الـــجـــــمـــهوريـــــة الـــــجزائـــــريـــــــــة الديـــــمقراطـــية الـــشعـــــبية
PEOPLES DEMOCRATIC REPUBLIC OF ALGERIA

وزارة الـــــتـــعـــليـــم الـــعـــالـــــي و الـــــــبـحـث الـــعلـــمي
Ministry of Higher Education and Scientific Research

جـــــــامعة غـــردايـــــة
University of Ghardaia

كـلية الـعلـوم و الـــــتكنولـــوجيا
Faculty of Science and Technology

قـــسم الرياضيـــات و الإعـــلام الآلـــي
Department of Mathematics and Computer Science

THESIS

Presented for the degree of Masters

In Computer Science

Specialty Intelligent Systems For Knowledge Extraction

By Safa batoul korichi and Karim aimene

Theme

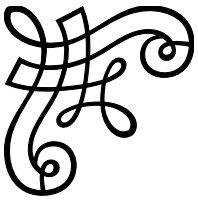
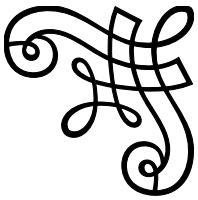
Automatic Image Caption Generation: study and implementation

Jury Members

M. MAHDJOUB Youcef	MAA	Univ. Ghardaia	Supervisor

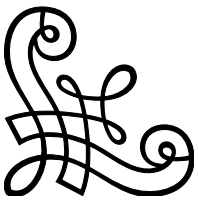
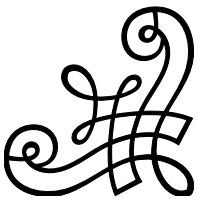
College Year 2020/2021


بسم الله الرحمن الرحيم

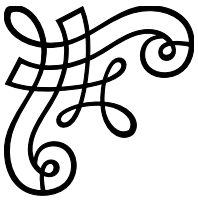



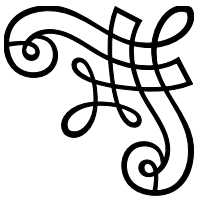
Dedicated

Id to dedicate my work to
My dear father, may God rest his soul, how much I wished he had been with me on this day and that he was proud of me.
my life mother the crown of my head, who completed her reign in my upbringing and morality and tired with me to this day whose words of encouragement and pressure for perseverance ring in my ear that did not leave my side. Her belief in my abilities is the reason for my success and the moral support that has always given me hope. May God prolong your age.
My dear sisters one by one, for their constant encouragement and moral support.
My beloved, my sister's children's, and their husbands especially (Yacine).
A special dedication to my partner friend Karim Aimene for his patience and hard work throughout the work period. and all my friends of the university and my life

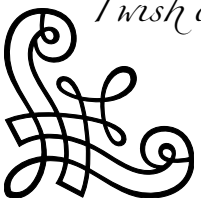



safa batoul korichi





To my beloved mother for all her prayers and benedictions and help.,for her endless love and encouragement.
My father, Thanks for providing for my needs, and Thanks for all the guidance and encouragement.
My dear sisters one by one, for their constant encouragement and moral support.
To my friend and partner safa for her hard work and lots of support.
I wish all the best for her. To all of my colleagues and my friends



Karim Aimene

Acknowledgment

First thanks to God Almighty, for giving us the strength and courage to complete this thesis.

we'll say thanks to our parents for all of their efforts and their Confidence.
And we would like to express our thanks and gratitude to we supervisor Mr, Mahdjoub Youcef who have contributed and helped in the realization of this master's thesis
Thank you so much, to all the respected teachers who have taught us all the years we've spent in The university.
Thank you all.

Abstract

Artificial Intelligence (AI) is currently moving increasingly towards multimodal learning which involve build system that can process information from multiple sources, such as text, images or audio. Image captioning is one of the main visual-linguistic tasks that requires generating captions to a specific image. The challenge is to create a unified Deep Learning (DL) model, suitable to describe an image in a correct sentence. To do so, we need to understand the proper way to visualize the text in a certain space. We used the new term of Transformer that brings a new concept into a sequence to sequence mechanism, we also include the power of modern GPU in processing data in an efficient and faster manner. In this path, we have experimented with a Transformer-based approach and applied it to the image captioning problem using MS COCO dataset.

Keywords: Multimodal Learning, Image captioning, Deep Learning (DL), Transformer, Sequence to sequence, MS-COCO.

Résumé

Mots clés : L'intelligence artificielle (IA) s'oriente actuellement de plus en plus vers l'apprentissage multimodal, qui consiste à construire un système capable de traiter des informations provenant de sources multiples, telles que du texte, des images ou du son. La description d'images est l'une des principales tâches visuelles-linguistiques qui nécessite de générer des légendes pour une image spécifique. Le défi consiste à créer un modèle unifié d'apprentissage profond (AP), capable de décrire une image avec des expressions correctes. Pour ce faire, nous devons comprendre la manière appropriée de visualiser le texte dans un certain espace. Nous avons utilisé le nouveau terme de Transformer qui apporte un nouveau concept dans un mécanisme de séquence à séquence, nous avons également exploité la puissance des GPU modernes afin de traiter les données d'une manière efficace et plus rapide. Dans ce parcours, nous avons expérimenté une approche basée sur le Transformer et nous l'avons appliquée au problème de la description d'images en utilisant le jeu de données MS-COCO.

Mots-clés: Apprentissage Automatique (AA), Apprentissage Profond (AP), Réseaux de Neurones convectif(RNC), Réseaux de Neurones Artificiels (RNA), TRANSFORMER, MSCOCO.

ملخص

الذكاء الاصطناعي يتجه حاليا نحو التعلم متعدد الاشكال من اجل بناء نظام قادر على التعامل مع مختلف المداخل صور نصوص او صوت. وصف الصور واحدة من اساسيات مهام الصورة واللغة من اجل انشاء تعليق على الصورة المحددة يكمن التحدي في انشاء نموذج تعلم عميق قادر على وصف الصورة بخبرة واسعة وجيدة، من اجل ذلك يجب فهم الطريقة الصحيحة لتمثيل النص في الفضاء. لقد استخدمنا المصطلح الجديد المتحول الذي يجلب طريقة جديدة لمتسلسل متسلسل، لقد استخدمنا قوة معالجات وحدة الرسوميات من اجل معالجة جيدة وسريعة في طريقنا في هدا النموذج اختبرنا تصميم مبني على متحول واختبرناه على مشكلة التعليق على الصور في مجموعة البيانات MSCOCO

كلمات مفتاحية: : نموذج متعدد , وصف الصور, التعلم العميق , المتحول , متسلسل لمتسلسل ,MSCOCO

.

Contents

List of Tables

List of Figures

List of Abbreviations

Adam Adaptive Moment Estimation Algorithm

ANN Artificial NeuralNetwork

BERT Bidirectional Encoder Representations from Transformers

BLEU BLilingual Evaluation Understudy

CE Cross Entropy

CNN Convolutional Neural Network

CReLU Concatenated ReLU

DL Deep learning

ELU Exponential Linear Unit

FFNN Feed Forward Neural Network

GD Gradient Decent

GRU Gated Recurrent Units

llSVRC ImageNet LargeScale Visual Recognition Challenge

LSTM Long Short Term Memory

METEOR Metric for Evaluation Translation with Explicit ORdering

Microsoft Common Objects in Context Microsoft CommonObjectsContext

ML Machine Llearning

NLP Natural Language Processing

ReLU Rectified Linear Unit

Resnet RESidual NETworks

RL Reinforcement Learning

RNN Recurrent Neural Network

ROUGE Recall Oriented Understudy Gisting

SOAT State Of The Art

SVM Support Vector Machine

TRIC TRansfomer Image Ccaptioing

VGG Visual Geometry Group

Introduction

Introduction

Artificial intelligence is in a set of algorithms and intelligence to try to mimic human intelligence. Machine learning is one of them, and deep learning is one of those machine learning techniques. a recent study on Deep Learning shows that it is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Deep Learning (DL) and Neural Network (NN) are currently driving some of the most ingenious inventions

In today's century. Their incredible ability to learn from data and the environment makes them the first choice of machine learning scientists. Deep Learning and Neural networks lie in the heart of products such as self-driving cars, image recognition software, recommender systems, Image captioning, etc... Being in a powerful algorithm, it is highly adaptive to various data types as well. Generating a description of an image it's image captioning. captioning of the image requires recognizing the important objects, their attributes, and their relationships in an image. It also needs to generate syntactically and semantically correct sentences. Deep learning techniques are capable of handling the complexities and challenges of image captioning. In this thesis, we aim to present a comprehensive review of existing deep learning-based image captioning. Our different approaches, We also discuss the datasets and the evaluation metrics popularly used in deep learning-based automatic image captioning, we explore the most used architecture for constructing image caption Generation with the training of dataset and their experimentation. This thesis includes three chapters:

- chapter 1:

  this chapter contained some definitions and concepts about AI, ML, and some forms of it. Moreover, we will explain Deep Learning and some of its tools and techniques in a more profound manner. and the data-sets with the evaluation metrics popularly used in deep learning.

- Chapter 2:

  this chapter firstly briefly describe some vision task models with a definition of the image captioning field. Then, explore some state-of-the-art existing deep learning-based image captioning.

- Chapter 3:

  In this chapter, we will try to implement the image captioning project on the Flicker data set. Then, We will discuss the architecture and explain the software, hardware, and the results we had on this experiment.

Motivation

In the Internet age, various long or short videos flood our lives. Video understanding has become an essential task. A video is composed of several frames of images to regard image comprehension as low-level video comprehension. When watching a video, we sometimes cannot obtain a high-level understanding of the video from the captions. Video captioning and image captioning technology can help us further understand their content. Compared with the factual caption, a caption can better help us understand videos or images. Captioning is for ordinary audiences, but captioning technology can greatly help the visually impaired understand videos and images. will be useful in cases/fields where text is most used, and with the use of this, you can infer/generate text from images. As in, use the information directly from any particular image in a textual format automatically. There are many NLP applications right now, which extract insights/summary from a given text data or an essay etc. The same benefits can be obtained by people who would benefit from automated insights from images. A slightly (not-so) long-term use case would be, explaining what happens in a video, frame by frame. Lots of applications can be developed in that space. Social Media. Platforms like Facebook can infer directly from the image, where you are (beach, cafe, etc), what you wear (color), and more importantly what you're doing also (in a way). See an example to understand it better.

CHAPTER 1

MULTI MODEL DEEP LEARNING CONCEPTS AND BACKGROUND

1.1 Introduction

Artificial intelligence is a type of computer technology which is concerned with making machines work in an intelligent way, similar to the way that the human mind works.Nowadays, AI became among the most interesting fields of computer science, due to its efficiency in solving very complex problems and automating day to day tasks that were once considered hard, rather impossible for a machine to comprehend. A lot of the credit for that goes to ML and DL.
In this chapter, we see some definitions and concepts about ML and some forms of it. Moreover, we will explain DL and some of its amazing tools and techniques in a more profound manner.

1.2 Machine Learning

ML is a part of AI, which uses several techniques to enable computers to learn from data and predict results.The term machine learning was popularized by Arthur Samuel in 1959 as:
"... the field of study that gives computers the ability to learn without being explicitly programmed"[2].
Moreover, a widely quoted definition was proposed by Tom Mitchell who stated that:
"

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" [3].

ML create a new generation of programming concept for solving problems. Unlike classical programming where we feed the program rules and data expecting answers (e.g., the sum of two integers), ML algorithms receive data and answers as labeled data[1], and to figure out the rules using statistical methods based on the data (Figure 1.1). Additionally, the algorithm is tested to judge its performance using the answers provided earlier.

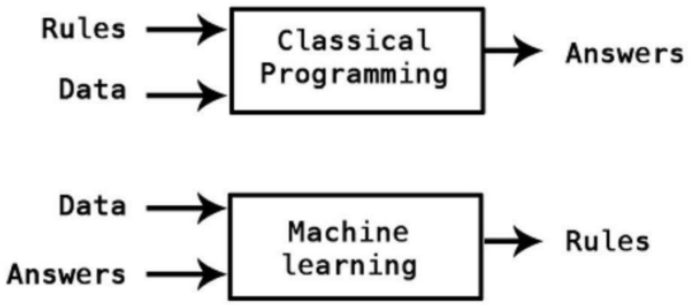


Figure 1.1: The difference between ML and classical programming.

1.2.1 Types Of Machine Learning

Supervised Learning Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.[2] $Y = f(x)$. The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training data-set can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance. Basically there is tow categories

[1] some time we don't need to knew the answers we only need to identify the pattern from the data. this will be discussed in subsection 1.2.1

[2] The input could be any form text, image, signal

- Classification: A classification problem is when the output variable is a category, such as "red" or "blue" or "disease" and "no disease".
- Regression: A regression problem is when the output variable is a real value, such as "dollars" or "weight"

Some popular examples of supervised machine learning algorithms are:

- Linear regression for regression problems.
- Random forest for classification and regression problems.
- Support vector machines for classification problems.

Unsupervised Learning Unsupervised learning is where you only have input data (X) and no corresponding output variables, the goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data. Unsupervised learning problems can be further grouped into :

- Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Reinforcement Learning Reinforcement Learning (RL) is the science of decision making, it is about learning the optimal behavior in an environment to obtain maximum reward. This optimal behavior is learned through interactions with the environment and observations of how it responds. Similar to children exploring the world around them and learning the actions that help them achieve a goal. Any real-world problem where an agent must interact with an uncertain environment to meet a specific goal is a potential application of RL. Here are a few RL examples :

- Robotics(Reinforcement Learning to Aerobatic Helicopter Flight) [4].
- games (masters chess, shogi and Go through self-play with RL) [5].

1.3 Deep learning

" For decades, in order to get computers to respond to our requests for information, we had to learn to speak to them in a way they would understand, "Tom Wilde CEO at Indico Data Solutions. [3].
This meant having to learn things like Boolean query language[4], or how to write complex rules that our computer can understand it and action on it. This forced a major imposition on the user and meant that only a relatively few skilled people could successfully retrieve information from computer-based information systems.

Deep learning is the one category of machine learning that emphasizes training the computer about the basic instincts of human beings, in deep learning a computer algorithm learns to perform classification tasks directly on any form of complex data like:

- images(image recognition[6])
- text(language translation[7])
- sound(speech recognition[8]).

This make deep learning has become a buzzword in the tech community and ironically, abstract and formal tasks that are among the most difficult mental undertakings for a human being are among the easiest for a computer

[3]Tom have 25 years of experience in solving the complex problems of digital content, the forefront of innovation in Enterprise AI.
[4]Boolean search could be hotel AND New York. This would limit the search results to only those documents containing the two keywords

1.4 Artificial Neural Network

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by biological nervous systems, similar as the brain. In 1957 Rosenblatt Frank introduce the first term of ANN it call him Perceptron [9]

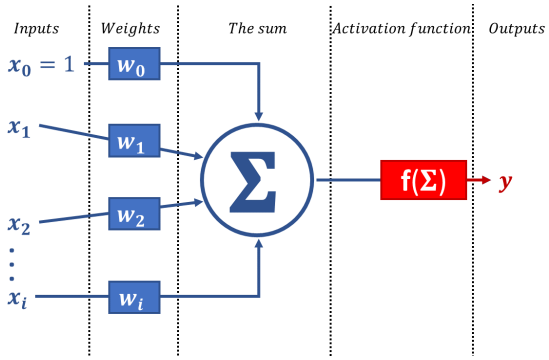


Figure 1.2: The structure of simple ANN .

As shown in figure 1.2 An artificial single neuron is represented by a mathematical function. It takes i inputs x, and each of them usually has its own weight W. The neuron calculates the sum and it is passed through the activation function to the network further, that make it formula: 1.1

$$y = \phi(\sum_{i=1}^{n} w_i x_i) \tag{1.1}$$

where ϕ is an activation function, we will talk about a few activation function in next section 1.5

1.5 Activation Functions

Activation functions define the output of the neuron in a range of values, there are different types of activation functions, means that they are responsible for calculating the sum of the product of the various weights and inputs with the bias[5] to determine the final output value for the current hidden layer, which would be the input for the next layer, in next subsection we will discuss some of theme and their characteristics

1.5.1 Rectified Linear Unit

Rectified Linear Unit (ReLU) is very popular equation calculated by 1.2:

$$y = f(z) = \max(0, z) \tag{1.2}$$

This function output is 0 if the input is negative, if the inputs is positive the outputs is the same Relu is a amazing function because of it advantages like cheap computational cost and low training time, eliminating the vanishing gradients problems [6]. Figure 1.3 shows the plot for the ReLU function.

[5] bias value allows the activation function to be shifted to the left or right, to better fit the data.
[6] There exists some variations for ReLU such as the Exponential Linear Unit (ELU)[10] , Concatenated ReLU (CReLU)[11] , etc.

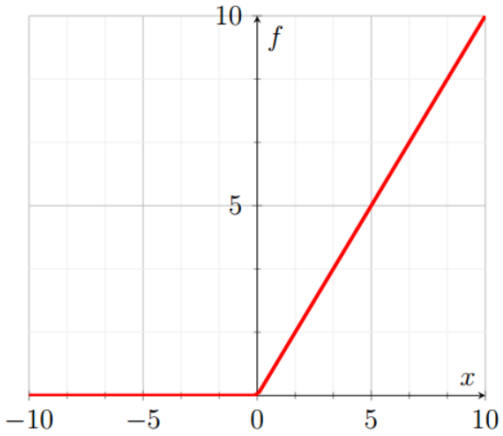


Figure 1.3: The ReLU function plot

1.5.2 Sigmoid

The Sigmoid function is a special form of the logistic function. It is often used in models that output a probability, which is a value around 0 and 1. let's say that we have a data and we need to estimate the prediction the Sigmoid is the solution says how much is the prediction and how much is the opposites
The Sigmoid function is defined mathematically by equation 1.3 and figure 1.4 represents its plot:

$$y = f(z) = \frac{1}{1 + e^{-z}} \tag{1.3}$$

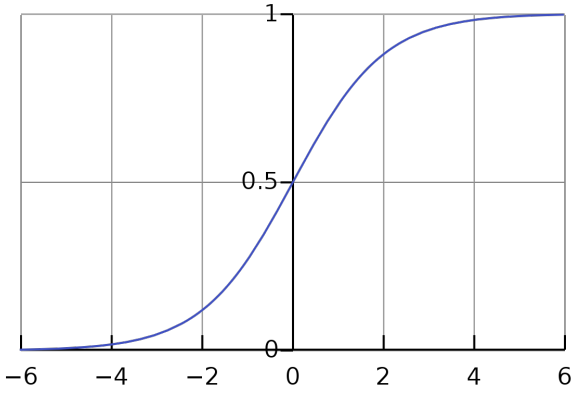


Figure 1.4: The Sigmoid function plot

One of its benefits is that its is non linear, thus it can be used to classify non linear data properly. However, In areas where z is very large or very low, the y value barley reacts to changes of z, this causes the gradient to be very small, leading to the vanishing gradient problem which makes training slower and more complicated. Despite that, Sigmoid is still very popular in classification problems.

1.5.3 Softmax

in the Sigmoid function we say that output is confined between 0 and 1 and the class is more then two the sigmoid wont take the probability of the whole class the solution is applying Softmax. The Softmax

function is defined by equation 1.4:

$$y = f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{1.4}$$

This function basically takes the input vector x and calculates a sum of the exponentials of its components $\sum_j e^{x_j}$. Then, to compute the first value of the output vector x_1 it just divides the exponential of x_1 by the first sum, and so on(equation 1.4). The final output is a vector of normalized probabilities that all add up to 1. This vector represents a trivial tool to determine the highly favorable class according to the model, as a trained model will output a vector in which the target class has the highest probability.
In the next section we will discuss some of the famous neural network types, starting with Feed-Forward Networks.

1.6 Model of ANN

1.6.1 Feed Forward Neural Network

A Feed Forward Neural Network is an artificial neural network in which the connections between nodes does not form a cycle. The opposite of a feed forward neural network is a recurrent neural 1.6.4 network, in which certain pathways are cycled. Unlike other forms of neural networks, feed-forward can only process information in one direction. although data may go through multiple hidden nodes it always flows in one direction, and never backwards.
A FNN is commonly seen in its simplest form as a single layer perceptron as shown in figure **??**

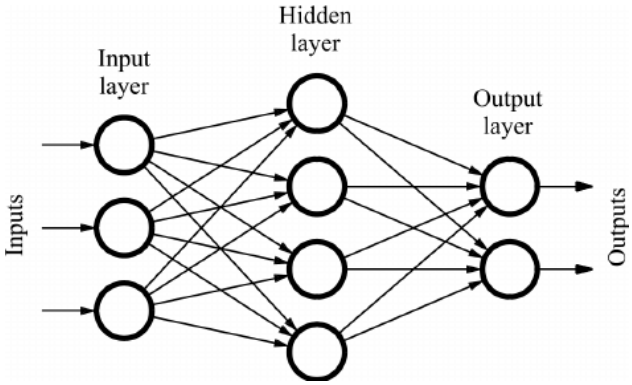


Figure 1.5: Sample of a feed forward neural network.

In this model as shown in figure 1.6, a series of inputs enter the layer and are multiplied by the weights. Each value is then added together to get a sum of the weighted input values. If the sum of the values is above a specific threshold, usually set at zero, the value produced is often 1, whereas if the sum falls below the threshold, the output value is −1. The single layer perceptron is an important model of feed forward neural networks and is often used in classification tasks. Furthermore, single layer perceptrons can incorporate aspects of machine learning. Using a property known as the delta rule, the neural network can compare the outputs of its nodes with the intended values, thus allowing the network to adjust its weights through training in order to produce more accurate output values. This process of training and learning produces a form of a gradient descent.

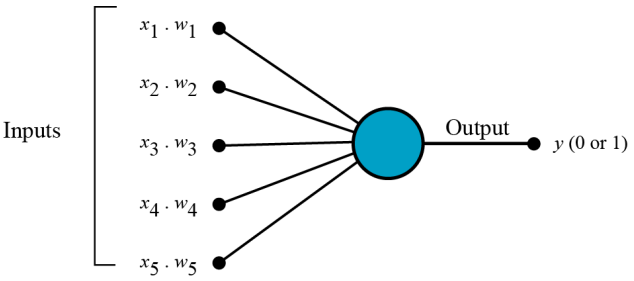


Figure 1.6: Multiplying inputs with weights for 5 inputs.

In multi-layered perceptrons, the process of updating weights is nearly analogous, however the process is defined more specifically as back-propagation. In such cases, each hidden layer within the network is adjusted according to the output values produced by the final layer.

1.6.2 Convectional Neural Networks

CNNs were inspired by the studies of the brain cortex in 1980 by Dr. Kunihiko Fukushima [12] after 10 years Yann LeCun et al.[13] present the new term of CNN in the field of machine learning

Convolutional Neural Network (CNN) is a widely used structure in an image processing, classification, segmentation and also for other auto correlated data field. It eliminates the need for manual feature extraction by taking advantage of learnable filters able to detects various image characteristics such as edges and shapes. There are many variants of CNNs but all of them share 3 main building blocks: convolution operation, convolutional layers, and pooling layers.

convolution operation : element-wise multiplication between filter and input, in the case of images, it is best to present a convolution operation on the 2-dimensional matrices for visualization purposes see Figure 1.7 the filter or kernel is a small matrix of weights. This kernel "slides" over the 2D input data.

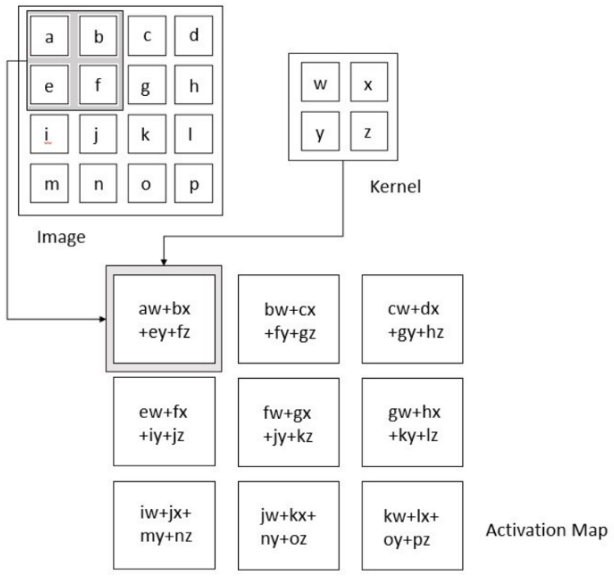


Figure 1.7: Convolution operation on 2-dimensional matrices

Many filters are used in CNN like

- Valid Convolution: Using a filter of an odd size causes the feature map to be smaller then the input image. This particular kind of convolution is called valid convolution.
- Same Convolution to have the same height and width as the input layer, it is common to add zeros around the input image. It is called zero padding and the operation becomes same convolution

convolutional layers : are the most important building block for CNNs. Unlike fully connected layers, a neuron in a convolutional layer is not connected to every neuron from the previous layer. Instead, it is connected to only a group of neurons called the receptive field

pooling layers : Similar to the Convectional layer the polling is responsible of reducing the spatial size of the convulsed features, also in a way tis decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model. There are two types of Pooling:

- Max Pooling returns the maximum value from the portion of the image covered by the Kernel.
- Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

their are some of Modern Convolutional Neural Networks that have use in task of classification, in the next subsection we will talk about few of them.

1.6.3 Modern Convolutional Neural Networks

Visual Geometry Group VGG is a CNN model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper [14] The model achieves the first and the second places in the challenge of(IlSVRC) [7] [8]

It is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million parameters.as shown in figure 1.8.

[7] (ILSVRC) is an annual competition helped between 2010 and 2017 in which challenge tasks use subsets of the ImageNet1.6.3 dataset.
[8] dataset spans 1000 object classes and contains 1,281,167 training images, 50,000 validation images and 100,000 test images

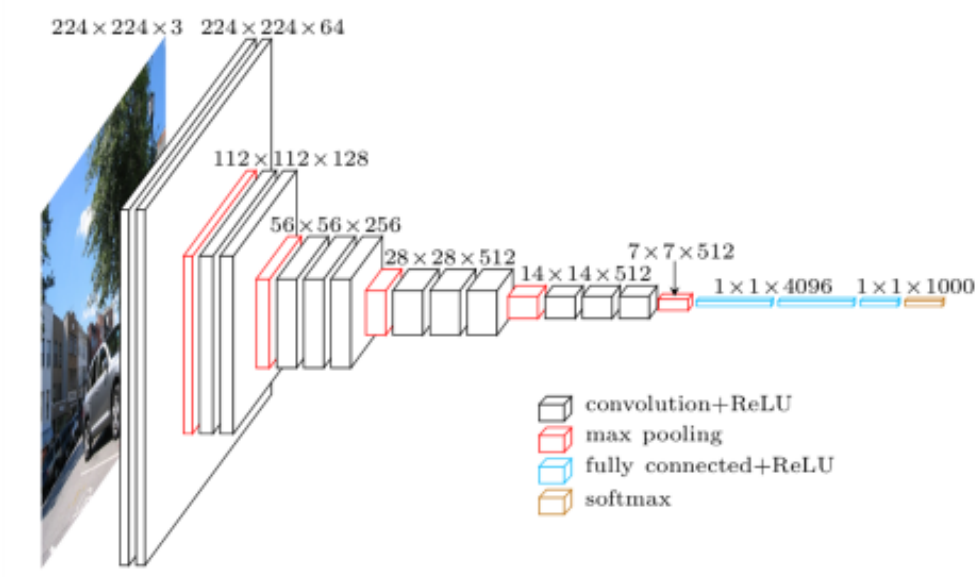


Figure 1.8: Architecture of VGG16

Unfortunately, It is slow to train and The network architecture weights themselves are quite large which make it not perfect to classification

Resnet Recently proposed residual networks (ResNets) get state of the art(SOTA) performance on the ILSVRC 1.6.3 2015 classification task (Deng et al., 2009 [15]) and allow training of extremely deep networks up to more than 1000 layers. Similar to highway networks, residual networks make use of identity shortcut connections that enable flow of information across layers without attenuation that would be caused by multiple stacked non-linear transformations, resulting in improved optimization . In residual networks, shortcut connections are not gated and untransformed input is always transmitted. While the empirical performance of ResNets in is very impressive, current residual network architectures have several potential limitations: identity connections as implemented in the current ResNet leads to accumulation of a mix of levels of feature representations at each layer, even though in a deep network some features learned by earlier layers may no longer provide useful information in later layers . [16]

A hypothesis of the ResNet architecture is that learning identity weights is difficult, but by the same argument, it is difficult to learn the additive inverse of identity weights needed to remove information from the representation at any given layer. The fixed size layer structure of the residual block modules also enforces that residuals must be learned by shallow subnetworks, despite evidence that deeper networks are more expressive (Bianchini et Scarselli, 2014 [17]). show below a generalized residual architecture that combines residual networks and standard convolutional networks in parallel residual and non-residual streams. We show architectures using generalized residual blocks retain the optimization benefits of identity shortcut connections while improving expressivity and the ease of removing unneeded information. and then show presentation of novel architecture, ResNet in ResNet (RiR), which incorporates these generalized residual blocks within the framework of a ResNet.

Generalization Residual Network Architecture :

The modular unit of the generalized residual network architecture is a generalized residual block consisting of parallel states for a residual stream, r, which contains identity shortcut connections and is similar to the structure of a residual block from the original ResNet with a single convolutional layer ($parametersWl,rr$), and a transient stream, t, which is a standard convolutional layer

(Wl,tt). Two additional sets of convolutional filters in each block

(Wl,rt,Wl,tr) also transfer information across streams.

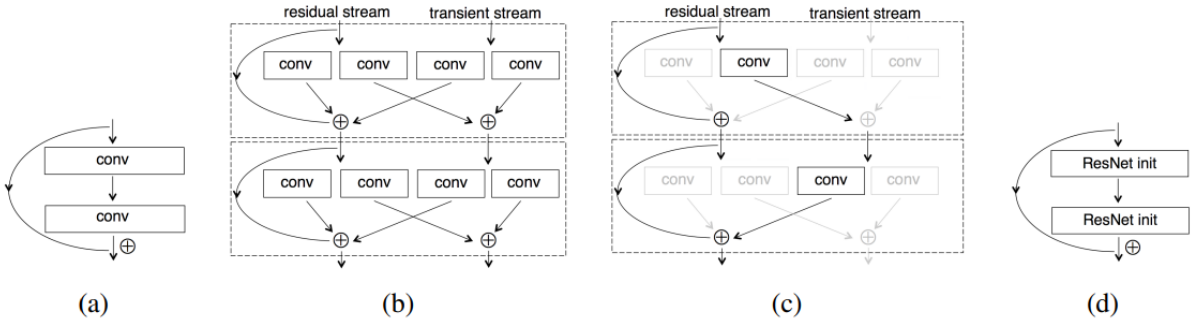


Figure 1.9: (a) 2-layer ResNet block. (b) 2 generalized residual blocks (ResNet Init). (c) 2-layer ResNet block from 2 generalized residual blocks (grayed out connections are 0). (d) 2-layer RiR block.

	generalized residual blocks		
with shortcut connections		N	Y
	N	CNN	ResNet Init
	Y	ResNet	RiR

Figure 1.10: Relationship between standard CNN, ResNet, ResNet Init, and RiR architectures.

$$rl+1 = (conv(rl, Wl, rr) + conv(tl, Wl, tr) + shortcut(rl))$$
$$tl+1 = (conv(rl, Wl, rt) + conv(tl, Wl, tr))$$

The generalized residual block can act as either a standard CNN layer (by learning to zero the residual stream) or a single-layer ResNet block (by learning to zero the transient stream). By repeating the generalized residual block several times, the generalized residual architecture has the expressivity to learn anything in between, including the standard 2-layer ResNet block (Figure 1.6.3 c).

This architecture allows the network to learn residuals with a variable effective number of processing steps before addition back into the residual stream, which we investigate by visualizations. The generalized residual block is not specific to CNNs, and can be applied to standard fully connected layers and other feedforward layers. Replacing each of the convolutional layers within a residual block from the original ResNet (Figure 1.6.3 a) with a generalized residual block (Figure 1.6.3 b) leads us to a new architecture we call ResNet in ResNet (RiR) (Figure 1.6.3 d). In figure 1.10, its summarize the relationship between standard CNN, ResNet Init, ResNet, and RiR architectures.

Inception from ImageNet challenge in 2014 (ILSVRC14) Szeged and his group in their paper Going deeper with convolutions [18] they came up with this architecture including a filters with multiple sizes operate on the same level, this would make the network "wider" rather than "deeper". in the Figure 1.11 below is the "naive" inception module. It performs convolution on an input, with 3 different sizes of filters (1x1, 3x3, 5x5). Additionally, max pooling is also performed. The outputs are concatenated and sent to the next inception module.

Then to reduce and limit the number of input channels they add an extra 1x1 convolution before

the 3x3 and 5x5 convolutions. Though adding an extra operation may seem counter intuitive, 1x1 convolutions are far more cheaper than 5x5 convolutions, and the reduced number of input channels also help. Do note that however, the 1x1 convolution is introduced after the max pooling layer, rather than before.

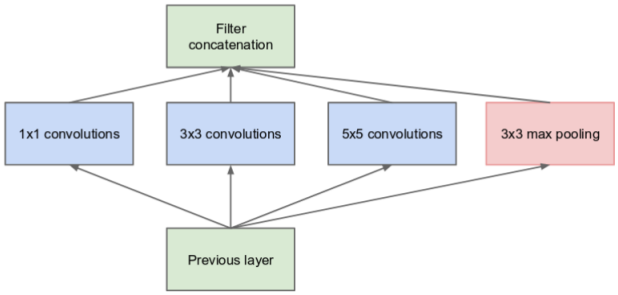


Figure 1.11: Architecture of naive version

1.6.4 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is the opposite of FFNN 1.6.1 Due to their temporal memory, RNNs can process sequences of inputs.this architecture is used for working with sequences. RNN processes data sequentially, encoding information in hidden state ht which is passed through every time step 1.12

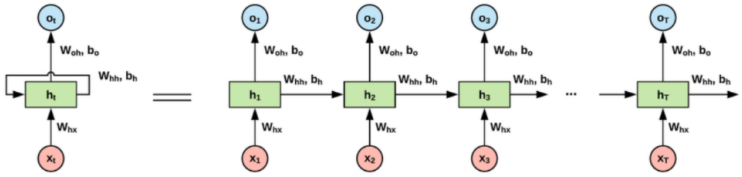


Figure 1.12: RNN architecture unfolded for every time step

Formally it can be defined as follows:

$$h_t = g_1(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \tag{1.5}$$

$$o_t = g_2(W_{oh}h_t + b_0) \tag{1.6}$$

where ht is the hidden state, xt is the input, bh, $b0$ are biases, ot is the output and $g1()$, $g2()$ are activation functions.

This structure can be extended to many layers because the output ot can be an input to another RNN layer. Stacking up many layers causes two problems: vanishing and exploding gradients. Gradients of RNNs output concerning the parameters of early layers are becoming very small (vanishing gradients) or very big (exploding gradients). One of the solutions to that problem is the modification of network architecture to the Long Short-Term Memory network which is presented in the next subsection.

1.6.5 Modern Recurrent Neural Networks

LSTM It was 1997 when Hochreiter and Schmidhuber in their paper[19] introduced the RNN architecture as a solution to the vanishing gradient problem. the idea is the ability to remove or add information to the cell state, carefully regulated by structures called gates. 1.13

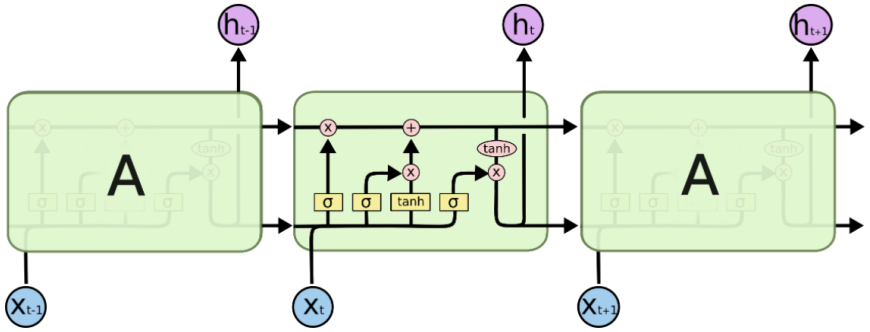
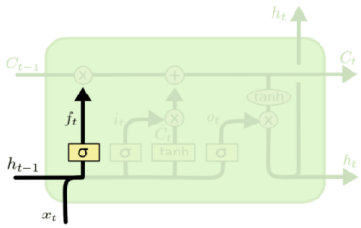


Figure 1.13: The repeating module in an LSTM contains four interacting layers.

The Forget Gate : This gate is to determine which parts of the information should be erased, the output of this gate $f_t$is forwarded to an element wise multiplication with the long term cell state from the previous time step C_{t-1} to modify it accordingly.The vector f_t is calculated by equation 1.7

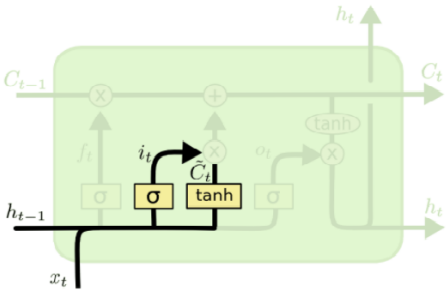


$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t] + b_f) \tag{1.7}$$

Figure 1.14: forget gate.

Tanh activation function This is not necessarily a gate but its job is to analyze the current inputs X_t and the previous short term state h_{t-1} to produce a vector of new candidate values for C_t. $\tilde{C}_t$ is computed by equation 1.9.

Input Gate This sigmoid layer decides what information from the previous vector (candidate values) to add to the cell state. i_t is computed by equation 1.8.



$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i) \tag{1.8}$$

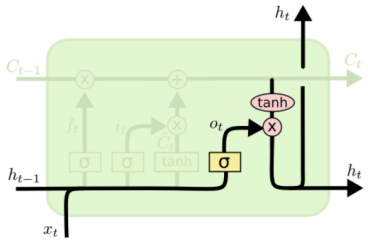
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, X_t] + b_C) \tag{1.9}$$

Figure 1.15: Input Gate.

All what is left is to perform the updates to the previous long term state to get the current cell state C_t according to equation 1.10:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{1.10}$$

Output Gate This gate filters the long term state C_t and decides what should be read and output at this time step h_t. It is calculated by equation 1.11, where o_t is the output of the output gate and its given by equation 1.12:



$$h_t = o_t * tanh(C_t) \tag{1.11}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o) \tag{1.12}$$

Figure 1.16: Output Gate.

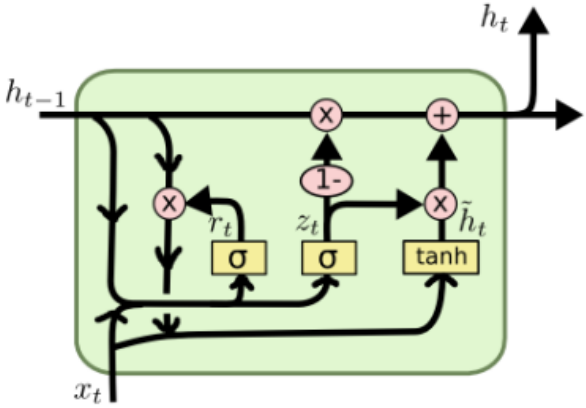
GRU In 2014 Cho, et al. in their paper [20] brought the vanishing gradient solution and similar to LSTM 1.6.5 in some cases, the working technique is very simple "update gate" and "reset gate"1.17

Update gate : Starting with calculating gate z_t for time step by formula **??** which mean the input X_t is multiplied by a weight W_{zx} then previous output H_{t-1} which hold information from previous units multiplied by a weight W_{zh} then applying sigmoid function 1.3 to squeeze the output between 0 and 1.1.13

Reset gate : Calculating the update gate R_t at time step t starting by multipling input X_t by a weight W_{rx} then multipling previous output H_{t-1} which hold information from previous units by a weight W_{rh} at the end sigmoid on them1.14

Current memory content : Input X_t is multiplied by a weight W_x, applying element-wise multiplication to the reset gate R_t and the previous output H_{t-1}, this allows to pass only the relevant past information then both are added together and a tanh function 1.11 is applied1.15

Final memory at current time step : at the end the unit has to calculate the H_t vector which holds information for the current unit and it will pass it further down to the network. Key role in this process plays the update gate Z_t. Applying element-wise multiplication to the update gate Z_t and $H_t$then Apply element-wise multiplication to one minus the update gate $1-Z_t$ and H_t.then soming together 1.16



$$z_t = \sigma(W_z[h_{t-1}, x_t]) \tag{1.13}$$

$$r_t = \sigma(W_r[h_{t-1}, x_t]) \tag{1.14}$$

$$r_t = \tanh(W[r_t * h_{t-1}, x_t]) \tag{1.15}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t \tag{1.16}$$

Figure 1.17: The structure of GRU.

If the vector Z_t is close to 0 then current content will be not consider since it is irrelevant for our prediction. At the same time since $Z_t$will be close to 0 at this time step, $1-Z_t$ will be close to 1 allowing the majority of the past information to be in counting

Encoder-Decoder Architecture In some problem like machine translation1.6.5 [9] our input and output to RNN layer is a sequence of text where the sequence is variable-length, to handle this type of inputs and outputs, we can design an architecture with two major components(two RNN). The first component is an encoder: it takes a variable-length sequence as the input and transforms it into a state with a fixed shape. The second component is a decoder: it maps the encoded state of a fixed shape to a variable-length sequence. This is called an encoder-decoder architecture, which is depicted in 1.18

strating from google in 2014 Ilya Sutskever et al.[21] introduce the term of a sequence to sequence model that aims to map a fixed-length input with a fixed-length output where the length of the input and output may differ.1.18 the model is containing three main parts as following

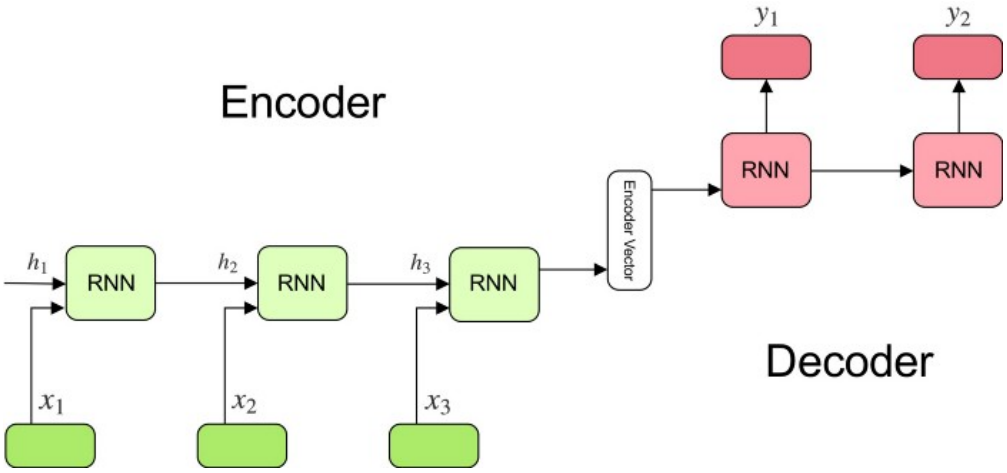


Figure 1.18: The encoder decoder architecture.

Encoder : A bunch of several recurrent units (LSTM1.6.5 or GRU1.6.5 cells for better performance) where each accepts a single element of the input sequence, collects information for that element and move it forward by calculating hidden state by formula $h_t = f(W^{hh}h_{t-1} + W^{hx}x_t)$

Encoder Vector : after using the formula above1.6.5 we produce a vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions.

Decoder : A stack of several recurrent units where each predicts an output y_t at a time step t, each one of them accepts a hidden state from the previous unit and produces output as well as its own hidden by formula 1.6.5 state.$h_t = f(w^{hh}h_{t-1})$ then using a softmax on the hidden state we will get the output at Y_t at time step.

Applications of Encoder-Decoder are vast in field like machine Translation(English to French), image Captioning, the above explanation just covers the simplest sequence to sequence model and, thus, we cannot expect it to perform well on complex tasks. The reason is that using a single vector for encoding the whole input sequence is not capable of capturing the whole information, This is why multiple enhancements like attention mechanism are being introduced.

1.6.6 Natural Language Processing

Natural Language Processing is broadly defined as the automatic manipulation of natural language, like speech and text, by software. as we say in previous section1.6.5 about sequence it becomes vital to enable computers to understand our mother language to offer assistance or make decisions based on it . NLP studies interactions between computers and humans using natural languages.

to make computer understand our sentences we need to start by teaching them the word first, to do so the main idea is text representation. the start of this work begin with basic idea of one-hot encoding.[10]. For

[9] Machine translation refers to the automatic translation of a sequence from one language to another

[10] A sentence is represented as a matrix of shape (NxN), where N is the number of unique tokens in the sentence

example, in the figure 1.19 under, each word is represented as sparse vectors (mostly zeroes) except of one cell (could be one, or the number of occurrences of the word in the sentence), unfortunately this approach has two significant drawbacks: the huge memory capacity issues, (because of the sparse representation matrix).and lack of semantic understanding. It can't understand relationships between words (e.g. school and book).

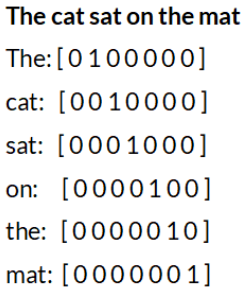


Figure 1.19: one-hot encodings for sentence "the cat set on the mat".

After that the DL1.3 algorithms helps the NLP problems by making word embedding[11] more effective in text representation.
in fact Word embedding is a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning. Key to the approach is the idea of using a dense distributed representation for each word.
The distributed representation is learned based on the usage of words. This allows words that are used in similar ways to result in having similar representations, naturally capturing their meaning. This can be contrasted with the crisp but fragile representation in a bag of words model where, unless explicitly managed, different words have different representations, regardless of how they are used. we can reviews three techniques that can be used to learn a word embedding from text data or a sequence .

Embedding Layer : An embedding layer, for lack of a better name, is a word embedding that is learned jointly with a neural network model on a specific natural language processing task, such as language modeling or document classification.
It requires that document text be cleaned and prepared such that each word is one-hot encoded. The size of the vector space is specified as part of the model, such as 50, 100, or 300 dimensions. The vectors are initialized with small random numbers. The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Backpropagation algorithm.
The one-hot encoded words are mapped to the word vectors. If a multilayer Perceptron model is used, then the word vectors are concatenated before being fed as input to the model. If a recurrent neural network is used, then each word may be taken as one input in a sequence.
This approach of learning an embedding layer requires a lot of training data and can be slow, but will learn an embedding both targeted to the specific text data and the NLP task.[22]

Word2Vec Word2Vec is a statistical method for efficiently learning a standalone word embedding from a text corpus. It was developed by Tomas Mikolov, et al. at Google in 2013 as a response to make the neural-network-based training of the embedding more efficient and since then has become the de facto standard for developing pre-trained word embedding.
Additionally, the work involved analysis of the learned vectors and the exploration of vector math on the representations of words. For example, that subtracting the "man-ness" from "King" and adding "women-ness" results in the word "Queen", capturing the analogy "king is to queen as man is to woman".[23]
Two different learning models were introduced that can be used as part of the word2vec approach to learn the word embedding; they are:

[11] A word embedding is a learned representation for text where words that have the same meaning have a similar representation.

- Continuous Bag-of-Words, or CBOW model.
- Continuous Skip-Gram Model.

The CBOW model learns the embedding by predicting the current word based on its context. The continuous skip-gram model learns by predicting the surrounding words given a current word. The continuous skip-gram model learns by predicting the surrounding words given a current word.

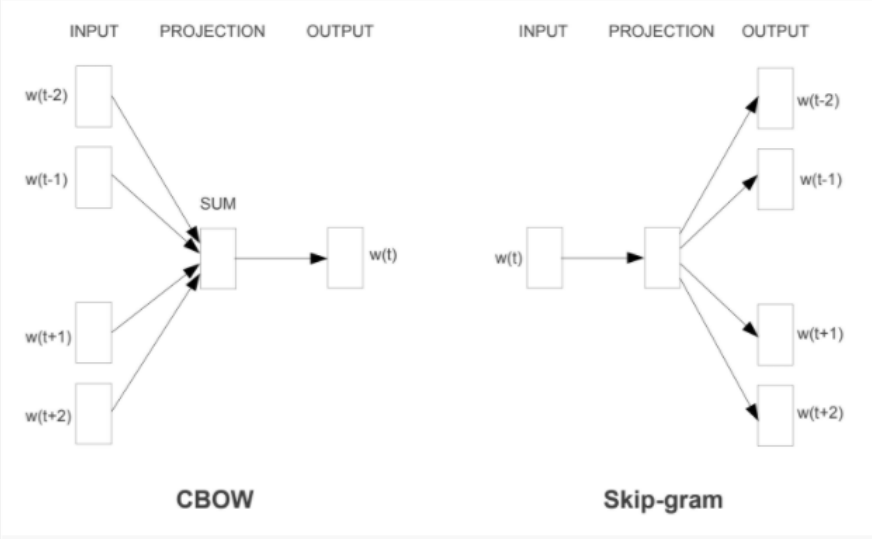


Figure 1.20: Word2Vec Training Models Taken from Efficient Estimation of Word Representations in Vector Space, 2013

Both models are focused on learning about words given their local usage context, where the context is defined by a window of neighboring words. This window is a configurable parameter of the model. The key benefit of the approach is that high-quality word embeddings can be learned efficiently (low space and time complexity), allowing larger embeddings to be learned (more dimensions) from much larger corpora of text (billions of words).[22]
Word vectors are vectors used to represent words, and can also be considered as feature vectors or representations of words. The technique of mapping words to real vectors is called word embedding. The word2vec tool contains both the skip-gram and continuous bag of words models.
The skip-gram model assumes that a word can be used to generate its surrounding words in a text sequence; while the continuous bag of words model assumes that a center word is generated based on its surrounding context words.

Tf-idf Tf-idf [12]stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.
One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model.
Tf-idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification.
the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that

[12] http://tfidf.com/

document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.
TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:
TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:
$IDF(t) = log_e(Totalnumberofdocuments/Numberofdocumentswithtermtinit)$.

Global Vector The Global Vectors for Word Representation, or GloVe, algorithm is an extension to the word2vec method for efficiently learning word vectors, developed by Pennington, et al. at Stanford. Classical vector space model representations of words were developed using matrix factorization techniques such as Latent Semantic Analysis (LSA) that do a good job of using global text statistics but are not as good as the learned methods like word2vec at capturing meaning and demonstrating it on tasks like calculating analogies (e.g. the King and Queen example above). GloVe is an approach to marry both the global statistics of matrix factorization techniques like LSA with the local context-based learning in word2vec.and GloVe, is a new global log-bilinear regression model for the unsupervised learning of word representations that outperforms other models on word analogy, word similarity, and named entity recognition tasks. Rather than using a window to define local context, GloVe constructs an explicit word-context or word co-occurrence matrix using statistics across the whole text corpus. The result is a learning model that may result in generally better word embeddings.[24]. The center word vector and the context word vector are mathematically equivalent for any word in GloVe.

GloVe can be interpreted from the ratio of word-word co-occurrence probabilities.

1.7 Attention

Once again in our goal to mimic human brain one of algorithm in a their way to perfecting seq2seq model is attention mechanism which is attempt to implement the same action of selectively concentrating on a few relevant things, while ignoring others in deep neural networks

1.7.1 Attention model

the start of this work begin with Bahdanau et al. in their paper [25] they came up with a simple but elegant idea where they suggested that not only can all the input words be taken into account in the context vector, but relative importance should also be given to each one of them. So, whenever the proposed model generates a sentence, it searches for a set of positions in the encoder hidden states where the most relevant information is available. This idea is called 'Attention'.

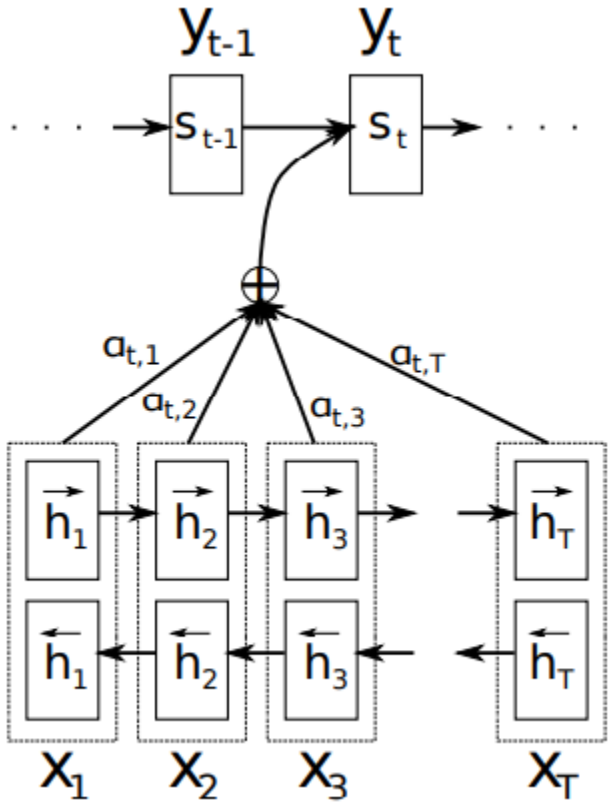


Figure 1.21: The graphical illustration of the proposed model of Bahdanau et al.

In the figures 1.21, This is the diagram of the Attention model shown in Bahdanau's paper. The Bidirectional LSTM [13] used here generates a sequence of annotations (h1, h2,....., hTx) for each input sentence. $h_j = [\overrightarrow{h_j^T}; \overleftarrow{h_j^T}]$ All the vectors h1,h2.., etc., used in their work are basically the concatenation of forward and backward hidden states in the encoder which means all the words are taking in consideration not just the last one as in encoder decoder model They did this by simply taking a weighted sum of the hidden states.the weights are also learned by a feed-forward neural network and by formula1.17 .

$$c_i = \sum_{j=0}^{T_x} \alpha_{ij} h_j \tag{1.17}$$

then $\alpha_{ij}$is computed by softmax 1.4 using

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_k^{T_x} \exp e_{ik}} . e_{ij} = a(S_{i-1}, h_j) \tag{1.18}$$

e_{ij} is the output score of a feedforward neural network described by the function a that attempts to capture the alignment between input at j and output at i. Basically, if the encoder produces Tx number of "annotations" (the hidden state vectors) each having dimension d, then the input dimension of the feedforward network is (t_x , 2d) (assuming the previous state of the decoder also has d dimensions and these two vectors are concatenated). This input is multiplied with a matrix w_d of (2d, 1) dimensions (of course followed by addition of the bias term) to get scores $e_i j$ (having a dimension (Tx , 1)). On the top of these e_{ij} scores, a

[13] It is RNN has duplicating the first recurrent layer in the network so that there are now two layers side-by-side, then providing the input sequence as-is as input to the first layer and providing a reversed copy of the input sequence to the second

tan hyperbolic function is applied followed by a softmax to get the normalized alignment scores for output j:
$E = I[T_x * 2d] * W_a[2d * 1] + B[T_x * 1]$.
$\alpha = softmax(tanh(E))$.
$C = IT * \alpha$.
So, α is a $(T_x, 1)$ dimensional vector and its elements are the weights corresponding to each word in the input sentence. Let α is [0.2, 0.3, 0.3, 0.2] and the input sentence is "this is a try". Here, the context vector corresponding to it will be: C=0.2*this"this" + 0.3*this"is" + 0.3*this"a" + + 0.3*this"try" [Ix is the hidden state corresponding to the word x].

Soft and hard attention later in 2015 Luong et al. [26] called this "global attention" with some modification in their model because all the inputs are given importance, when applying the "global" attention a lot of computation is happen. This is because all the hidden states must be taken into consideration, concatenated into a matrix, and multiplied with a weight matrix of correct dimensions to get the final layer of the feedforward connection.

As a solution to this problem and inspiring from computer vision problem Kelvin et al.[27] bring the new terms of "Soft attention"1.7.1 and "Hard attention"1.7.1 [14][15] into the field, as we can see Soft Attention is the global Attention but that not mean that hard attention is "Local attention", it is a blend of both the concepts, where instead of considering all the encoded inputs, only a part is considered for the context vector generation. This not only avoids expensive computation incurred in soft Attention but is also easier to train than hard Attention. in the figure 1.22 we can see taht global attention considers all hidden states (blue) whereas local attention considers only a subset:

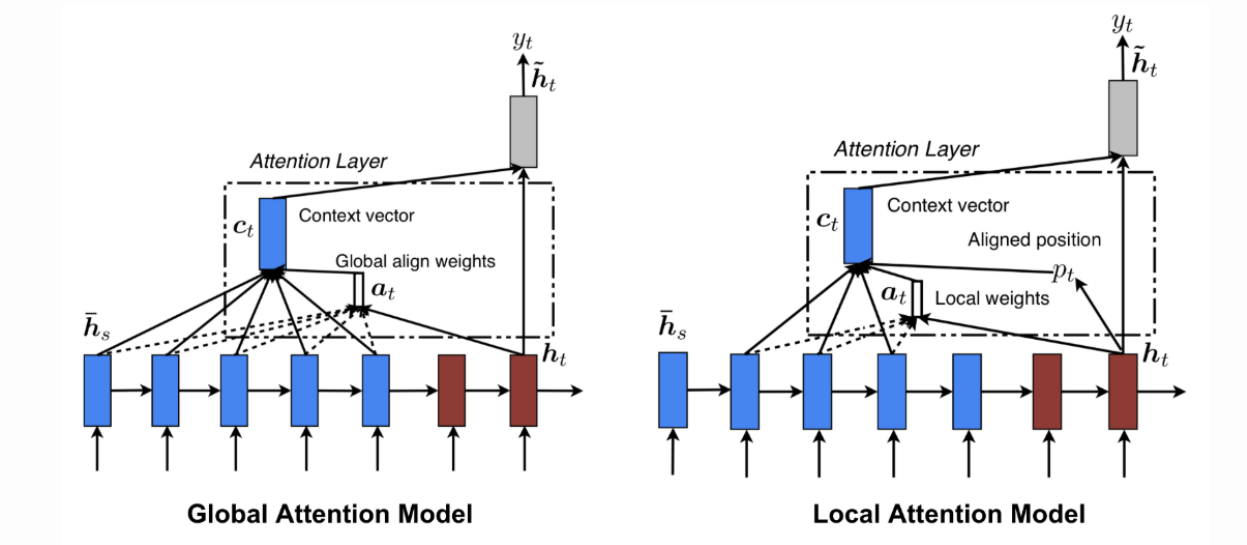


Figure 1.22: the difference between the Global and Local Attention mechanism

the model tries to predict a position p_t = in the sequence of the embedding of the input words. Around the position p_t, it considers a window of size, say, 2D. Therefore, the context vector is generated as a weighted average of the inputs in a position $[p_t D, p_t + D]$ where D is empirically chosen. Furthermore, there can be two types of alignments:

Monotonic alignment : where pt is set to t, assuming that at time t, only the information in the neighborhood of t matters

Predictive alignment where the model itself predicts the alignment position as follows : $p_t = S \cdot Sigmoid(v_p^t \tanh(W_p h_t))$

[14] Soft Attention is where all image patches are given some weight
[15] hard Attention, only one image patch is considered at a time.

in 2016 Cheng et al.present their paper [28] and make a great jump in attention mechanism from their paper they make whats call a "Self attention" which is the mechanism of relating different positions of a single sequence or sentence in order to gain a more vivid representation

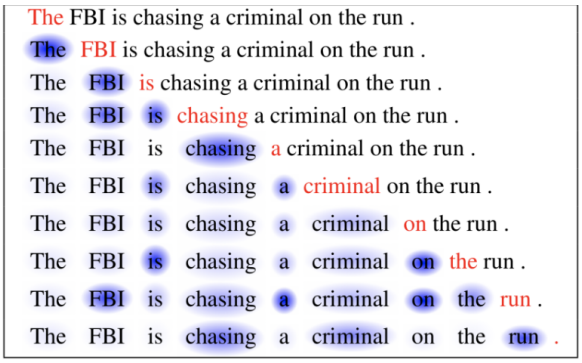


Figure 1.23: Illustration of Cheng et al. model

as shown in figure 1.23 Color red represents the current word being fixated, blue represents memories. Shading indicates the degree of memory activation, when processing the sentence word by word, where previously seen words are also emphasized on, is inferred from the shades, and this is exactly what self-Attention in a machine reader[16] does. the mechanism of score calculation is that each embedding of the word should have three different vectors corresponding to it, namely Key, Query, and Value, These vectors are trained and updated during the training process. Next, we will calculate self-attention for every word in the input sequence to do so using the vector query with dot product of keys vectors then dividing the score on the dimension of the key vector, next these scores are normalized using the softmax activation function, These normalized scores are then multiplied by the values vectors and sum up the resultant vectors to arrive at the final vector z

$$attention(Q,K,V) = Softmax(\frac{QK^T}{\sqrt{d_k}})v \tag{1.19}$$

this may take a while to calculate but the results is remarkable and significant.

1.8 Transformer

As we see in the section 1.6.5 the Encoder-Decoder with attention model is great in sequence to sequence or in vector to sequence[17] model but the problem of the this model is slow to train rather then the input of data must be sequentially because the current state depends on the previous state as a solution to this Vaswani et al. in 2017 they present in their paper [29] the brilliant term of transformer.

As shown in figure 1.24 the transformer is solely based on attention mechanisms in generally ("self attention"), similar to encoder decoder architecture transformer can input sequence can be passed in parallel.

the mechanism is very simple, starting with input embedding as we say in NLP section 1.6.6 computer do not understand words they get numbers, vectors and matrices.

the idea is to map every word to a point in space where similar words in meaning are physically closer to each other the space in which they are present is called an embedding space we could pretraine this embedding space, or even just use an already pretrained embedding space like GLOVE 1.6.6 this embedding space Maps a word to a vector but the same word in different sentences may have different meanings this is where

[16] Machine reader is an algorithm that can automatically understand the text given to it
[17] input is a vector and output is sequence

positional encoders come in it's a vector that has information on distances between words and the sentence the original paper uses a sine $PE_{pos,2i} = \sin(pos/1000^{2i/d_model})$ and cosine $PE_{pos,2i+1} = \cos(pos/1000^{2i/d_model})$ labeleq:pos function to generate this vector, the next step is to put the vector into the Encoder1.24

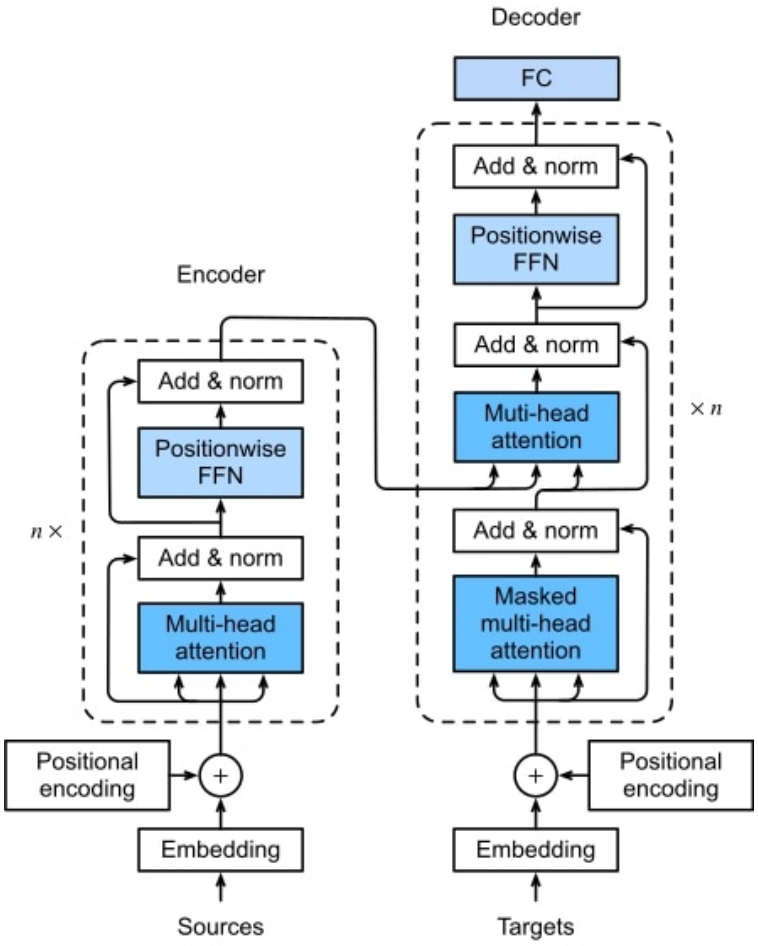


Figure 1.24: The transformer architecture.

Self attention

Multi-headed Attention : An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key"[29].as shown in figure 1.25 this is the same as in self attention 1.7.1 the only difference that the self attention is applied not once but multiple times in the Transformer's architecture, in parallel and independently. It is therefore referred to as Multi head Attention after that the attention vectors are passed to the normalization layer similar to batch normalization

Feed Forward Neural Network : after normalizing the vector we pass it to the FFNN, it's normal FNN where the vector become adjustable to Decoder part

Let's talk about the Decoder 1.24, the output sequence is also get the embedding space and add a positional Encoding to get the notion of context once again multi-headed Attention is applied to this vectors.

for now we have two vectors the Encoder and the Decoder both of them pass into a another attention block,next the vector is passed into FFNN layer to make it formally digestible to the linear layer applying a soft max to get the prediction of next sequence .

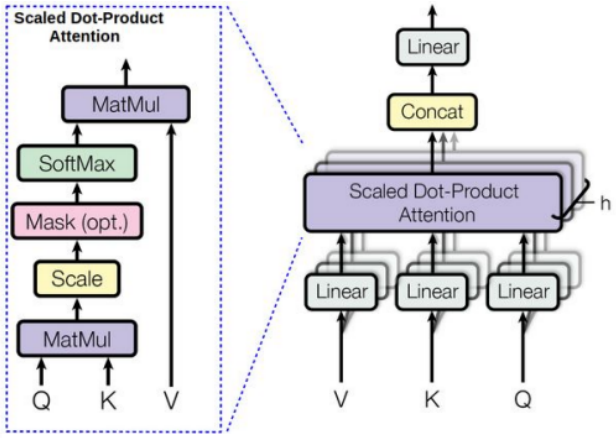


Figure 1.25: multi-headed Attention.

This approach bring a lot of potential especially in seq2seq model starting from the text, image, frames and a lot and because long-term dependencies are also important to improve Computer Vision tasks,
But it comes with its own share of limitations:

- Attention can only deal with fixed-length text strings. The text has to be split into a certain number of segments or chunks before being fed into the system as input.
- This problem causes context fragmentation. For example, if a sentence is split from the middle, then a significant amount of context is lost. In other words, the text is split without respecting the sentence or any other semantic boundary

to deal with this problem folks who worked with Transformer came up with Transformer-XL.

1.8.1 Transformer models

Transformer-XL in the paper [30] the model was very simple and amazing, the hidden states obtained in previous segments are reused as a source of information for the current segment. It enables modeling longer-term dependency as the information can flow from one segment to the next.
During the training phase in Transformer-XL, the hidden state computed for the previous state is used as an additional context for the current segment. This recurrence mechanism of Transformer-XL takes care of the limitations of using a fixed-length context.

Googles BERT Bidirectional Encoder Representations from Transformer(BERT) From Google AI The BERT framework, a new language representation model, uses pre-training and fine-tuning(trasfer learning) to create SOTA (state of the art) models for a wide range of tasks. These tasks include question answering systems, sentiment analysis, and language inference

BERT uses a multi-layer bidirectional Transformer encoder. Its self-attention layer performs self-attention in both directions. Google has released two variants of the model

- BERT Base: Number of Transformers layers = 12, Total Parameters = 110M. BERT Large: Number of Transformers layers = 24, Total Parameters = 340M.

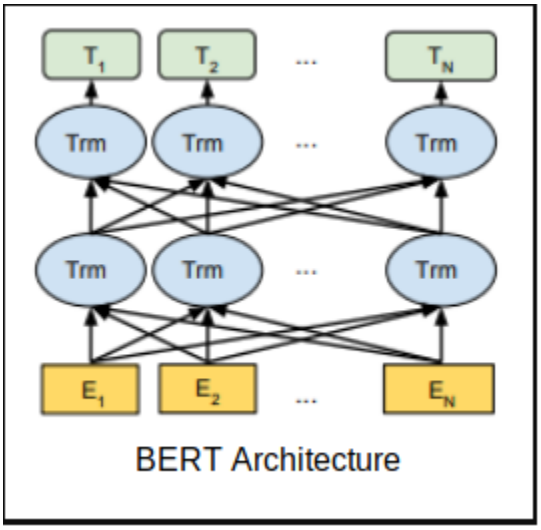


Figure 1.26: overview of BERT

1.9 Transfer learning

Humans have an inherent ability to transfer knowledge across tasks, we don't learn everything from scratch when we attempt to learn new aspects or topics we simply have basics and We transfer and leverage our knowledge from what we have learnt in the past.
Conventional machine learning and deep learning algorithms, so far, have been traditionally designed to work in isolation. These algorithms are trained to solve specific tasks. The models have to be rebuilt from scratch once the feature-space distribution changes. Transfer learning is the idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones
We cloud benefit transfer learning on any of the Modern Convolutional Neural Networks or the RNN model or any type of ANN

1.10 Evaluation Metrics

From modern problem of generating sequence from vector in ou case (image captioning)is the evaluation metrics or how to say that our model is better then others even humans could be precise of generating a new sequences. The evaluation of this models is generally performed using metrics such as BLEU1.10.1, METEOR, ROUGE or CIDEr,SPICE, all of which mainly measure the word overlap between generated and reference captions. The recently proposed SPICE measures the similarity of scene graphs constructed from the candidate and reference sentence, and shows better correlation with human judgments.

Metrics based on measuring word overlap between candidate and reference captions find it difficult to capture semantic meaning of a sentence, therefore often lead to bad correlation with human judgments. Secondly,each evaluation metric has its well-known blind spot, and rule-based metrics are often inflexible to be responsive to new pathological cases.[31]

As a start we need to make some concept like n-gram,precision to clarify,starting with the meaning of n-gram let's say that we have the sentence "karim is trying to make image captioning" see the table under 1.1

Table 1.1: unigram, bigram, and trigram for the sentence, karim is trying to make image captioning .

Unigram -	**Bigram** -	**Trigram** -
karim	karim is	karim is trying
is	is trying	is trying to
trying	trying to	trying to make
to	to make	to make image
make	make image	make image captioning
image	image captioning	-
captioning	-	-

that make the n-gram is a sequence of words occurring within a given window where n represents the window size.next the precision we can define the precision as follow 1.20

$$precision = \frac{\text{No. the word that predict and exist in reference}}{\text{t}otalNo.of word} \tag{1.20}$$

Next we 'll look to something call " n-gram modified" and it work like that

starting with Count clip Counting the maximum number of times a candidate n-gram occurs in any single reference translation; this is referred to as Count, For each reference sentence, counting the number of times a candidate n-gram occurs, Taking the maximum number of n-grams occurrences in any reference count, Taking the minimum of the Count and Max Ref Count. Also known as Count clip as it clips the total count of each candidate word by its maximum reference count finally the formula 1.21

$$P_n = \frac{C \in \sum_{Candidates} \sum_{ngram \in C} Count_{clip}(n-gram)}{\sum_{C \in Candidates} \sum_{n-gram} Cont(n-gram')} \tag{1.21}$$

,now we are ready to start with Evaluation matrix

1.10.1 Bilingual evaluation understudy

The BLEU method (Papineni et al., 2001) [32] was proposed as a rapid way for evaluating and ranking Machine Translation systems. Its robustness stems from the fact that it works with several reference texts (human-made translations), against which it compares the candidate text. The procedure is the following:

1. Count how many N-grams from the candidate text appear in any of the reference text (for different values of N). The frequency of each N-gram is clipped with the maximum frequency with which it appears in any reference.

2. Combine the marks obtained for each value of N, as a weighted linear average.

3. Apply a brevity factor to penalise the short candidate texts (which may have many N-grams in common with the references, but may be incomplete).

The use of several references, made by different human translators, increases the probability that the particular words and their relative order, in the automatic translation, will appear in any reference. On the other hand, the procedure is very sensitive to the choice of the reference translations [**?**]. we details more of this terms in section with example 1.1. A BLEU score might not be the perfect judge of Machine Translation quality. Yet it does have several advantages over other metrics and methods:Fast and affordable, Easy to understand, Independent of language, Results are close to human evaluation, It is in common usage.

This means that, once you understand BLEU scores, you can use them and know you are getting at least some kind of accurate understanding of the quality a Machine Translation engine can produce.

+

1.10.2 Recall-Oriented Understudy for Gisting Evaluation

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [33] is a set of metrics that are used for measuring the quality of text summary. It compares word sequences, word pairs, and n-grams1.10 with a set of reference summaries created by humans. Different types of ROUGE such as ROUGE-1, 2, ROUGE-W, ROUGE-SU4 are used for different tasks. For example, ROUGE-1 and ROUGE-W are appropriate for single document evaluation whereas ROUGE-2 and ROUGESU4 have good performance in short summaries. However, ROUGE has problems in evaluating multi-document text summary.

1.10.3 Metric for Evaluation of Translation with Explicit Ordering

METEOR (Metric for Evaluation of Translation with Explicit ORdering) [34] is another metric used to evaluate the machine translated language. Standard word segments are compared with the reference texts. In addition to this, stems of a sentence and synonyms of words are also considered for matching. METEOR can make better correlation at the sentence or the segment level.

1.11 Optimizers

To train any model of ANN 1.6 that mean to reduce the loss function and the model be more accurate. to do so we need to change the weight, learning rate in the model constantly this is called "Optimization"

The most commonly used algorithm for training a network is Gradient Decent (GD)

1.11.1 Gradient Decent

It is a first-order optimization algorithm. This means it only takes into account the first derivative when performing the updates on the parameters,it calculates that which way the weights should be altered so that the function can reach a minimal. using the formula $\theta = \theta\eta.\bigtriangledown J(\theta)$ " note that the α shouldn't be too small or too big"

Through back propagation, the loss is transferred from one layer to another and the model's parameters also known as weights are modified depending on the losses so that the loss can be minimized the GB is easy to implement ,calculate and understandable its may trap the local minimal and requires large memory to calculate for the whole data-set

There are different variations for GD, we will try explained in the next few subsections.

1.11.2 Stochastic Gradient Descent

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i\left(x_{i,} y_i, W\right) + \lambda R\left(W\right). \tag{1.22}$$

$$\bigtriangledown_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \bigtriangledown_W L_i\left(x_{i,} y_i, W\right) + \lambda \bigtriangledown_W R\left(W\right). \tag{1.23}$$

Full sum expensive when N is large! for example, we're using the image net dataset N could be like 1.3 million .so actually computing this loss could be very expensive and require computing perhaps Millions of evaluations of this function so that could be slow. such as this problem we have the Stochastic Gradient Descent.

Instead of calculating the gradients for all of your training examples on every pass of gradient descent, it's sometimes more efficient to only use a subset of the training examples each time. Stochastic Gradient Descent is an implementation that either uses batches of examples at a time or random examples on each pass.

What's called Stochastic Gradient Descent1.11.2where rather than computing the Loss 1.12 and Gradient 1.11.1.Over the entire training set instead at every iteration sample some small set and actually because the Gradient is linear operator .

Stochastic Gradient Descent because you can view this Monte Carlo estimate of some expectation of the true value. [18] [19]

1.11.3 Adagrad

Adagrad adapts the learning rate specifically to individual features; that means that some of the weights in your dataset will have different learning rates than others. This works really well for sparse datasets where a lot of input examples are missing. Adagrad has a major issue though: The adaptive learning rate tends to get really small over time. Some other optimizers seek to eliminate this problem.

1.11.4 RMSprop

RMSprop is a special version of Adagrad developed by Professor Geoffrey Hinton in his neural nets class. Instead of letting all of the gradients accumulate for momentum, it only accumulates gradients in a fixed window. RMSprop is similar to Adaprop, which is another optimizer that seeks to solve some of the issues that Adagrad 1.11.3 leaves open.

1.11.5 Adam

Adaptive Moment Estimation (Adam)[35] is a method of gradient optimization. It keeps a decaying average of past squared gradients v_t just like RMS Prop And AdaDelta, but also another decaying average of past gradients m_t that plays the role of a Momentum[20]. m_t and v_t are given by equations 1.24 and 1.25 respectively:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{1.24}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{1.25}$$

Because the authors observed that these averages were biased towards zero, they applied bias correction, so they proposed $\hat{m}_t$ and $\hat{v}_t$ which are calculated using equations 1.26 and 1.27 respectively:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{1.26}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{1.27}$$

The update rule is slightly different then AdaDelta and RMSProp and is given by equation 1.28:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \tag{1.28}$$

The authors recommend a β_1 of 0.9, a β_2 of 0.999 and an ϵ of 10^{-8}, because they show empirical improvements in practice and better performance in comparison with to other algorithms[37].

[18] fhttp://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf

[19] feifeili@stanford.edu

[20] Momentum is also a method that helps accelerate SGD in the direction of the best minimum. For detailed explanation refer to the original paper [36]

1.12 Loss Functions

Neural networks are trained using stochastic gradient descent and require that you choose a loss function when designing and configuring your model. .After training our model wee need"Loss Function" a loss function is incredibly simple: It's a method of evaluating how well our algorithm models doing on data-set. If our predictions are totally off, then loss function will output a higher number. If they're pretty good, it'll output a lower number. in next subsection we will discuss few loss function

1.12.1 Cross Entropy

Entropy is the measure of uncertainty associated with a given probability distribution. For example, supposing that X is the random variable that describes the class of a randomly picked image from a dataset. If all dataset images are of the same class then the entropy of X is 0, because there is no uncertainty about the class of any randomly picked image.

The Cross Entropy (CE) is a measure for estimating the difference between two probability distributions over a random variable. Usually, the first distribution is a target denoted as P, and the second is an approximation of the target distribution, and it is denoted Q. CE can be used as a loss function for ANNs, considering the label of an image, a target distribution P, and the model's prediction for that image, an approximated distribution Q. Given a random image from a dataset, the CE between a prediction vector Q and a label vector P is given by the formula 1.29:

$$H(P,Q) = -\sum_{i=1}^{C} P(i)\log(Q(i)) \tag{1.29}$$

1.12.2 Log-Likelihood

The softmax function [38]gives us a vector y' , which can interpret as estimated conditional probabilities of each class given any input x , e.g., $y^1 = P(y = catx)$. Suppose that the entire dataset X,Y has n examples, where the example indexed by i consists of a feature vector x^i and a one-hot label vector y^i . can compare the estimates with reality by checking how probable the actual classes are according to our model, given the features:

$$P(Y|X) = \prod_{i=1}^{n} P\left(y^i|x^i\right). \tag{1.30}$$

According to maximum likelihood estimation, we maximize P(Y X) , which is equivalent to minimizing the negative log-likelihood:

$$-logP(Y|X) = \sum_{i=1}^{n} -logP\left(y^i|x^i\right) = \sum_{i=1}^{n} l\left(y^i|y'^i\right). \tag{1.31}$$

where for any pair of label y and model prediction y' over q classes, the loss function l is:

$$l\left(y^i|y'^i\right) = -\sum_{j=1}^{q} y_j \log y'_j. \tag{1.32}$$

1.12.3 Mean squared error

Mean squared error (MSE) is the workhorse of basic loss functions; it's easy to understand and implement and generally works pretty well. To calculate MSE, you take the difference between your predictions and the ground truth, square it, and average it out across the whole dataset.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - _i)^2. \tag{1.33}$$

21

[21] https://algorithmia.com/blog/introduction-to-loss-functions

1.13 DATA-SETS

A number of data-sets are used for training,testing,and evaluation of the image captioning methods.The data-sets differ in various perspective such as the number of images,the number of captions per image,format of the captions,and image size. Three data-sets:Flicker30k,and MS COCO Data-set are popularly used.These data-sets together with others are described in Section 1.13. A number of evaluation metrics are used to measure the quality of the generated captions compared to the ground-truth.Each metric applies it's own technique for computation and has distinct advantages. [39] The commonly used evaluation metrics are discussed in Section 1.10.1

Microsoft Common Objects in Context Dataset The MS COCO (Microsoft Common Objects in Context) [22] dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images. Splits: The first version of MS COCO dataset was released in 2014. It contains 164K images split into training (83K), validation (41K) and test (41K) sets. In 2015 additional test set of 81K images was released, including all the previous test images and 40K new images.[40] Based on community feedback, in 2017 the training/validation split was changed from 83K/41K to 118K/5K. The new split uses the same images and annotations. The 2017 test set is a subset of 41K images of the 2015 test set. Additionally, the 2017 release contains a new unannotated dataset of 123K images.

The dataset has annotations for :

- object detection: bounding boxes and per-instance segmentation masks with 80 object categories,
- captioning: natural language descriptions of the images (see MS COCO Captions),
- key points detection: containing more than 200,000 images and 250,000 person instances labeled with keypoints (17 possible key points, such as left eye, nose, right hip, right ankle),
- stuff image segmentation – per-pixel segmentation masks with 91 stuff categories, such as grass, wall, sky (see MS COCO Stuff),
- panoptic:full scene segmentation, with 80 thing categories (such as person, bicycle, elephant) and a subset of 91 stuff categories (grass, sky, road),
- dense pose: more than 39,000 images and 56,000 person instances labeled with Dense Pose -annotations – each labeled person is annotated with an instance id and a mapping between image pixels that belong to that person body and a template 3D model. The annotations are publicly available only for training and validation images.

Flickr30K Dataset

Flickr30K [41] Is a dataset for automatic image description and grounded language understanding. It contains 30k images collected from Flickr with 158k captions provided by human annotators. It does not provide any fixed split of images for training, testing, and validation. Researchers can choose their own choice of numbers for training, testing, and validation. The dataset also contains detectors for common objects, a color classifier, and a bias towards selecting larger objects. Image captioning methods use this dataset for their experiments. dataset.[23] -example of image from Flickr30K dataset

[22] http://mscoco.org
[23] https://paperswithcode.com/dataset/flickr30k

the white and brown dog is running over the surface of the snow .

a white and brown dog is running through a snow covered field .

a dog running through snow .

a dog is running in the snow

a brown and white dog is running through the snow .

Figure 1.27: example from Flicker data-set with caption

1.14 Conclusion

In this chapter mentioned above, we present some of the Machine learning used as a means of implementing AI solutions. and As a class of machine learning, we introduced the definition of machine learning, and our types and representational learning focuses on how to automatically find the appropriate way to represent data. Deep learning is multi-level representation learning through learning many layers of transformations. as discussed and explaining the definition of deep learning we including the basics of neural network architectures, and our methods of computer vision Finally, we set some data and our method of evaluation

In the next chapter we discuss the main problem of we subject image caption and See the background of them

CHAPTER 2

IMAGE CAPTIONING - SOME PREVIOUS WORKS

2.1 Introduction

When using an artificial neural network, the capabilities of machines have risen to levels as unseen before. The machine can now understand the context of a scene and filter out the noise to process only the most important information.

The ability to automatically describe the content of a document using properly formed English or any language sentences is a very challenging task, for instance by helping visually impaired people better understand the content of images on the web

In this chapter, we see some briefly describe generalities of Vision Language Task and type of image captioning.

We investigate and explore Study some main about stat of the art methods and approaches has an achievement by connects computer vision and natural language processing to Image Caption Generation and The most prominent data used for it with their methods of evaluations.

In this chapter we will experiment some DL based methods for image captioning.

2.2 Vision Language Task

Computer vision is an umbrella term for a set of techniques used to interpret image-based data by computers. While image processing has been possible for a long time, computers were unable to interpret those images in any way, under the big umbrella is vision language models and vision language tasks this tasks is :

Image-Sentence Retrieval :The goal is to retrieve relevant sentences given an image, or to retrieve relevant images given a sentence.

Phrase Grounding :In phrase grounding the task is to find the location of a phrase given an image it is known to exist in.

Text-to-Clip :For text-to-clip, the goal is to locate the temporal region (i.e. the video clip) that is described by a query

Visual Question Answering : the goal is to produce a free-form natural language answer given an image and question[42]. This open-ended task consists of three types of questions: yes/no, number and other

2.3 Image captioning

From the Vision language task image captioning it's a challenging artificial intelligence problem as it requires both techniques from computer vision to interpret the contents of the photograph and techniques from natural language processing to generate the textual description.

2.4 Definition of the problem

Captioning image involves generating human readable textual description given an image, such as a photograph, it is easy problem for a human, but very challenging for machine as it involves both understanding the content of image and how to translate this into natural language.

The literature on caption generation can be divided into three main groups,we ll provide some background on previous work in three but focusing on the last one

2.5 State of The Art

2.5.1 Template-based approaches

This approaches first detect objects, actions, scenes and attributes,then fill them in a fixed sentence template, e.g. using a subject-verb-object template. These methods are intuitive and can work with out of the box visual classification components.

However, they require explicit annotations for each class. Given the typically small number of categories available, these methods do not generate rich enough captions. Moreover, as they use rigid templates the generated sentence is less natural.

In 2010 A.Farhadi et al. [43] According to them the methods can pool sentence from image ,and described a system that can compute a score linking an image to a sentence.

This score can be used to attach a descriptive sentence to a given image, or to obtain images that illustrate a given sentence. The score is obtained by comparing an estimate of meaning obtained from the image to one obtained from the sentence. Each estimate of meaning comes from a discriminating procedure that is learned using data.

For this methods they work with a data-set that images and corresponding sentences and also labels for their representations of the meaning space.they retrieved PASCAL data-set 2008 2.5.1 [1][2][44]

[1] http://www.pascal-network.org

[2] The Pascal VOC 2008 data-set is created for visual object classifications and detection.

their model assumes that there is a space of Meanings that comes between the space of Sentences and the space of Images, to evaluate the results of their work miser the similarity between a sentence and image.

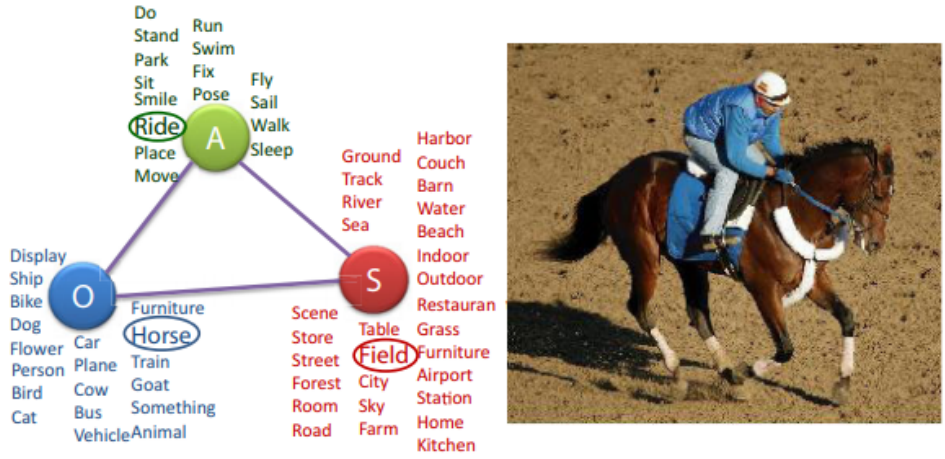


Figure 2.1: representation of the space of the meanings model of A.Farhadi et al.

in conclusion the sentences are not generated but searched from a pool of candidate sentences this wasn't efficient enough to make new caption for unseen problem

In 2011 another paper take the same road, Yang et al.[45] they proposed a sentence generation strategy that describes images by predicting the most likely nouns(names), verbs, scenes and prepositions in image that make up the core sentence structure.

the model detect objects and scenes using trained detection algorithms , to keep the math simple, they limit the elements of space into just 4 dimension (Nouns-Verbs-Scenes-Prepositions)

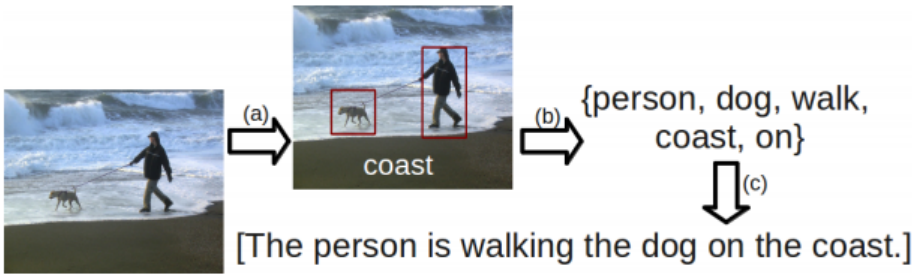


Figure 2.2: Overview Yang et al. model

they used a language model trained from the English Gigaword[3][4] corpus to obtain their estimates together with probabilities of co-located nouns,scenes and prepositions.

finally the experiment results show that the strategy of combining vision and language produces readable and descriptive sentences compared to naive strategies that use vision alone.[45]

In addition, the sentence that is generated for each image is limited to at most two objects occurring in a unique scene.

[3] https://catalog.ldc.upenn.edu/LDC2003T05

[4] English Gigaword was produced by Linguistic Data Consortium (LDC) is a comprehensive archive of newswire text data in English that has been acquired over several years

as a critique to this work the group says that their model strategy is fails to predict the appropriate verbs or nouns and this is due to object/scene detection's can be wrong and noise from the corpus.
Template-based approaches Critiques

Template-based methods can generate grammatically correct captions. However, templates are predefined and cannot generate variable-length captions moreover they cannot generate image specific and semantically correct captions

2.5.2 Retrieval-based approaches

In retrieval-based approaches, captions are retrieved from a collection, of existing captions. Retrieval based methods first find the visually similar images with their captions from the training data set. These captions are called candidate captions.The captions for the query image are selected from these captions pool

In 2011 Ordonez et al. [46] have described an effective caption generation method for general web images. This method relies on collecting and filtering a large data set of images from the internet to produce a novel webscale captioned photo collection. they present two variations on thier approach, one that uses only global image descriptors which is achieved this by computing the global similarity of a query image to large web-collection of captioned images, C. then find the closest matching image (or images) and simply transfer over the description from the matching image to the query image. to compose captions, and one that incorporates estimates of image content for caption generation.

2.5.3 Novel image caption generation

Most DL based image captioning methods fall into this category,the main idea is to generate captions from the visual content using a language model which mean to generate new caption for every image by it self also mean that the semantic of sentence is more accurate the the previous strategy (2.5.2) (2.5.1)

in figure 2.3 we can review taxonomy of image caption models.

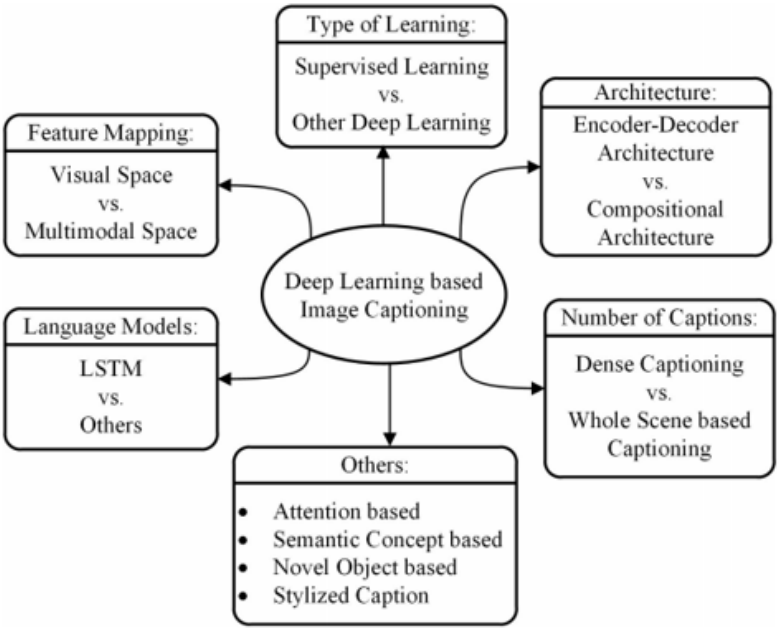


Figure 2.3: overview of Deep Learning model in image captioning

In supervised learning, training data come with desired output called label, the progress of (SL) make the researchers interested in using them in automatic image captioning.

Grouping the image captioning deep learning model is underling on the model used in the architect, we will discuss a few algorithms of supervised strategy and tring to talk about their state of the art (SOTA)

Multi-modal space based the main idea in this is three part contain Encoder for language,vision,multi-model space and Decoder for language. The process is very easy and famous the field starting with extracting the image features with CNN 1.6.2 model then process the language with Encoder 1.6.4 to learn dense feature both of the features forwards into the multi-model space to map the image features with the word features in a common space the hole process in shown in figure 2.4.

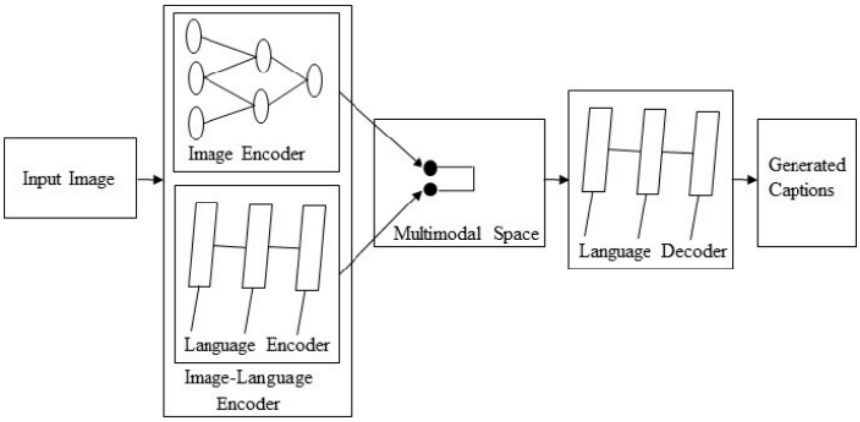


Figure 2.4: diagram of multi-modal space-based image captioning.

in the (SOTA) Kiros et al.[47] in 2014 the group apply a CNN feature extraction with a multimodal neural language models as Modality-Biased Log-Bilinear Model (MLBL-B) and the Factored 3-way [39] unlike the other approach this model doesn't relay on any template but relay on the image features and and word representations learned from deep neural networks and multimodal neural language models respectively Unfortunately the language models have limitations to handle a large amount of data and are inefficient to work with LSTM

In the table 2.1 under some of the work in multi-model space based note" (VS=Visual Space, MS=Multimodal Space, SL=Supervised Learning,DC=Dense Captioning, WS=Whole Scene, EDA=Encoder-Decoder Architecture, AB=AttentionBased).

Reference	Image Encoder	Language Model	Category
Karpathy et al. 2015 [48]	VGGNet	RNN	MS,SL,WS,EDA
Park et al. 2017 [49]	ResNet	LSTM	VS,SL,WS,EDA,AB

Table 2.1: An overview of the deep learning-based approaches for image captioning

Dense captioning and Whole scene :

In dense captioning, captions are generated for each region of the scene. This method use CNN to identify the region in image then pass the region features into language Decoder to generate caption [39].

In 2015 [50] proposed an image captioning method called DenseCap, which convolutional network, a dense localization layer, and an LSTM to localizing the dence they use a spatial soft attention mechanism [51, 52] and a and bilinear interpolation [51] This process help to back-propagation through the network and smoothly select the active regions.

In Whole scene their is multiple model Captions for the whole scene ex: **??**

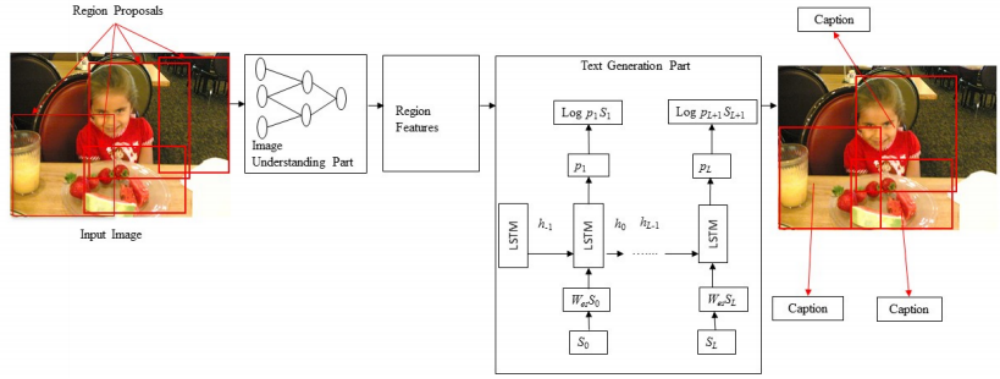


Figure 2.5: process of Dense caption

Compositional Architecture-based this is actually has multiple model and forms starting from CNN to extract features from image then through a language block to get candidate captions in the final step re-ranking the candidate to get a more accurate model see in figure 2.6

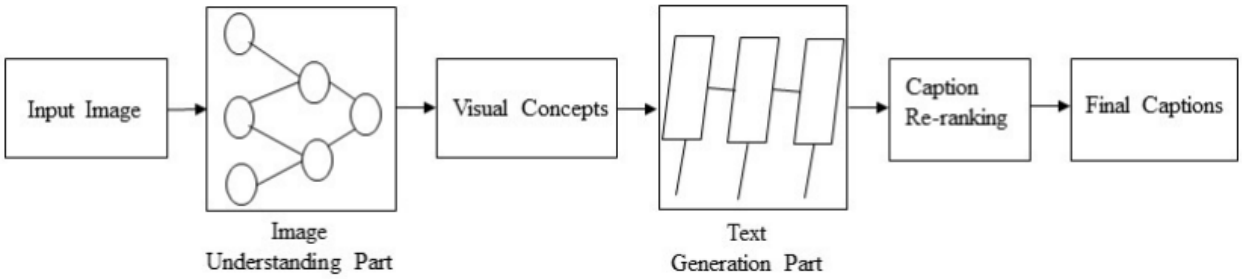


Figure 2.6: Compositional Architecture

In 2015 fang et al. [53] present the paper From Captions to Visual Concepts and Back, the caption generation pipeline see figure 2.7 detects a set of words that are likely to be part of the image's description. These words may belong to any part of speech, including nouns, verbs, and adjectives. they determine vocabulary V using the 1000 most common words in the training captions, next using a CNN model to get features from sub-regions next mapping the word along with features using MLP [54] (Multiple instance learning) passing this map to language model to generate image captions then calculate the rank by a linear weighting of sentence features then using MERT (Minimum Error rate training) is using to knew the Similarity between image and sentence

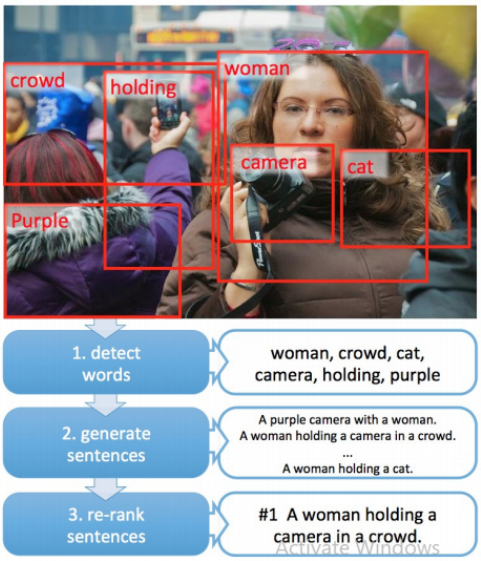


Figure 2.7: fang et al. pipeline model

It achieves a significant improvement in choosing high quality image captions.

Attention based inspiring by the success of attention in translation machine this was brilliant idea to bring this in image description, the way to this is to initially with a features extraction of visual image then pass it into language block then in each time step the model focus on the silent part of image to be more accurate in description

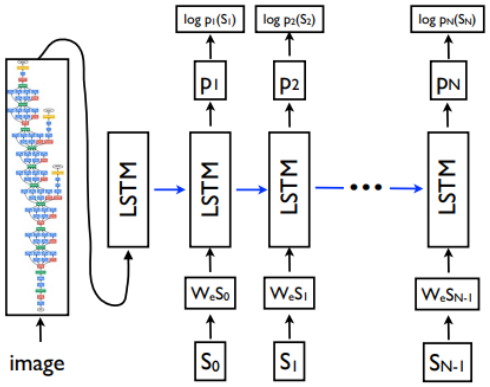


Figure 2.8: LSTM model combined with a CNN image embedding

in 2015 xu et al.[55] present show attend and tell, the first term of attention in image description the whole process in too simple starting with extracting features in the blind regions in image then using stochastic hard attention and deterministic soft attention see in the chapter 1 (1.7.1) (1.7.1) this attention bring details into caption generated.

While attention-based methods look to find the different areas of the image at the time of generating words or phrases for image captions, the attention maps generated by these methods cannot always correspond to the proper region of the image. It can affect the performance of image caption generation [39].

Semantic Concept-Based basically this process relaying on the good understanding of the image starting by encoding the image features along with semantic concepts then feed it into different part of hidden state of the language block to describe the image with semantic aspect as shown in figure 2.9.

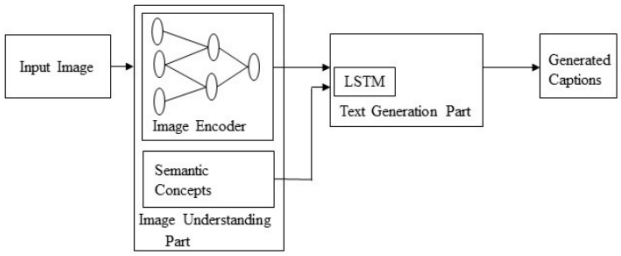


Figure 2.9: semantic model in image captaining

Karpathy et al. introduce the paper [48] This method employs a novel combination of CNN over the image regions, bidirectional RNN over sentences, and a common multi-modal embedding that associates the two modalities.as shown in figure 2.10

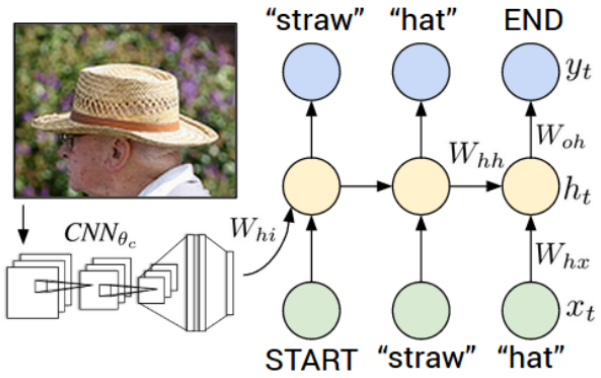


Figure 2.10: Karpathy et al.model

"an RNN considers the current word and the contexts from all the previously generated words for estimating a probability distribution of the next word in a sequence this method extends it for considering the generative process on the content of an input image. This addition is simple but it makes it very effective for generating novel image captions' [39]

Also in the same model of image capting their are multiple model we didn't discuss like Novel Object-based, Stylized Caption, LSTM models and unsupervised learning Genrative adversarial network caption

2.6 Conclusion

In this chapter, we introduce firstly small definitions of vision task model then we investigate the state of the art methods in different model architecture and approaches for Image Caption Generation

In the next chapter we'll try to experiment and implement some improvement in one of the deep learning-based models of image captioning

CHAPTER 3

IMPLEMENTATION

3.1 Introduction

In this chapter, we will try to implement a photo caption project on flicker1.13. We will discuss architecture and explain the programs, hardware, and results we have obtained in this experiment.

3.2 Architecture

Transformer-based architectures represent the state of the art in sequence modeling tasks like machine translation and language understanding. Their applicability to multi-modal contexts like image captioning, however, is still largely underexplored. With the aim of filling this gap we present the TRIC(transformer image captioning)

3.2.1 Image captaining based on transformer

.
through the process of making a TRIC, TRIC consist of Three main parts see figure 3.1

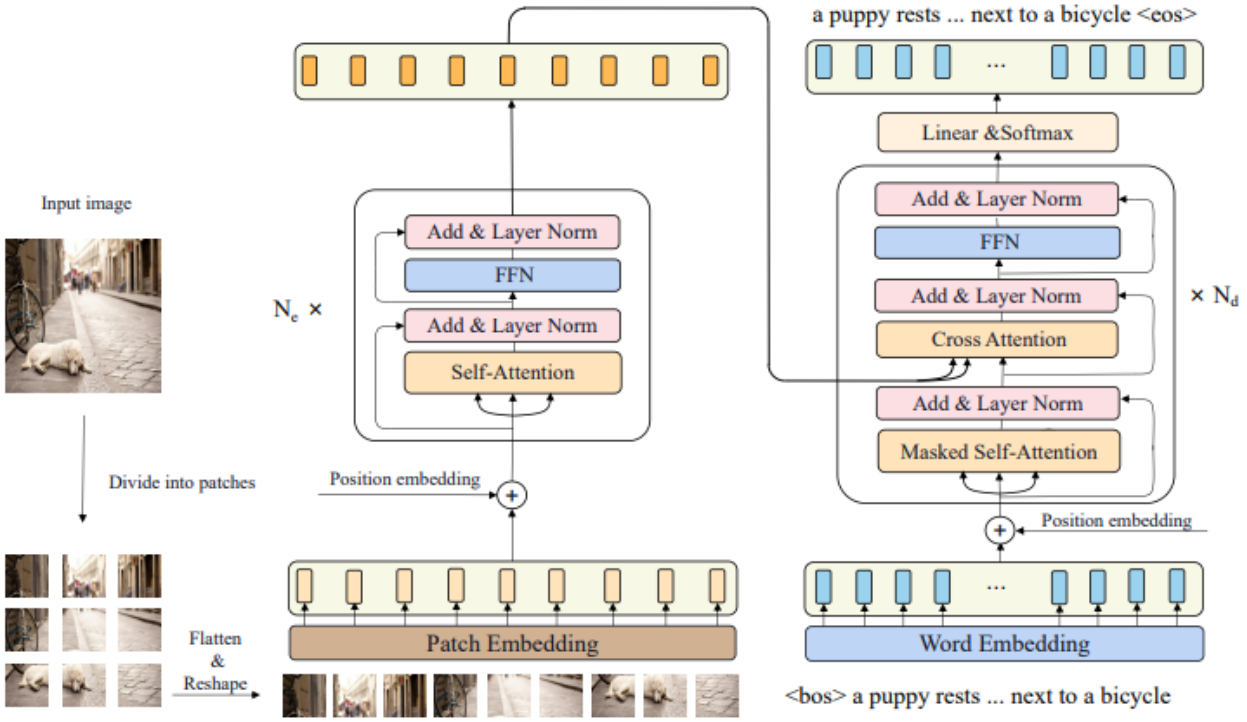


Figure 3.1: Overview of the whole model

Transformer Encoder Instead of using a pretrained CNN or Faster R-CNN model to extract spatial features or bottom-up features, we choose to sequentialize the input image and treat image captioning as a sequence-to-sequence prediction task , we resize the input image into a fixed resolution then divide the resized image into N patches, where $N = \frac{H}{P} \text{Œ} \frac{W}{P}$ and P is the patch size,then we flatten each patch and reshape them into a 1D patch sequence

The encoder of CPTR consists of Ne stacked identical layers, each of which consists of a multi-head self-attention mechanism 1.7, and the a simple, position-wise fully connected feed-forward network.

in this Encoder we use:

- One Embedding layer
- 4 layers, each layer contains:
  - 8 Multi-Head Attention block
  - one Normalization layer for Attention block
  - one Feed-Forward Network block
  - one Normalization layer for FFN block

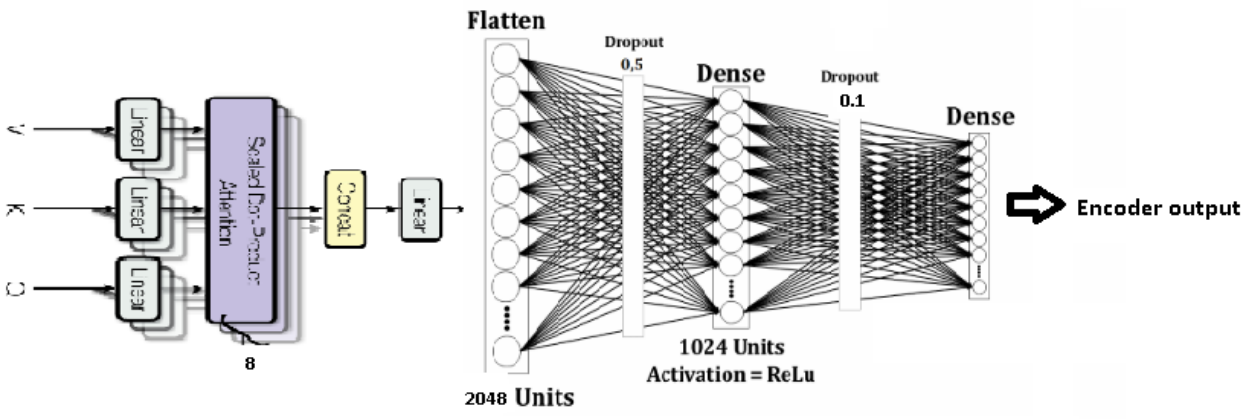


Figure 3.2: Overview of the Encoder part.

For every word, we can have an attention vector generated that captures contextual relationships between words in a sentence. Multi-headed attention in the encoder applies self-attention. the input of FNN is 2048 from the attention wight that applies the relu function on the positional encoder see figure 3.2

Transformer Decoder So, we have created the Encoder. We'll go ahead to tackle the Decoder, which is very similar to the Encoder, except that there are two Multi-Head Attention blocks in one layer, one for the target sequences and one for the Encoder's output also the bottom Multi-Head Attention is masked.

the table under shows the hyper parameter using in the original paper and our hyper parameter based on it **??**

Table 3.1: Summary hyper parameter in our architecture and compere to original paper of Wei Liu et al. [1]

hyper parameter	our **architecture**	**Wei Liu et al.[1]**
Number of layer	**3**	6
Number of head attention	**4**	8
Dimensionality of input and output	256	256
FFN inner-layer dimensionality	**1024**	2048
Number of parmter	**72126522**	83959866
Size of vocabulary	**10000**	30522

3.3 Software

3.3.1 Python

Python[1] is an interpreted, object-oriented, high-level programming language launched in December 1989 by Guido Van Rossum. Python is used in several domains like web development and creating software

[1] https://www.python.org

prototypes. But most importantly, the vast majority of AI and ML practitioners use it to implement their models due to its simplicity and consistency, plus the access to a great number of pre-implemented libraries and frameworks. These libraries make the coding part much easier for the developer. Granted, Python offers a platform independence which allows the developer to implement projects and run them on different platforms regardless of their type[2].

3.3.2 Pytorch

Pytorch[3]is an optimized tensor library for deep learning using GPUs and CPUs. open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR). It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.
A number of pieces of deep learning software are built on top of PyTorch, including Tesla Autopilot, Uber's Pyro, HuggingFace's Transformers, PyTorch Lightning, and Catalyst.

PyTorch provides two high-level features:

1. Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU)
2. Deep neural networks built on a type-based automatic differentiation system

3.3.3 Keras

Keras[4] is a high-level API, which represents a simple interface, that minimizes the number of user actions required for common use cases.
Keras is easy to use and focused on user experience. It also makes it easier to run experiments, as it can easily scale up to large clusters of GPUs or entire TPU pods. One of its advantages is having the low-level flexibility to implement arbitrary research ideas but at the same time, it offers optional high-level convenient features to speed up experiment deployment, meaning that all the tools that you might need are modules that can be called and fine tuned very easily with minimum amount of coding.

3.3.4 Google Colab

Colab [5] is a free notebook environment that runs entirely in the cloud. It lets you and your team members edit documents, the way you work with Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

Google is quite aggressive in AI research. Over many years, Google developed AI framework called TensorFlow and a development tool called Colaboratory. Today TensorFlow is open-sourced and since 2017, Google made Colaboratory free for public use. Colaboratory is now known as Google Colab or simply Colab .
Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reasons for making it free for public could be to make its software a standard in the academics for teaching machine learning and data science. It may also have a long term perspective of building a customer base for Google Cloud APIs which are sold per-use basis. Irrespective of the reasons, the introduction of Colab has eased the learning and development of machine learning applications [6]
If you have used Jupyter [7] notebook previously, you would quickly learn to use Google Colab. To be precise, Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members -

[2] https://medium.com/towards-artificial-intelligence/why-is-pythons-programming-language-ideal-for-ai-and-data-science-e
[3] https://pytorch.org/
[4] https://keras.io
[5] https://www.tutorialspoint.com/google_colab/google_colab_introduction.html
[6] https://colab.research.google.com/notebooks/intro.ipynb
[7] https://jupyter.org

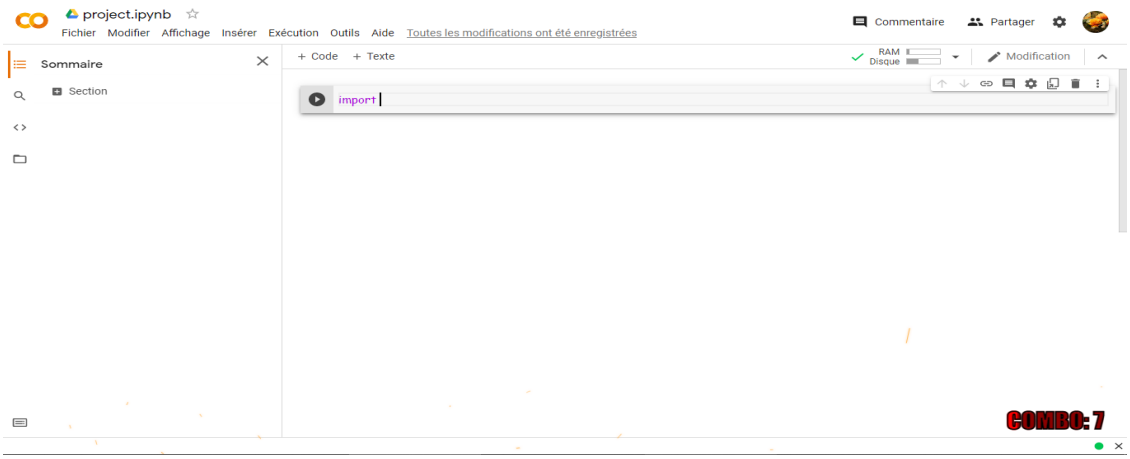


Figure 3.3: Interface of google colab

just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

3.4 Hardware

During the implementation process, we use Google Colab Because it provides us with frame work equipment for free to use. As follows:

Table 3.2: The hardware used for the implementation.

CPU	Intel(R) Xeon(R) CPU @ 2.00GHz
GPU	Tesla P100-PCIE-16GB
RAM	25.51 GB
Disk	68.40 GB

3.5 Training and Results

In general training Transformer-based models aren't cheap. To take advantage of the parallelization, one has to utilize the power of GPU computing and because of the policy of using google colab about their run-time we had to make multiple breaks in training(we solve this through checkpoints)

As a start we try to use 30000 images with 5 captions on each sample, the training for 20 epochs lead to this result in accuracy and loss function 3.73.8.

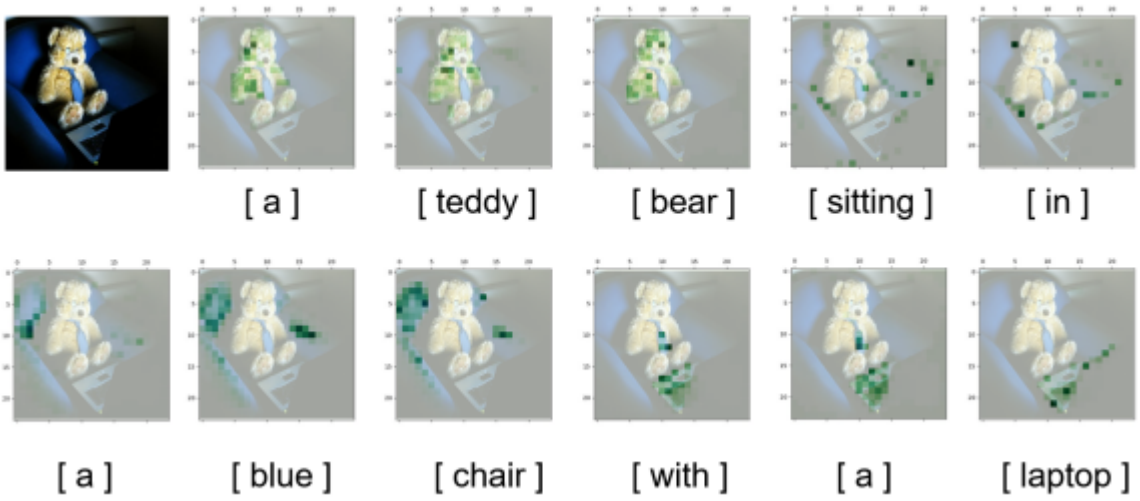


Figure 3.6: Visualization of the attention weights computed by the words-to-patches cross attention in the last decoder layer. A teddy bear sitting in a blue chair with a laptop is the caption generated by our model

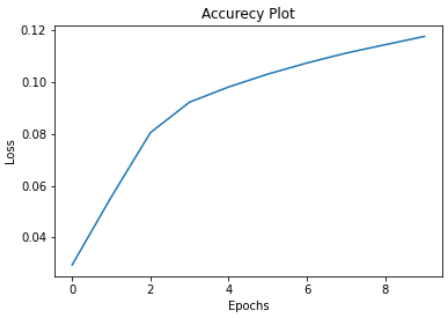


Figure 3.4: Accuracy Plot epoch 20, 30000 image

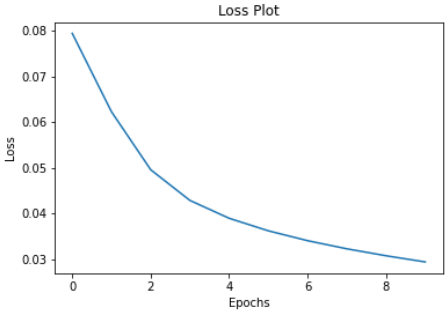


Figure 3.5: Loss plot epoch 20, 30000 image

the attention vector from encoder see figure 3.6

the problem we had is storage wasn't enough for the whole data-set even we bay google drive space, then we need to replace data-set into flicker, in the flicker 8 we get apply same hyper parameter 3.1.

We got this accuracy after 20 epoch on GPU and see the results in blue Rouge

results	score
BLEU-1	29.41
BLEU-2	54.23
BLEU-3	69.27
BLEU-4	73.64

Table 3.3: our model achievement in different blue scores

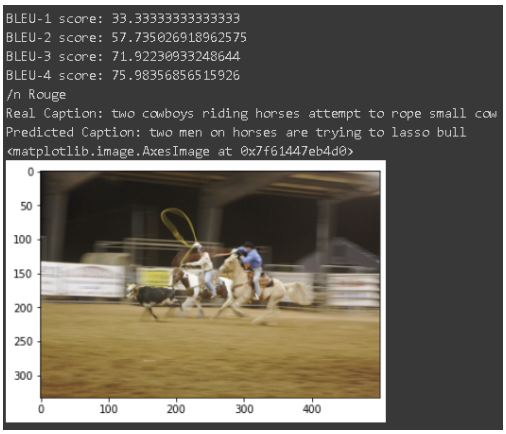


Figure 3.7: blue evaluation with image

Figure 3.8: blue evaluation with image

our model achieved:

3.6 Discussion of Results

let's say that this architecture is a brilliant idea and the transformation from NLP field into the image processing brings a lot of attention into the computer vision

According to results **??** we can say that transformer is better then the CNN+LSTM and CNN+Transformer and better then the visual attention, obviously the results was for 30 epoch only and with the reducing the hyper parameter 3.4

Table 3.4: A comparison between the results of original approach and similar approach .

Author	Technique	Blue1	Blue2	Blue3	Blue4	Rouge
Xiujun Li et al. [56]	**OSCAR**	-	-	-	**41.7**	-
Ours	TRIC	81.8	66.5	51.8	39.5	59.2
Ting Yao et al. [57]	**CNN-LSTM-A**	78.7	62.7	47.6	35.6	56.4
Steven J. Rennie et al. [58]	**SCST**	78.1	61.9	47.0	35.2	56.3

3.7 Conclusion

In this chapter we had implement a model of transformer with some bargaining with original model, though this still a good deal(better then CNN+LSTM)

in order to take advantage of the transformer attention and the parallelization of the data in training the cost is a GPU and a lot of data to learn

Conclusion

Conclusion

This research aims to explore a new mechanism in the field of deep learning and computer vision which is transformer the new model make a great jump in the research area of NLP and bringing this to computer vision field helps the researcher in finding a new way to represent data no matter their shape.

Based on the model presented in the third chapter, we conclude that using a transformer in image captioning is a huge jump because is a simple yet scalable approach that can be applied to any kind of data if it is modelized as a sequence of embedding,
We also conclude that transformer is definitely produce SOTA in the computer vision filed due a global understanding of images.

Much like any other task in CV, training time was a bit long. However, that was a problem we could work around with reducing the hyper parameter of the model, also with the need for reasonable storage space for large data-sets

It was a great experience to handle with the transformer in image captioning and it help us to consider applying the model in other territory, also trying with different hyper parameter and other data-set.

Bibliography

[1] Wei Liu, Sihan Chen, Longteng Guo, Xinxin Zhu, and Jing Liu. Cptr: Full transformer network for image captioning, 2021.

[2] A. L. Samuel. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3):210–229, 1959.

[3] Tom M Mitchell et al. Machine learning. 1997.

[4] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. Advances in neural information processing systems, 19:1, 2007.

[5] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science, 362(6419):1140–1144, 2018.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, 2017.

[7] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. arXiv preprint arXiv:1412.2007, 2014.

[8] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In 2011 IEEE Workshop on Automatic Speech Recognition & Understanding, pages 196–201. IEEE, 2011.

[9] F. Rosenblatt. The Perceptron - A Perceiving and Recognizing Automaton, 1957.

[10] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

[11] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units, 2016.

[12] Kunihiko Fukushima, Sei Miyake, and Takayuki Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. IEEE transactions on systems, man, and cybernetics, (5):826–834, 1983.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[16] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029, 2016.

[17] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. IEEE transactions on neural networks and learning systems, 25(8):1553–1565, 2014.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[20] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[21] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112, 2014.

[22] Yoav Goldberg. Neural network methods for natural language processing. Synthesis lectures on human language technologies, 10(1):1–309, 2017.

[23] Tomáš Mikolov, Wen-tau Yih, and Geoffrey 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.

[24] Jeffrey Pennington, Richard Socher, and Christopher D 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.

[25] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

[26] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

[27] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention, 2016.

[28] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading, 2016.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[30] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[31] Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge Belongie. Learning to evaluate image captioning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5804–5812, 2018.

[32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pages 311–318, 2002.

[33] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, pages 74–81, 2004.

[34] Satanjeev Banerjee and Alon 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.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[36] Ning Qian. On the momentum term in gradient descent learning algorithms. Neural networks, 12(1):145–151, 1999.

[37] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

[38] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. arXiv preprint arXiv:2106.11342, 2021.

[39] MD Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. ACM Computing Surveys (CsUR), 51(6):1–36, 2019.

[40] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer, 2014.

[41] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In Proceedings of the IEEE international conference on computer vision, pages 2641–2649, 2015.

[42] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. Vqa: Visual question answering, 2016.

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

[44] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[45] Yezhou Yang, Ching Teo, Hal Daumé III, and Yiannis Aloimonos. Corpus-guided sentence generation of natural images. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 444–454, 2011.

[46] Vicente Ordonez, Girish Kulkarni, and Tamara Berg. Im2text: Describing images using 1 million captioned photographs. Advances in neural information processing systems, 24:1143–1151, 2011.

[47] Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. Multimodal neural language models. In Eric P. Xing and Tony Jebara, editors, Proceedings of the 31st International Conference on Machine Learning, volume 32 of Proceedings of Machine Learning Research, pages 595–603, Bejing, China, 22–24 Jun 2014. PMLR.

[48] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3128–3137, 2015.

[49] Cesc Chunseong Park, Byeongchang Kim, and Gunhee Kim. Attend to you: Personalized image captioning with context sequence memory networks, 2017.

[50] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning, 2015.

[51] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. Advances in neural information processing systems, 28:2017–2025, 2015.

[52] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation, 2015.

[53] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back, 2015.

[54] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. Advances in neural information processing systems, pages 570–576, 1998.

[55] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In International conference on machine learning, pages 2048–2057. PMLR, 2015.

[56] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks, 2020.

[57] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. Boosting image captioning with attributes, 2016.

[58] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7008–7024, 2017.