République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Département: Génie électrique Et Informatique Industrielle

Mémoire de projet de fin d'études

pour l'obtention du diplôme
D'ingénieur d'état en Télécommunications

Spécialité
Systèmes Télécommunications et Réseaux

Thème

Automated Image Description Based on Multimodal Deep Learning

Réalisé par

Meriem BENYOUCEF

Présenté et soutenu publiquement le: 06/07/2022

Composition du Jury:

Présidente	Mme. Souhila BOUTARFA ,	M.C.B	ENST
Promotrice	Mme. Seloua CHOUAF,	M.A.A	ENST
Examinatrice	Mme. Samira BOUCHAMA,	M.C.B	ENST
Examinateur	M. Belkacem ABBADI,	M.C.B	ENST

Année Universitaire 2021 – 2022

Dedication

Before all else, I shall thank God Almighty for the strength, and the unwavering drive he has evoked in me to complete this humble work.

I would like to dedicate this thesis to those who helped shape me into who I am today and nurtured my love for learning: my dear parents.

To my little sisters: Yasmine, Nour El Houda and Hadjer, who brighten up my days.

In my quest to reach this level of my academic life, I have exhausted many efforts. None of this would have been possible without the constant support of my closest friends: Nouha, Rym and Afif, to whom I wish a life of success and joy.

Acknowledgments

My utmost gratitude goes out to Ms. CHOUAF Seloua for accepting to lead me through the entirety of this journey, for her precious advice and wise judgment, for the ambitious spark she lit in me.

Sincere thanks to the members of the jury, for I am honored by them taking the time to review my work, and inspiring me to strive for improved projects in the future.

Table of contents

List of figures ---------- I
List of tables ---------- III
List of symbols---------- V
List of acronyms ----------VII
General introduction---------- 1
Chapiter I. Project context and state of the art
I.1 Introduction---------- 4
I.2 Automated image description ---------- 4
I.2.1 Importance and applications of automated image description---------- 4
I.2.2 Describable images ---------- 5
I.3 Human-like textual generation---------- 5
I.3.1 Natural Language Processing (NLP) ---------- 5
I.3.2 Natural Language Understanding (NLU)---------- 6
I.3.3 Natural Language Generation (NLG) ---------- 6
I.3.4 NLP, NLU and NLG - difference and relationship---------- 6
I.4 Multimodal systems---------- 7
I.4.1 Use cases---------- 7
I.4.2 Advantages - Contribution---------- 8
I.5 Artificial intelligence---------- 8
I.5.2 Deep Learning---------- 9
I.6 Conclusion ---------- 9
Chapiter II. Theoritical background
II.1 Introduction---------- 11
II.2 Convolutional Neural Networks ---------- 11
II.3.1 Normalization layer---------- 11
II.3.2 Convolutional layers---------- 12
II.3.3 Pooling layer ---------- 14
II.3.4 Activation Function---------- 14
II.3.5 Fully-Connected layer---------- 16
II.3.6 VGG-16 ---------- 17
II.3.7 Inception –V3---------- 19
II.3.8 ResNet-50---------- 22
a) Architecture ---------- 24
II.3.9 Xception---------- 25
II.3 Recurrent Neural Networks RNN ---------- 28
II.3.1 Long Short Term Memory (LSTM) ---------- 28
II.4 Global Architecture ---------- 30

II.5 Implementation ---- 32
II.5.1 Presentation of tools ---- 32
II.5.1.1 Hardware specifications ---- 32
II.5.1.2 Software specifications ---- 32
II.5.2 Steps of the experimentation ---- 32
II.5.2.1 Step 1: Training ---- 33
II.5.2.2 Step 2: testing ---- 34
II.6 Conclusion ---- 35

Chapiter III. Experimentation and results

III.1. Introduction ---- 37
III.2. Dataset presentation ---- 37
III.3. Study of different encoder architectures ---- 38
III.3.1 Experiment 01: Functionality test ---- 40
III.3.1.1 Training ---- 40
III.3.1.2 Testing ---- 43
III.3.1.3 Generated descriptions ---- 42
III.3.1.4 Conclusion ---- 44
III.3.2 Experiment 02: Study of convergence ---- 45
III.3.2.1 Training ---- 45
III.3.2.2 Testing ---- 49
III.3.2.3 Generated descriptions ---- 48
III.3.2.4 Conclusion ---- 50
III.4. Study of dataset size's influence on different models' performances ---- 50
III.4.1 Training ---- 50
III.4.2 Testing ---- 54
III.4.3 Generated descriptions ---- 53
III.4.4 Conclusion ---- 54
III.5. Study of the data choice influence ---- 55
III.5.1 Experiment 01: Study over the entire Flicker 8K dataset ---- 55
III.5.1.1 Training ---- 55
III.5.1.2 Testing ---- 58
III.5.1.3 Generated descriptions ---- 57
III.5.1.4 Conclusion ---- 59
III.5.2 Experiment 02: studying 10 random splits of the dataset ---- 59
III.5.2.1 Training ---- 60
III.5.2.2Testing ---- 66
III.5.2.3 Generated descriptions ---- 64
III.5.2.4 Conclusion ---- 69

General Conclusion ---------- 71
Bibliography ---------- 72

List of Figures

Figure I.1: Example of textual description types……..…....5
Figure II.1: General architecture of a convolutional neural network…….....11
Figure II.2: An image before and after size normalization……12
Figure II.3: The three channels of an RBG image……..12
Figure II.4: Gray scale image as matrix of numbers……...12
Figure II.5: Convolution procedure……...…….13
Figure II.6: Output images of applying different convolution filters to a given image……..13
Figure II.7: Illustration of the pooling techniques…….....14
Figure II.8: Feed forward propagation process……...…15
Figure II.9: Illustration of the fully connected layer……...16
Figure II.10: Detailed VGG-16 architecture……18
Figure II.11: Detailed Inception V3 architecture……....20
Figure II.12: Detailed architecture of ResNet-50……...23
Figure II.13: Details of each layer of the ResNet-50 architecture……24
Figure II.14: Detailed Xception architecture……..…25
Figure II.15: Modified depth wise separable convolution……..……....26
Figure II.16: Conventional depth wise separable convolution……....26
Figure II.17: Details of each layer of the Xception architecture…….…....27
Figure II.18: Unrolled recurrent neural network architecture…….28
Figure II.19: Encoder-decoder merge architecture…….…….30
Figure II.20: Process of sequence generation…….…….31
Figure II.21: Results of the four architectures performance on ImageNet dataset…...……...31
Figure II.22: Forward and back-propagation in a neural network……..…..……33
Figure III.1: Flickr 8K dataset overview…….…...37
Figure III.2: Images from the Flickr 8K dataset with their attributed descriptions……38
Figure III.3: Flowchart of the experimental procedure……...39
Figure III.4: An overview on the global architecture…….…….38
Figure III.5: Loss results of the four studied models trained for 10 epochs……...41
Figure III.6: Accuracy results of the four studied models trained for 10 epochs…….……...41
Figure III.7: Examples of generated descriptions by the four models……42
Figure III.8: Examples of generated descriptions by the four models……43
Figure III.9: Natural language metrics evaluation for the four models…….....44
Figure III.10: Loss results of Xception and Inception-V3 models trained for 50 epochs…...46

Figure III.11: Accuracy results of Xception and Inception-V3 models……..46
Figure III.12: Examples of generated descriptions by Inception-V3 and Xception …..…….48
Figure III.13: Natural language metrics evaluation for Xception and Inception-V3 models.49
Figure III.14: Loss results of the four models trained on one split of the dataset……...51
Figure III.15: Accuracy results of the four models trained on one split of the dataset……...52
Figure III.16: Generated description by the four models trained on one split of the dataset..53
Figure III.17: Evaluation metrics results for the four models trained on one split ……54
Figure III.18: Loss results of Xception trained on the original and shuffled set…….56
Figure III.19: Accuracy results of Xception trained on the original and shuffled set……..…56
Figure III.20: Examples of generated descriptions by Xception after being trained on the original and shuffled datasets……57
Figure III.21: Examples of generated descriptions by Xception after being trained on the original and shuffled datasets……58
Figure III.22: Natural language metrics evaluation for Xception on the original and the shuffled set……58
Figure III.23: Minimum, maximum and average loss results obtained by Xception over ten splits ……..……61
Figure III.24: Minimum, maximum and average loss results obtained by Inception-V3 over ten splits of the dataset……..62
Figure III.25: Minimum, maximum and average accuracy results obtained by Xception over ten splits of the dataset……..62
Figure III.26: Minimum, maximum and average accuracy results obtained by Inception-V3 over ten splits of the dataset……..63
Figure III.27. Examples of generated descriptions by Xception after being trained on the original and shuffled datasets……....65
Figure III.28: Natural language evaluation metrics results of Xception trained on ten splits of the dataset……..66
Figure III.29: Natural language evaluation metrics results of Inception-V3 trained on ten splits of the dataset……67
Figure III.30: Average, minimum and maximum evaluation metrics values of Xception after being trained over ten splits of the dataset…….…….....68
Figure III.31. Average, minimum and maximum evaluation metrics values of Inception-V3 after being trained over ten splits of the dataset……68

List of Tables

Table II.1: Activation functions……..……....15
Table II.2: Summary of calculations results……...……..…17
Table II.3: Details of each layer of the VGG-16 architecture…….....19
Table II.4: Details of each layer of the Inception-V3 architecture……...22
Table III.1: Summary of training settings……....40
Table III.2: Summary of time consumption on training by the four models……..……....40
Table III.3: Summary of best loss and accuracy results for each model……..…42
Table III.4: Natural language evaluation metrics' results for each model……..……44
Table III.5: Summary of training settings……...…….45
Table III.6: Summary of time consumption on training by Xception and Inception-V3……...45
Table III.7: Best loss and accuracy results achieved by Xception and Inception-V3…..…....47
Table III.8: Comparison between the first, tenth and convergence loss results.....47
Table III.9: Comparison between the first, tenth epoch and convergence accuracy result…..47
Table III.10: Natural language metrics evaluation for Xception and Inception-V3…….……49
Table III.11: Summary of training settings…….…….50
Table III.12: Summary of time consumption results……..……...……...51
Table III.13: Best loss and accuracy results of each model…….52
Table III.14: Evaluation metrics results for the four models trained on one split……..…....54
Table III.15: Summary of training settings……...55
Table III.16: Best accuracy and loss results achieved by the Xception model trained on the original and shuffled training set ……..57
Table III.17: Natural language metrics evaluation for Xception and Inception-V3……..59
Table III.18: Summary of training settings…….……..…60
Table III.19: Summary of training sets vocabulary for the ten splits…..……...…...…60
Table III.20: summary of time consumption on training by Xception and Inception-V3 models on 10 splits…….....…61
Table III.21: best, average, maximum and minimum results of accuracy and loss values of Xception and Inception-V3 trained on ten splits of the dataset…….……63
Table III.22: Natural language evaluation metrics results of Xception trained on ten splits of the dataset……...…..…….……67
Table III.23: Natural language evaluation metrics results of Inception-V3 trained on ten splits of the dataset……..……...…….67

Table III.24: Average, minimum and maximum evaluation metrics values of Xception after being trained over ten splits of the dataset……...……...……68
Table III.25: Average, minimum and maximum evaluation metrics values of Inception-V3 after being trained over ten splits of the dataset……...……...…69
Table III.26: Comparison of metrics results of the Xception model trained on one split and on the entire dataset…….....……..……69

List of symbols

×	Product
+	Addition
−	Subtraction
\	Division
%	Percentage
<	Inferior
>	Superior
-	Underscore
~	Tilde
Y	Activation function
∂	Standard deviation
$\sum$	Summation
α	Added weight
β	Number of unigrams in reference sentence
γ	Number of unigrams in a candidate sentence
τ	Number of unigrams in the candidate sentence also found in reference sentence
Ω	N-gram
Tanh x	Hyperbolic tangent function
Sin x	Sine function
$\mathbf{e^x}$	Exponential function
Max (x,y)	Maximum function
Log (x)	Logarithmic function
BP	Multiplicative factor
RS	Reference sentence
CS	Candidate sentence
RSL	Reference sentence length
CSL	Candidate sentence length
A	Learning rate
R	Recall
P	Precision
c	Total length of candidate translation corpus
Y	Activation function
x	Output of the previous filter and input of the activation function
i	Index
j	Index
t	Stride

d	Padding
u	Input size
s	Filter size
$\mathbf{P_n}$	Precision of an n-gram
r	Average length of all references
$\mathbf{W_t}$	New weight
$\mathbf{W_{t-1}}$	Old weight
$\mathbf{C_{t-1}}$	Old cell state
$\mathbf{f_t}$	Ft vector of the values to be deleted from the cell state
$\mathbf{C_{t\sim}}$	Vector of new candidate values to the state of the cell state
$\mathbf{C_t}$	New cell state
$\mathbf{x_t}$	Input of the present cell
$\mathbf{h_t}$	Output of the present cell
$\mathbf{x_{t-1}}$	Input of the old cell
$\mathbf{h_{t-1}}$	Output of the old cell
$\mathbf{x_{t+1}}$	Input of the future cell
ht+1	Output of the future cell
$\boldsymbol{i_i}$	Vector of values to be updated

List of acronyms

AGI	Artificial General Intelligence
AI	Artificial Intelligence
ANI	Artificial Narrow Intelligence
ANN	Artificial Neural Network
ASI	Artificial Super Intelligence
BLEU	Bilingual Evaluation Understudy
BRNN	Bidirectional recurrent neural networks
CIDEr	Consensus-based Image Description Evaluation
CNN	Convolutional Neural Networks
DL	Deep Learning
GRU	Gated Recurrent Units
LCS	Longest Common Subsequence
LSTM	Long Short Term Memory
METEOR	Metric for Evaluation for Translation with Explicit Ordering
ML	Machine Learning
MSE	Mean Squared Error
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
RBG	Red Blue Green
RNN	Recurrent Neural Networks
ROUGE	Recall Oriented Understudy for Gisting Evaluation
SGD	Stochastic Gradient Descent
TF-IDF	Term Frequency Inverse Document Frequency
VGG	Visual Geometry Group
FC	Fully Connecte Layer
Conv	Convolutional Layer
Conv Sep	Separable Convolution Layer
Avg-Pool	Average Pooling Layer
Max-Pool	Max Pooling Layer
Concat	Concatenation Of The Inputs

Add	Addition Of The Inputs
FF	Flattened Feature Vector
Relu	Rectified Linear Unit
GPU	Graphics Processing Unit
CPU	Central Processing Unit
RAM	Random Access Memory
Ghz	Gigahertz
GB	Gigabyte
DDR4	Double Data Rate
2D	Two Dimentional

General introduction

The researchers have benefited from very large data samples including images and videos to enrich their experiments and provide more accurate results. Furthermore, the creation of data has grown exponentially in the past few decades, resulting in billions of images and videos uploaded daily to the World Wide Web. Despite the benefits of these big amounts of data, the tasks of organizing and searching for specific photo collections amongst thousands has become a challenging and immensely time consuming task for humans. Thanks to the invention of robust machines with high computational power which enable us to process faster the huge amounts of data and the revolutionary breakthroughs that have been achieved in various artificial intelligence fields, these tasks can now be automated using deep learning-based models that efficiently use image information to generate accurate descriptions. These systems that have shown high success rate in object identification, image classification, and scene recognition can be further extended to the generation of relevant descriptions of those images content. This idea can have a wide variety of applications including: virtual assistance, image indexing, information retrieval, human-robot interaction and more.

Nevertheless, the creation of such systems remains challenging since it requires not only the identification of objects or recognizing scenes in a given image, but also to analyze their states, understand the relationship among them and generate a semantically and syntactically correct sentence. Furthermore, the model should be optimized to bear large amounts of data without being time consuming.

Using a computer to automatically generate a natural language description for an image, is connecting two research communities. First, the computer vision community contributes by creating a model that identifies the objects and the relationships between them and transforms that into a useable form of data. Second, natural language processing community contributes with a model that is able to generate a sequence of words that describes in a human-like form the relationship between the former identified objects.

With the fast development of deep neural networks, employing more powerful network structures to improve the performance of image description generation models, many architectures have been created. In this work, we will study how convolutional neural networks (CNN) and recurrent neural networks (RNN) can be employed to solve image description challenges.

The main goal of this project is to create a deep learning-based model that is able to generate syntactically and semantically correct natural language descriptions based on a given image. Then, the quality of the generated descriptions will be evaluated.

Thanks to the various use cases of automated image description in the past few years, the attention of many researchers has been drawn leading them into competition to develop different architectures to better fulfil the task. Most of these architectures are based on the encoder-decoder structure which will be adopted in this project.

For the encoder part, CNN architectures encode the information displayed by the content of the image into a useable form of data. After that, a RNN-based model is used as a language decoder to produce, word by word, a descriptive sentence.

In our experiments, we chose to use, as encoders, Xception, Inception-V3, ResNet-50 and VGG-16 architectures, being the four most common CNN architectures, whereas, a LSTM-based architecture is used in the decoding part.

In order to evaluate the generated descriptions, different natural language evaluation metrics have been adopted, namely: BLEU, ROUGE, METEOR and CIREr.

To present the work carried out, this document will be organized as follows:

- The first chapter introduces the contextual generalities related to the project, the various use case of automated image description and the main underlying goals of this thesis, along with an overview on previous works.
- The second chapter encloses a presentation of the necessary vocabulary related to the topic, the different fields of artificial intelligences, deep learning including CNN and RNN, in addition to natural language processing. Also, this chapter will mention the global structure of the proposed architecture and the different methods that are used to evaluate its performances.
- The third chapter describes the experimental part of this project. It will start with the introduction of the used dataset, the steps to follow during the training and testing processes to finally present the various experiments carried out and discuss the results that have been recorded.
- This document is completed with a summary of the main conclusions to retain and some perspectives to guide future works.

Chapter I. Project context and State of the art

I.1 Introduction

The major problem associated with visual data is information load, organizing, identifying and accessing images can be an inefficient and time consuming task for humans. Natural language has the advantage that it permits the condensation of visual data at different levels of detail according to the operation-specific demands. Therefore, building a CNN and RNN-based automatic description generator is one of the applications that uses natural language advantages to automatically generated textual descriptions from images which is a promising solution for the this problem. This chapter is presentation of the theoretical background of the concepts used in this study and the related works.

I.2 Automated image description

The process of generating textual descriptions based on images has been recently considered as a fundamental topic of research that combines the effort of different communities acting in the field of artificial intelligence and more precisely the ones interested in deep learning.

I.2.1 Importance and applications of automated image description

Deep learning has accelerated the speed of research work in various fields. Development of convolutional neural network-based deep learning architectures has played a major role in providing better solutions to computer vision-based problems. Similarly, varieties of recurrent neural networks are used to solve problems in natural language processing. Although there are some problem statements which exist in the intersection of computer vision and natural language processing domain, one such problem statement is image captioning where the objective is to generate textual descriptions for the input image. Designing better deep learning-based solutions for this problem statement relies heavily on developments done in both domains.

In this section, we have picked some interesting use cases where automated image description can be both helpful and profitable.
Image description is of a great help for visually impaired people to better understand the content of images if an additional text-to-voice module in integrated to make the generated descriptions audible.

In addition, it can be used in the medical field to accelerate the identification process of certain health conditions using automatic image interpretation such as skin cancer diagnosis.

Textual image description can be used in real time monitoring and surveillance where the relevant suspicious descriptions can be made to fire-up alarms and take the appropriate corrective actions or to guide self-driven cars in their surroundings analysis.

It can also help search engines to retrieve suitable results over the captured annotations or keywords and it may be useful in automatic product tagging in online catalogs.

The automatic generation of image content descriptions can be valuable for the identification of the users' uploaded images in social media platforms or to categorize personal multimedia collections.

I.2.2 Describable images

Digital images are used to capture all kinds of visuals, spaces, objects, people, and more. Textual descriptions can be automatically generated from most detailed and informative visuals such as: logos, videos, pictures, art, graphs, maps and charts. However, it is not destined to describe aesthetic and decorative visuals since they don't convey any meaningful information [1].

Images can be presented in many textual forms namely: alt-texts, captions and descriptions, each of them is destined for a different use case.

- **Alt-text:** It is a short textual explanation of an image's content, mostly used in the coding of digital files, to serve as an alternative to the image in case it does not appear.
- **Caption:** It provides a brief explanation accompanying an image or an illustration.
- **Image description:** a detailed explanation of an image that provides more complete information about its content.
  The example below demonstrates how each method describes the given image.



Alt text: A picture of baby lions and their father lion.

Caption: A photo of baby lions taken during a zoo visit.

Image description: four baby lions are sitting on a large rock, beside an old male lion with plants in the background.

Figure I.1. Example of textual description types.

I.3 Human-like textual generation

In this section, concise definitions are mentioned to install a certain familiarity with the fundamental notions related to the natural language field.

I.3.1 Natural Language Processing (NLP)

Natural language processing (NLP) is the branch of artificial intelligence that combines computational linguistics-rule based modeling of human language with statistical and deep learning models to help computers analyze, interpret and manipulate human language in the form of text or voice data. This field aims to make the machines understand human natural language. The importance of this field lays in its ability to process efficiently large volumes of textual data, organize unstructured data by resolving ambiguity in language and adds useful numeric structure to the data for many downstream applications, such as speech recognition or text analytics.

NLP systems perform tokenization, parsing, lemmatization, stemming, part-of-speech tagging, language detection and identification of semantic relationships in order to break down language into elemental pieces. This will enable the machine to understand relationships between the pieces to create meaning. A vocabulary of unique words is then built into the NLP systems in addition to a set of grammatical rules, using text vectorization

to transform text into a useable form of data such as vectors of numbers where each unique number refers to a specific word in the vocabulary. Machine learning algorithms are fed large datasets of images with their respective expected textual outputs. The system learns to make associations between input and the textual representation that better describes it [2].

I.3.2 Natural Language Understanding (NLU)

NLU systems aim to interpret intent, resolve context and word ambiguity. These systems are used to tackle the extremely complex problem of semantic interpretation, understanding the intended meaning of spoken or written language, with all the context and inferences a human is able to comprehend [2].

I.3.3 Natural Language Generation (NLG)

Text generation is the task of generating text with the goal of appearing indistinguishable to human-written text. This task is more formally known as NLG in the literature.
In other terms, NLG uses computational linguistics and deep learning-based models to automatically generate natural language outputs based on non-linguistic inputs.

From a historical point of view, in the 1970s, Goldman firstly announced its work in the text generation field from a deep conceptual base [3]. Further researches were pursued by McDonald that tackled the text generation problems as decision making problem [4]. An applet on language planning was presented by McKeown in 1980's [5]. In the 1990s, a generic architecture for text generation was discussed by Reiter [6] and Hovy [7].

Lately, the NLG problem is addressed with Markov processes or deep generative models like Long Short-Term Memories. Text generation systems are evaluated either through human ratings or automatic evaluation metrics like METEOR, ROUGE, and BLEU (which will be introduced later).

I.3.4 NLP, NLU and NLG - difference and relationship

NLP translates what a human says or writes into textual data. An NLG system uses that data to generate a language. But what if the machine's linguistic response makes no sense? That's where the NLU comes in.

While NLP systems are responsible for transforming natural human language into a useable form of data and NLG systems use such data to produce natural language text, NLU models analyze the data produced by NLP to understand the meaning of words and contextual nuances, and makes sure the NLG model generates human-like sentences that are grammatically and semantically correct.

I.4 Multimodal systems

Being one of the most familiar means of human interaction, natural language can significantly reduce the training effort required to enable communication with machines. On the other hand, the coverage of current natural-language dialogue systems is still strongly limited.

The integration of non-verbal media often improves the usability and acceptability of natural language components as they can help compensate for the deficiencies of current natural language technology. In fact, empirical studies show that properly designed multimodal systems have a higher degree of stability and robustness.

Before going any further, the term mode or modality is used to refer to different kinds of perceptible entities while the term medium relates to the carrier of information, to the different kinds of physical devices and to the information types.

We call multimodal system the system able to analyze and generate multimodal information or provide support in accessing digital resources of multiple media [8].

Based on the observation that a lot of information is encoded redundantly, a number of research projects rely on the linguistic sources (e.g. transcribed speech or closed captions) when analyzing image/video material. Indeed a number of projects, such as the Broadcast News Navigator [9], have shown that the use of linguistic sources in multimedia retrieval may help overcome the so-called semantic gap, i.e. the discrepancy between low-level features and higher-level semantic concepts [8].

I.4.1 Use cases

Natural language processing methods are crucial tools to facilitate access and use of enormous amount of multimedia information, these tools can optimize these tasks in three main ways:

- **Multimedia analysis**

Information can often be retrieved more easily from meta-data, audio, or images thanks to NLP tools. A number of projects have shown that the use of linguistic sources in multimedia retrieval may help overcome the discrepancy in semantic concepts.

Natural language-based video or image processing systems can be applied in tasks that don't require complete syntactic and semantic analysis such as: image and video classification, and video segmentation.

Researchers took advantage of different kinds of user-provided natural-language content for image classification as presented in their paper "Product Images Classification with Multiple Feature Combination" [10]. Furthermore, information extracted from online resources can be exploited for image or video analysis tasks, for example: resolving ambiguities of text accompanying an image or to refine queries for an image retrieval system [11].

- **Access to multimodal information**

Natural language access to visual data is often much more convenient since it allows for a more efficient formulation of queries by using simple keywords. Such keywords may specify a subject matter, such as 'sports' [12], but also subjective impressions, such as 'sad movie [13].

- **Summaries of multimedia information**

An early example is the Columbia Digital News System [12] that provides summaries over multiple news articles, and uses image classification tools to select the images that are the most relevant to the generated summary.

Other NLP systems start from visual information and transform it into natural language [14] such as ROCCO II, which generates natural-language commentaries for games of the RoboCup simulator league, by analyzing visual scenes and recognize spatial relations and motion patterns.

I.4.2 Advantages - Contribution

Multimodal/multimedia systems pose significant challenges for natural-language processing, which focuses on the analysis or generation of one input or output modality/medium only. Methods for natural-language processing may be extended in such a way that they become useful for the broader context of multimodal/multimedia systems as well. While unification-based grammars have proven useful for modality orchestration and analysis, text planning methods have been successfully applied to multimedia content selection and structuring.

The works done in the area of multimedia information retrieval demonstrates that the integration of natural language methods enables a deeper analysis of the underlying multimedia information and thus leads to better search results.

I.5 Artificial intelligence

Artificial Intelligence is the incorporation of human intelligence into computers through an algorithm in order to mimic how the human brain works. The machine than would be able to provide solutions for different tasks without a human intervention. This field can be primarily divided into three subfields:

- **Artificial Narrow Intelligence (ANI):** which is the only type of AI that exists in our world today, it is goal oriented and programmed to perform a single task efficiently such as: computer-aided diagnosis to assist physicians interpret medical images, natural-language-based user interfaces that involve using human language as input for interacting with computers, processing and analyzing big data and learn from the outcome without being explicitly programmed.
- **Artificial General Intelligence (AGI):** it is known as the future of AI, it is about machines becoming conscious, driven by emotions and able to think, strategize and perform multiple tasks even under uncertain conditions.
- **Artificial Super Intelligence (ASI):** which is a hypothetical level of AI in which machines will be able to supersede human intelligence in all aspects.

I.5.1 Machine Learning

Machine learning (ML) is about providing the machine with a proper dataset and self-learning algorithms to spot possible patterns in the given dataset and make better decisions performing a specific task. The learning process takes place automatically without being explicitly programmed, the improvement happens through experiences. Some use cases of ML are: natural language analytics, sentiment analysis, object detection and classification tasks. ML algorithms can be classified into three categories:

- **Supervised learning:** where both input and output variables are known to the machine. The task of the machine is to learn the mapping from input to output and to create a classification model that predicts the classification of new data.
- **Unsupervised learning**: t is when the machine is left to learn from unlabeled data, find similarities and patterns to make its' future decisions.
- **Reinforcement learning**: The machine relays on its' past experiments to make future decision thanks to an interactive environment that provides it with rewards for performing correctly and penalties for performing incorrectly.

I.5.2 Deep Learning

Deep Learning (DL) is the subfield of ML that uses programs based on artificial neural networks, which are complex multi-layered networks. Through these networks the information is processed then transferred from one layer to another over connecting channels, until it reaches the last layer that produces the outputs. This mimics the work way of neurons in our brain.

We mainly classify the neural networks architectures into Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

I.6 Conclusion

Throughout this chapter, we have first presented a gentle introduction to the major goals of the automated image description passing by the presentation of the human-like textual generation. In this part, we tried to introduce the reader to the main natural language processing applications. Next, we broadly presented the multimodal systems, their use cases and contributions. Finally, we explained the differences between artificial intelligence, machine learning and deep learning to prepare the reader to the next chapter where we have the intent of discussing the different deep learning-based architectures and tools that will be used in the experimental part.

Chapter II. Theoretical background

II.1 Introduction

This chapter will include a discussion of the different deep learning-base architectures that will be used in the experimental part, in addition to the necessary vocabulary related to this work. Then, the general architecture of our system will be presented. Finally, the tools used to realize and evaluate this project will be listed.

II.2 Convolutional Neural Networks

A convolutional neural network is a category of artificial neural networks that has proven to be very effective in processing structured arrays of data such as images. The main use cases of this category is image recognition and classification. The first CNN was created in 1989 by Yann Lecun and it was called LeNet.

The power of this type of networks lays in its ability to identify patterns in the given dataset very efficiently by operating on raw images without the necessity of preprocessing and recognize the various aspects of a given image such as lines, gradients and circles. Then it assigns to each of them a unique number called bias. The information is transferred from one layer to another over connecting channels which have a value attached to each of them called weight. The bias is added to the weighted sum of inputs reaching the neuron to which then an activation function is applied. The result of the function determines if the neuron gets activated. Every activated neuron passes on information to the following layer and this continues up to the second last layer. Finally, the last layer produces outputs for the program. The figure below will demonstrate the general architecture [15].

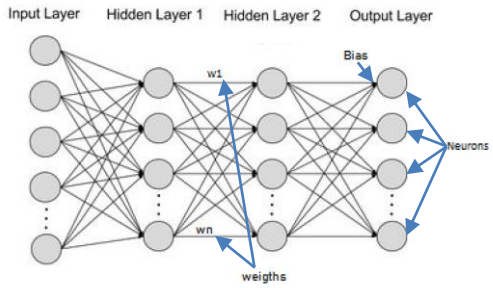


Figure II.1. General architecture of a convolutional neural network

The hidden layers consist of four types of layers which are the basic building blocks of every Convolutional Neural Network:

- Convolutional layers.
- Pooling layers.
- Fully connected layers.
- Normalization layers.

II.3.1 Normalization layer

This layer performs a normalization of the input image's size, in the following example, the original image has been modified and turned into 32×32×1 pixels, this layer doesn't count as one of the network's hierarchical layers and each CNN have its' own input normalization[15].

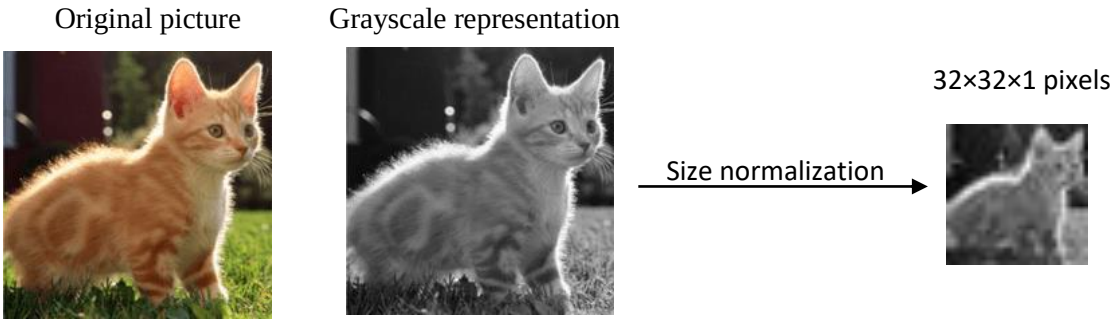


Figure II.2. An image before and after size normalization

II.3.2 Convolutional layers

Standard digital colored images are obtained thanks to three sensors which are electronic devices that convert the optical image created by the camera lens into a digital file that is stored in a device's memory, these three sensors capture the intensity of red, green and blue light hitting a small area of the projected image called a pixel, each sensor stores the intensity values of its specific color in a 2D matrix called a channel producing three channels. Stacking these channels over each other forms the 2D image. A grayscale image is composed of one channel only. The figure below demonstrate the matrix representation.

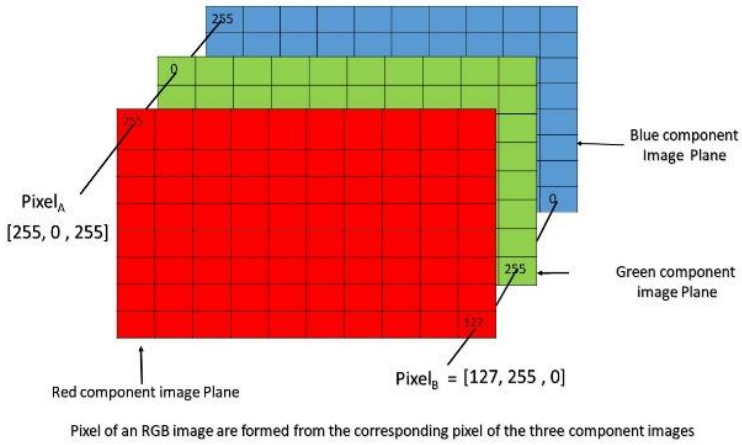


Figure II.3. The three channels of an RBG image

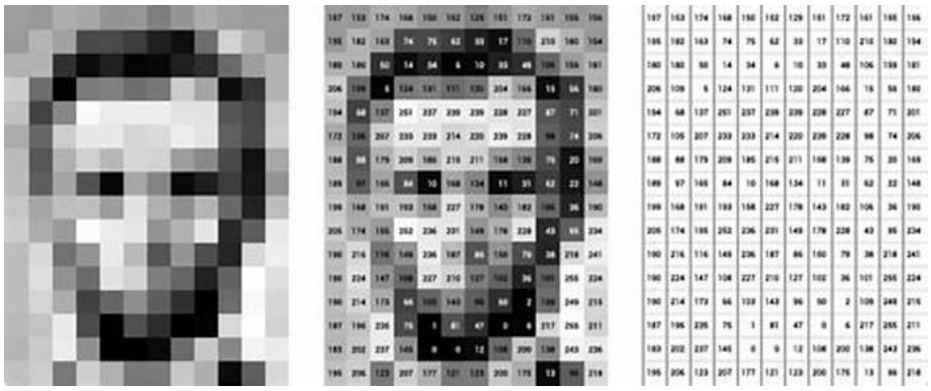


Figure II.4. Gray scale image as a matrix of numbers

Convolutional layers contain many convolutional filters called kernels, a kernel is a smaller matrix for example: a 3×3 matrix, which slides over the original image pixel by pixel calculating the element-wise multiplication of the kernel's elements and the original image's sub matrix elements than add the results of all products, the result will be the element of the output image. The figure below demonstrate the process.

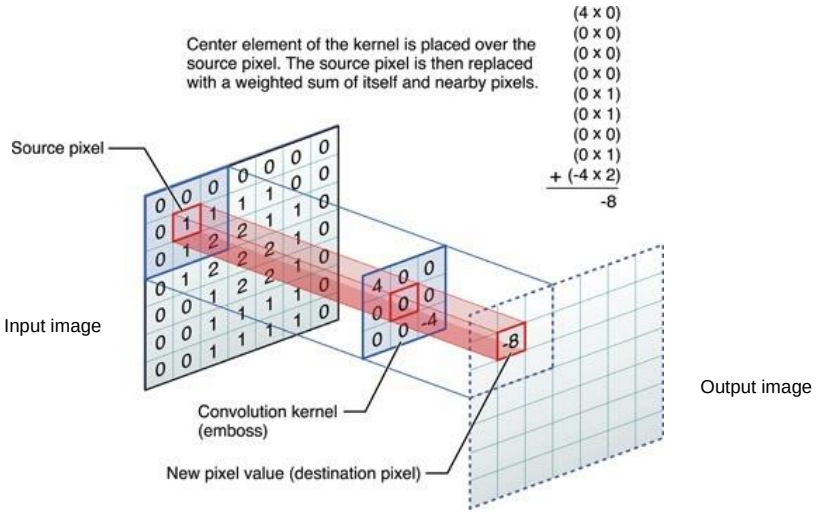


Figure II.5. Convolution procedure

Just like the human visual cortex processes incoming images by identifying their features progressively, convolutional layers stacked one on top of the other enable CNN architectures to recognize more sophisticated shapes and extract more complex features. With three or four convolutional layers it is possible to recognize handwritten digits and with twenty five layers it is possible to distinguish human faces [15]. The figure below shows on the right side the output images of applying different types of convolutional filters to the pictures on the left side.

Original image



Output images of applying different convolutional filters

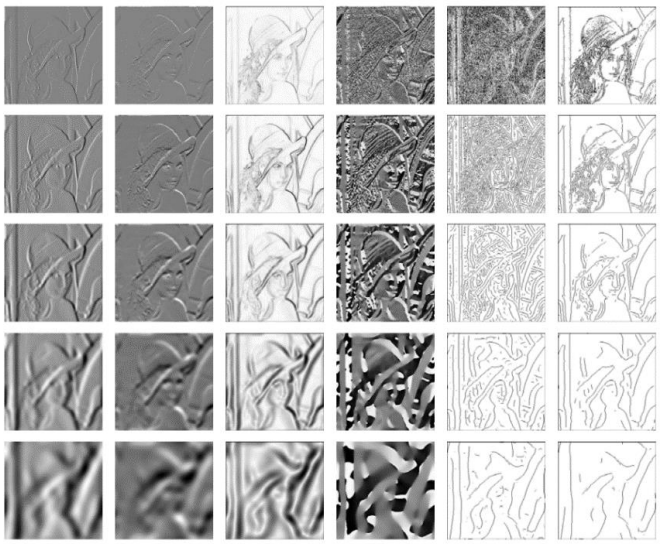


Figure II.6. Output images of applying different convolution filters to a given image

II.3.3 Pooling layer

The pooling layer, also called down sampling layer or subsampling layer, reduces the spatial size of the convolved image while preserving its important features, this has the advantage of decreasing the computational power required to process the data by effectively training the model on dominant features which should be rotational and positional invariant [15].

The process is similar to the convolution, by sliding a window of a smaller size, for example a 2×2 window, moves over the original image applying a pooling technique amongst the following:

- **Average pooling:** The elements of the image portion that is covered by the window are averaged and the result is stored in the output image [15].
- **Max pooling:** The maximum value of the elements of the image portion that is covered by the window presents one element of the output image. This technique is widely used because it doesn't only reduces the dimension of the image, it also discards the noisy activations. It is generally said that max pooling performs a lot better than average pooling [15].

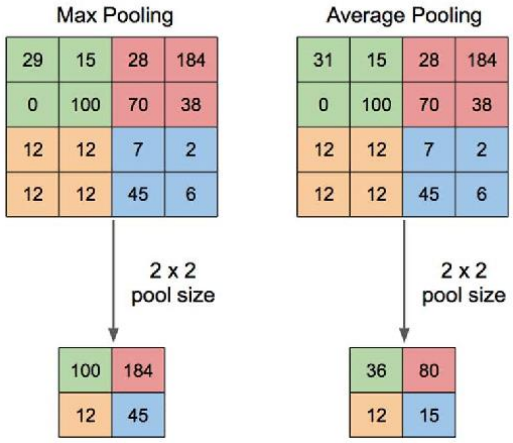


Figure II.7. Illustration of the pooling techniques

II.3.4 Activation Function

Activation or transfer functions lie at the core of every convolutional neural network from shallow to deep. Their properties and characteristics shape the output range of each layer, and thus, their capabilities.

The role of the activation function is deciding whether the neuronal set of input is important. The activation of the neuron or not in the process of predicting an output to be fed to the next hidden layer, using simple mathematical operations. This function adds non-linearity to the network during the forward propagation. Without the activation function, all neurons will perform linear transformation on the inputs using the weights and biases because the composition of two linear functions is a linear function, this makes learning any complex task impossible and the result would be just a linear regression model.

During the feed-forward propagation, the flow of information occurs in the forward direction. The input is used to calculate some intermediate function in the hidden layer, which is then used to calculate the output. In this case, the activation function is a mathematical gate in between the input feeding the current neuron and its output going to the next layer. [15] The figure below illustrates the feed-forward propagation process.

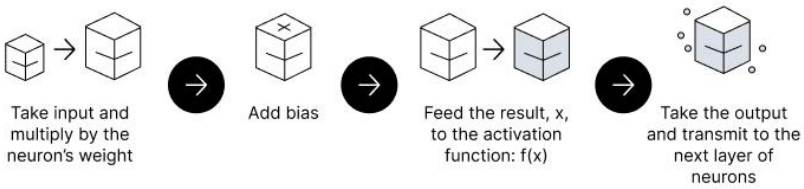
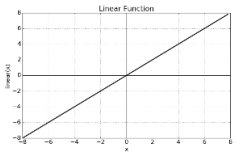
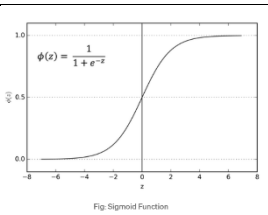
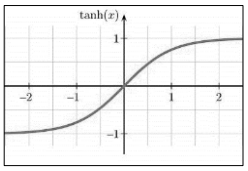
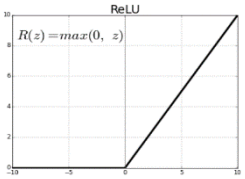
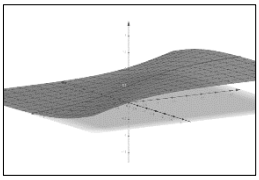


Figure II.8. Feed forward propagation process

The table below summarizes some of the most used activation functions.

Table II.1. Activation functions

Function	Nature	Equation	Range	Use	Shape
Linear	Linear	$Y = a \times x$	]-∞,+∞[	Output layers for regression problems	
Sigmoid	Non-linear	$Y = \frac{1}{1+e^{-x}}$	[0,1]	Usually used in output layer of a binary classification	
Tanh	Non-linear	$Y = \tanh x$ $Y = \frac{2}{1+e^{-2x}}$ $Y = 2 \times sigmoid(2x)$	[-1,+1]	Usually used in hidden layers of a neural network, it helps bringing mean close to 0	
RELU	Non-linear	$Y = \max(0, x)$	[0,+∞[	Implemented in hidden layers, widely used because it is less computationally expensive and learns faster than tanh and sigmoid because it involves simpler mathematical operations.	
Softmax	Non-linear	$Y_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$	[0,1]	Mostly used in the output layer of the classifier when trying to handle multiple classes.	

II.3.5 Fully-Connected layer

After the series of shape and size changes applied to the input image throughout the different layers, it is finally flatten into a column vector, and all its neurons are connected to all neurons of the fully connected layer. This layer is also called a dense layer, it has a shape of a multi-layer perceptron that maps feature vectors generated by the convolutional and pooling layers to the outputs using a classification technique such as softmax.

The figure below illustrates the process.

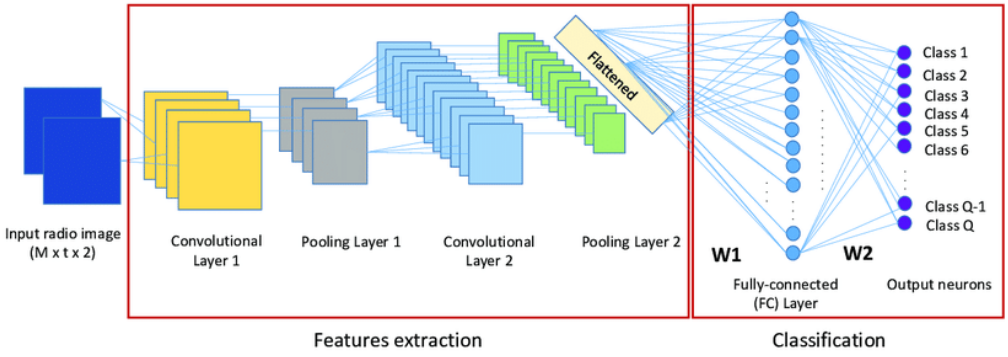


Figure II.9. Illustration of the fully connected layer

The example below will demonstrate different layers process images in the network.

➢ Considering a convolutional layer of six kernels with 5×5 size each.

- The input shape is by the height and width of the image 32×32.
- The output will be a feature map and its' shape is calculated using the formula below.

$$\text{Width or Height} = \frac{u - s + 2 \times d}{s} + t \qquad \textbf{(II.1)}$$

Where:

- t: stride (by default is set to 1)
- d: padding (by default is set to 0)
- u: input size
- s: filter size

- Each filter produces a channel or an image.
- The number of weights is the number of cells in a filter's matrix multiplied by the depth of the input image.

$$\text{Number of weights} = \text{filter's height} \times \text{filter's width} \times \text{depth} \qquad \textbf{(II.2)}$$

- A bias of one is added for each filter applied, so the bias will be set to 1.

$$\text{Total number of parameters} = (\text{number of wiegths} + \text{bias}) \times \text{number of filters} \qquad \textbf{(II.3)}$$

- The total number of parameters in a convolutional layer is calculated as follows:
- Each pixel in this convolutional layer is connected to 25 pixels plus a bias pixel in the input image.
  The total number of connections is calculated as follows:

$$\text{Number of connections} = (\text{Number of weights} + \text{bias}) \times \text{number of filters} \times \text{number of pixels in the output image} \qquad \textbf{(II.4)}$$

- Assuming that the convolutional layer is followed by a pooling layer with the following settings:
    - Window size: 2×2
    - Number of windows: 6
    - Pooling type: average pooling
- The output shape is calculated using the previously given formula with the following settings:
    - Stride: 2
    - Padding: 0
- The sum of the pixels covered by the 2×2 window in the output images of the convolutional layer is multiplied by a weight coefficient plus an offset, and then the result is mapped in the output image.
- Each filter produces a channel or an image so the number of channels in this case is 6 and the output shape will be 14×14×6.
- A bias of one is added for each filter applied, so the bias will be set to 1.
- This layer has got no learnable parameters.
- The total number of connections is calculated using the formula below:

$$\text{Number of connections} = (\text{Number of weights} + \text{bias}) \times \text{number of filters} \times \text{number of pixels in the output image} \quad \textbf{(II.5)}$$

The table below summarizes the results.

Table II.2. Summary of calculations results

Layer	Convolutional layer	Pooling layer
Input shape	32×32×1	28×28×6
kernel shape	5×5	2×2
Number of kernels	6	6
Output feature map shape	28×28×1	14×14×1
Number of weights	25	0
Learnable parameters	156	0
Number of neurons	28×28×6	14×14×6
Number of connections	122304	5880

Convolutional neural networks are basically seen as a repeated combination of convolutional layers, pooling layers and activation functions, aiming to extract more and more relevant features, increasing the depth and the complexity of these networks to solve more challenging tasks, this resulted in the creation of various CNN architectures.

As part of the proposed project, we are interested in the study of four CNN architectures selected among the most commonly used models, namely: Inception-V3, Xception, VGG-16 and ResNet-50. These models will be presented in detail.

II.3.6 VGG-16

The Visual Geometry Group (VGG) architecture was firstly introduced by Simonyan and Zisserman in their 2014 paper "Very Deep Convolutional Networks for Large Scale Image Recognition" [16]. The contribution from this paper is the designing of deeper networks by

stacking more layers and use smaller size filters. The “16” stands for the number of weighted layers in the network which was considered very deep in 2014.

a) Architecture

The figure below illustrate the specifications related to each layer.

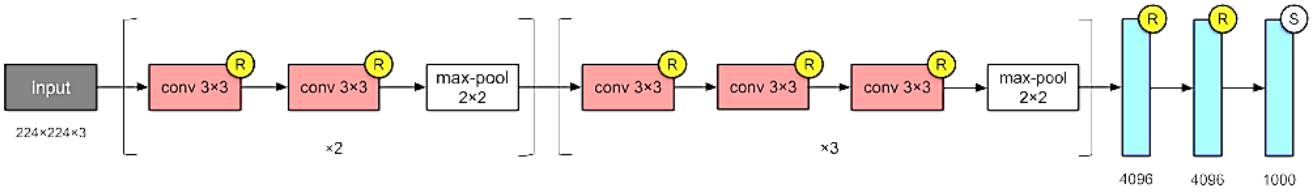


Figure II.10. Detailed VGG-16 architecture

The input of this architecture has a fixed size of 224×224×3. It is preferable to perform an additional normalization process to the image by subtracting the mean value from each of its pixels.
The image pass through the first stack of two convolutional layers followed by a ReLU activation. Each of these two layers contains 64 filters. This configuration preserves the spatial resolution, the size of the output activation map is the same as the input image dimensions. The activation maps are then passed through spatial max pooling. This halves the size of the activations. The output size is 112×112×64.

The resulting activations flow through another 2 convolutional layers stack, with 128 filters each. The size after the second stack becomes 56×56×128. This is followed by the third stack with three convolutional layers of 256 filters each and a max pooling layer, making the output size of the stack 28×28×256. This is followed by two stacks of three convolutional layers, each containing 512 filters. The output at the end of both these stacks will be 7×7×512. The stacks of convolutional layers are followed by three fully connected layers with a flattening layer in between. The first two have 4,096 neurons each, and the last fully connected layer serves as the output layer and has 1,000 neurons corresponding to the 1,000 possible classes for the ImageNet dataset. The output layer is followed by the Softmax activation layer used for categorical classification.

The table below include the details of each layer.

Table II.3. Details of each layer of the VGG-16 architecture[17]

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

b) **Advantages and drawbacks**

VGG-16 can be used for Image Recognition or Classification like disease diagnosis and road or traffic signs recognition from a moving vehicle. With its simple architecture, VGG16 performed really well in image detection. In fact, it was the winner of ImageNet detection challenge in 2014. Due to its depth and number of fully-connected nodes, this architecture is painfully slow to train, and its weights are quite large which makes deploying it a tiresome task.

II.3.7 Inception -V3

Inception-V3 architecture was created by a group of Google developers. It has been presented in a paper called "Rethinking the Inception Architecture for Computer Vision" [18] in 2016. It was an attempt to improve the previous inception versions by avoiding representational bottlenecks, in other words, this architecture contributes in drastically reducing the input dimensions of the next layer and have more efficient computations thanks to factorization methods [18].

a) **Architecture**

This architecture is clearly deeper and more complex than the previously presented architecture. It is composed of a stem module followed by three iterations of the inception-A module and a reduction-A module. The outputs flow through four iterations of inception-B module followed by reduction-B module in addition to an auxiliary classifier, this time the outputs flow through a global average pooling layer followed by two fully connected layers.

The figure below illustrate in detail the Inception-V3 architecture which will be explained later on

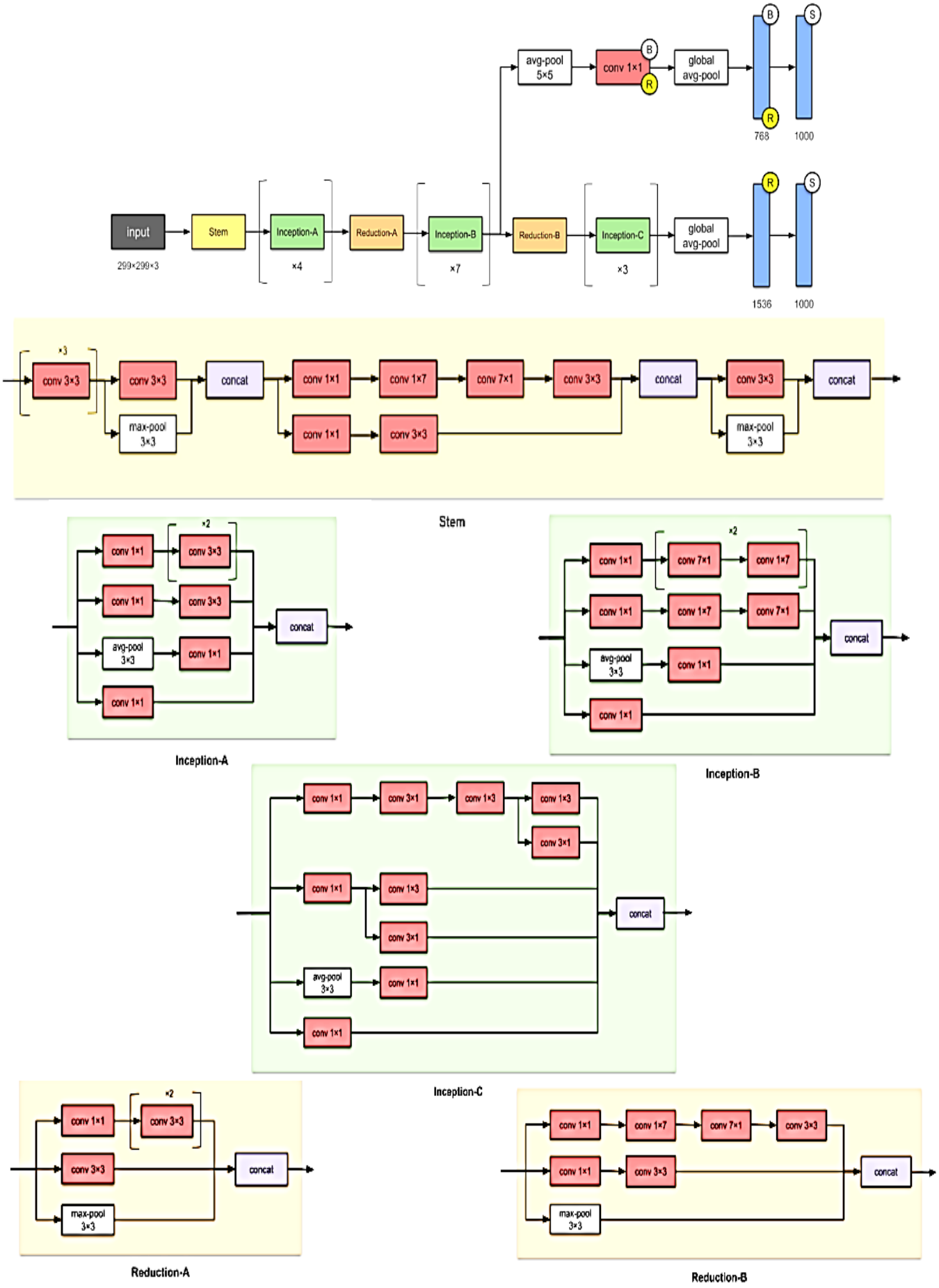


Figure II.11. Detailed Inception-V3 architecture

- **Stem module:** This layer includes a stack of three convolutional filters of 3×3 in size followed by a max pooling layer of 3×3 window size. The activations then flow through another stack of two convolutional filters, the first is 1×1 in size and the last is 3×3 in size followed by pooling layer with 3×3 window size and pass to the next layer [18].

- **Inception-A module:** Inception modules act as "multi-level feature extractors" by computing many convolutions within the same module using different sizes and shapes of kernels. The outputs of these filters are then stacked along the channel dimension before being fed into the next layer which reduces the computational power noticeably. In the Inception-A module, the convolution is performed on the input using two different sizes of symmetrical filters. It includes several 3×3 and 1×1 filters in addition to a max pooling with a window size of 3×3. The factorization of convolutional filters into 1×1 filters offers a generous dimension reduction by reducing the number of parameters and, thus, economizing the computational costs [18].

- **Inception-B module:** To make the model more efficient, several asymmetric convolutions are performed within this module using 7×1 filters followed by 7×1 filters in addition to several 1×1 filters and max pooling with a 3×3 window size [18].

- **Inception-C module:** The asymmetrical convolution is applied once again in this module using 3×3 which are easier to calculate and add to the depth of the architecture.

- **Reduction-A:** In order to reduce the grid size efficiently, the activation dimension of the network filters is expanded instead of using traditional poling techniques. For example, if we have a d×d grid with k filters after reduction, it results in a d/2 × d/2 grid with 2k filters. This is insured by two parallel blocks of convolutional filters and a pooling layer. In the reduction-A module, the convolutional filters used are symmetrical 3×3 and 1×1 filters, added to a 3×3 window of max pooling. The resulting outputs are concatenated and sent to the next layer.

- **Reduction-B:** The same technique is applied using asymmetrical 7×1, 1×7, 3×3 and 1×1 filters in addition to max pooling with 3×3 window size.

- **Utility of Auxiliary classifiers:** These classifiers can be represented by a small CNN inserted between layers during the training step. The loss incurred is added to the main network loss. They are used to improve the convergence of very deep neural networks. Thus, the auxiliary classifiers act as a regularizer in this architecture [18].

The specifications related to each layer are listed in the table below.

Table II.4. Details of each layer of the Inception-V3 architecture

TYPE	PATCH / STRIDE SIZE	INPUT SIZE
Conv	3×3/2	299×299×3
Conv	3×3/1	149×149×32
Conv padded	3×3/1	147×147×32
Pool	3×3/2	147×147×64
Conv	3×3/1	73×73×64
Conv	3×3/2	71×71×80
Conv	3×3/1	35×35×192
3 × Inception	Module 1	35×35×288
5 × Inception	Module 2	17×17×768
2 × Inception	Module 3	8×8×1280
Pool	8 × 8	8 × 8 × 2048
Linear	Logits	1 × 1 × 2048
Softmax	Classifier	1 × 1 × 1000

b) **Advantages and drawbacks**

Inception-V3 is a 48 layer deep CNN, and it has been proved that it is more efficient in terms of the number of parameters generated and memory and hardware resources in when compared to VGG-16, it produces smaller weights than both VGG-16 and the forthcoming ResNet-50. Inception-V3 was adapted for several use cases such as: image classification and image captioning.

II.3.8 ResNet-50

Over the years, stacking more and more layers and building deeper networks helped experts to attain better results in solving complex problems. However, this resulted in undesired side effects. It has been observed that the training becomes more difficult in for deeper networks, since the appearance of a degradation problem which can be expressed as the saturation of accuracy levels that slowly decrease after a certain point. Until 2016, when a group of experts proposed a solution for this matter in their paper entitled “Deep Residual Learning for Image Recognition” [19] where they introduced “ResNet architecture” which stands for Residual Network architecture. The specificity of this architecture is that it includes skip connections which are also known as shortcut connections or residuals while building deeper models. The number 50 indicates the number of weighted layers present in the architecture.

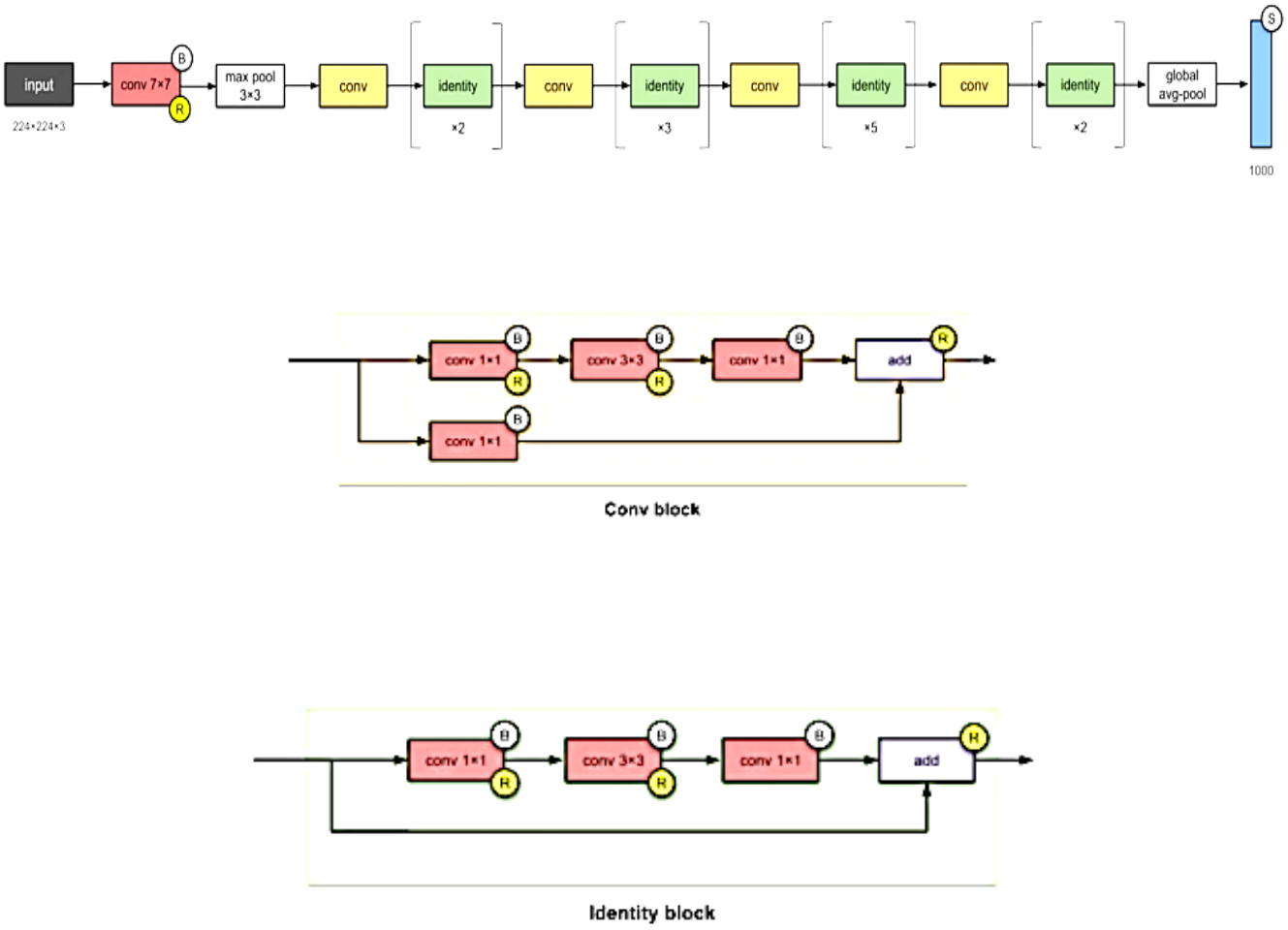


Figure II.12. Detailed architecture of ResNet-50

a) **Architecture**

The figure above shows in detail the architecture of ResNet-50. The first layer includes 7×7 size convolutional filters followed by a max pooling layer with 3×3 window size. In the next step, alternations of convolutional blocks and repeated identity blocks will take place four times, the identity block is repeated two times in the first alternation, three times in the second, five times in the third and two times in the fourth. After that, average pooling is performed followed by a fully connected layer and a softmax function at the end.

- **Convolutional block:** The input of this block flow respectively through 1×1, 3×3 filters, then 1×1 convolutional filters. In parallel, 1×1 kernel filters are applied to the input of this layer and the results are added at the end of the block before being passed to the next layer.

- **Identity block:** This block is similar to the convolutional block, the input flow through the same arrangement of 1×1 and 3×3 as the convolutional block filters except that the input is added directly to the output without performing an additional 1×1 convolution, creating a skip connection.

The ResNet architecture aims to make the inputs and outputs have the same size. If they already have the same size, the input is added to the output layer directly through the skip connections. If they are not the same size, a convolutional block is added to the shortcut path to make their sizes equal, either by padding with extra zero entries to increase the dimensions of the input volume or performing 1×1 convolutions.

In Neural Networks, the error gradient gets computed at the end which gets back propagated through previous layers and multiplied with the multiple terms until the first layer. There can be multiplication with small fractions at different points when there is a long chain which makes the outcome smaller and smaller. As a result, the first layers' coefficient aren't updated since they receive very small gradient and the training becomes slow with no changes in the parameters. ResNet uses identity functions to back propagate the gradients. The gradient would not be affected as it is multiplied by 1.

ResNet creators presented several versions of this architecture increasing the number of layers. The table below includes the details of filters present in each layer.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure II.13. Details of each layer of the ResNet-50 architecture

b) **Advantages and drawbacks**

Even though ResNet is much deeper than VGG-16, the model size is actually substantially smaller due to the usage of global average pooling rather than fully-connected layers. This architecture can be used for image classification, localization and detection also for image description purposes.

II.3.9 Xception

Xception is a deep convolutional neural network architecture introduced in a paper entitled "Deep Learning with Depth wise Separable Convolutions" [20] published in 2017 by Francois Chollet, a researcher at Google and the creator and chief maintainer of the Keras library.

a) **Architecture**

The figure below shows in detail the architecture of Xception.

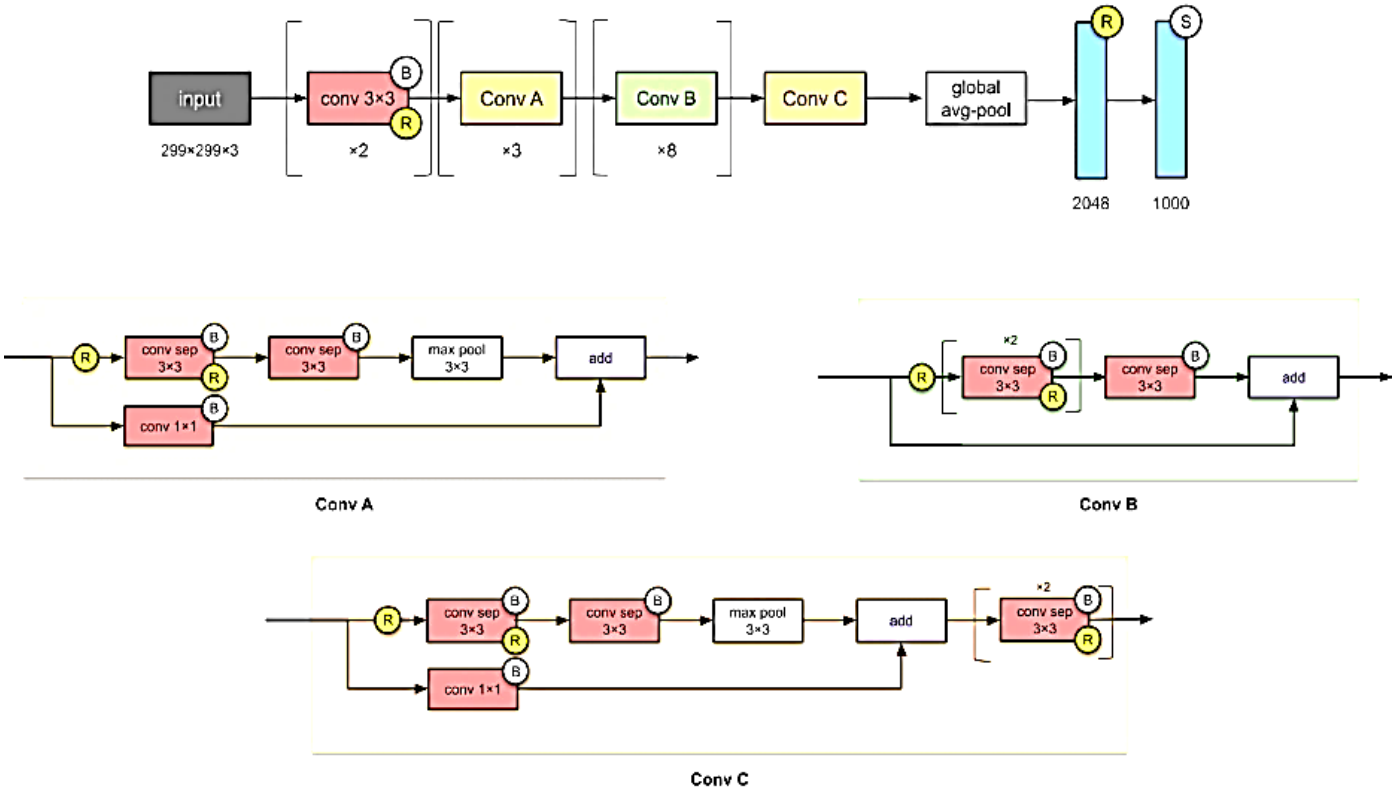


Figure II.14. Detailed Xception architecture

Xception's architecture relies on two main points:

- Modified depthwise separable convolution inspired by the inception architecture.
- Skip connections between convolution blocks inspired by the ResNet architecture.

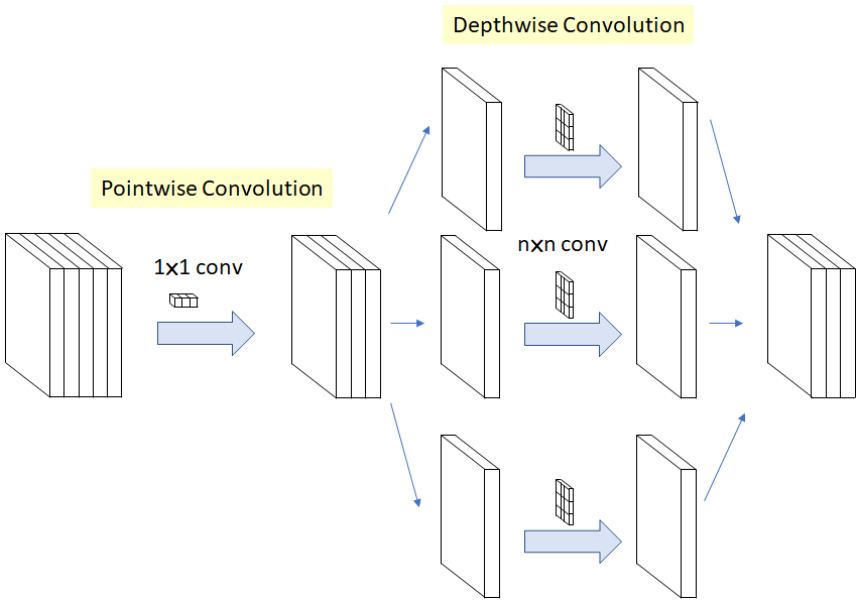


Figure II.15. Modified depth wise separable convolution

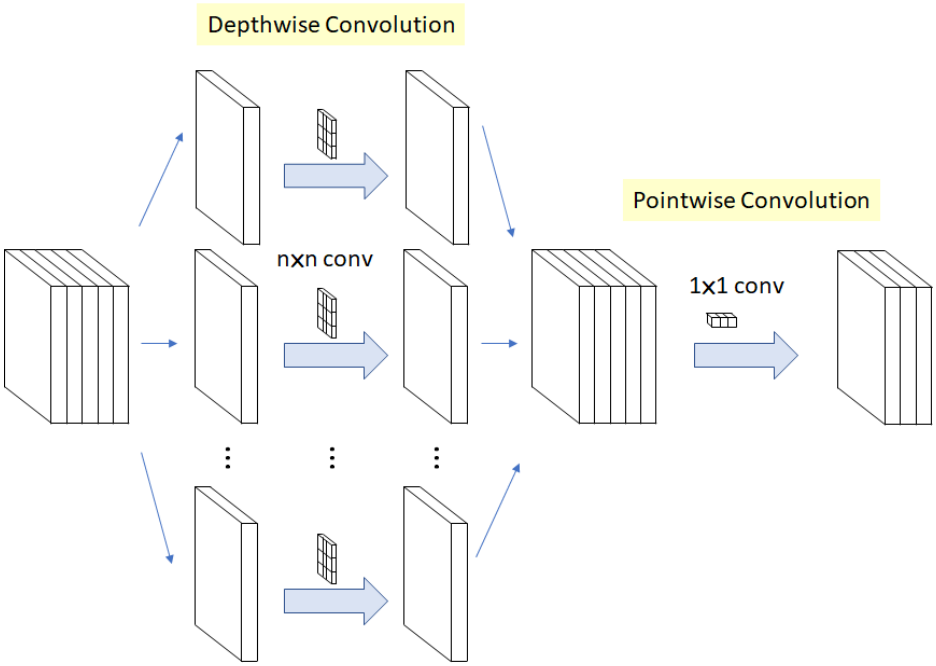


Figure II.16. Conventional depth wise separable convolution

This modification is motivated by the inception modules in Inception-V3 in which 1×1 convolution is applied before the n × n spatial convolutions. The modified depthwise separable convolution is the pointwise convolution insured by 1×1 convolution to change the dimension followed by a depthwise convolution which is the channel-wise n×n spatial convolution**.** Compared to conventional convolution where depth wise convolution is performed before pointwise convolution, convolution doesn't have to be performed across all channels. Thus, the number of connections is reduced [20].

The figure below describes the before mentioned architecture in more detail.

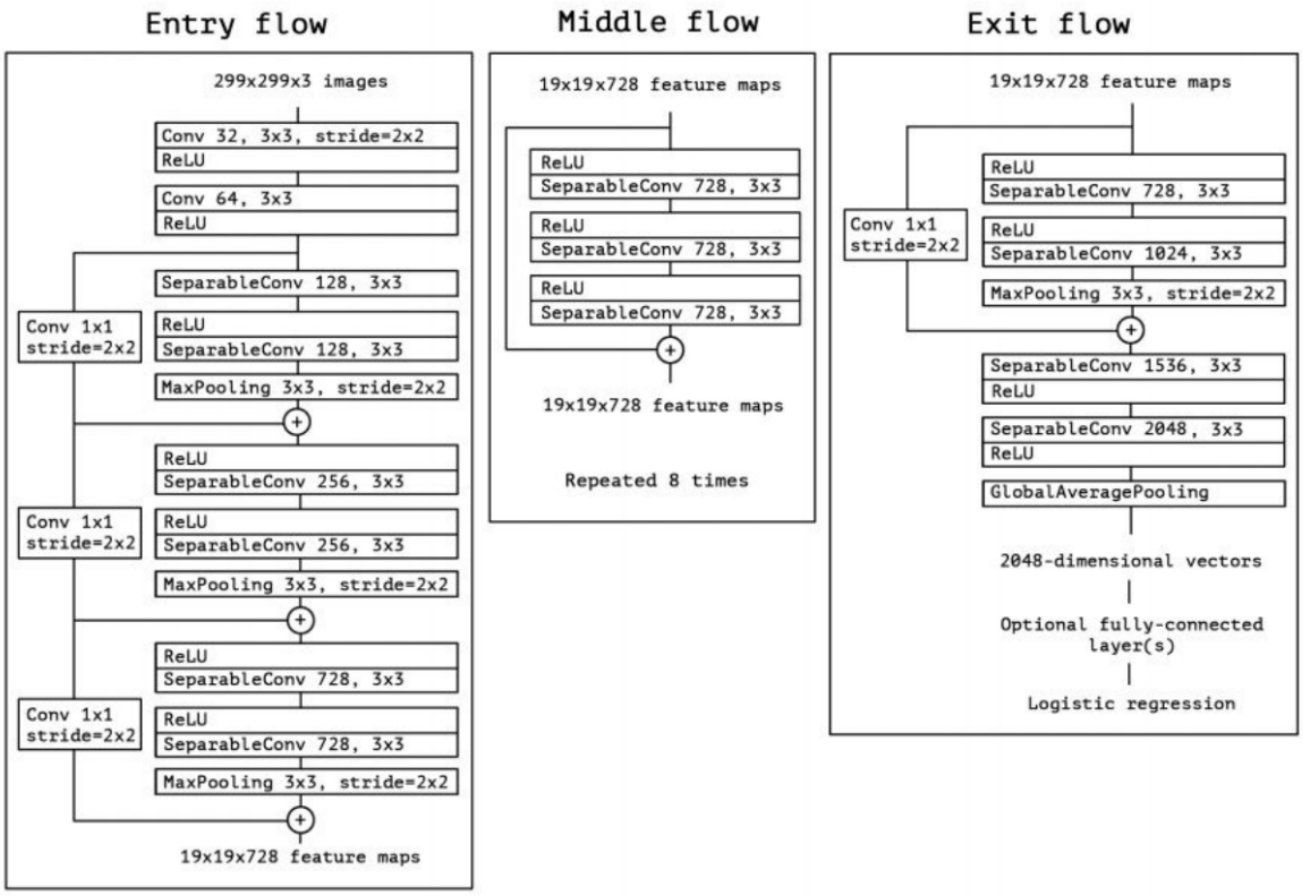


Figure II.17. Details of each layer of the Xception architecture

As shown in the figure above, the data goes first through the entry flow, then through the middle flow which is repeated eight times, to finally pass through the exit flow. All convolution and separable convolution layers are followed by batch normalization. The separable convolutions are named "SeparableConv" in the figure, they are the modified depthwise separable convolution which are treated as Inception modules and placed throughout the whole deep learning architecture. There are residual or skip connections placed for all flows [20].

b) **Advantages and drawbacks**

Xception is the deepest among the previously mentioned architectures including 71 layers, but thanks to the combination of all the good features present in Inception-V3 and ResNet architectures, it could be the deepest and still maintain the lightest weights of all the four architectures presented, less parameters calculated and less material consumption spent. Xception can be used for image classification, localization, detection and finally for image description.

II.3 Recurrent Neural Networks RNN

Convolutional neural networks are designed to process inputs which are independent of each other in one time step without temporal context. However, processing a sequence of interdependent data, such as time series and natural language phrases, the output depend on the previous computations, this requires another type of architectures which have the concept of memory that would store the states or information of previous inputs in order to generate the next output of the sequence. These architectures are called recurrent because they perform the same task for every element of the sequence. RNNs are used in a variety of tasks: transforming a static input into a sequence as in image description, processing sequences into a static output as in video labelling, or transforming sequences into sequences as in automatic translation [21]. The Hopfield Network, which was introduced in 1982 by J.J. Hopfield, can be considered as the first network with recurrent connections.

There are different variations of RNNs that are being applied practically in machine learning problems, we particularly cite: Bidirectional recurrent neural networks (BRNN), Gated Recurrent Units (GRU) and Long Short Term Memory (LSTM).

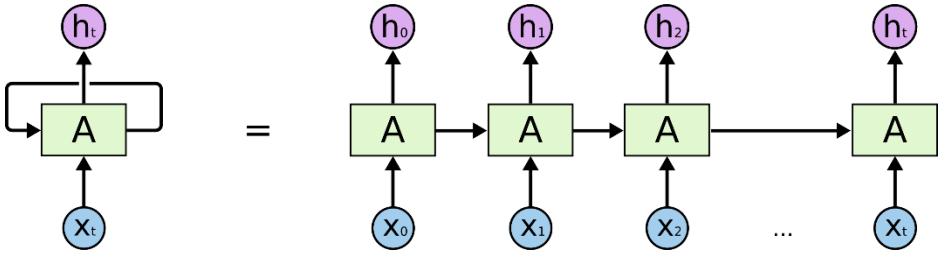
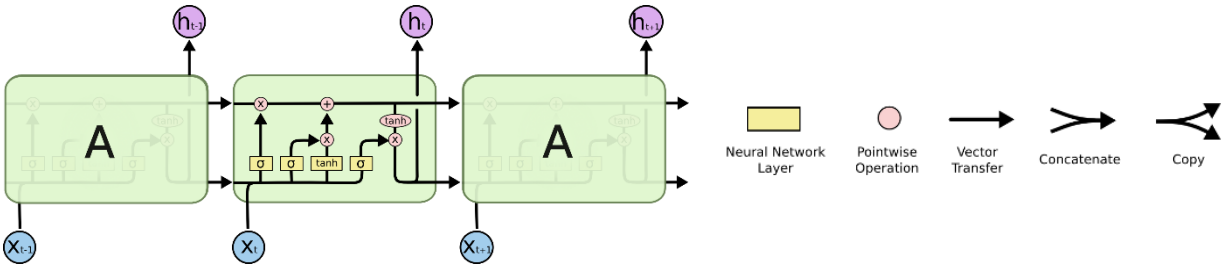


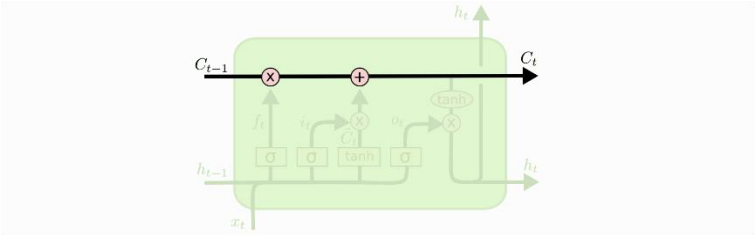
Figure II.18. Unrolled recurrent neural network architecture

II.3.1 Long Short Term Memory (LSTM)

LSTMs are a type of RNN networks, they were designed to solve the long-term dependency problem and remembering information for long periods of time. They have a chain-like structure and the repeating module has four different layers [22].



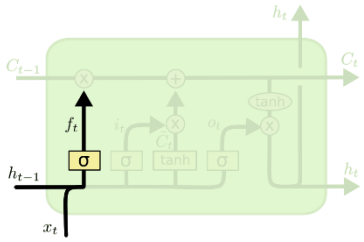
The horizontal line running through the top of the diagram is the cell state, it runs straight down the entire chain, with only some minor linear interactions [22].



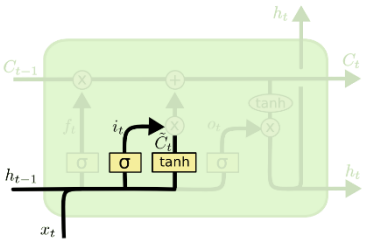
The LSTM have the ability to remove or add information to the cell state, carefully regulated by structures called gates [22].

- **Forget Gate:** conditionally decides what information to throw away from the block.
- **Input Gate:** conditionally decides which values from the input to update the memory state.
- **Output Gate:** conditionally decides what to output based on input and the memory of the block.

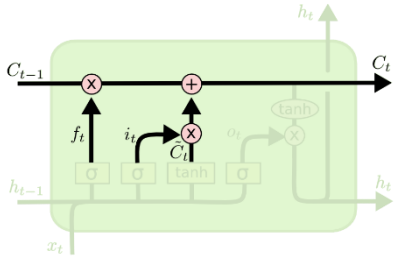
Each unit is like a mini-state machine where the gates of the units have weights that are learned during the training procedure.



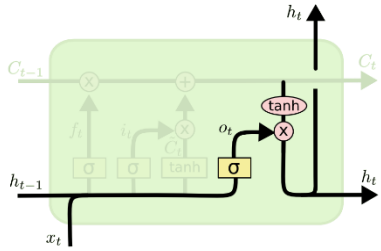
The first step is to decide what information to keep and what to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It takes h_{t-1} and x_t as inputs and outputs f_t which is a number between 0 and 1 for each number in the old cell state C_{t-1}. 1 represents information to keep while a 0 represents information to throw away [22].



The next step is to decide what new information to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" that decides which values are to be updated. Next, a tanh layer that creates a vector of new candidate values $C_t^{\sim}$ that could be added to the state. In the next step, these two vectors are combined to create an update to the state [22].



The old cell state C_{t-1} is multiplied by f_t to throw away the unnecessary information decided in the first step and the result is added to $i_t * C\sim_t$. This is the new candidate values, scaled by how each state value is to be updated [22].



The sigmoid layer is applied first in order to decide what parts of the cell state are going to be the outputs. Next they go through tanh layer to obtain values between −1 and 1. Finally, they are multiplied by the output of the sigmoid gate to output only the parts that were decided [22].

a) Advantages and drawbacks

LSTMs became popular for their various fields of application such as: Language modelling, text generation, speech and handwriting recognition, translation and more. On one hand, they are able to solve the problem of vanishing gradients, on the other hand, the structure of the cell become quite complex with the additional features which results in longer time to train a high memory-bandwidth.

Our goal in this work is to build a model that is able to generate a description from a given image. For this matter, we will be using a deep learning model based on convolutional neural networks (CNN) and recurrent neural networks (RNN). To insure that the generated words are as close to human language as possible, we will proceed to evaluate these descriptions using various natural language evaluation metrics.

II.4 Global Architecture

In order to create an image description system, an encoder-decoder based architecture will be used.

- **Encoder:** A network model that reads the image input and encodes the content into a fixed-length vector. This task is insured by convolutional neural networks.
- **Decoder:** A network model that reads the encoded image and generates the textual description output. This task is insured by recurrent neural networks.

There are many ways to combine the CNN and RNN network in order to create an image description model [23]. We will be using the merge architecture described by Marc Tanti, et al. in their papers [23 - 24] since it has been proven efficient solving image description problems.

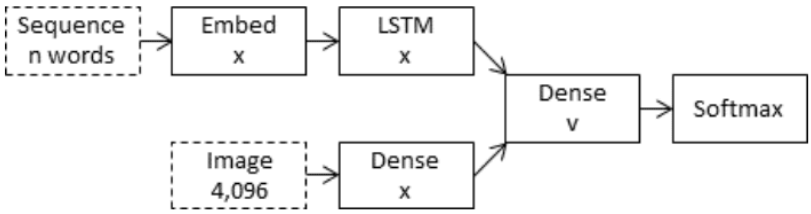


Figure II.19. Encoder-decoder merge architecture

The proposed model is trained with a set of images and their corresponding descriptions called “the training set”. These sentences are written by people and refer to a particular but unknown region of the image.

First, The CNN has to learn the relation between a sentence snippets and the visual image regions by extracting features from the images and encoding them into a fixed-length vector that can be stored in a file for later use. The last two fully connected layers destined for image classification are deleted to obtain the features vector. Afterwards, the RNN is fed by the feature vector and the description sentences to discern which feature best represents what part of the

text. It gets trained with the previous outputs and learn how to generate sentences. The model is trained to combine a word and the previous context to predict a new word [25].

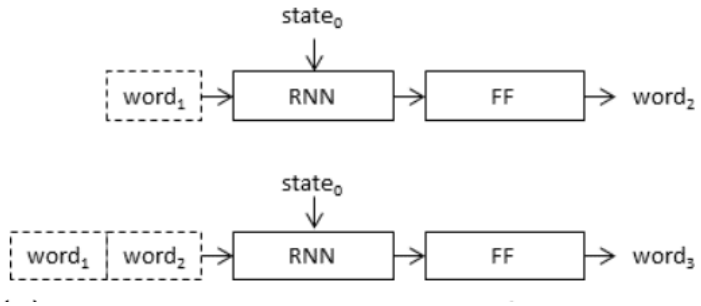


Figure II.20. Process of sequence generation

As it is shown by the figure above, the first recurrent neuron outputs "word$_2$". The previous outputs of neurons will form the input of the next until the end of the sequence.

Repeating this procedure on every image of the training set, the model learns how to associate image assets with their proper text description and generates a meaningful result.

The number of CNN and RNN based architectures increases each year, introducing deeper and more complex architectures. In the present project, we suggest studying four CNN architectures which are: Inception-V3, Xception, VGG-16 and ResNet-50 and one type of RNN networks which is Long Short Term Memory networks.

The CNN architectures previously presented were chosen because they are commonly used and available on libraries, and for their success in "The ImageNet Large Scale Visual Recognition Challenge". ImageNet is a benchmark dataset in object classification and detection with hundreds of object categories and millions of images. The challenge takes place annually from 2010 to present, attracting participation from more than fifty institutions.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
VGG16	528 MB	0.713	0.901	138,357,544	23
InceptionV3	92 MB	0.779	0.937	23,851,784	159
ResNet50	98 MB	0.749	0.921	25,636,712	-
Xception	88 MB	0.790	0.945	22,910,480	126
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
ResNeXt50	96 MB	0.777	0.938	25,097,128	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

Depth refers to the topological depth of the network. This includes activation layers, batch normalization layers etc.

Figure II.21. Results of the four architectures performance on ImageNet validation dataset

II.5 Implementation

II.5.1 Presentation of tools

The Experimental settings and software used in our experiments are shown below:

II.5.1.1 Hardware specifications

- CPU: Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz.
- RAM: 12 GB DDR4.

II.5.1.2 Software specifications

- **TensorFlow:** It is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that simplifies the process of building ML powered applications such as training deep neural networks. It was developed by Google and firstly released in 2015. It can be used in a wide variety of programming languages such as Python, JavaScript, and C++. We used the TensorFlow 2.8.0 version [26].
- **Keras**: it is an open-source high-level Neural Network library written in Python. It can be described as an interface for TensorFlow. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly and extensible, to simplify the experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination. It has been designed to work with Python, quickly, and easily. We used the Keras 2.8.0 version [27].
- **JupyterLab:** It is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning [28].
- **Google Colaboratory:** Colab is a free Jupyter notebook environment that runs entirely in the cloud, it grants the users access to GPUs, combine executable code and rich text in a single document, along with images, HTML, LaTeX and more with zero configuration required and for free. It also enable the user to save and access their drive files [29].
- **Python programing language:** Python is a high-level, general-purpose programming language. It is being used mainly in web development and Machine Learning applications. We will be using the 3.10 python version.

II.5.2 Steps of the experimentation

The first step is training our models on a dataset of images and their corresponding descriptions, this step is evaluated by monitoring loss and accuracy values, and the model with the best results is saved. The second step is testing, the models generate descriptions from new images which is a test set. The generated descriptions are then evaluated using the different evaluation metrics.

II.5.2.1 Step 1: Training

Before presenting the learning process, we will define the necessary vocabulary that will be useful to understand this step:

- **Epoch:** it is one forward and backward pass of all the training samples through the entire network.
- **Batch size:** it is the number of training samples that are fed into the network before updating its weights.
- **Step:** it is to process one batch of data.
- **Learning rate:** it is a hyper-parameter that controls by how much we are adjusting the weights of our network with respect the loss gradient.
- **Optimizer:** it is an algorithm that is used to minimize an error function by finding the partial derivatives of this function with respect to the weights. Those derivatives are then used by an algorithm that can iteratively minimize a given function then it adjusts the weights up or down, depending on which decreases the error, insure the training with minimum resources and provide efficient results in lesser time. There are many optimizers such as: Stochastic Gradient Descent (SGD), Momentum and Adam, in our experiments we will be using the Adam optimizer since it has proven itself to be the most efficient compared to the other optimizers [30].

The expression below demonstrates the relation between the different parameters:

$$W_t = W_{t-1} - A \times \left(\frac{\partial\, Error}{\partial\, W_{t-1}}\right) \qquad \text{(II.6)}$$

W_t: New weight
W_{t-1}: Old weight
A: learning rate

Training neural networks relies on two major steps which are forward-propagation and backward-propagation. Forward-propagation is the process of passing the inputs throughout the network and get the final output of the model then check if this output is correct or incorrect by calculating the error. Back-propagation is going backwards through the neural network and adjust the weights up or down using an optimization algorithm that can iteratively minimize the error.

The image below illustrates the concept of forward and back-propagation in a neural network:

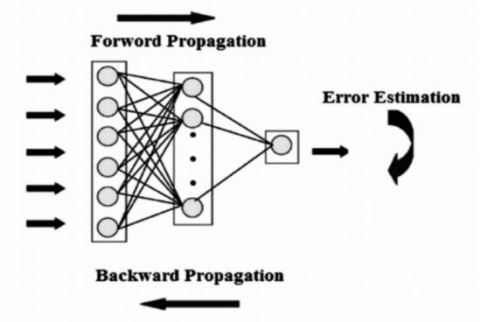


Figure II.22. Forward and back-propagation in a neural network

- The model receives image features, a sequence of words and the output word. To avoid reaching RAM and GPU limits, we create a data generator for the training set to generate sequences of the size of the batch progressively for one image at a time.
- Finally, we estimate the training accuracy and loss to supervise the model's learning.

The performance of our models is evaluated by the values of two main functions which are loss and accuracy:

- **Accuracy:** it is the count of correct predictions divided by the total number of predictions, it can be expressed as a percentage.
- **Loss function:** In neural networks we seek to minimize the error using an optimization algorithm, the function used to evaluate the set of weights is referred to as the objective function which is referred to as a loss function. This function aims to distill all aspects of the model down into a single number referred to as loss in such a way that improvement in that number are a sign of a better model. This number is a summation of the errors made for each sample in the training set. Therefore, the goal is to minimize this value. Mean Squared Error (MSE) is a loss function used in regression problems where real-value quantity is to be predicted. Cross-entropy is another loss function used in multi-class classification problems which we will be using to evaluate our training process.

II.5.2.2 Step 2: testing

To measure the quality of the generated descriptions by the different models, we will be using different standard evaluation metrics that will be presented below.

a) BLEU (Bilingual Evaluation Understudy)

The BLEU [31] metric is the most popular metric for evaluating machine translation system. It is used to evaluate various NLP tasks such as Machine-translation, text generation, paraphrase generation, and text summarization. For our evaluations we will be using BLEU-1, BLEU-2, BLEU-3 and BLEU-4.

Precision is approximated by Modified n-gram precision. An n-gram precision is the fraction of n-grams in the candidate text which are present in any of the reference texts, the candidate's n-grams are matched as many times as they are present in any of the reference texts. To include all the n-gram precision scores in our final precision, we take their logarithmic averaging to represent all values fairly.

$$\boldsymbol{P = e^{\sum_{n=1}^{N} \frac{1}{n} log\, p_n}} \quad \textbf{(II.7)}$$

Recall is approximated by best match length, a longer candidate text is more likely to contain a larger fraction of some reference than a shorter candidate. This is done by adding a multiplicative factor BP with the modified n-gram precision as follows with: c is total length of candidate translation corpus and r is average length of all references.

$$BP = \begin{cases} 1, & \text{if } c > r \\ \exp(1 - \frac{r}{c}), & \text{otherwise} \end{cases} \quad \textbf{(II.8)}$$

b) ROUGE (Recall Oriented Understudy for Gisting Evaluation)

ROUGE [32] is mostly used for summary evaluation and it comes in many types from which we mention: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S Depending on the feature used for calculating recall. For our evaluations we will be using ROUGE-L.

ROUGE-L is based on the longest common subsequence (LCS). It uses an F-score which is the harmonic mean of precision and recall values. Assuming the candidate is of length CSL and the reference is of length RSL, precision and recall are calculated as follows:

$$\boldsymbol{P = \frac{lSC(CS, RS)}{CSL}} \quad \textbf{(II.9)} \qquad \boldsymbol{R = \frac{lSC(CS, RS)}{RSL}} \quad \textbf{(II.10)} \qquad \boldsymbol{F = 2 \times \frac{RP}{R + P}} \quad \textbf{(II.11)}$$

c) METEOR (Metric for Evaluation for Translation with Explicit Ordering)

METEOR [33] is used for evaluating machine translation system and it claims to have better correlation with human judgment and allows synonyms and stemmed words to be matched with a reference word. It replaces the precision and recall calculated by the BLEU metric by a weighted F-score based on mapping unigrams and a penalty function for incorrect word order.

Weighted F-score is calculated by finding the largest subset of mappings that can form an alignment between the candidate and reference and we assume that the number of mapped unigrams between the two is τ. Precision, recall, the F score and the penalty function are calculate as follows:

$$\boldsymbol{P = \frac{\tau}{\gamma}} \quad \textbf{(II.12)}$$

$$\boldsymbol{R = \frac{\tau}{\beta}} \quad \textbf{(II.13)}$$

$$\boldsymbol{F = \frac{10 \times PR}{R + 9P}} \quad \textbf{(II.14)}$$

$$\boldsymbol{Penalty = 0.5 \times \left(\frac{\gamma}{\tau}\right)^3} \quad \textbf{(II.15)}$$

Where γ is the number of unigrams in a candidate sentence, β is the number of unigrams in reference sentence and τ is the number of unigrams in the candidate sentence also found in reference sentence.

d) CIDEr (Consensus-based Image Description Evaluation)

CIDEr is a metric for evaluating image descriptions that uses human consensus. All words in the sentences (both candidate and references) are first mapped to their stem or root forms. That is, "fishes", "fishing" and "fished" all get reduced to "fish." We represent each sentence using the set of n-grams present in it. An n-gram ω_k is a set of one or more ordered words. A measure of consensus would encode how often n-grams in the candidate sentence are present in the reference sentences. Finally, n-grams that commonly occur across all images in the dataset should be given lower weight, since they are likely to be less informative. To encode this intuition, we perform a Term Frequency Inverse Document Frequency (TF-IDF) weighting for each n-gram [34].

II.6 Conclusion

This chapter aimed to familiarize the reader with the necessary theoretical concepts and vocabulary used in this project. We began by introducing the conventional and recurrent neural networks together with a detailed explanation of each one of the architectures that will be used later on. Following that we presented the general architecture of our model and finally the tools that will be used to carry the different experiments.

In the next chapter, we will present the results of training and testing the different architectures in different experimental settings.

Chapter III. Experimentation and results

III.1. Introduction

Many researches and studies have been carried in the framework of developing and improving automated image description models resulted in creating numerous CNN and RNN architectures with different ways to combine them in order to create a model capable of generating a description from a given image automatically.

This chapter will start with a presentation of the dataset that will be used throughout our experiments. Next, the training and testing processes and their results of each experiment will be presented, discussed and concluded by samples of generated descriptions.

III.2. Dataset presentation

There are many datasets available to download online and for free, they come in different sizes and offer a variety of themes, and each is suitable for adifferent use case.

For our various studies we will be using the Flickr8K dataset to train and test our image description models, since it is relatively small (about 1 Gigabytes in size) so it can be downloaded and used to carry our experiments easily on our workstation.

This dataset includes a collection of 8,092 colored and realistic images of different sizes stored in a JPG format that capture people, animal, locations and scenes from every day's life. Each image is associated with five different captions, each of them focuses on different aspects of the described scene, or use different linguistic constructions [35].

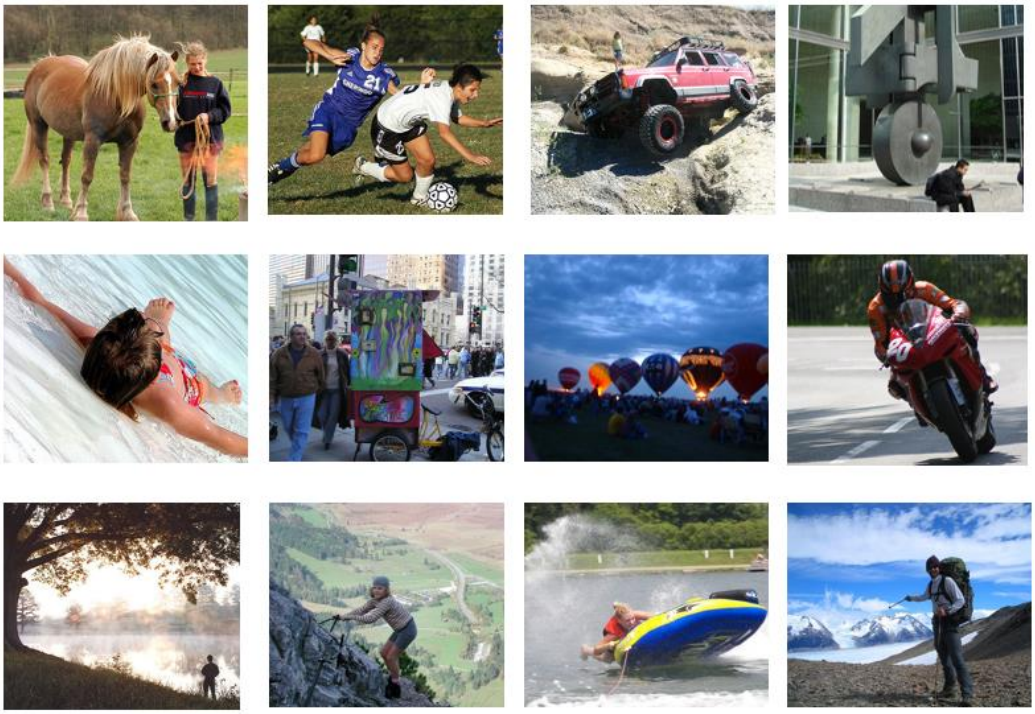


Figure III.1. Flickr 8K dataset overview



A boy wearing a red shirt and jeans is doing a flip on his bike.
A person flipping a bicycle upside down.
A person flips on a bike.
A person in a red shirt doing tricks on a bicycle.
A person is show upside down on his bicycle over a large field.



A hiker makes his way downhill
A man carrying sticks descends a rocky hill
A man is hiking on a mountaintop on a cloudy day
Hiker walking along high rocks in front of a blue sky with puffy white clouds
The man in a blue sweater is climbing down the rocks

Figure III.2. Images from the Flickr 8K dataset with their attributed descriptions

III.3. Study of different encoder architectures

To realize a model that is able to automatically generate descriptions of a given image, we will use four encoder-decoder architectures based on CNN for the encoder part and RNN for the decoder part. In order to compare the performances of CNN models, we will use four different CNN networks and LSTM as a type of RNN. The architectures are:

- Inception-V3 + LSTM
- Xception + LSTM
- VGG-16 + LSTM
- ResNet-50 + LSTM

Finally, in order to evaluate the performances of the these architectures during the training, we will be monitoring loss and accuracy values and use natural language evaluation metrics to evaluate the quality of the generated descriptions. The natural language evaluation metrics that will be used are: BLEU, ROUGE, METEOR and CIDEr.

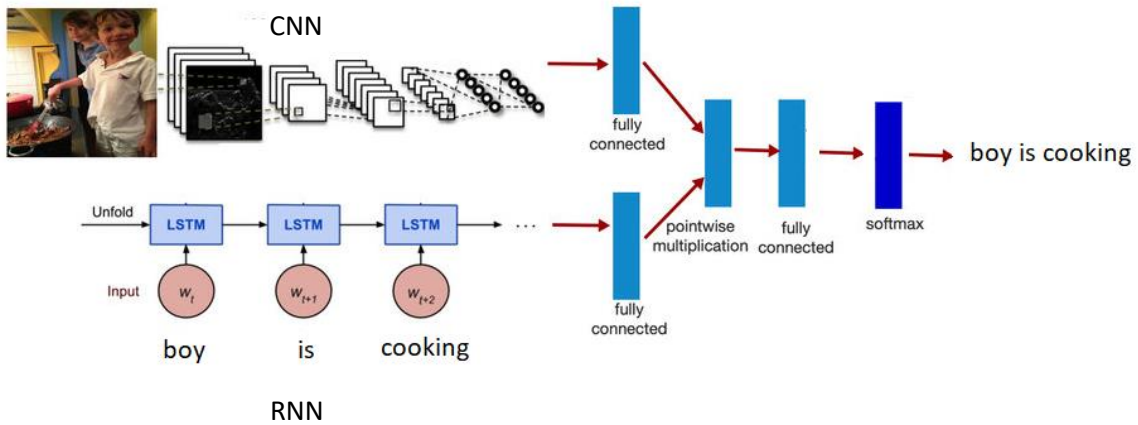


Figure III.3. An overview on the global architecture

Before starting the listing of the experiments held, we will present a flowchart summarizing the reflection principle that yielded to the definition of the overall experimental procedure.

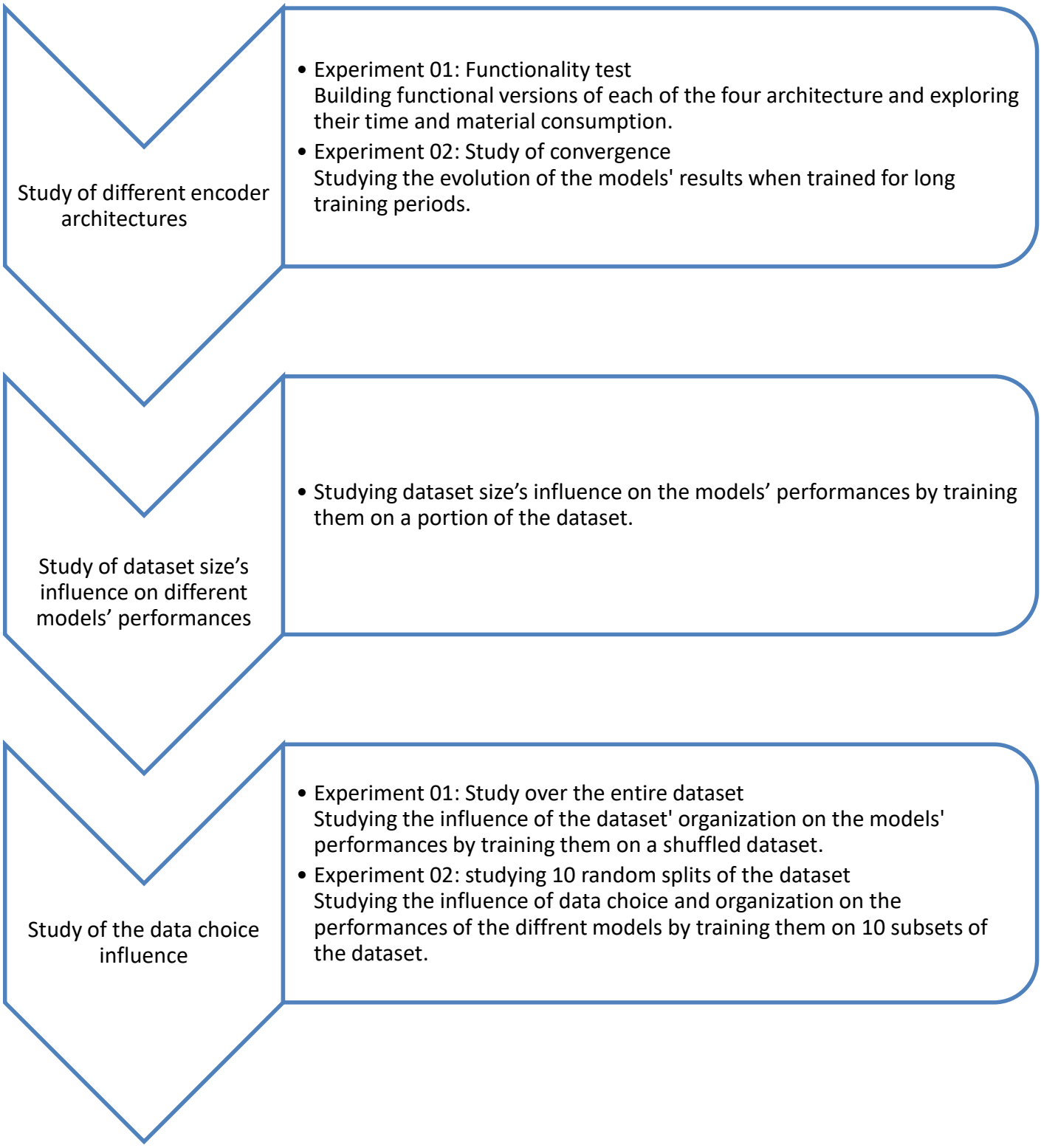


Figure III.4. Flowchart of the experimental procedure

III.3.1 Experiment 01: Functionality test

For our first experiment, we launch the training for a small number of epochs to test the functionality of our architectures and to gain a global view on time consumption and the evolution rate of different results.

III.3.1.1 Training

a) **Settings**

The table below indicates the values we chose for the main parameters.

Table III.1. Summary of training settings

Optimizer	Adam
Batch size	32 (default value)
Learning rate	0,001 (default value)
Loss function	Categorical cross entropy
Metrics	Accuracy
Training set size	6000 images
Training set's vocabulary size	7577
Description length	32 characters
Number of epochs	10

b) **Presentation of training results**

The training step will be evaluated in terms of time, loss and accuracy as presented below.

The table below shows an estimation of time spent by each model to complete the training.

Table III.2. Summary of time consumption on training by the four models

Model	Time per epoch	Total time spent on training
Xception	About 30 minutes	Around 5 hours
VGG-16	About 1h 30 minutes	Around 15 hours
Inception-V3	**About 20 minutes**	**Around 4 hours**
ResNet-50	About 35 minutes	Around 6 hours

Discussion

The previous table recorded different execution times for the different models considered. These results show that Inception-V3 was the fastest model followed by Xception and ResNet-50. The VGG-16 ranks last consuming a much longer period to complete ten epochs of training.

The graphs below shows the evolution of loss and accuracy values respectively over the ten epochs of training.

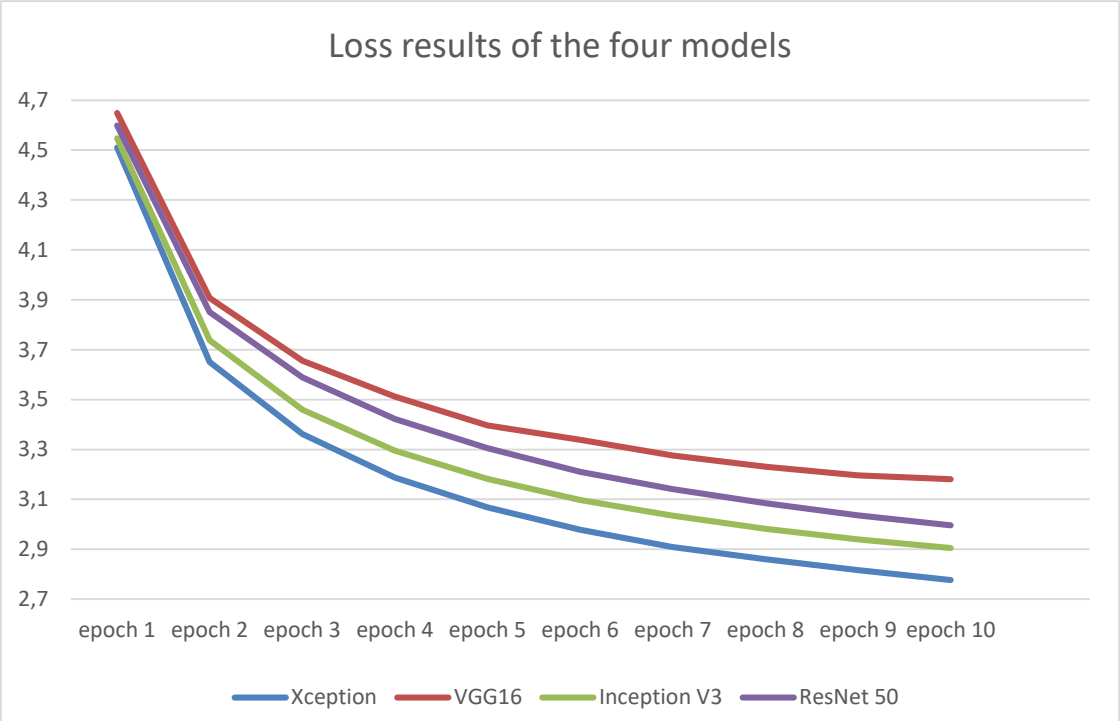


Figure III.5. Loss results of the four studied models trained for 10 epochs

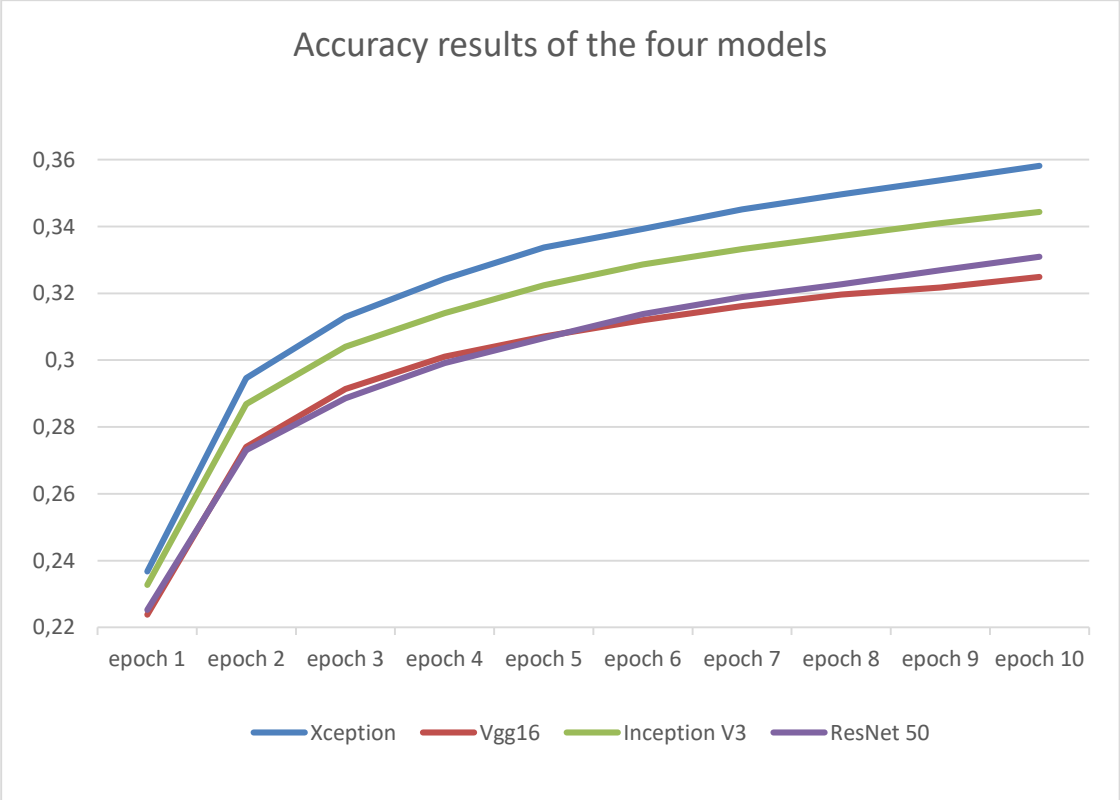


Figure III.6. Accuracy results of the four studied models trained for 10 epochs

The table above summarizes the best accuracy and loss results achieved by each model by the end of the training, results of the best model are marked in bold.

Table III.3. Summary of best loss and accuracy results for each model

Model	**Xception**	VGG-16	Inception-V3	ResNet-50
Loss	**2,777**	3,1806	2,905	2,9955
Accuracy	**0,3582**	0,3249	0,3444	0,331

Discussion

Graphs above show a fast decrease in loss results and increase in accuracy results for the four models in the first three epochs (decrease by more than 0.5 per epoch in loss and increase by more than 0.03 per epoch in accuracy values), the values then improve following a slower rate until the end of the training period (decrease by less than 0.1 per epoch in loss and increase by less than 0.01 per epoch in accuracy values).

The estimation of the loss revealed results that have been improved along with the training. The best results that have been collected at the tenth epochs fluctuate between 2.777 and 3.1806 for the loss values and between 0.3582 and 3.1806, this enable us to rank the models' performance decreasingly as it follows: Xception, Inception-V3, ResNet-50 and finally VGG.

III.3.1.2 Generated descriptions

In the table below, we present the descriptions generated by the four models after ten epochs of training based on an image randomly picked beforehand from the dataset along with its' five attributed descriptions that we will consider as references, in order to judge the validity of the generated ones. Each common word between the generated or one of reference descriptions will be marked in bold and considered correct. The quantitative study is made by the different metrics of the NLP while the qualitative is made on the basis of the personal appreciation of the similarity between generated sentences and references but especially between generated sentences and image.

Image	***Model***	***Generated descriptions***
	Xception	**Brown dog is** running **through the** grass
	Inception-V3	**Brown dog is** running **through the water**
Reference descriptions	VGG-16	**Dog is** running **through the water**
A brown dog is swimming through water with an orange ball in its mouth		
A dark brow dog holds an orange ball in his mouth while walking through water		
A dog wading in the water with a ball in his mouth	ResNet-50	Two dogs are running **through the** grass
A wet, brownish-red dog with an orange ball in his mouth is swimming in some water		
Brown dog in water, with orange and BLEU ball in mouth		

Figure III.7. Examples of generated descriptions by the four models

Image	*Model*	*Generated descriptions*
	Xception	Two men **are** playing **with** red **ball**
	Inception-V3	Two **people are** sitting on the edge **of** dock
Reference descriptions	VGG-16	Man **in** red shirt **is** riding bike on the beach
A group of people are jumping into the air with their arms held high		
A group of people in a desert jumping , one throwing a ball		
A group of six people have jumped into the air in the middle of nowhere and one is catching a ball	ResNet-50	Man **in** red shirt **is** riding bike on the beach
Six people are jumping in the air , one of them throwing a ball		
Six people jump for ball		

Figure III.8. Examples of generated descriptions by the four models

Discussion

- the four models could identify the presence of the main element of the picture which is the dog in the first example, the Inception-V3 and VGG-16 models could identify the background which is water while the other two models mistaken it for grass
- the Inception-V3 produced the best description of the all four models since it generated the longest descriptions with only one word not matching any of the references', ResNet-50 produced the worst description amongst the our since it identified two dogs and grass as a background
- In the second example, the quality of descriptions decreased noticeably, the Xception and Inception V3 model could identify the main subject of the image which is the presence of more than one man while the other models couldn't. Xception could also identify the ball and the relation between the different subjects which is playing with the ball.
- These results were achieved after only ten epochs of training and can be improved if the models were trained enough

III.3.1.3 Testing

In order to evaluate the descriptions generated by the four models based on the 1000 pictures dedicated for testing, we use the four most commonly used natural language metrics which are: BLEU, ROUGE – L, METEOR and CIDEr

The figure below shows the results of metric evaluation for the four models

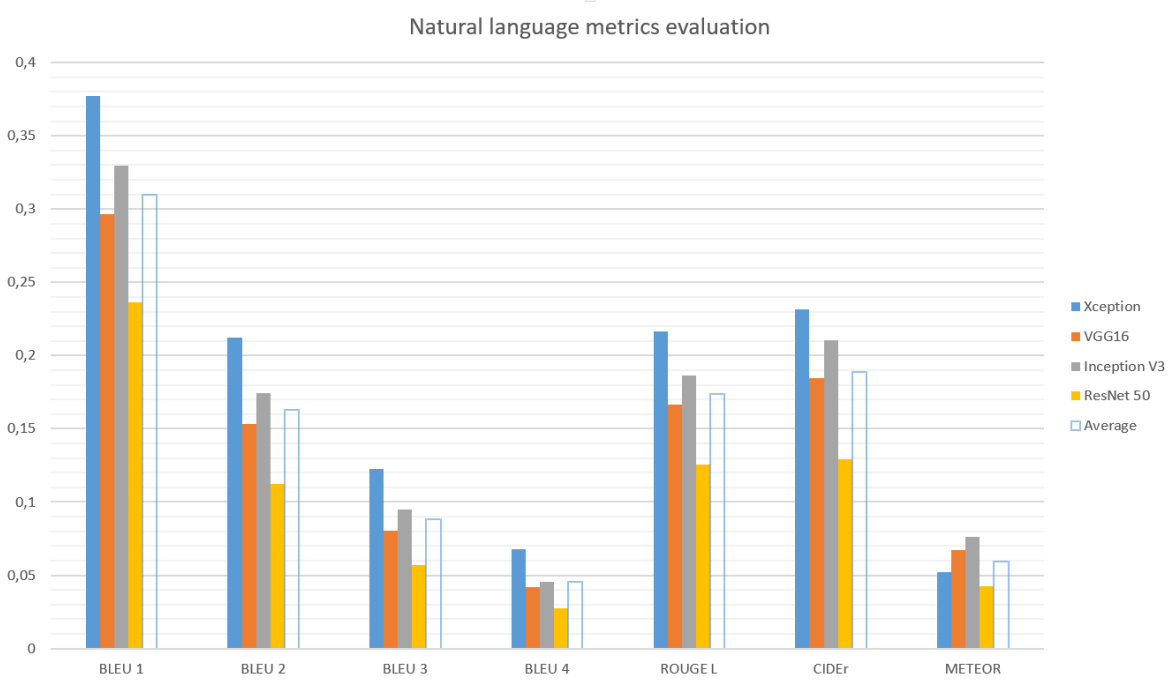


Figure III.9. Natural language metrics evaluation for the four models

The table below summarizes the natural language metrics' results for each model, the best results are highlighted in bold.

Table III.4. Natural language evaluation metrics' results for each model

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	ROUGE L	CIDEr	METEOR
Xception	**0,377294**	**0,2123175**	**0,122415**	**0,067639**	**0,216328**	**0,231274**	0,052406
VGG-16	0,296432	0,153234	0,080434	0,042198	0,166696	0,184815	0,066922
Inception-V3	0,329641	0,174141	0,094623	0,045752	0,186669	0,210589	**0,076057**
ResNet-50	0,23615	0,112578	0,056738	0,027297	0,125333	0,129085	0,042607
Average	0,30987925	0,16306763	0,0885525	0,0457215	0,1737565	0,18894075	0,059498

Discussion

The Xception model scored the best results in most of the natural language evaluation metrics followed by Inception-V3 then the VGG-16 and finally the ResNet-50.

III.3.1.4 Conclusion

In this first experiment, we trained four models with different encoders which are Xception, Inception-V3, VGG-16 and ResNet-50 on the Flickr 8k training set of 6000 images for 10 epochs and evaluated the training process in terms of time consumption, loss and accuracy; then we evaluated the generated descriptions based 1000 images of the testing set by the four models using four natural language evaluation metrics which are BLEU, ROUGE – L, METEOR and CIDEr, finally we compared the descriptions generated by each model based on the same picture.

Inception V3 and Xception models are the most efficient amongst the four studied models since they achieved the best results in terms of loss, accuracy and natural language metrics while consuming the shortest periods of time to complete the training and produced accurate descriptions.

The training process took around 30 hours be completed also was material consuming since it was done locally on the workstation and the inability to train more than one model at a time due to memory limitations.

In the light of these results, we choose to study the convergence of Xception and Inception-V3 models as a next step.

III.3.2 Experiment 02: Study of convergence

In this experiment, we will study the convergence of Inception-V3 and Xception models on the same dataset and compare their results to the ones achieved in the previous study. In other words, we will train the models until they reach their stability by showing no improvement at all or an insignificant change in the results.

III.3.2.1 Training

a) Settings

We will keep the same settings from the previous experiment and only change the number of epochs to 50, the table below indicates the values we chose for each parameter.

Table III.5. Summary of training settings

Optimizer	Adam
Batch size	32 (default value)
Learning rate	0,001 (default value)
Loss function	Categorical cross entropy
Metrics	Accuracy
Training set size	6000 images
Training set's vocabulary size	7577
Description length	32 characters
Number of epochs	50

b) Presentation of training results

The table below shows an estimation of time spent by Xception and Inception-V3 models to complete the training

Table III.6. Summary of time consumption on training by Xception and Inception-V3 models

Model	Time per epoch	Total time spent on training
Xception	About 30 minutes	Around 25 hours
Inception-V3	**About 20 minutes**	**Around 17 hours**

Discussion

Reaching convergence with both models took a very long time (almost two days in total) especially that we weren't able to train them simultaneously due to memory limitations. The Inception-V3 model took a shorter period of time to reach the convergence comparing to the Xception model.

The graphs below shows the evolution of loss and accuracy values respectively for the Xception and Inception-V3 models over 50 epochs.

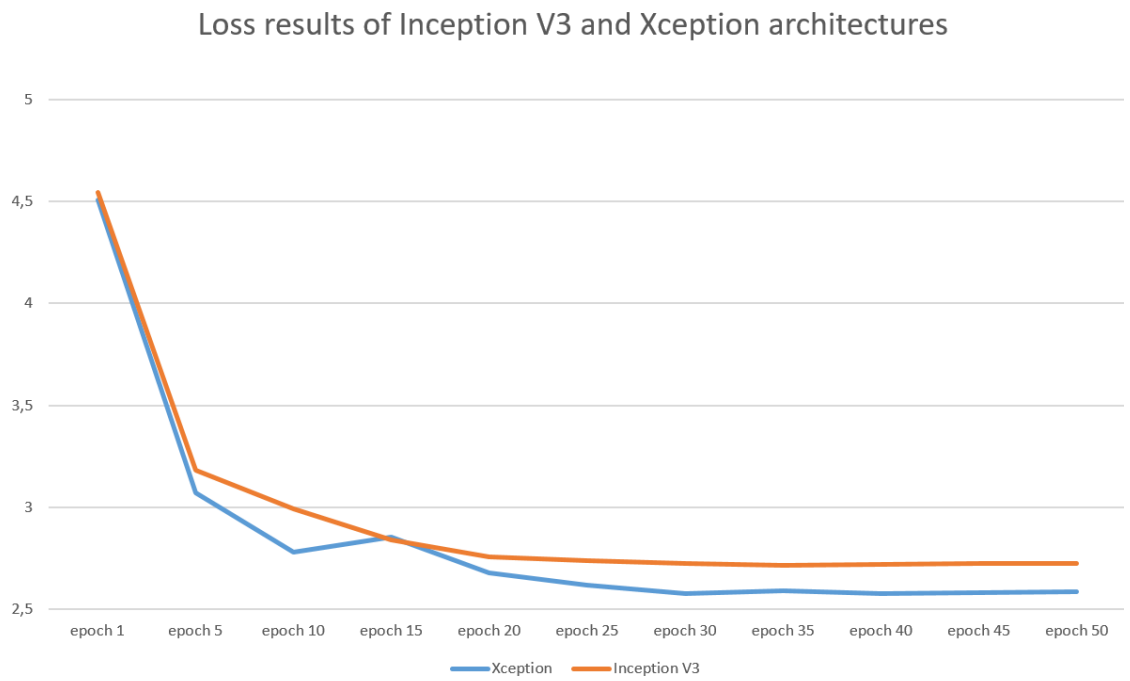


Figure III.10. Loss results of Xception and Inception-V3 models trained for 50 epochs

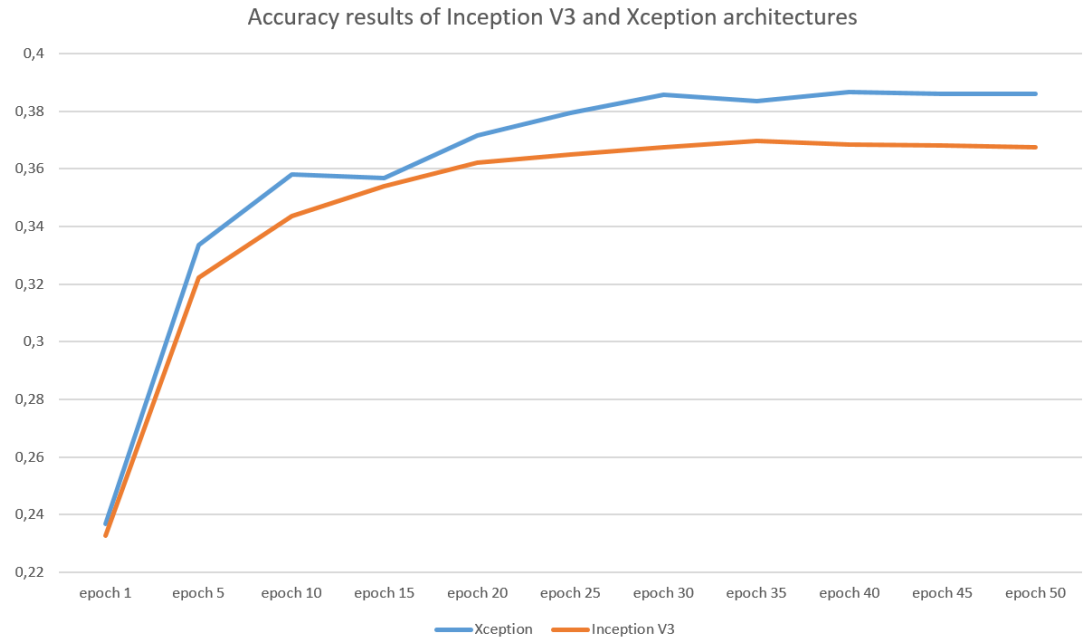


Figure III.11. Accuracy results of Xception and Inception-V3 models trained for 50 epochs

The table below summarizes the best results in terms of loss and accuracy achieved by both models.

Table III.1. Best loss and accuracy results achieved by Xception and Inception-V3 models

Model	Epoch of best result	Loss	Accuracy
Inception-V3	37	2.7129	0.3692
Xception	40	**2.5738**	**0.3867**

Discussion

Graphs above show the evolution of loss and accuracy values of Inception-V3 and Xception model over fifty epochs of training, a fast improvement rate in both loss and accuracy values for both models during the first five epochs followed by a slower rate until about the twentieth epoch for the Inception-V3 model and around the thirtieth epoch for Xception in terms of loss, and from the tenth epoch until around the fortieth epoch for the Xception and Inception-V3 model in terms of accuracy, afterwards, both models reach relative convergence until the end of training period.

Xception model's best results were achieved by the fortieth epoch with a loss value of 2.5738 and an accuracy value of 0.3867. The inception-V3 model's best result is reached by the thirty seventh epoch with a loss value of 2.7129 and an accuracy value of 0.3692, although the results remain relatively close, the Xception model achieved better results than Inception-V3 in terms of loss and accuracy.

c) Result comparison

The table below shows a comparison of results we have obtained from training both models for 10 epochs in the previous experiment and training them until they reach the convergence, the best results are marked in bold:

Table III.2. Comparison between the first, tenth epoch and convergence loss results

Model	Loss				
	After 1 epoch	After 10 epochs	Best result	Difference in loss from first to 10^{th} epoch	Difference in loss from 10^{th} epoch to best result
Xception	**4.5105**	**2.777**	**2.5738 in epoch 40**	**- 1.7335**	- 0.2032
Inception-V3	4.5473	2.9929	2.7129 in epoch 37	- 1.5544	**- 0.28**

Table III.3. Comparison between the first, tenth epoch and convergence accuracy results

Model	Accuracy				
	After 1 epoch	after 10 epochs	Best result	Decrease rate from first to 10^{th} epoch	Decrease rate from 10^{th} epoch to best result
Xception	**23.67 %**	**35.82%**	**38.67 %**	**12.15 %**	**2.85 %**
Inception-V3	23.27 %	34.35%	36.92 %	11,08 %	2.57%

Discussion

- The tables above shows that the decrease rate of loss values for both models from the first to the tenth epoch is around nine times greater than the decrease rate from the tenth epoch to the best result for the Xception model and around six times greater for the Inception-V3 model. As for accuracy results, the increase rate from the first to tenth epoch is around four times greater than the increase rate from the tenth epoch to the best result for both models
- Loss decrease rates and accuracy rates of the two models are close yet Xception achieved better rates in most cases.

III.3.2.2 Generated descriptions

Below, descriptions generated by the Xception and Inception-V3 models after being trained for 10 epochs and by their best models based on the same picture. The correct words will be marked in bold.

<table>
<tr><th>Image</th><th>Model</th><th colspan="2">Generated descriptions</th></tr>
<tr><td rowspan="3"></td><td colspan="3">After 10 epochs</td></tr>
<tr><td>Xception</td><td colspan="2">brown dog is running through the grass</td></tr>
<tr><td>Inception-V3</td><td colspan="2">brown dog is running through the water</td></tr>
<tr><td>Reference descriptions</td><td colspan="3">With the best model</td></tr>
<tr><td>A brown dog is swimming through water with an orange ball in its mouth</td><td rowspan="2">Xception</td><td colspan="2" rowspan="2">dog is running through the water</td></tr>
<tr><td>A dark brow dog holds an orange ball in his mouth while walking through water</td></tr>
<tr><td>A dog wading in the water with a ball in his mouth</td><td rowspan="3">Inception-V3</td><td colspan="2" rowspan="3">brown dog is running through the water</td></tr>
<tr><td>A wet, brownish-red dog with an orange ball in his mouth is swimming in some water</td></tr>
<tr><td>Brown dog in water, with orange and BLEU ball in mouth</td></tr>
</table>

Figure III.12. Examples of generated descriptions by Inception-V3 and Xception after ten epoch and after convergence

Discussion

- After training the two models over fifty epochs, the Xception model has shown an improvement in its generated description and making a correct prediction about the background being water.
- The Inception-V3 showed no improvement since it generated the exact same descriptions after ten and thirty seven epochs.

III.3.2.3 Testing

We evaluate the descriptions generated by the two models based on the test set using the same previously mentioned natural language evaluation metrics.

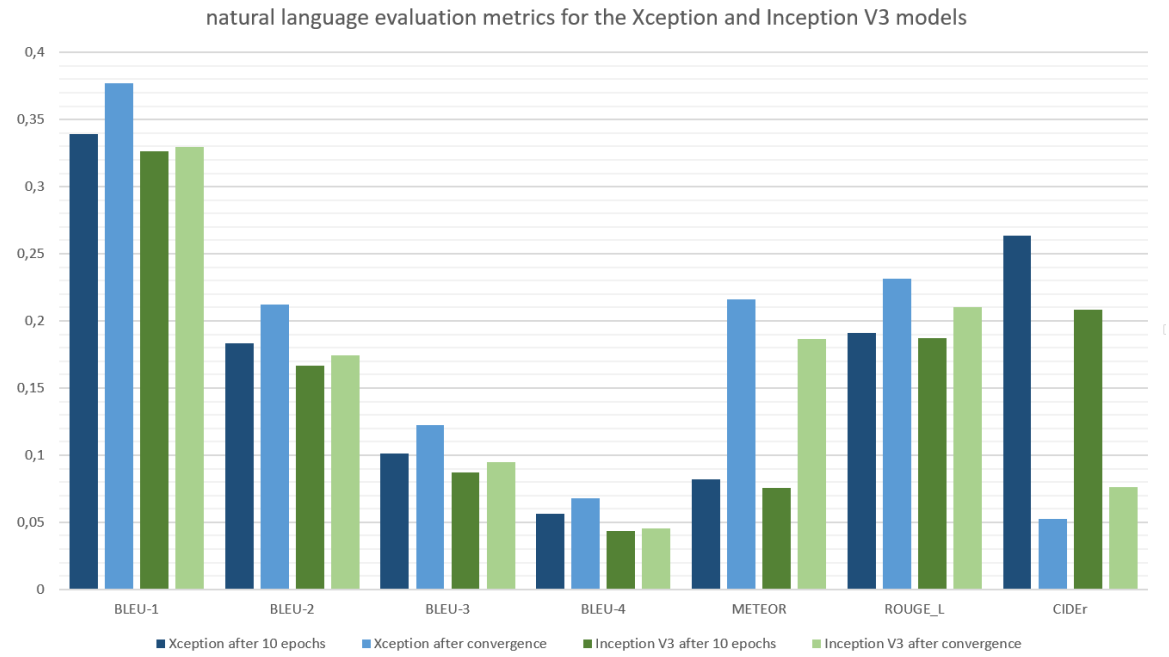


Figure III.13. Natural language metrics evaluation for Xception and Inception-V3 models

The table below summarizes the natural language metrics' results for the two models after 10 epochs of training and after convergence, the best results are marked in bold.

Table III.4. Natural language metrics evaluation for Xception and Inception-V3 models

Model	**Xception**		Inception V3	
Metrics	After 10 epochs	After convergence	After 10 epochs	After convergence
BLEU-1	0,339237	**0,377294**	0,32643	0,329641
BLEU-2	0,183126	**0,2123175**	0,166886	0,174141
BLEU-3	0,10156	**0,122415**	0,08703	0,094623
BLEU-4	0,056171	**0,067639**	0,043488	0,045752
METEOR	0,081707	**0,216328**	0,075743	0,186669
ROUGE_L	0,190813	**0,231274**	0,187297	0,210589
CIDEr	**0,263356**	0,052406	0,208635	0,076057

Discussion

- The natural language evaluation shows a slight increase in most metrics' results for both models.
- The Xception model scored better results than the Inception-V3 after ten epochs and after convergence in all metrics.

III.3.2.4 Conclusion

In this experiment, we studied the convergence of the Xception and Inception-V3 models by training them for 50 epochs on the same dataset used previously. To get a more precise idea about the evolution of the training process, we compared the results obtained after 10 epochs with results of the best model in terms of loss, accuracy and their evolution rate. Finally, we evaluated the descriptions generated by the two models in both cases using the four previously mentioned natural language evaluation metrics.

Although the achieved results remain close, the Xception model surpassed Inception-V3 in most of the evaluations highlighting the superiority of the Xception model over other models thanks to its promising results.

We think it is worthy to remind that the tasks performed and previously described were time consuming and took more than 40h to be concluded showing a negligible improvement regarding loss and accuracy results.

In the next experiment, we will study the influence of the dataset size on the different architectures' performances.

III.4. Study of dataset size's influence on different models' performances

The goal of this experiment is to study the influence of dataset's size on the four models' performances, we select a split of the dataset containing 800 images in total, and we dedicate 600 images for training, 100 images for validation and 100 images for testing.

III.4.1 Training

We launch the training of each one of the four models and we study and evaluate their convergence.

a) Settings

The table below indicates the values we chose for the main parameters.

Table III.5. Summary of training settings

Optimizer	Adam
Batch size	32 (default value)
Learning rate	0,001 (default value)
Loss function	Categorical cross entropy
Metrics	Accuracy
Training set size	600 images
Description length	32 characters
Number of epochs	Variable

b) Presentation of training results

Once again we will use the advantage of Google collab's cloud services to lunch the training of all four models simultaneously.

The table below shows an estimation of time spent by the four models to complete the training.

Table III.6. Summary of time consumption results

Model	Time per epoch	Total number of epochs	Total time spent on training
Xception	**About 2 minutes**	**250**	**Around 8 hours**
Inception-V3	About 2,5 minutes	300	Around 13 hours
VGG-16	About 2,5 minutes	200	Around 9 hours
ResNet-50	About 3 minutes	350	Around 18 hours

Discussion

The Xception model consumes the shortest period of time to reach its' convergence followed by the VGG-16 than the Inception-V3 and finally ResNet-50.

The figures below show loss and accuracy values of the four models after being trained on split of the dataset.

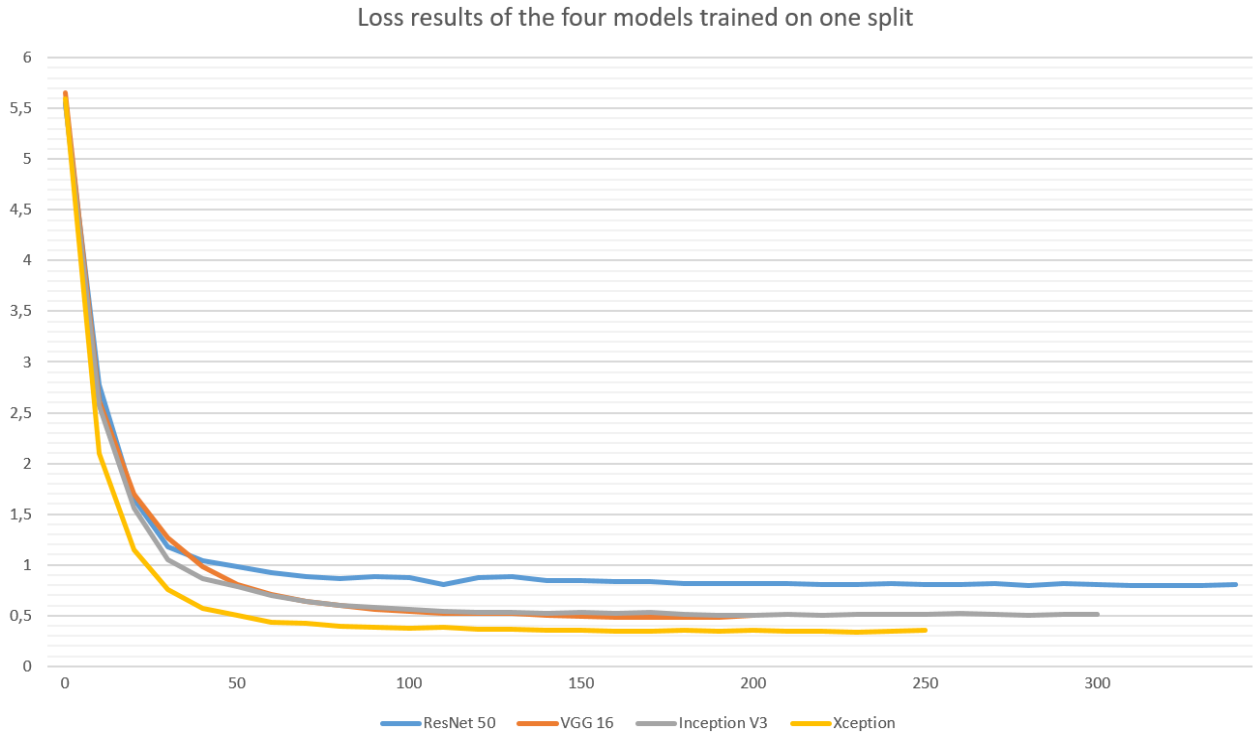


Figure III.14. Loss results of the four models trained on one split of the dataset

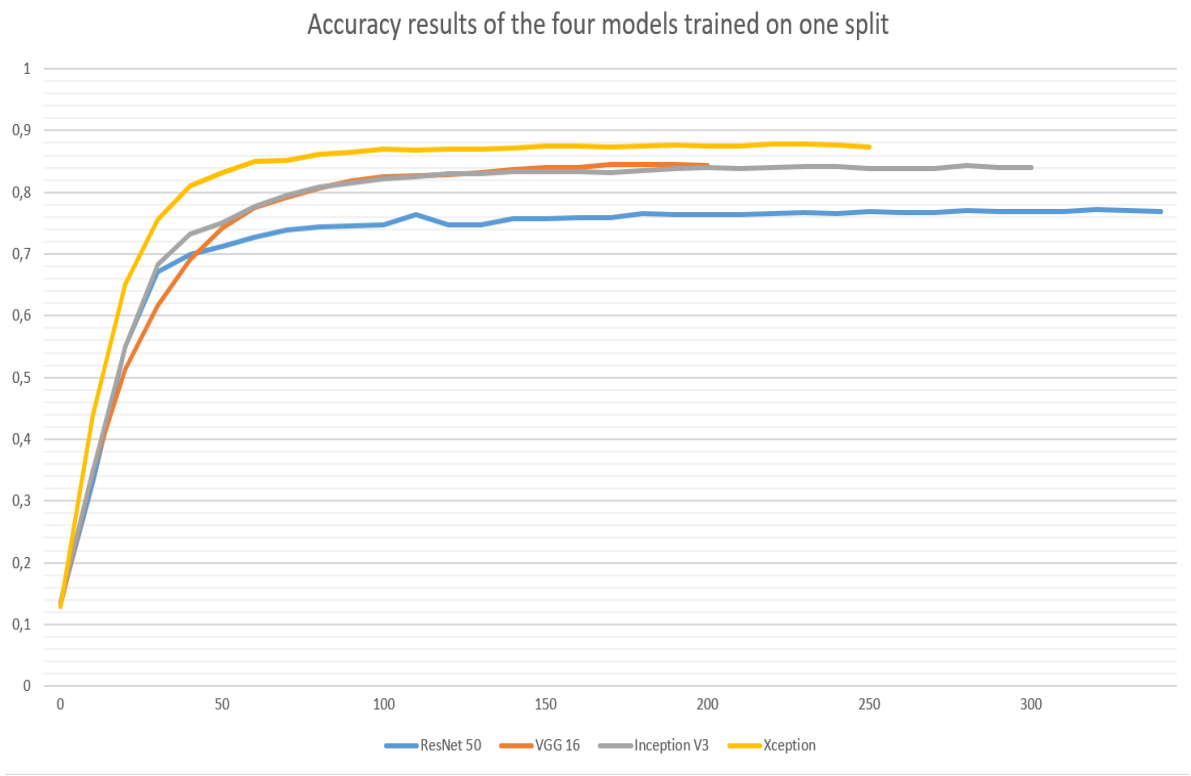


Figure III.15. Accuracy results of the four models trained on one split of the dataset

The table below includes the best loss and accuracy values obtained by each model.

Table III.7. Best loss and accuracy results of each model

Model	Loss	Accuracy
Xception	**0,3339**	**0,88**
VGG-16	0,4734	0,8473
Inception-V3	0,4985	0,8419
ResNet-50	0,7938	0,7705

Discussion

- The graphs show the evolution of accuracy and loss values for the four models trained on one split of the previously studied dataset until they have reached their convergence.
- The results display a fast regression rate in loss and increase rate in accuracy values from the first till about fiftieth epoch followed by a slower rate until the last epoch and each of the four models show a different progression rate in accuracy and loss, the Xception model is the fastest to reach its convergence (around the 50^{th} epoch), followed by VGG-16 and Inception-V3 (around the 100^{th} epoch), and finally ResNet-50 (around the 150^{th} epoch).
- From the first to the fiftieth epoch: accuracy increased by about 0.7 (which is a 70% increase) for Xception, by about 0.65 (which is a 65% increase) for VGG-16 and inception -V3, by about 0.6 (which is a 60% increase) for ResNet-50.
- From the fiftieth epoch till each model's convergence: accuracy increased by about 0.01 (which is a 1% increase) for Xception, by about 0.03 (which is a 3% increase) for VGG-16 and inception-V3, by about 0.07 (which is a 7% increase) for ResNet-50.

- From the first to the fiftieth epoch : loss values decreased by about 5 for Xception, by about 4.9 for VGG-16 and inception-V3, by about 4.7 for ResNet-50
- From the fiftieth epoch till each model's convergence: loss values decreased by about 0.2 for all four models.
- The accuracy and loss from the fiftieth epoch and on can be considered negligible compared to the first 50 epochs which allow to say that the fiftieth epoch is the start of their convergence.
- The Xception model ranks first by achieving the best results in terms of loss and accuracy, followed by the VGG-16 and Inception-V3 and finally the ResNet-50 model.
- These results show an improvement in loss and accuracy compared to the results achieved in the previous experiment over the entire dataset.

III.4.2 Generated descriptions

The table below includes descriptions generated by each of the four models after being trained on one split of the previously studied dataset based on the same picture. The correct words will be marked in bold.

<table>
<tr><th>Image</th><th>Model</th><th>Generated descriptions</th></tr>
<tr><td rowspan="2"></td><td>Xception</td><td>group of people riding rock ski ski mountain</td></tr>
<tr><td>Inception-V3</td><td>man in yellow shirt and black shorts poses in front of mountain</td></tr>
<tr><td>Reference descriptions</td><td rowspan="3">VGG-16</td><td rowspan="3">man in red jacket is standing in the snow next to gray mountain</td></tr>
<tr><td>A group of backpackers lay on the dry ground</td></tr>
<tr><td>A group of hikers are resting on the ground in front of some mountains</td></tr>
<tr><td>People camp with the mountains in the background</td><td rowspan="3">ResNet-50</td><td rowspan="3">the man is leaning on rocks close to water</td></tr>
<tr><td>Several hikers rest with their gear in front of a mountain</td></tr>
<tr><td>The group of hikers is resting in front of a mountain</td></tr>
</table>

Figure III.16. Generated description by the four models trained on one split of the dataset

Discussion

- Only the Xception model could identify the two main assets of the picture which are the group of people and the mountain in front of them, the inception-V3 and VGG-16 models could only identify the mountain correctly and the ResNet-50 model didn't identify any of them correctly.
- The descriptions generated by the four models are close to human language except for Xception's which includes a repetition of "ski" work which is wrong.

III.4.3 Testing

The diagram and table below include the natural language evaluation metrics' results obtained by each of the four models.

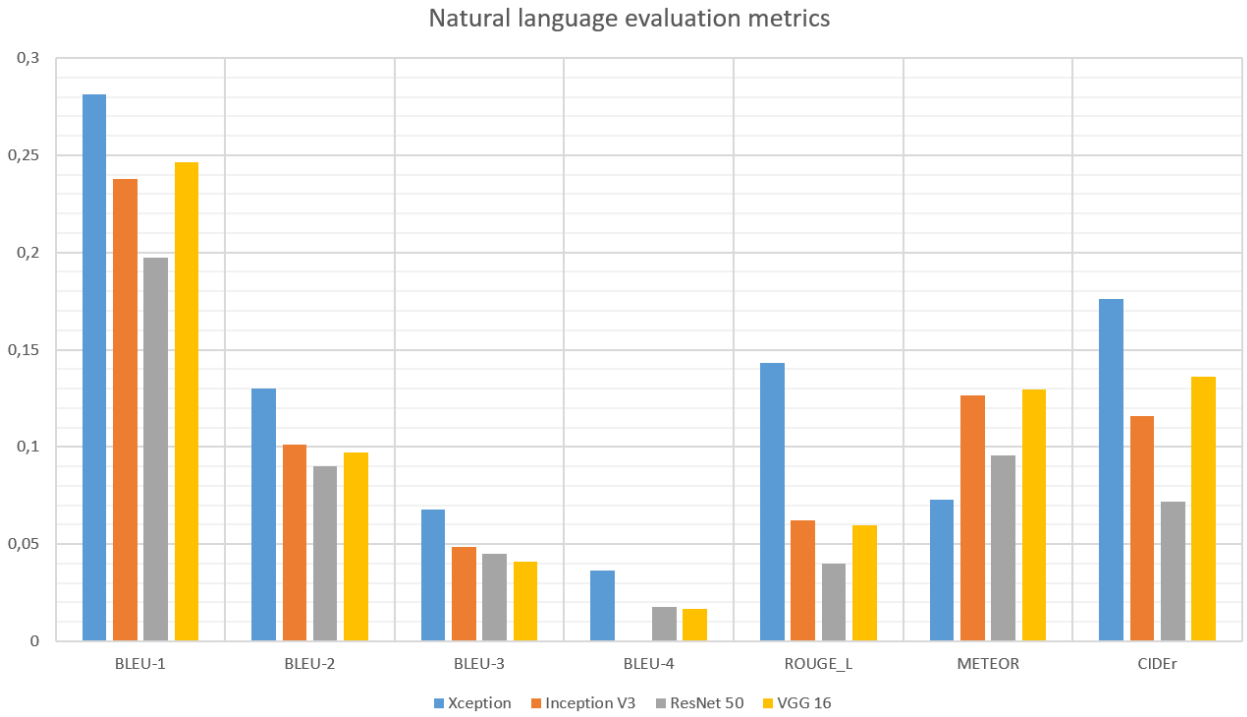


Figure III.17. Evaluation metrics results for the four models trained on one split of the dataset

Table III.8. Evaluation metrics results for the four models trained on one split of the dataset

	Xception	Inception-V3	ResNet-50	**VGG-16**
BLEU-1	**0,281305**	0,238055	0,197441	0,246209
BLEU-2	**0,129812**	0,10113	0,090198	0,09693
BLEU-3	**0,067862**	0,048299	0,044778	0,040676
BLEU-4	**0,036218**	0	0,017894	0,0165
ROUGE_L	**0,1429521**	0,06203	0,039872	0,059753
METEOR	0,072661	0,126289	0,09549	**0,129436**
CIDEr	**0,176001**	0,115864	0,0719	0,136075

Discussion

- After training the models on one split, Xception achieved the best results in most metrics followed by Inception-V3 and VGG-16 which achieved very close results and finally comes the ResNet-50 model.

Comparing these results to results obtained after training the models on the original dataset, we notice a decrease in most natural language evaluation results.

III.4.4 Conclusion

In this experiment, we studied the convergence of four models with different encoder architectures which are: Xception, Inception-V3, VGG-16 and ResNet-50 over a split of the dataset including 800 images which were divided into: 600 images for training, 100 images for testing and 100 images for development, we have evaluated their performances in terms of time, loss, accuracy and natural language metrics.

Once again, this experiment confirms the superiority of the Xception model over the other models and despite the change in dataset size, this model proves itself to be robust and efficient in all aspects of evaluation.

The results also helped us see the influence of dataset size reduction on the quality of generated descriptions and natural language metrics' results with four different models which helped us prove our previous conclusions.

III.5. Study of the data choice influence

In this study, we will study the influence of data choice and organization on the different models' performances.

III.5.1 Experiment 01: Study over the entire Flicker 8K dataset

In order to study the dataset's composition and the influence of how images are divided between the training, testing and development sets on the results, We shuffle the entire 8000 pictures of the flickr8K dataset using a code that does this task automatically and produce a new random splitting, dedicating 6000 pictures for training, 1000 pictures for validation and 1000 for testing.

III.5.1.1 Training

For our first experiment, we will study the convergence of the Xception model over this new repartition and compare results of different evaluations to results of the previous study.

a) **Settings**

The table below indicates the values we chose for the main parameters.

Table III.15. Summary of training settings

Optimizer	Adam
Batch size	32 (default value)
Learning rate	0,001 (default value)
Loss function	Categorical cross entropy
Metrics	Accuracy
Training set size	6000 images
Description length	32 characters
Number of epochs	50
Original training set's vocabulary size	7577
Shuffled training set's vocabulary size	7583

b) **Presentation of training results**

The time spent by the Xception model to reach its' convergence in both cases shows no difference consuming in each case around 25h to complete the training spending around 30 minutes per epoch.

The graphs below shows the evolution of loss and accuracy values respectively for the Xception model trained on the original and shuffled set over 50 epochs.

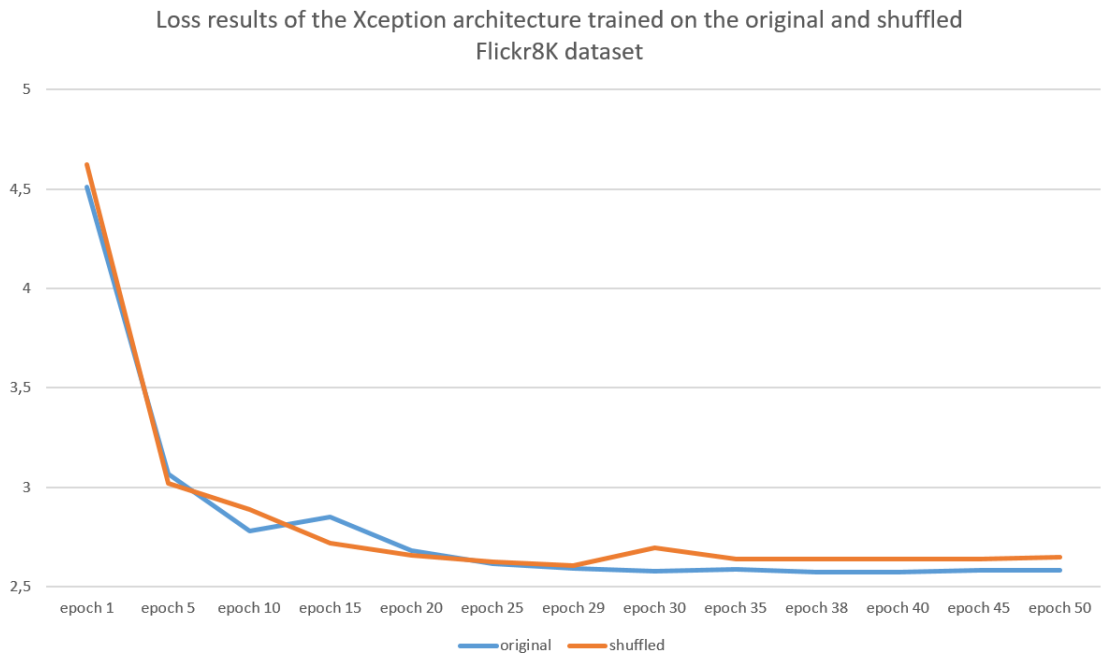


Figure III.18. Loss results of Xception trained on the original and shuffled set for 50 epochs

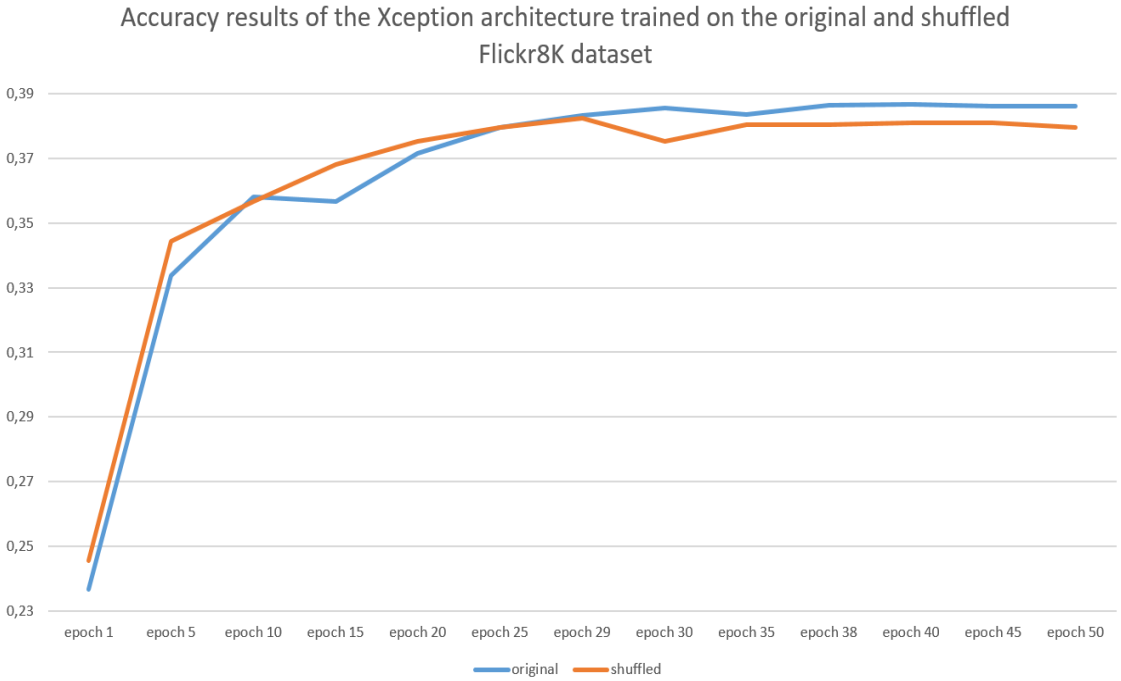


Figure III.19. Accuracy results of Xception trained on the original and shuffled set for 50 epochs

The table above shows the best results achieved by the Xception model after being trained on the shuffled dataset along with the previous convergence study's results to help us compare the results.

Table III.16. Best accuracy and loss results achieved by the Xception model trained on the original and shuffled training set

Dataset	Loss	Accuracy	Epoch of best result
Original	2.5738	0.3867	40
Shuffled	2.6071	0.3824	30

Discussion

Loss and accuracy curves obtained after training the Xception model for fifty epochs have the same overall shape with some minor fluctuations in their values between the fifth and thirty-fifth epochs, afterwards the model settle around a stable value starting of 2.5738 on the original and 2.6071 on the shuffled set in terms of loss, and a value of 0.3867 on the original and 0.3824 on the shuffled set in terms of accuracy. The best results were achieved by the thirtieth epoch on the original set and y fortieth epoch on the shuffled set we shall highlight the fact that shuffling the dataset didn't bring noticeable improvement into loss and accuracy values.

III.5.1.2 Generated descriptions

Below, descriptions generated by the Xception model after being trained on the original and shuffled dataset based on the same picture. The correctly predicted words will be marked in bold.

Image	***Dataset***	***Generated descriptions***
	Original	**Young boy in swim trunks** is jumping **into** pool
Reference descriptions	Shuffled	**boy in swimming trunks is** splashing **in the water**
A boy in orange and white swimming trunks is falling into water at a beach or lake		
A boy in orange shorts is playing in the shallow part of the ocean		
A boy in swim trunks in the water at the beach		
A boy wearing board shorts falls backwards in the water		
a young man wearing a red and white floral swinsuit falling into the beach		

Figure III.20. Examples of generated descriptions by Xception after being trained on the original and shuffled datasets

Image	*Dataset*	*Generated descriptions*
	Original	Two people are **riding** their **dirt** bike on the be ach
Reference descriptions	Shuffled	Two people are parked on **dirt road**
A blue rally car pulls away from the spectators watching from the side of the track		
Blue race car riding on dirt path with onlookers		
A racing car goes through a dirt course while fans watch		
A race car zooms while onlookers watch		
A car is being driven on a trail while people on both sides of the road look on		

Figure III.21. Examples of generated descriptions by Xception after being trained on the original and shuffled datasets

Discussion

- In the first example, the model could identify the main subject of the picture which is a boy in his swimsuit, and the presence of water in the background producing satisfactory descriptions.
- In the second example, the model couldn't identify the main asset of the picture which the race car, however it could identify the presence of people in both cases and the dirty road after being trained on the shuffled set.
- Training the model on different images with different descriptions produced different results.

III.5.1.3 Testing

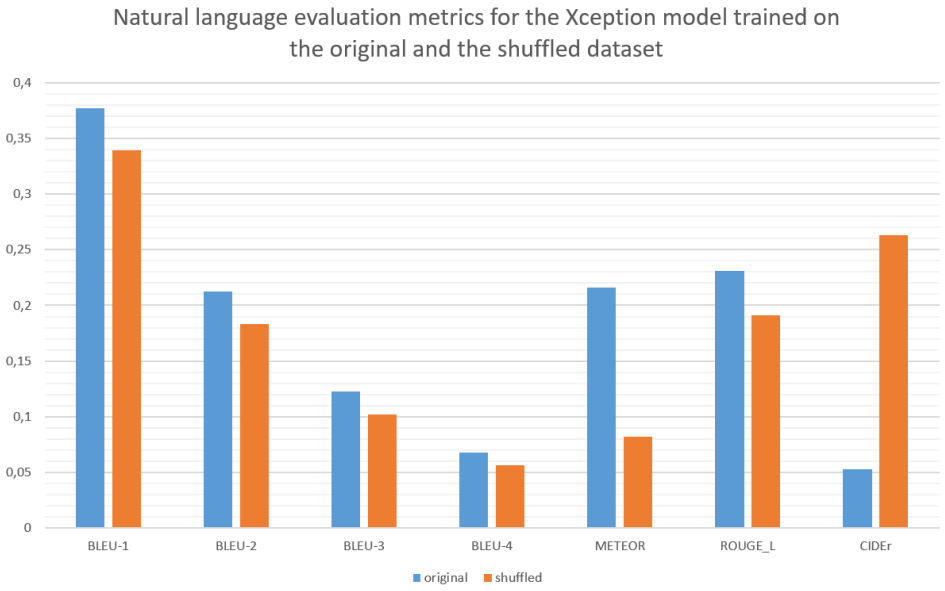


Figure III.22. Natural language metrics evaluation for Xception on the original and the shuffled set

The table below summarizes the natural language metrics' results for both cases.

Table III.17. Natural language metrics evaluation for Xception and Inception-V3 models

Metric	Original	Shuffled
BLEU-1	0,377294	0,339237
BLEU-2	0,2123175	0,183126
BLEU-3	0,122415	0,10156
BLEU-4	0,067639	0,056171
METEOR	0,216328	0,081707
ROUGE_L	0,231274	0,190813
CIDEr	0,052406	0,263356

Discussion

The results show some changes in the values obtained after evaluation the descriptions generated by the Xception model after being trained of the original and shuffled dataset.

The BLEU and ROUGE scores show minor changes, on the other hand, the METEOR and CIDEr made noticeable changes this can be due to the fact that BLEU-n is based on evaluating whether all consecutive n words in the generated candidate are in the reference but it does not check if all words in reference are covered and does not take into account semantic similarity. ROUGE-L score is based on the longest sequence of words that is shared between both but not necessarily consecutive, but still in order. On the other hand, METEOR and CIDEr score takes into account the length of common sequence, word order and semantics.

III.5.1.4 Conclusion

In this experiment we aimed to study the influence of the dataset's organization on the model's results, we proceeded to Xception model of a shuffled version of the previously studied dataset, we evaluated its performances in terms of loss, accuracy, natural language metrics and compared the results to the ones previously obtained based on the original dataset, finally, we compared descriptions generated by models trained on the shuffled and original training sets based on the same picture.

We haven't obtained any noticeable improvement in loss, accuracy and evaluation metrics' values after shuffling the dataset. However, the natural language evaluation using the different metrics showed some noticeable changes in the values which means that choice of samples can infer the quality of the generated descriptions even if the size of the vocabulary is almost the same since we have 7583 words in the original training set and 7577 in the shuffled training set's vocabulary with the same maximum length of the descriptions which is 32 characters.

Based on these conclusions, in next experiment we will divide the dataset into ten random splits, launch the training on each of them to calculated average values instead of making conclusions based on a unique subset.

III.5.2 Experiment 02: studying 10 random splits of the dataset

In this study, we divide the shuffled flickr8K dataset into 10 splits containing each: 600 pictures dedicated to training, 100 pictures to testing and 100 for development using a code that does this task automatically.

III.5.2.1 Training

We launch the training on each of the ten splits using the two architectures which achieved the best results in the previous experiments which are: Xception and Inception-V3 and we study and evaluate their convergence.

a) Settings

The table below indicates the values we chose for the main parameters:

Table III.18. Summary of training settings

Optimizer	Adam
Batch size	32 (default value)
Learning rate	0,001 (default value)
Loss function	Categorical cross entropy
Metrics	Accuracy
Training set size	600 images
Description length	32 characters
Number of epochs	Variable

The table below includes the vocabulary sizes in each split's training set.

Table III.19. Summary of training sets vocabulary for the ten splits

Split	1	2	3	4	5	6	7	8	9	10
Xception	2525	2395	2447	2392	2475	2447	2542	2413	2443	2553
Inception-V3	2525	2395	2447	2392	2475	2447	2542	2413	2443	2553

The size of vocabulary calculated in each split for both models is exactly the same but changes from split to split which can be explained by the difference of pictures present in each split and so the deference of the words in their descriptions.

The reduction in dataset size resulted in a reduction in vocabulary size.

b) Presentation of training results

After testing the time consumption of both models to reach their convergence on the first split, we noticed that it takes around 7 hours for Xception and around 8 hours for Inception-V3, due to the number of splits considered and because of the time consumption and the memory available on the PC, an independent and online platform has been exploited thanks to the parallelism that it may offer.

The table below shows an estimation of time spent by Xception and Inception-V3 models to complete the training.

Table III.20. summary of time consumption after training by Xception and Inception-V3 models on 10 splits

Model	Time per epoch	Total time spent on training
Xception	**About 2 minutes**	**Around 28 hours**
Inception-V3	About 2,5 minutes	Around 32 hours

Discussion

Exploring the cloud services enabled us to reduce the time consumption but reaching convergence with both models on ten splits took a long period of time, nevertheless, Xception took lesser time comparing to the Inception-V3 model to complete the task.

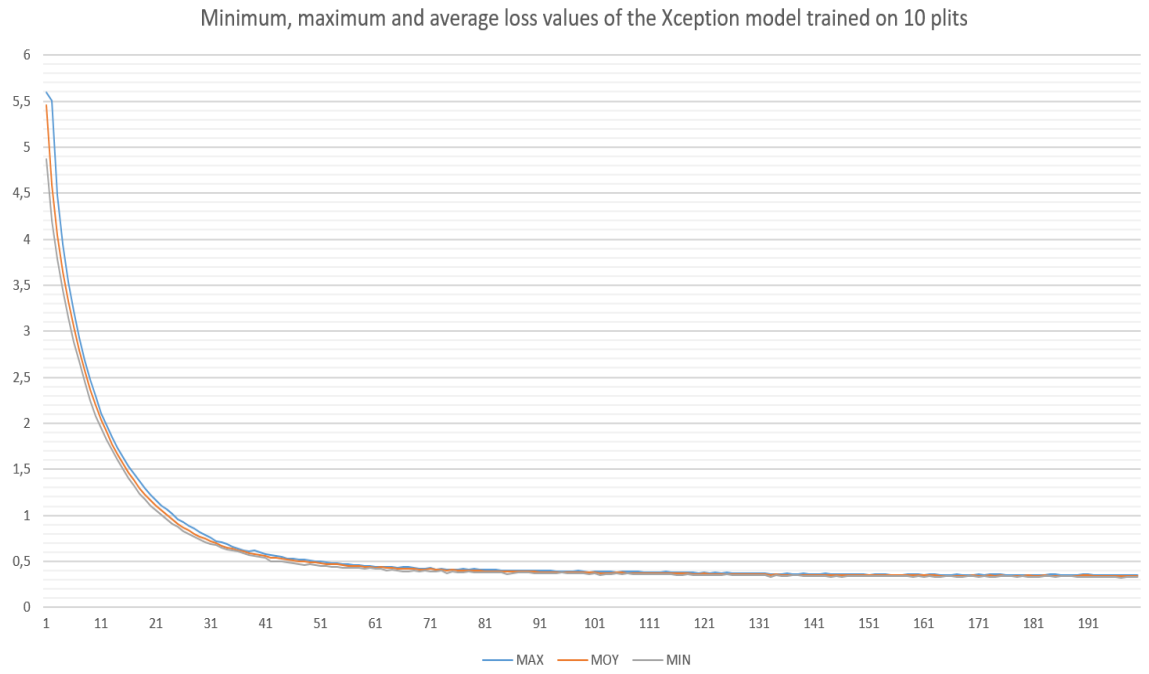


Figure III.23. Minimum, maximum and average loss results obtained by Xception over ten splits of the dataset

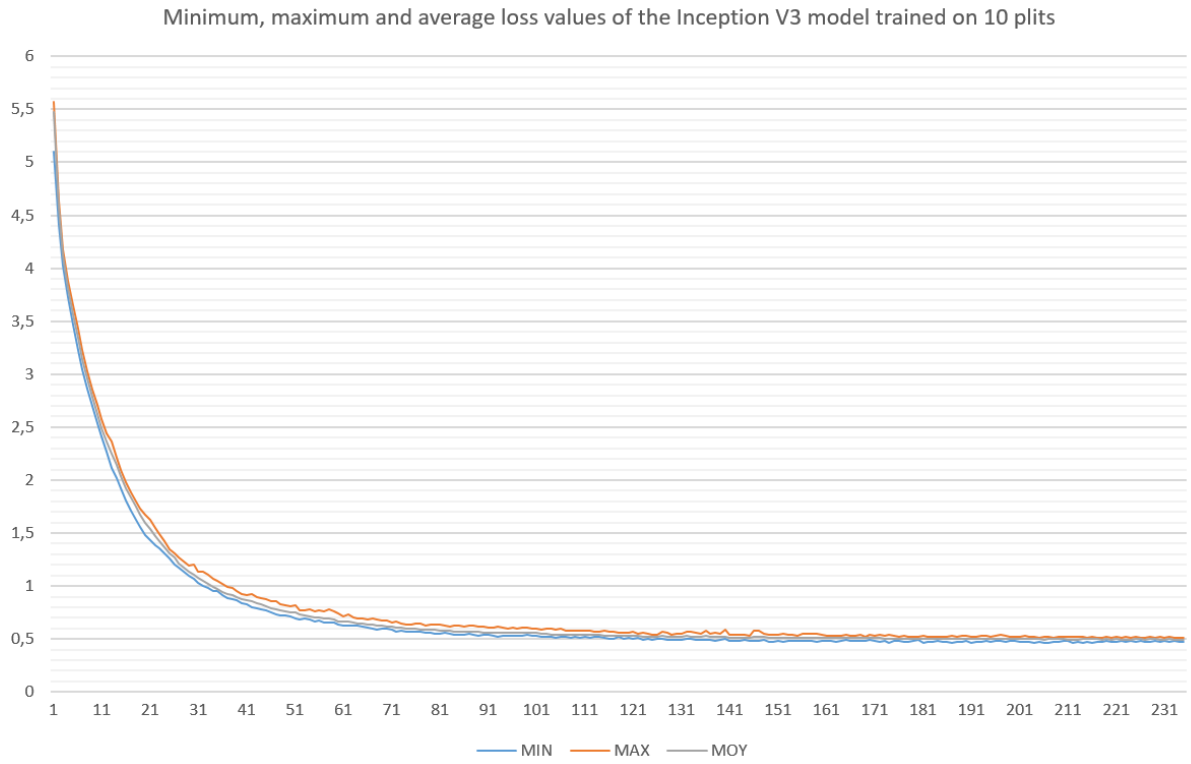


Figure III.24. Minimum, maximum and average loss results obtained by Inception-V3 over ten splits of the dataset

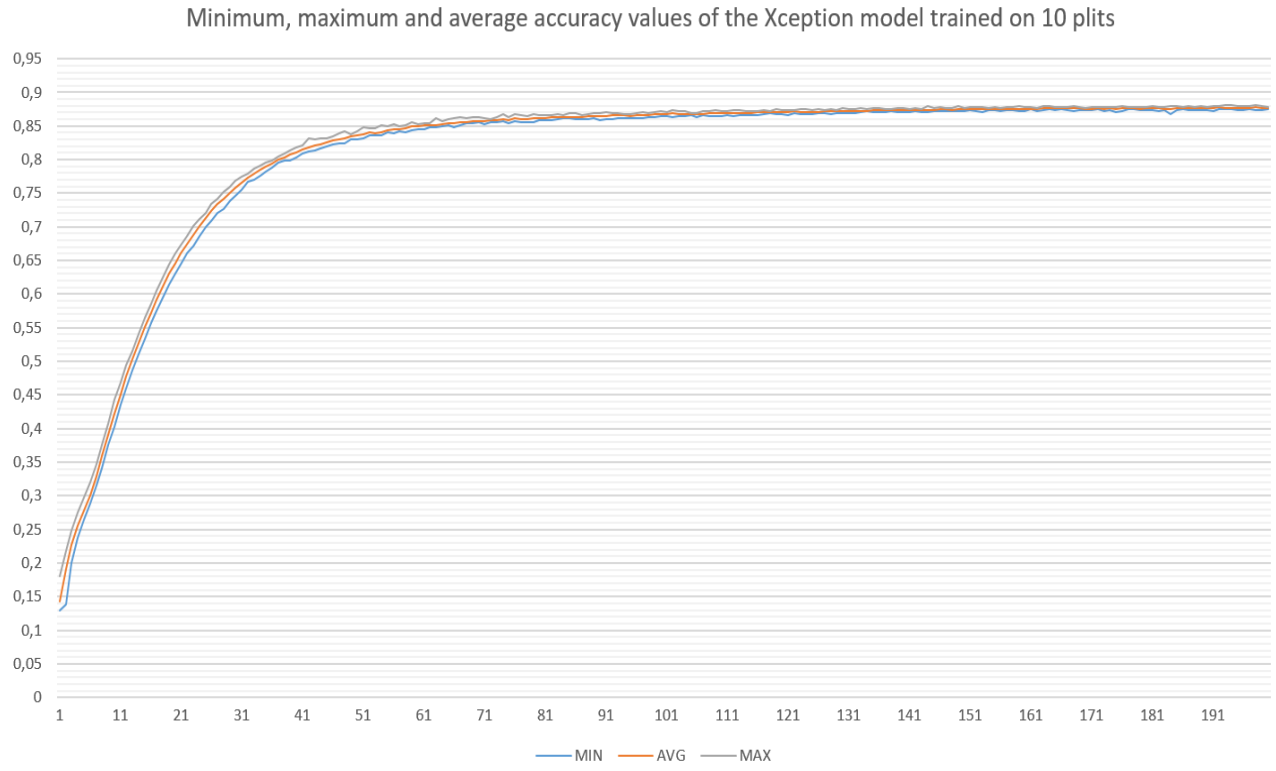


Figure III.25. Minimum, maximum and average accuracy results obtained by Xception over ten splits of the dataset

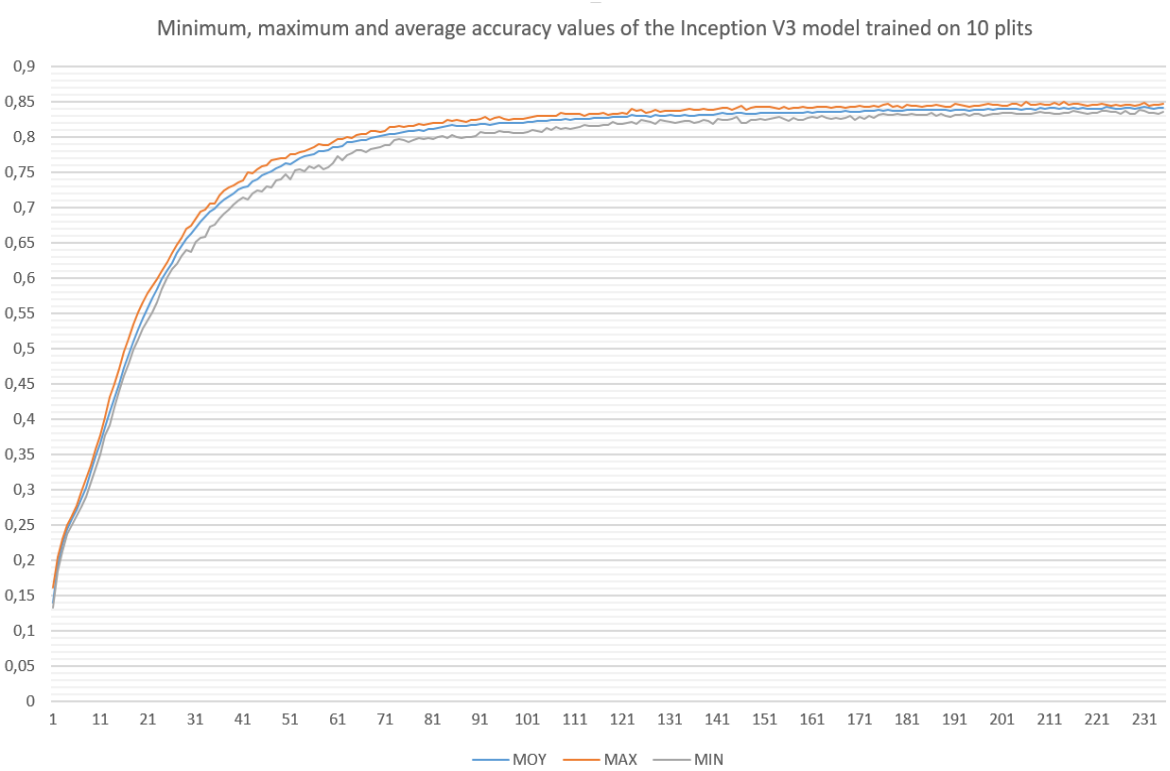


Figure III.26. Minimum, maximum and average accuracy results obtained by Inception-V3 over ten splits of the dataset

The table below summarizes the best, average, maximum and minimum results of accuracy and loss values of both models.

Table III.21. best, average, maximum and minimum results of accuracy and loss values of Xception and Inception-V3 trained on ten splits of the dataset

Split	Epoch of best results	Loss	Accuracy	Epoch of best results	Loss	Accuracy
1	185	0,367	0,8721	251	0,4985	0,8419
2	176	0,3404	0,8587	230	0,4801	0,8486
3	195	0,3357	0,8787	191	0,4759	0,845
4	174	0,3353	0,878	235	0,4904	0,8424
5	194	0,3304	0,8791	227	0,499	0,8389
6	167	0,3395	0,8786	212	0,4753	0,8438
7	197	0,3312	**0,8805**	210	0,4902	0,843
8	202	0,3328	0,8803	205	**0,4586**	**0,8504**
9	173	**0,3262**	0,8788	233	0,5025	0,8394
10	192	0,3277	0,8802	185	0,4757	0,8452
MOY		0,3366	0,8765		0,4846	0,84386
MIN		0,3262	0,8587		0,4586	0,8389
MAX		0,367	0,8805		0,5025	0,8504

Discussion

- The graphs show the evolution of the average, maximum and minimum accuracy and loss values calculated from Xception and Inception-V3 results on each split over 200 epochs.
- Both models' results display a fast regression rate in loss and increase rate in accuracy values from the first till about fiftieth epoch followed by a slower rate until the last epoch.
- Xception's accuracy increased by about 0.7 (which is a 70% increase) from the first to the fiftieth epoch whereas it increased by about 0.03 (which is a 3% increase) from the fiftieth epoch till the end of training. Inception-V3 accuracy results increased by about 0.6 (which is a 60% increase) from the first to the fiftieth epoch whereas it increased by about 0.06 (which is a 6% increase) from the fiftieth epoch till the end of training.
- Xception's loss results decreased by about 5 from the first to the fiftieth epoch whereas, it decreased by about 0.3 from the fiftieth until the last epoch, and Inception-V3 results decreased by about 5 from the first to the fiftieth epoch whereas it decreased by about 0.3 from the fiftieth until the last epoch.
- The improvement in loss and accuracy results for both models from the fiftieth epoch and on can be considered negligible compared to the first 50 epochs which allow to consider the fiftieth epoch as the start of convergence.
- In the beginning, the difference between the average, the maximum and minimum in accuracy is about 0.05 and about 0.5 in loss and by the end of the training, the difference is reduced to about less 0.02 in accuracy and 0.1 in loss for both models which is a small variation.
- These results show an improvement in loss and accuracy compared to the previous results obtained from training the models on a bigger training set.

III.5.2.2 Generated descriptions

The descriptions generated by the Xception and Inception-V3 models based on the same picture after being trained on 10 splits dataset are summarized in the table below.

	Split	*Descriptions generated by the Xception model*
	**1**	**dog** is **running** along **the grass**
	2	**the** brown **and white** great **dog** plays with his hand in the hills
	3	**two dogs run** next to get yellow and onlookers in brown **dog**
	4	**dog** runs **in the grass** with its mouth open and **running**
	5	an **dog** is in front
	6	**two dogs** play with each other sitting together
	7	**black dog** runs with soccer ball
	8	**dog** is playing in grassy field
	9	**two dogs** wrestle outdoors
	10	brown **and white dog on** trampoline with forest in the background
Proposed descriptions	**Split**	***Descriptions generated by the Inception-V3 model***
A black and a black dog are running	1	man looks mysterious in blue shirt and red stick in the background
A black and white dog chases a bigger dog across the grass	2	**dog** catches flying disk in yard with **grass**
	3	little boy pushing another little boy up the pole
One dog chases another in the grass	**4**	**black and white dog** is **running through the grass**
	5	man in an orange shirt plays in field of blue plank of bird
Two brindle dogs running in the grass	6	**two dogs** playfully bite third dog which has its tongue sticking out
	7	man **and white dog play** tug of white foliage
	8	**two black dogs** play or fight together
Two dogs run through the grass	9	brown **dog** with collar runs in **the** dead **grass** with his tongue hanging out to the side
	10	**dog running** on **the** sand

Figure III.27. Examples of generated descriptions by Xception after being trained on the original and shuffled datasets

Discussion

- All descriptions generated by the Xception models include the main asset of the picture which is the presence of a dog, while three of the Inception-V3 models didn't.
- Overall, the Xception models achieved better results than Inception-V3's comparing their score in each split.
- The quality of descriptions decreased noticeably for both models comparing to the descriptions obtained in the previous experiments after training the models on bigger training sets.

III.5.2.3 Testing

The tables and diagrams below display the natural language evaluation metrics' results of Xception and Inception-V3 models after being trained over ten splits of the dataset

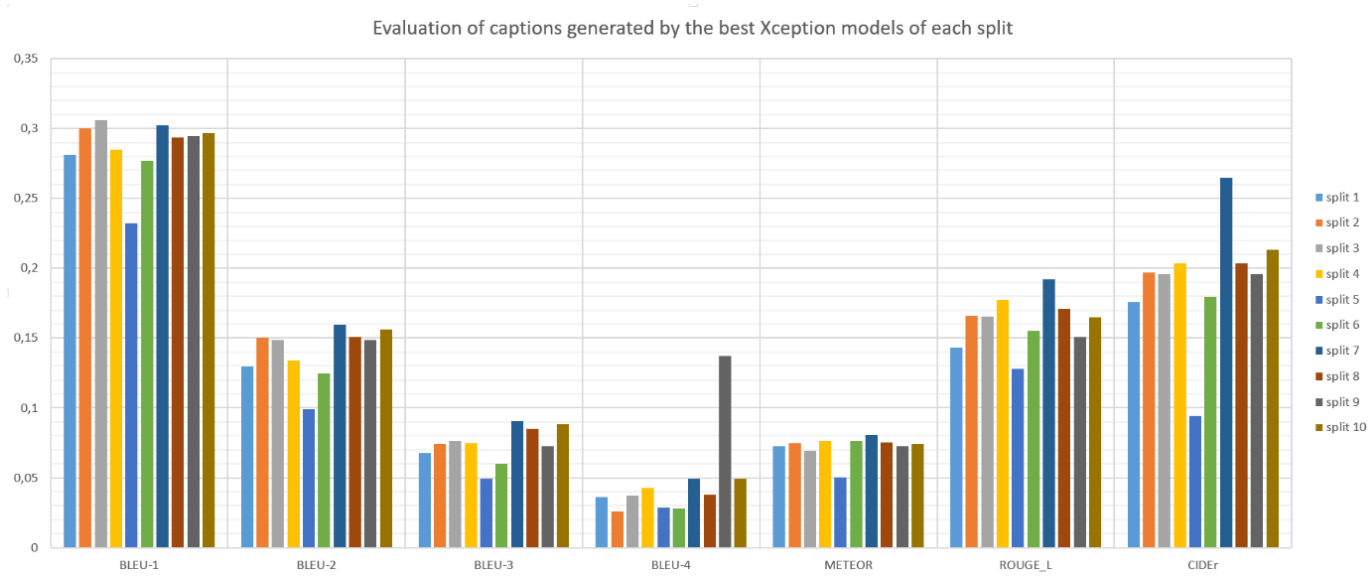


Figure III.28. Natural language evaluation metrics results of Xception trained on ten splits of the dataset

Table III.22. Natural language evaluation metrics results of Xception trained on ten splits of the dataset

split	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE_L	CIDEr
split 1	0,281305	0,129812	0,067862	0,036218	0,072661	0,142952	0,176001
split 2	0,300087	0,149974	0,074153	0,025758	0,075006	0,165720	0,197104
split 3	**0,305893**	0,148445	0,076546	0,037304	0,069558	0,165544	0,195947
split 4	0,284653	0,133801	0,074898	0,042928	0,076358	0,17711	0,203673
split 5	0,232496	0,099245	0,049023	0,028482	0,050202	0,127864	0,094285
split 6	0,276596	0,124567	0,059864	0,028272	0,076319	0,155171	0,179455
split 7	0,302139	**0,159373**	**0,090415**	0,04914	**0,080731**	**0,191858**	**0,264607**
split 8	0,293863	0,150917	0,085003	0,037551	0,075186	0,171082	0,203691
split 9	0,294737	0,148607	0,07248	**0,13694**	0,072764	0,151030	0,196026
split 10	0,29697	0,156417	0,088277	0,049408	0,074021	0,164863	0,213205

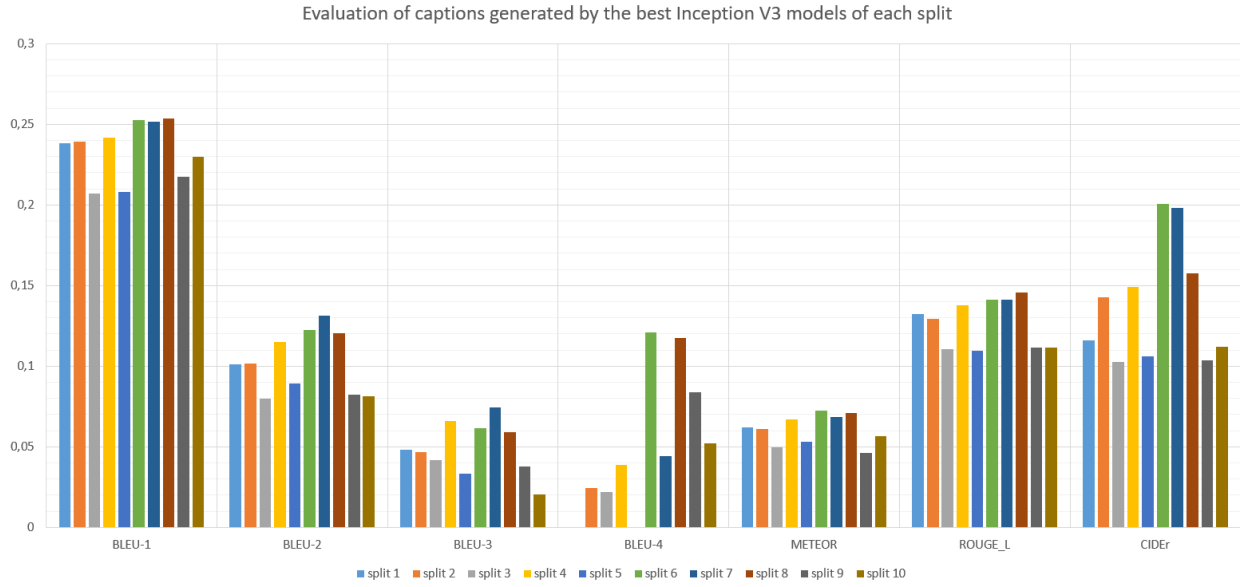


Figure III.29. Natural language evaluation metrics results of Inception-V3 trained on ten splits of the dataset

Table III.17. Natural language evaluation metrics results of Inception V3 trained on ten splits of the dataset

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE_L	CIDEr
split 1	0,238055	0,10113	0,048299	0	0,06203	0,13249146	0,115864
split 2	0,239051	0,101539	0,046539	0,024233	0,061155	0,12934443	0,142843
split 3	0,207192	0,079975	0,041717	0,02183	0,049526	0,11043909	0,102491
split 4	0,241599	0,115158	0,065966	0,03895	0,066948	0,13791049	0,148966
split 5	0,208042	0,089237	0,033419	0	0,053201	0,10959181	0,106109
split 6	**0,252766**	0,122615	0,061659	**0,121152**	**0,07244**	0,14100854	**0,200651**
split 7	0,251371	**0,131069**	**0,074581**	0,044293	0,06824	0,14121055	0,198121
split 8	0,253794	0,120316	0,059203	0,117478	0,071048	**0,14547747**	0,157413
split 9	0,217617	0,082348	0,037937	0,083856	0,046148	0,11163571	0,103819
split 10	0,229583	0,081474	0,020218	0,052056	0,056457	0,11163571	0,112104

The tables and diagrams below summarizes the maximum, the minimum and the average values calculated over the ten splits for both models.

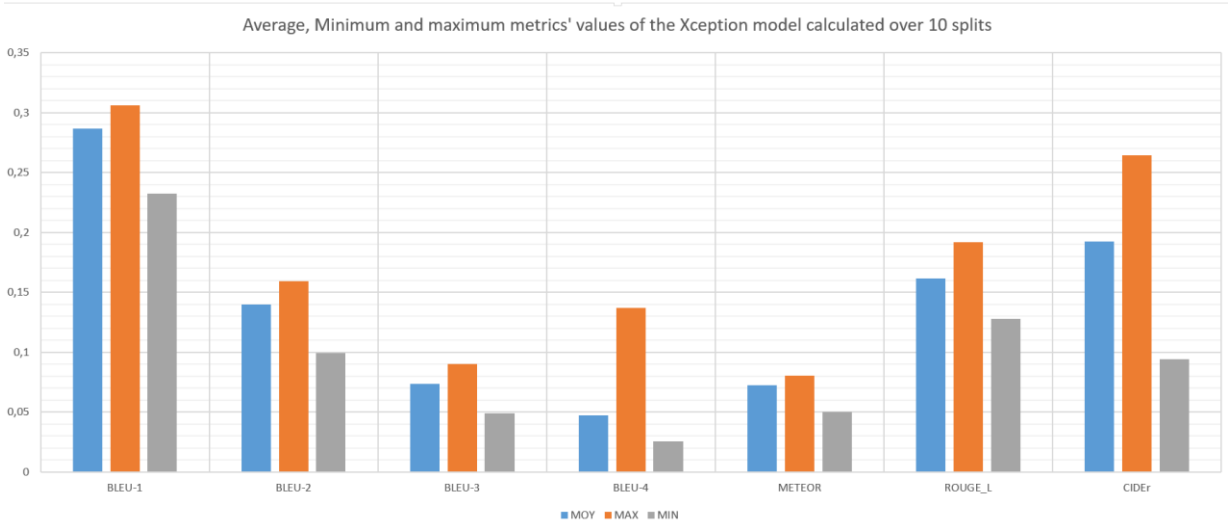


Figure III.30. Average, minimum and maximum evaluation metrics values of Xception after being trained over ten splits of the dataset

Table III.18. Average, minimum and maximum evaluation metrics values of Xception after being trained over ten splits of the dataset

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE_L	CIDEr
MOY	0,2868739	0,1401158	0,0738521	0,0472001	0,0722806	0,161320	0,1923994
MAX	0,305893	0,159373	0,090415	0,13694	0,080731	0,191858	0,264607
MIN	0,232496	0,099245	0,049023	0,025758	0,050202	0,127864	0,094285

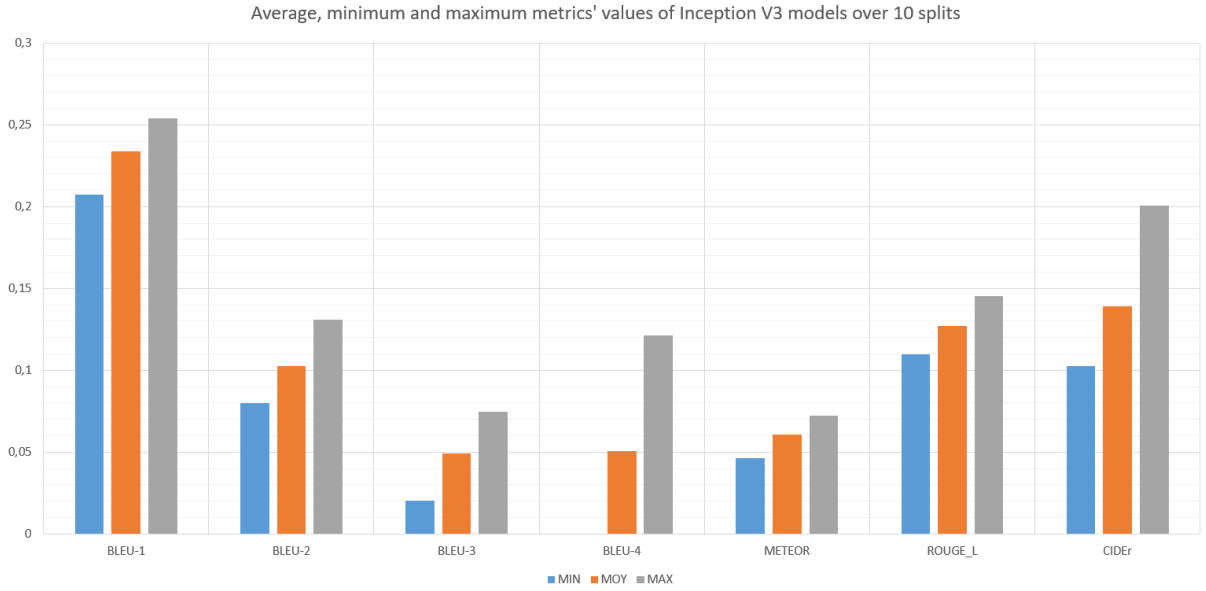


Figure III.31. Average, minimum and maximum evaluation metrics values of Inception-V3 after being trained over ten splits of the dataset

Table III.25. Average, minimum and maximum evaluation metrics values of Inception-V3 after being trained over ten splits of the dataset

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE_L	CIDEr
MIN	0,207192	0,079975	0,020218	0	0,046148	0,10959181	0,102491
MOY	0,233907	0,1024861	0,0489538	0,0503848	0,0607193	0,12707452	0,1388381
MAX	0,253794	0,131069	0,074581	0,121152	0,07244	0,14547747	0,200651

The table below will help us compare the results of natural language metrics' results obtained after training the two models on the entire dataset and on one split of it only.

Table III.26. Comparison of metrics results of the Xception model trained on one split and on the entire dataset

	The entire dataset		One split of the dataset	
Metrics	**Xception**	Inception-V3	**Xception**	Inception-V3
BLEU-1	**0,377294**	0,329641	**0,281305**	0,238055
BLEU-2	**0,2123175**	0,174141	**0,129812**	0,10113
BLEU-3	**0,122415**	0,094623	**0,067862**	0,048299
BLEU-4	**0,067639**	0,045752	**0,036218**	0
METEOR	**0,216328**	0,186669	**0,1429521**	0,06203
ROUGE_L	**0,231274**	0,210589	0,072661	**0,126289**
CIDEr	0,052406	0,076057	**0,176001**	0,115864

Discussion

- The difference between the maximum and minimum values remain small for each metric which mean that the results obtained over the ten splits are close enough and that the model was trained and evaluated on images close in nature.
- Comparing the metrics' results of the two models trained on the entire dataset and one split of it we notice that this decrease in size resulted in a noticeable de decrease in metric's result.

III.5.2.4 Conclusion

In this experiment, we studied the performances of two models with different encoders which are Xception and Inception-V3 models after training them on ten random splits of the previously studied dataset, than we compared the results obtained from all splits.

The results confirm the superiority of the Xception model over the other models in all aspects of evaluation.

Dataset size reduction resulted in: on one hand, a decrease in natural language evaluation metrics' results which can be due to the smaller vocabulary or the training sets didn't include enough data for the models to learn the nuance between all the images' assets. On the other hand we have an improvement in loss and accuracy values which can be due to giving the models lesser information to learn since the dataset size is reduced and increasing the number of epoch to give it enough time to better fit the data.

The results of ten splits are close which lead us to conclude that the data organization doesn't affect the results because the dataset in varied enough that the same nature of images in present in the training and testing sets.

The reduction in dataset and vocabulary sizes results in a decrease of natural language evaluation metrics results which means that the data choice affects the quality of the generated descriptions.

In all the experiments we have carried along this chapter, the Xception model has proven itself to be the most efficient model amongst the four.

General Conclusion

The main goal of this project was to develop an architecture that is able to generate an accurate description of a given image and compare different deep learning-based architectures.

We firstly assured the good functioning of our systems by training the models for ten epochs on the Flickr 8k dataset and compared their time consumption and evaluation results, this experiment showed a lot of time and material consumption by the four models with the efficiency of the Xception and Inception-V3 models. As a result, we carried a convergence study for the two best models by training them for fifty epochs, this study showed little improvement in both models performances. We proceeded then to study the influence of dataset's size on the four models' performances by training the models on one split of the dataset until their convergence, this resulted in lesser time consumption with noticeable changes in the evaluation results of all models with Xception and Inception-V3 taking the lead. These results lead us to study the influence of data choice and organization on the best two models' performances, this was achieved by training Xception and Inception-V3 models on ten subsets of the dataset for fifty epochs, this experiment showed the importance of data choice and its influence on the generated descriptions, also it confirmed the superiority and efficiency of the Xception model compared to the other architectures.

The huge time and material consumption were the main limitation of this project since the training of the models on a bigger training set produced better results than the smaller subsets but took very long period of time to achieve. Thus, the necessity of more powerful hardware to fulfil the task.

Nevertheless, the achieved results remain satisfactory and enabled us to reach our main goal

Perspectives:

- This project can be further developed by the study of the different RNN architectures and their influence on the decoder part of the model.
- Implementing attention mechanisms to the encoder-decoder architecture.
- Carrying studies on other dataset to further investigate the data choice's influence.

Bibliography

[1] https://www.americananthro.org/ImageDescriptions. Consulted: June 2022.

[2] https://www.sas.com/en_us/insights/analytics/what-is-natural-language-processing-nlp.html#:~:text=Natural%20language%20processing%20%28NLP%29%20is%20a%20branch%20of,the%20gap%20between%20human%20communication%20and%20computer%20understanding. Consulted: June 2022.

[3] Goldman N.M. 1974. Computer Generation of Natural Language from a Deep Conceptual Base. Ph.D. thesis, Stanford University, CA.

[4] McDonald D.D. 1980. Natural Language Production as a Process of Decision Making Under Constraint. Ph.D. thesis, MIT Artificial Intelligence Laboratory, Cambridge, MA.

[5] McKeown K.R. 1985. Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text. Cambridge University Press, Cambridge, England.

[6] Reiter E. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In Proceedings of the Seventh International Workshop on Natural Language Generation, pages 163–170.

[7] Hovy E.H. 2000. Language Generation. Entry for Encyclopedia of Cognitive Science (article 86). London: McMillan. (Invited encyclopedia article)

https://www.cs.cmu.edu/~hovy/papers/00ECS-article-NLG-Hovy.pdf. Consulted: July 2022.

[8] Merlino A., Morey D. and Maybury M. 1997. Broadcast news navigation using story segmentation. In Proceedings of the fifth ACM international conference on Multimedia MULTIMEDIA '97. Association for Computing Machinery, New York, NY, USA, pages 381–391.

[9] André E. and Martin J.C. 2014. Multimodal Systems. In The Oxford Handbook of Computational Linguistics (2nd edn), chapter 45. Oxford University Press, Oxford, pages 1071–1090.

[10] Jia SJ., Kong, X.W., Fu HY. and Jin G. 2010. Product Images Classification with Multiple Feature Combination. Proceedings of the 1st International Conference on E-Business Intelligence, Atlantis Press, pages 446-469.

[11] Kliegr, T., Chandramouli, K., Nemrava, J., Svatek, V. and Izquierdo, E. 2008. Combining Image Captions and Visual Analysis for Image Concept Classification. In Proceedings of the 9th International Workshop on Multimedia Data Mining, at the 2008 International Conference on Knowledge Discovery and Data Mining, pages 8-17.

[12] Aho A.V., Chang SF., McKeown, K.R., Radev D.R. and Smith J.R. 1998. Columbia digital news project: an environment for briefing and search over multimedia information. Int J Digit Library. Pages 377–385.

[13] Chan C.H. and Jones G.F.J. 2005. Affect-based indexing and retrieval of films. In Proceedings of the 13th annual ACM international conference on Multimedia (MULTIMEDIA '05). Association for Computing Machinery, New York, NY, USA, pages 427–430.

[14] André E., Rist T., Mulken S.V., Klesen M. and Baldes S., 2000, The Automated Design of Believable Dialogues for Animated Presentation Teams, In Embodied Conversational Agents, MIT Press, pages 220—255.

[15] Kliegr T., Chandramouli K., Jan N., Svátek V. and Izquierdo E. 2008. Combining Image Captions and Visual Analysis for Image Concept Classification. In: 9th International Workshop on Multimedia Data Mining (MDM/KDD 2008). Las Vegas: ACM, 2008, s. pages 8–17. ISBN 978-1-60558-261-0.

[16] Simonyan K. and Zisserman A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, 7-9 May 2015, pages 1-14.

[17] https://www.analyticssteps.com/blogs/common-architectures-convolution-neural-networks. Consulted: June 2022.

[18] Szegedy C., Vanhoucke V., Ioffe S., Shlens J., and Wojna Z. 2016. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818-2826.

[19] Kaiming H., Zhang X., Ren S., and Sun J. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770-778.

[20] Chollet F. 2017. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition. Pages 1251-1258.

[21] Sherstinsky A. 2020. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network, Physica D: Nonlinear Phenomena, Volume 404.

[22] https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Consulted: June 2022.

[23] Tanti M., Gatt A. and Camilleri K. 2018. Where to put the image in an image caption generator. Natural Language Engineering.

[24] Tanti M., Gatt A. and Camilleri K. 2017. What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator? In Proceedings of the 10th International Conference on Natural Language Generation, Santiago de Compostela, Spain, pages 51-60.

[25] https://wiki.tum.de/display/lfdv/Recurrent+Neural+Networks+-+Combination+of+RNN+and+CNN, consulted: June 2022.

[26] https://www.tensorflow.org/. Consulted: June 2022.

[27] https://keras.io/. Consulted: June 2022.

[28] https://jupyter.org/. Consulted: June 2022.

[29] https://colab.research.google.com/. Consulted: June 2022.

[30] https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6. Consulted: July 2022.

[31] Papineni K., Roukos S., Ward T., and Zhu WJ. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, Association for Computational Linguistics, pages 311–318.

[32] Lin CY. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In Text Summarization Branches Out, Barcelona, Spain. Association for Computational Linguistics, pages 74–81.

[33] Banerjee S. and Lavie L. 2005. 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, Ann Arbor, Michigan. Association for Computational Linguistics, pages 65–72.

[34] Vedantam R., Zitnick C.L., and Parikh D. 2015. Cider: Consensus-based image description evaluation. In CVPR, pages 4566–4575.

[35] https://www.kaggle.com/datasets/adityajn105/flickr8k, consulted: July 2022

ملخص

تهدف هذه المذكرة إلى المساهمة في إنشاء نموذج وصف آلي يعتمد على معماريات التعلم العميق لإنشاء أوصاف نصية شبيهة بالبشر للصور. يجمع هذا العمل بين الشبكات العصبية التلافيفية المختلفة (CNN) لجزء استخراج الميزات، والذاكرة طويلة المدى (LSTM) لجزء التوليد النصي. أسفر اختبار هذه النماذج من خلال سلسلة من التجارب عن نتائج مختلفة من حيث الخسارة والدقة ومقاييس تقييم اللغة الطبيعية. الكلمات المفتاحية: التعلم العميق، الشبكات العصبية التلافيفية (CNN)، الشبكات العصبية المتكررة (RNN)، الذاكرة الطويلة والقصيرة (LSTM)، معالجة اللغة الطبيعية (NLP)، الوصف الآلي للصور.

Abstract

This dissertation aims to contribute to create an automated description model based on deep learning architectures to generate human-like textual descriptions of images. This work combines different convolutional neural networks (CNN) for the feature extraction part, and long-term memory (LSTM) for the textual generation part. Testing these models through a series of experiments yielded different results in terms of loss, accuracy and natural language evaluation metrics.

Keywords: Deep learning, Convolutional neural networks (CNN), Recurrent neural networks (RNN), Long Short Time Memory (LSTM), Natural Language Processing (NLP), Automated Image Description.

Résumé

Ce mémoire vise à contribuer à créer un modèle de description automatisé basé sur des architectures d'apprentissage en profondeur pour générer des descriptions textuelles d'images. Ce travail combine des différents réseaux de neurones convolutifs (CNN) pour la partie extraction de caractéristiques, et réseaux de neurones récurrents à longue mémoire à court terme (LSTM) pour la partie génération textuelle. L'évaluation de ces modèles à travers une série d'expériences a donné des résultats différents en termes de perte, de précision et de métriques d'évaluation du langage naturel.

Mots-clés : Apprentissage profond, Réseaux de neurones convolutifs (CNN), Réseaux de neurones récurrents (RNN), Réseaux de neurones récurrents à longue mémoire à court terme (LSTM), Traitement automatique du langage naturel (NLP), Description automatisée des images.