical-coherent model.

We'll now examine at whether Chatterjee and Schwing's [3] technique generates sentence coherence. After the subject vectors have been created, the balanced total of them is saved in a global topic vector G:

$$G = \sum_{i=1}^{M} \alpha_i T_i \tag{3.3}$$

$$\alpha_i = \frac{||T_i||_2}{\sum_j ||T_j||_2} \tag{3.4}$$

To assist in preserving sentence coherence, the coherence vector Ci1 is computed. The concealed state of the WordRNN is obtained after the final word of the current phrase has been produced in order to compute the integrity vector. The authors supply this hidden state representation using a two-layer totally connected net known as a coherence network. Ci1 is the result of this network. As the baseline coherence vector, C0 is set to zero.

G is the worldwide topic vector, Ti is the topic vector, and Ci-1 is the coherence vector. Using these three vectors, a binding unit may be used to calculate the final topic vector

T_i^'. The index i-1 of the coherence vector relates to the previous sentence, but the index I of the subject vector and the index I of the final topic vector correspond to the sentence about to be created.

$$T_i^C = \frac{\alpha T_i + \beta C_{i-1}}{\alpha + \beta} \tag{3.5}$$

In this approach, the squared norm of the difference between the fused topic-vector and each of its inputs is minimized:

$$T_i^C = \arg(min\ \alpha) \tag{3.6}$$

The constants and are assigned to different values depending on whatever dataset the project is trained on. On the Stanford-Paragraph dataset, the values for training are α = 1.0 and β = 1.5.

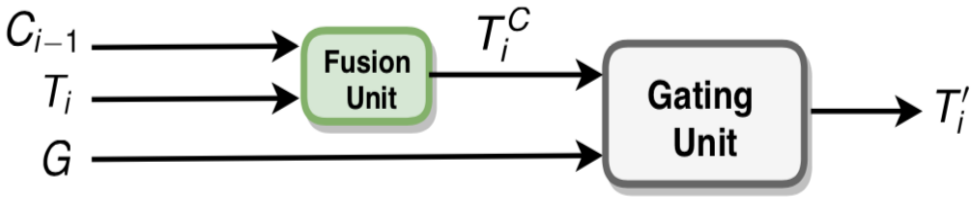


Fig 3.7: Coupling unit combining the global topic vector, current topic vector and the coherence vector C_{i-1} originating from the previous sentence [3].

The fused vector is then transmitted to the gating unit, which is constructed as a single GRU cell with the universal topic vector G determining the cell's initial internal state. The cell's result is the end topic vector T i' after a data set, which is then used as the initial feed to the WordRNN, which creates the following sentence. This process is seen in Figure 3.8.

Connecting the final created word of the previous phrase to the first intake of the following sentence enhanced relative coherence between the produced sentences, according to the model's authors. In a similar vein to vector G, Li et al. [30] investigate monitoring a global, paragraph-level environment. They do, however, use a third level of RNN hierarchy to simulate paragraph level context. In the coupling unit, the GRU

cell is used in a similar fashion to this previous work, however it is now made in a more compact manner.

The gating unit knows to manage how much of the worldwide background G to "let through" for constructing the next phrase instead of relying on the transitory context vector T i^C , which is more local to the present sentence. The GRU cell in the gating unit does not simulate a recurrence since it is only run for a single time-step.

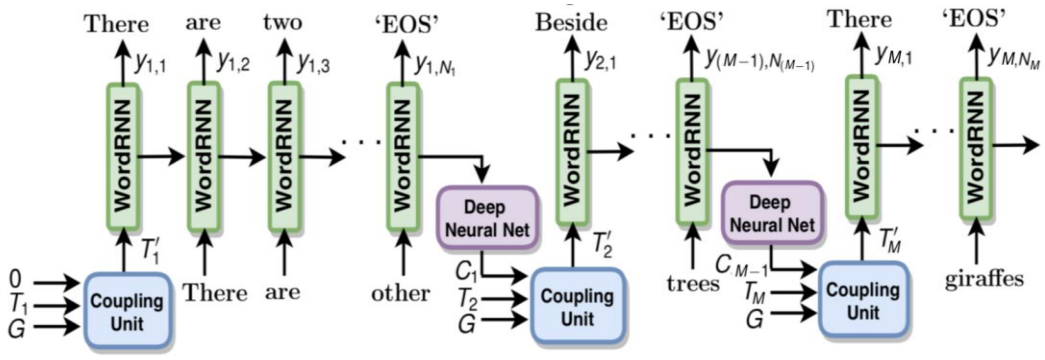


Fig. 3.8: The coupling unit sends subjects to WordRNN [9]

3.4.1 Hierarchical-Coherent Decoder

The structure is employed by changing the fundamental hierarchical model. The only changes to our hierarchical structure are that more components must be arranged at each phrase time-step g to match PyTorch's Packed Sequence based RNN input set-up specified in Sections 3.2.1 and 2.3.3. The dimensions of the input embeddings given at each WordRNN instance are represented by P-dimensional vectors J_i, H, A_{i-1}, $\hat{J}_i^A$ and K_i'.

For attaining coherence vector A_{i-1}, we utilise wordRNN's second last concealed output at sentence i-1. The solution obtained differs from what was observable in sample scripts by original developers. They utilize internal layer of WordRNN from the previous instance after the phrase is done creating and the <END> symbol has been created. This work conducted the same approach in tests, but the outcomes were constantly poorer on traditional measurements.

Activation function utilised by the original authors [3] is the Scaled exponential Linear Unit (SeLU) [29], that is analogous to the regularly used ReLU. Since SeLU could not

give good results ReLU irregularity was used both for coherence vector generation and topic generation.

3.5 Extensions to Paragraph Captioning

There have been many refinements to the model base model proposed by [3]. Some of these modifications include:

The "Regions-Hierarchical" model proposed by Krause et albaseline [4] has been refined several times. Besides to the previously noted inclusion of coherence vectors and accompanying paragraph coupling, these advancements include:

- Attention mechanisms on Linguistic and perceptual aspects [11],
- GAN based architecture making use of an objective function which relies on adversarial networks.objective function utilising a GAN architecture [27, 11],
- In addition to the current methodology that makes use of RNN at sentence level as well as word level, a paragraph RNN can be introduced[11].
- ED architecture based on VaE. [3].

Attention in hierarchical linguistic models was introduced by [30]. The usage of attention in the picture tagging[31] and human language interpretation domains[38] inspired their attention technique. To tackle the closely related problem of paragraph length video captioning, [36] proposes employing emphasis on image elements gathered inside a video frame. It also deploys progressive attention while creating captions for static pictures, which is less significant when creating subtitles for video frames.

[11] offer to include different enhancements for paragraph generation. A comparable model was created by [4]. The model feeds off of DenseCap features, while the execution focuses on image feature vector along with words in dense subtitles created by the very same DenseCap system that produced the image features. They also have a duplicating [28] function that allows them to take words from the visited region descriptions and paste them into their own documents. In their GAN-based solution to paragraph captioning, [27] suggest hierarchical grader.

[11] includes an adversarial training mechanism. For training on paragraphs without graphics, semi-supervised components are paired with acquiring single-sentence labels on the MS-COCO Captions database.

[3] illustrate how using a VaE-based framework, in which a section of source space of subtitles model is tested at run-time, basically creating a few more order of

unpredictability within outcomes of the model, can enhance the diversity of generated captions as a substitute to using an adversarial objective.

CHAPTER 4
Experimental Analysis

External, pre-computed image characteristics pre trained on classification of images and on Dense-Captioning were used in our models. A RNN, pre–trained on image captioning provided network weights for our task of paragraph captioning. Description of datasets used is provided in Section 4.1.

The datasets utilized are described in Section 4.1. Section 4.2 discusses the sought-after automated assessment metrics used to evaluate outcomes and provides high -level evaluation criteria that humans might make use of. Section 4.3 goes into the specifics of the training. Finally, Section 4.4 gives experimental results followed by final thoughts and discussions in Section 4.5.

4.1 Datasets

The experiments were done with 3 different datasets. Before we start model trainings, we split each and every dataset into a training set, which comprises of photos that will be made use to train them, and a set of validation, that is in general a collection used to see how well they generalize to previously unknown data.

MSCOCO [10]: An image captioning dataset that comprises of images as well as captions analogous to each image. There are in total 82, 783 images available for training and 40, 504 images for validation. We use the MS COCO Captions c5 subset, which gives on an average, 5 single sentence captions corresponding to each image. There are 108, 077 photos in the Visual Genome Regions (VG Regions) [25] dataset. Every image in VG Regions has an average of 50 per-region annotations, each of which contains a brief region description as well as the coordinates for a bounding box encircling the described region. Each geographical description is on average 5 words long.this dataset contains 77,398 and 5000 training and validation images. The Stanford-Paragraph dataset [4] is a subset of 19, 551 VG Regions images annotated with multi-sentence descriptions, with each sentence averaging 11.91 words in length. This dataset is used to train our final model because it is currently the best accessible paragraph captioning dataset. We used the identical train/val split as the dataset's authors, with 14, 579 images trained and 2,490 images validated.

4.2 Evaluation Parameters

4.2.1 BLEU

The BLEU-1,2,3,4 (bilingual evaluation understudy) [20] evaluation parameter is dependent upon accuracy in n-gram. N- gram is basically the word combination being matched at that iteration. For a caption written like "A dog is running on grass," for example, we might write:

1-grams (or unigrams): “A”, “dog”, “is”, “running”, “on”, “grass”

2-grams (or bigrams): “A dog”, “dog is”, “is running”, “running on”, “on grass”

4-grams: “A dog is running”, “is running on grass”

It's worth noting that there's a related concept known as a "character n-gram," in which n denotes the count of characters rather than words. Current work is interested only in n-gram word.

We'll show how to use BLEU-1 with unigrams in the examples below, but the same approach may be used with all sorts of n-grams independent of their size . For the case where Count(a) denotes word count (or unigrams) in available image description, each repetition of a particular word increases the count. Every caption created gets a precision score(unigram) in the following way:

$$P_{Captions}=\frac{\sum_{a\in Captions} Count(a)}{L} \tag{4.1}$$

When we come across any particular word which was present in actual image description, the counter for T_p is incremented. Simple precision has the drawback that T_p counts may surpass the count in actual image description. Lets try to understand it with an example:

Ground truth: A dog is running on grass.

Caption 1 : A A A A A A A A .

Caption 2 : There is a dog running on grass.

The unigram precision score for Caption 2 is 5/6, but it is 6/6 for Caption 1, despite the fact that the second one is significantly relevant to the actual description.

By adopting modified precision, BLEU hopes to address this issue. For a single caption, P_1^*, the adjusted score of precision for unigrams, is derived just by trimming

the appearance of any word in the caption generated to the actual number in the available image description

$$P_1^* = \frac{\sum_{a \in Captions} Count_{clip}(a)}{L} \quad (4.2)$$

The values of all 4 scores come in range (0,1) which are then multiplied with 100 to bring values in (0,100) range.When it comes to reporting our findings, we take the same approach.

4.2.2 METEOR

When utilizing BLEU as an automated evaluation metric for image captioning, there are a few issues to consider. For starters, it ignores recall, as specified in Section 4.2.2. When used for image captioning, recall, when implemented for image captioning counts number of words from available image description appear in the description generated by the model. It, also ignores synonyms, and when working with bigger n-grams, such as while computing BLEU-4, penalizes descriptions having changed, yet right, order of words than the available image description. Meteor [32, 16] aims to solve the issues of matching synonyms and paraphrased sentences by combining precision and recall.

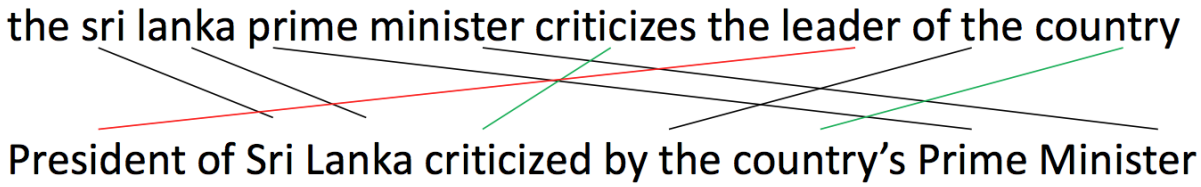


Fig 4.1 Meteor matches example [33].

The sentence generated is compared with expected sentence using matchers m_i :

- **Exact**, m_1 – trying to find exact word matches, similar to unigram.
- **Stem**, m_2 – trying to find words that have a common stem.
- **Synonym**, $m3$ – trying to compare each word with synonyms of it available in [19]
- **Paraphrase**, m_4 – trying to compare that particular set of words with a pre-defined phrase table. Figure 4.1 gives a demonstration of how matching is done within sentences. Black colored lines correspond represent "exact"

match, green colored lines indicate “stem” matches, and red colored lines are a representation of matching phrase

It considers words present in a particular language that occur more than a specified number of times to be function words. "about", "A,", "about," "the," "had," and "could" are examples of function words. Content words are those that are less common, such as "computer," "dog," "purple," "play," and so on. The function words in hypothesis and reference sets are abbreviated as *hf* and *rf* ,, respectively, and the content words are abbreviated as *hc* and *rc*.

Each matcher *mi* has a weight *wi* linked with it (see Table 4.3 for parameter values). The value of the parameter $\boldsymbol{\delta}$ determines the relative weighing of content and function words. The weighted precision *Pw* and weighted recall *Rw* are calculated using the matchers and their accompanying weights:

$$P_w = \frac{\sum_{i=1}^{4} w_i \,.\, (\delta.m_i(h_c) + (1-\delta).m_i(h_f))}{\delta.|r_c| + (1-\delta).|h_f|} \tag{4.3}$$

$$R_w = \frac{\sum_{i=1}^{4} w_i \,.\, (\delta.m_i(r_c) + (1-\delta).m_i(r_f))}{\delta.|r_c| + (1-\delta).|r_f|} \tag{4.4}$$

The numbers of content and function words in the hypothesis and reference captions are represented by |*hc*|, |*hf* |, |*rc*|, and |*rf* |.

The weighted precision and recall are then multiplied by the parameterized harmonic mean F_{mean}:

$$F_{mean} = \frac{P_w \,.\, R_w}{\alpha.P_w + (1-\alpha).R_w} \tag{4.5}$$

The parameter $\boldsymbol{\alpha}$ (see Table 4.3) is commonly chosen so that recall is prioritized over precision [14]. In order to reward longer consecutive matches, Meteor introduces "chunks," which are defined as sequences of matches that are in the same order in both the hypothesis and the reference. For example, in Figure 4.1, "srilanka" and "prime minister" both represent a chunk made up of two matches. Because the word order between the reference and the hypothesis gets more comparable as the chunk length grows, the Meteor score improves; nevertheless, as the chunk length increases, the hypothesis becomes more "fragmented."

The fragmentation penalty (P) is determined as follows: If m is the total number of word matches (averaging the hypothesis and reference to account for paraphrase), and k is the total number of chunks

$$\mathrm{P} = \Upsilon.\left(\frac{k}{m}\right)^{\beta} \tag{4.6}$$

The fragmentation penalty is then applied to the harmonic mean of the weighted precision and recall to arrive at the final Meteor score:

$$\text{Score} = (1 - \mathrm{P}) \cdot F_{mean} \tag{4.7}$$

Table 4.3 shows the default values for the English version of Meteor, which have been tweaked to improve the connection between human judgement and Meteor-scores [16]. Overall, Meteor has been demonstrated to have a better correlation with human judgement than BLEU [16].

To demonstrate, we use only the exact matches, m_1, to Calculate the Meteor score for the preceding section's sample sentences. A period character (".") is likewise regarded as a function word in this context. The reference and hypothesis captions are given to us, with content terms marked and the remaining words classified as function words:

Hypothesis: *There is a cat on the mat.*

Reference: *the cat is on the mat.*

Table 4.1: Evaluation Parameter values of Meteor for Sentences in English [16].

$\boldsymbol{\alpha}$	β	Υ	$\boldsymbol{\delta}$	w_1	w_2	w_3	w_4
0.85	*0.20*	*0.60*	*0.75*	*1.00*	*0.60*	*0.80*	*0.60*

.

The weighted precision and recall are then calculated using Eqs. 4.6 and 4.7. We only calculate the exact matches, *m1*:, to make our example basic.

$$P_w = \frac{w_1(\delta.m_1(h_c)+(1-\delta).m_1(h_f))}{\delta.|h_c|+(1-\delta).|h_f|} \tag{4.8}$$

$$R_w = \frac{w_1(\delta.m_1(r_c)+(1-\delta).m_1(r_f))}{\delta.|r_c|+(1-\delta).|r_f|} \tag{4.9}$$

4.2.3 CIDEr

Another popular automated assessment measure is CIDEr (Consensus-based Image Description Evaluation) [14]. CIDEr was built exclusively for analysing picture captions, unlike BLEU and Meteor, which were created to assess machine translation outputs. It works with datasets such as MS COCO Captions [34], which provide a large number of ground truth picture descriptions for each image. We will not utilise the feature for analysing multiple captions per picture in our paragraph captioning experiment since the Stanford-Paragraph [4] validation dataset only gives one reference description per image. CIDEr can refer to either a standard CIDEr score or a modified version known as CIDEr-D, which makes changes to the CIDEr score to prevent people from "gaming" the system by changing the captions to get unrealistically high scores.

4.3 Results

The outcomes on experimentation performed on three models introduced in Chapters 2 and 3: flat, hierarchical, and hierarchical-coherent are as follows. All three models rely on a WordRNN that has been pre-trained on VG Regions datasets or MS COCO Captions or. The results provided by the authors of each baseline are shown for the published baselines.

4.3.1 Image Captioning Experiments

A number of flat captioning algorithms are trained making use of MS COCO dataset. Another essential objective for this initial trial was to uncover great approaches for input feature pre-computationthat have been obtained from the pre-trained DenseCap and ResNet-152 models. Alternative combinations of input features was also attempted, as well as average and maximum pooling.

On the image captioning assignment, we discovered that our best-performing models utilised average pooled DenseCap features.[4] state that their DenseCap characteristics were pooled to the maximum, which is interesting. Average pooling, by "averaging over" the minor details, appears to be superior at annotating common data regarding image. Maximum pooling, on the other hand, may aid the model in capturing more particular details. It is not necessary that all of these image specific details are of any use for single sentence captioning.

All the best performing models for picture captioning employed a concatenation of ResNet-152 and DenseCap image characteristics. All subsequent trials used the vocabulary and concatenated picture attributes from the dataset that was brought to use for pre-training of WordRNN.

4.3.2 Flat Architecture Results

Existing pre-trained models for image captioning were tuned up with Stanford-Paragraph Dataset so as to evaluate the performance of flat captioning models when tasked with a rather complex task, generating a paragraph description.

We improved several of the pre-trained image captioning models stated in Section 4.4.1 by training them on the Stanford-Paragraph dataset to examine how well the flat model performs when confronted with the more difficult job of paragraph synthesis. Each paragraph was merged into a single caption before being sent to the model. The character "." at the end of each line is treated as a token in this caption. The period character (".") at the end of each word was considered as another token inside the caption. The resulting caption might be up to 80 words long. For flat paragraph captioning, all models used an LSTM-based RNN implementation.

Our flat models were pre-trained using the MS COCO Captions or VG Regions datasets. MS COCO utilises 9, 957 words, whereas VG Regions uses 19,804 words, based on the vocabulary of the dataset it was pre-trained on.

Table 4.2: Comparison Results for different models for paragraph captioning

Model Used	BLEU-3	BLEU-4	CIDEr	Meteor	BLEU-2	BLEU-1
Image- Flat [26,30]	12.20	7.71	11.06	12.82	19.95	34.04
Flat, MS COCO, avg-pooled DenseCap, E=256	12.73	7.57	18.64	14.73	21.52	36.84
Flat, MS COCO, E=1024	12.98	7.66	**20.17**	14.95	**21.97**	37.59
Flat, VG Regions, E=256	**13.03**	**7.77**	19.00	14.82	21.85	37.29
Flat, VG Regions, E=1024	12.77	7.40	17.84	**14.98**	21.87	**37.74**

Table 4.5 displays the results of our top-scoring flat models for producing paragraph captions, as well as the current non-hierarchical state-of-the-art, "Image-Flat" [5]. In addition to CIDEr (C) and Meteor (M) ratings, we show BLEU 1-4 (B1-4) findings. The model with the highest CIDEr score (in bold) was trained for 77 epochs. We chose to highlight (in bold) the highest results on a per-metric basis since there was no one flat model that beat all others on every criteria.

Following an examination of captions produced by multiple models, it became obvious that the model with the highest CIDEr score produced the best captions. When faced with the decision of which measure to use after evaluating the output of various models, we chose models with the highest or very high CIDEr scores for further study.

4.3.3 Hierarchical Architecture Results

It's time to introduce the SentenceRNN and WordRNN divisions of labour, as described in Section 3.2, after seeing the flat model's performance results. [4] is remarkably similar to our hierarchical approach. However, because we didn't have access to the source code for the original model, there are likely to be inconsistencies in implementation.

Table 4.3: Comparison Results for hierarchical models

Model Name	BLEU-3	BLEU-1	METEOR	BLEU-4	BLEU-2	CIDEr
Regions-Hierarchical [30]	14.23	41.9	15.95	8.69	24.11	13.52
Hierarchical, VG Regions, E=256	12.33	38.38	15.18	7.02	21.63	15.87
Hierarchical, VG Regions, E=1024	12.62	39.65	15.15	7.14	22.33	17.01
Hierarchical, VG Regions, GRU, E=256	12.46	39.69	15.15	7.02	22.14	17.49
Hierarchical, MS-COCO, E=256	12.33	40.09	15.09	6.79	22.19	16.99
Hierarchical, MS-COCO, avg-pooled, E=256	12.38	BLEU-1	15.11	6.91	22.19	17.79

Hierarchical, VG Regions+ MS COCO, E=1024	12.21	41.9		14.88	21.82	17.85

The difference was not substantial when compared to some of the other models, when WordRNN was pre-trained on a single dataset. We also noticed that utilising the "Reduce on Plateau" scheduler with validation loss and continuously using early-stopping + fine-tuning with manually tweaked learning rates was not as effective as using a single static LR for the whole training.

4.3.4 Hierarchical-Coherent Architecture Results

This model, which we described in Section 3.4, is the most complex of the models we tested, and tries to replicate results obtained by[3].The approach brings about continuity in sentences that make up the paragraph.. Increased fluency can also be regarded as a result of this improvement. Next section discusses about the performance of our hierarchical-coherent model performed on standard metrics, and we'll see the captions generated by the model in Section 4.4.6.

When training our model, we only employed 1024-wide embeddings because prior attempts on hierarchical captioning models suggested that broader embeddings offered somewhat better results. Furthermore, using ResNet-152, every model we evaluated includes maximum-pooled DenseCap input features. We utilised the same DenseCap feature arrangement as the "Diverse-Coherent" baseline to make our results comparable. Table 4.7 shows the baseline and results for the three best hierarchical coherent models we trained.

Table 4.4: Comparison of Diverse coherent and Hierarchical-Coherent models.

Model	BLEU-1	BLEU-4	METEOR	BLEU-3	CIDEr	BLEU-2
Diverse- Coherent [9]	42.12	9.05	17.81	14.74	19.95	25.18
Hierarchical Coherent, static-lr, GRU	17.46	22.73	12.71	40.52	7.04	15.29
Hierarchical Coherent, Cylical-lr,	19.23	22.96	12.89	40.80	7.12	15.45

Our results demonstrated an improvement over the conventional hierarchical model when utilising the hierarchical-coherent design, although the scores fell short of the

established baseline [3]. Based on what we learned from the authors' incomplete source code, mentioned in Section 3.4.1, there are likely some variations between our and the baseline implementation. Furthermore, the publicly available source code only covers a portion of the training process, leaving the exact details up to interpretation.

While we used LSTM RNNs in the majority of our tests, the GRU-based models we trained produced equivalent results. Both types of RNNs used to get equivalent scores had similar convergence durations, showing that LSTM and GRU-based language models operate similarly in hierarchical contexts. When comparing the output of many models that scored similarly, the LSTM-based models tended to give somewhat more accurate captions.

4.3.5 Comparison with State of the Art

Table 4.8 draws a comparative analysis of published baseline methods with the results obtained in this work.

Table 4.5: Comparison Results for different Paragraph Generation Models

Model Name	METEOR	BLEU-2	BLEU-4	BLEU-1	CIDEr	BLEU-3
Image- Flat [5,4]	12.82	19.95	7.71	34.04	11.06	12.2
RTT- GAN [11]	18.39	25.35	9.21	42.06	20.36	14.92
Diverse-Coherent [3]	17.81	25.18	9.05	42.12	19.95	14.74
Regions-Hierarchial [4]	15.95	24.11	8.69	41.9	13.52	14.23
Diverse-Coherent (with VaE) [3]	**18.62**	**25.52**	**9.43**	**42.38**	**20.93**	**15.15**
Human [4]	19.22	25.68	9.66	42.88	28.55	15.55
Flat (ours)	14.94	21.96	7.60	37.58	20.16	12.94
Hierarchical (ours)	15.05	22.09	6.82	39.57	17.69	12.30
Hierarchical – coherent (ours)	15.47	22.99	7.14	40.83	19.21	12.93

The remaining baselines are all hierarchical, with an E = 1024 embedding size and input given by max-pooled Dense-Cap features. The hierarchical coherent architecture used in this work bases upon "Diversified-Coherent" model without VaE; in their next model, the same authors are able to enhance diversity performance of the generated captions using VaE.

We identified the top performing model for each of the three model types we compared on the basis of values obtained on Meteor and CIDEr metrics and by drawing up comparison between captions generated by them for a specific image. The flat captioning model that has been used here can be described as follows: pre-training performed on MS-COCO dataset. It has DenseCap features that are max pooled and embedding size is given by E=1024.

The hierarchical model, on the other hand, involves pre-training with a WordRNN on VG Regions dataset.

The hierarchical coherent model underwent training with a cyclical learning rate, unlike our other models.

As previously stated, we were unable to achieve equivalency with the baseline models as entire code for either of the three models was unavailable. Also, the flat model used surpassed the baseline model for flat captioning on all but one evaluation parameters i.e., BLEU-4 and there too, its value lies very close to that of the baseline method. This tells us that paragraph captioning with non- hierarchical has still room for improvement.

4.4 Discussion

This thesis implements and compares performance of 3 model types for image captioning: flat model, hierarchical model and hierarchical- coherent model. Prior to training on Stanford- Paragraph Dataset, pre-training was also performed on two datasets. The results suggest that except for the fact that models that have been pre-trained on VG Regions scored well on BLEU-4, there was hardly any difference between models trained on VG regions and MS COCO. This can be justified as follows: Stanford-Paragraph dataset is basically an improvement over the VG regions [30]. All models used here profited from utilizing pre-calculated features, and based on our early trials, all models made use of pre-trained RNN weights.

4.4.1 Flat Versus Hierarchical Models

When it comes to automatic evaluation metrics scores, flat captioning model performs better than the hierarchical models. Based on our findings, it is not impossible to believe that if efforts are made to refine the flat captioning model, it will eventually achieve performance comparable to best systems. The flat model's main advantage is that it is built on a simpler architecture, making it easier to train and incorporate additional enhancements.

Looking at generated captions, it was found that he images in the validation dataset show a lot of variation; for some images, it is simpler to caption with paragraph length captions when compared to others. Shorter captions were generated by the flat approach, which frequently seemed to be an enhanced version of single sentence caption. In terms of quality, it appears that flat captioning model represents shorter descriptions and hierarchical-coherent model represents more extensive paragraph captions. However, it appears that greater linguistic complexity does not automatically imply greater value of fidelity, since almost all paragraphs contained some inaccuracies.

4.4.2 Possible Enhancements

A machine generated caption can be good if humans are unable to distinguish it from a human generated description So, depending on human evaluation criteria, how can we increase the performance of our models? A few ideas for improving scores attained by models are:

• Fidelity – Upgrading the process of input features extraction and incorporating a learning algorithm in which captions are generated based on inputs given.

• Intelligibility – Taking a more robust linguistic model that is trained on larger and diverse dataset;

• Adequacy – by making use of the caption scores obtained inside the loss function.

• Fluency – this has been worked upon in part by the coherence-based models. The improvement on this would require work on a number of parameters mentioned above.

CHAPTER 5
Conclusion and Future Work

This work, for paragraph image captioning, explores three different types of models. The most simple and basic is the flat model. It makes use of a CNN encoder and a RNN decoder in its native form. Hierarchical model introduces a top-level RNN. This helps in keeping a tab on the backdrop of the sentence. Hierarchical-coherent model makes use of coherence vectors to ensure that the gradient moves from endmost word of previous sentence to next sentence.

We used flat captioning, hierarchical captioning, and hierarchical-coherent captioning models in our experiments. Features taken from ResNet-152 [2] as well as DenseCap [1] features used in paragraph image captioning have been used as input. RNNs based on LSTM or GRU were utilised in the models. LSTM based RNN was employed for training most of the models. Performance comparisons revealed the existence of minute differences between methods involving LSTM and the methods involving GRU. Models were evaluated on CIDEr -D , Meteor and BLEU -1,2,3,4 . We did not detect a significant difference between GRU-based models and LSTM based models.

While several of our models came close to meet the baseline methods [3, 4], this work was not able to achieve scores as good as the scores mentioned by hierarchical models it is based upon. On all the following metrics, we outperformed the published Imageflat baseline model [5]. It has been demonstrated that it is relatively simple to obtain high evaluation metrics scores for paragraph captioning with any flat-model by making minor changes to image captioning model given by [6]. The performance of our flat model is quite near to baseline hierarchical captioning model in terms of CIDEr – D score.

Going ahead, approaches like attention learning and reinforcement leaning can be integrated into flat models as well as with hierarchical models. One such implementation can be the use of reinforcement leaning for improving Meteor and CIDEr scores. This will greatly improve the generative ability of the model. Yet another possible field of research would be to experiment with new models of dense captioning and look for the possibility of generating features better suited for paragraph captioning.

REFERENCES

1. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.
2. J. Johnson, A. Karpathy, and L. Fei-Fei. DenseCap: Fully convolutional localization networks for dense captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4565–4574, 2016.
3. M. Chatterjee and A. G. Schwing. Diverse and coherent paragraph generation from images. In Proceedings of the European Conference on Computer Vision (ECCV), pages 729–744, 2018.
4. J. Krause, J. Johnson, R. Krishna, and L. Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3337–3345. IEEE, 2017.
5. A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3128–3137, 2015.
6. O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3156–3164, 2015.
7. M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. arXiv preprint arXiv:1802.05365, 2018.
8. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
9. S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, page 3, 2017
10. X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft COCO captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325, 2015.
11. X. Liang, Z. Hu, H. Zhang, C. Gan, and E. P. Xing. Recurrent topic-transition GAN for visual paragraph generation. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 3362–3371, 2017.
12. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In NIPS- W, 2017.
13. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Largescale machine learning on heterogeneous systems, 2015. Software available from: https://www.tensorflow.org/.
14. R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4566–4575, 2015.
15. K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866, 2017.
16. M. Denkowski and A. Lavie. Meteor universal: Language specific translation evaluation for any target language. In Proceedings of the Ninth Workshop on Statistical Machine Translation, pages 376–380, 2014.
17. S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In Advances in Neural Information Processing Systems (NIPS), pages 493–499, 1996.
18. X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS), pages 249–256, 2010.
19. G. A. Miller. WordNet: a lexical database for English. Communications of the ACM, 38(11):39–41, 1995.
20. K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pages 311–318. Association for Computational Linguistics, 2002.

21. D. M. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.
22. J. White, T. O'Connell, and F. O'Mara. The ARPA MT evaluation methodologies: Evolution, lessons, and future approaches. In Proceedings of the First Conference of the Association for Machine Translation in the Americas, pages 193–205, 1994.
23. National Research Council (US). Automatic Language Processing Advisory Committee. Language and machines: Computers in translation and linguistics; a report, volume 1416. National Academies, 1966.
24. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In European Conference on Computer Vision (ECCV), pages 740–755. Springer, 2014.
25. R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision (IJCV), 123(1):32–73, 2017.
26. P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6077–6086, 2018.
27. B. Dai, S. Fidler, R. Urtasun, and D. Lin. Towards diverse and natural image descriptions via a conditional gan. arXiv preprint arXiv:1703.06029, 2017.
28. J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-tosequence learning. arXiv preprint arXiv:1603.06393, 2016.
29. G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In Advances in Neural Information Processing Systems (NIPS), pages 971–980, 2017.
30. Li, M.-T. Luong, and D. Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. arXiv preprint arXiv:1506.01057, 2015.
31. K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning (ICML), pages 2048– 2057, 2015.
32. S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, pages 65–72, 2005.
33. A. Lavie. Evaluating the output of machine translation systems, 2011. https://www.cs.cmu.edu/~alavie/Presentations/MT-Evaluation-MT-Summit-Tutorial-19Sep11.pdf, accessed 2019-03-19.
34. X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft COCO captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325, 2015.
35. R. Lin, S. Liu, M. Yang, M. Li, M. Zhou, and S. Li. Hierarchical recurrent neural network for document modeling. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 899–907, 2015.
36. H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu. Video paragraph captioning using hierarchical recurrent neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4584–4593, 2016.
37. R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. Transactions of the Association for Computational Linguistics, 2:207–218, 2014.
38. D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
39. O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 39(4):652–663, 2017.
40. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.
41. I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems (NIPS), pages 3104–3112, 2014.
42. I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. MIT press, 2016.

43. A. Alzu'bi, A. Amira, and N. Ramzan. Semantic content-based image retrieval: A comprehensive study. Journal of Visual Communication and Image Representation, 32:20–54, 2015.
44. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015.
45. S. J. Pan and Q. Yang. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359, 2010.
46. J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems (NIPS), pages 3320–3328, 2014.
47. A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS), pages 1097–1105, 2012.
48. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
49. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, 2015.
50. J. Lu, C. Xiong, D. Parikh, and R. Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 375–383, 2017.
51. Y. Bengio, P. Simard, P. Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.
52. T. Mikolov, M. Karafiát, L. Burget, J. Cernock`y, and S. Khudanpur. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association, 2010.
53. T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 746–751, 2013.
54. Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. Journal of Machine Learning Research, 3(Feb):1137–1155, 2003.
55. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems (NIPS), pages 3111–3119, 2013.
56. J. Mitchell and M. Lapata. Composition in distributional models of semantics. Cognitive Science, 34(8):1388–1429, 2010.
57. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.
58. J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.
59. A. Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.
60. S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.
61. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.
62. R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3-4):229–256, 1992.
63. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS), pages 2672–2680, 2014.
64. R. Shetty, M. Rohrbach, L. A. Hendricks, M. Fritz, and B. Schiele. Speaking the same language: Matching machine to human captions by adversarial training. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017.
65. Y. Pu, W. Yuan, A. Stevens, C. Li, and L. Carin. A deep generative deconvolutional image model. In Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics (AISTATS), pages 741–750, 2016. L. Wang, A. Schwing, and S. Lazebnik. Diverse and accurate image description using a variational auto-encoder with

an additive gaussian encoding space. In Advances in Neural Information Processing Systems (NIPS), pages 5756–5766, 2017.

66. D. P. Kingma and M. Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.
67. R. Kiros, R. Salakhutdinov, and R. S. Zemel, “Unifying visual-semantic embeddings with multimodal neural language models,” arXiv preprint arXiv:1411.2539, 2014.
68. L. Fei-Fei, A. Iyer, C. Koch, and P. Perona, “What do we perceive in a glance of a real-world scene?” Journal of vision, vol. 7, no. 1, pp. 10–10, 2007.
69. Y. Feng and M. Lapata, “Visual information in semantic representation,” in Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2010, pp. 91–99.
70. J. Fan, Y. Gao, and H. Luo, “Hierarchical classification for automatic image annotation,” in Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007, pp. 111–118.
71. B. Z. Yao, X. Yang, L. Lin, M. W. Lee, and S.-C. Zhu, “I2t: Image parsing to text description,” Proceedings of the IEEE, vol. 98, no. 8, pp. 1485–1508, 2010.