



The Republic of Tunisia  
Department of Higher Education and Scientific Research  
University of Tunis El Manar  
National Engineering School of Tunis



**National Engineering School of Tunis**  
Information and Communication Technologies Department

## **Graduation Project Report**

**Mohamed Aymen KTARI**

Presented for the title of  
**TELECOMMUNICATIONS ENGINEER**

**State-of-the-art deep Learning neural architectures  
comparison, explainability and interpretability**

Research and Development Host Organization

**Natural Solutions and LIS Centrale Marseille**



*Defended on 07 / 12 / 2019 :*

President of the Jury	:	M. Faouzi BAHLOUL
Reporter	:	Mme. Imène ELLOUMI
ENIT Supervisor	:	M. Taoufik AGUILI
Natural Solutions Supervisor	:	M. Makki VOUNDY
LIS Centrale Marseille Supervisor	:	M. Thierry ARTHIERES

Academic year: 2018/2019

# Signatures

Natural Solutions signature and stamp

A large, empty rectangular box with a thin black border, intended for the signature and stamp of Natural Solutions.

ENIT Supervisor: Pr. Taoufik Aguil

A large, empty rectangular box with a thin black border, intended for the signature of the ENIT Supervisor, Pr. Taoufik Aguil.

## ***Acknowledgement and Dedication***

“First, praises and thanks to god, the Almighty, for his blessings throughout my work. Second, I wholeheartedly dedicate this graduation project to my dear parents and my little brother for their endless love, limitless support and continuous believe and encouragements.

I would like to express our deep gratitude to my supervisors Mr. Makki VOUNDY and Mr. Thierry Artieres for their patient guidance, valuable and constructive suggestions, enthusiastic encouragements and useful critiques through the planning and the development of the whole project.

My grateful thanks are also extended to Mr. Hachem Kadri, the headmaster of the Qarma Deep Learning Research Team at Centrale Marseille and Mr. Olivier Revellotti, founder and CEO of Natural Solutions.

Last but not least, I specially devote the work of this internship to Pr. Taoufik Aguil, my ENIT supervisor and SysCom Research Unit President and to all the respectable and honorable teachers and colleagues who tremendously contributed to my academic professional progression and self-Development.”

# Contents

<b>List of figures</b> .....	IV
<b>Introduction</b> .....	1
<b>I. Context and objectives</b> .....	2
<b>Introduction</b> .....	2
<b>1. Host organization presentations</b> .....	2
1.1. Natural Solutions .....	2
1.2. Laboratoire d'Informatique et des Systèmes .....	5
1.3. Ecole Centrale de Marseille .....	6
<b>2. Project's Context</b> .....	7
2.1. Technical Background and problem statement .....	7
2.2. EcoBalade .....	8
2.3. Objectives and perspectives .....	10
<b>3. Internship's progression</b> .....	12
3.1. Software Enviroment, packages and frameworks .....	12
3.2. AGIL Methododology, Scrum .....	16
<b>Conclusion</b> .....	18
<b>II. Project's preliminary study and technical overview</b> .....	19
<b>Introduction</b> .....	19
<b>1. AI and Deep Learning</b> .....	19
<b>2. Computer Vision: Image Recognition</b> .....	22
<b>3. Convolutional Neural Networks</b> .....	24
<b>4. State-of-the-art neural networks preliminary study</b> .....	29
4.1. Mobilenet .....	29
4.2. Inception .....	32
4.3. Resnet .....	33
4.4. VGG.....	34
<b>Conclusion</b> .....	34
<b>III. Image Recognition Model: design, deploy, validation and tests</b> .....	36
<b>Introduction</b> .....	36
<b>1. Mobilenet</b> .....	36

1.1	Preprocessing: Data Set and Data Augmentation .....	36
1.2	Features extraction .....	38
1.3	Fine Tuning and transfert learning .....	39
1.4	Validations and tests .....	39
<b>2.</b>	<b>Inception v3 .....</b>	<b>42</b>
2.1.	Preprocessing: Data Set and Data Augmentation .....	42
2.2.	PaaS: Colab, Amazon Web Services and Kaggle .....	42
2.3.	Features extraction .....	43
2.4.	Bottleneck approach and transfert learning .....	44
2.5.	Validations and tests .....	45
<b>3.</b>	<b>Inception Resnet v2 .....</b>	<b>47</b>
3.1.	Preprocessing: Data Set and Data Augmentation .....	47
3.2.	PaaS: Colab, Amazon Web Services and Kaggle .....	47
3.3.	Features extraction .....	48
3.4.	Bottleneck approach and transfer learning.....	49
3.5.	Validations and tests .....	50
<b>4.</b>	<b>State-of-the-art neural architectures Comparison .....</b>	<b>51</b>
4.1.	Summurizing results .....	51
4.2.	Intersections and Differences .....	52
4.3.	Summary and perspectives .....	54
	<b>Conclusion .....</b>	<b>55</b>
	<b>General Conclusion .....</b>	<b>56</b>
	<b>References .....</b>	<b>57</b>

# List of figures

Figure 1: Natural Solutions Logo.....	2
Figure 2: Natural Solutions used and provided technologies.....	3
Figure 3: Laboratoire d’informatique et Systèmes Logo .....	5
Figure 4: Centrale Marseille Logo .....	6
Figure 5: EcoBalade Logo.....	9
Figure 6: Screenshots from the EcoBalade Application.....	9
Figure 7: Tensor Board GUI set-up and visualizations .....	14
Figure 8: tensor flow respectful results compared to concurrent frameworks and packages	16
Figure 9: Graphic simplified representation of the AGIL Methodology.....	17
Figure 10: Scrum Manager Sprints and Data Visualizations .....	18
Figure 11: The historical timeline of AI.....	20
Figure 12: AI Research and Development Timetable.....	21
Figure 13: Simplified representation of a binary neuron .....	25
Figure 14: CNN Layers and functions simplified representation .....	26
Figure 15: Different CNN architectures, layers and functions.....	28
Figure 16: MobileNet architecture simplified diagram.....	30
Figure 17: State-of-the-art neural architectures Accuracy-Density comparison.....	31
Figure 18.1: Inception v3 simplified diagram.....	32
Figure 18.2: Inception ResNet v2 simplified diagram.....	33
Figure 19: State-of-the-art neural networks Image per second Top-1 accuracies .....	34
Figure 20: State-of-the-art neural networks G-FLOPx operations Top-1 / Top-5 accuracies.	35
Figure 21: MobileNet Model diagonal Confusion Matrix .....	38
Figure 22: MobileNet Model’s Validation accuracy .....	40
Figure 23: Screenshot of MobileNet Model’s raw prediction while testing.....	40
Figure 24: Screenshot of MobileNet Model’s Tests.....	41
Figure 25: Inception v3 model’s diagonal Confusion Matrix.....	44
Figure 26: Screenshot of Inception Model’s Validation Accuracy .....	45
Figure 27: Screenshot of Inception Model’s raw prediction while testing.....	46
Figure 28: Screenshot of Inception Model’s Tests.....	46
Figure 29: ResNet Model’s diagonal confusion Matrix.....	49
Figure 30: Screenshot of ResNet Model’s Validation Accuracy.....	50

*Figure 31: Screenshot of ResNet Model's raw prediction while testing.....50*  
*Figure 32: Screenshot of ResNet Model's Tests.....51*  
*Figure 33: Comparison summarizing Table.....52*  
*Figure 34: Inception model summary.....53*  
*Figure 35: Resnet model summary.....53*  
*Figure 36: MobileNet model summary.....54*

# Introduction

Artificial Intelligence, Data Science, neural architectures and Deep Learning are ones of the top trending ICT engineering fields nowadays, making a deep technological intersection with a variety of disciplines such as mathematics, computer science and Telecommunications building the ultimate bridge to humanity's modernity, innovation and comfort.

In fact, the human continuous need for high performances and extremely accurate mission-critical results provides a set of problems, challenges and struggles making the whole focus of Data scientists, developers and ICT engineers in order to design, build, develop and optimize models and algorithms that fit with the technical objectives and assure the desired accuracy for a better quality of life and a heavy economic and political weight at a very concurrently ICT international market.

Therefore, Research and Development engineers are the real key to look toward linking the experimental work from the laboratory with the compromises and expectations from the firm, the project partners or the clients. R&D is the strongest path for a young freshly graduated engineer toward a better professional development, greater and clearer specialization and deeper understanding of the subject of interest.

The internship is divided into two big parts. The first part provides continuity to a previous work done on the Natural Solutions EcoBalade Application in which we try to design, build and evaluate an image recognition neural network model based on a huge 55 class Data Set of vegetal species. Through that model, we compare some state-of-the-art neural architectures in order to get familiar with their features, technical and mathematical background in order to finally choose the algorithm that fits the most with our needs in terms of accuracy, time delay and memory consumption.

The second part is mainly a research and development work in which we study the concept of explainability and interpretability, one of the most demanded innovations in terms of friability. The main purpose is obviously to create a secure transparent confidence between the machine and the human been. Interpretability is a post-graduation-project technical objective providing the main concern and focus of the internship's future perspectives.

# I. Context and objectives

## Introduction

The first chapter of this report is mainly introductive. In fact, it provides an overview about the host organizations Natural Solutions and Centrale Marseille, the project's main problematic and context, the provided solutions and a variety of technical objectives and perspectives.

### 1. Host Organizations presentations

#### 1.1 Natural Solutions

Natural Solutions is a digital agency specialized in environmental innovation. In fact, Natural Solutions develop and create custom digital solutions for nature, biodiversity, the environment, and land management: websites, mobile applications and softwares.

Moreover, the main responsibility is to deeply consider the specific problems related to ecology and environment while ensuring the sustainability of customer's data. Indeed, thanks to the multi-disciplinary nature of Natural Solutions teams and their expertise in data collection, management, storage and exploitation providing to the clients precious advising and consulting for their projects and demands.



*Figure 1: Natural Solutions Logo [3]*

#### Natural Solutions values:

- **Technicity:** Passion for our business, the web, IT and ICT.
- **Respect:** Customers, partners, cooperating laboratories.
- **Innovation:** curious and creative open minds
- **Ecology:** Respect for resources, motivation, passion and love for the Mother Nature.

## The team:

Natural Solutions team is made up of talented web and mobile developers sharing an open workspace with trainees and employers providing a beautiful professional environment for more creative and efficient work.

The Natural Solutions team profiles combine a variety of skills and experiences (GIS, Web, Mobile) to provide clients with solutions tailored to their needs in the field of nature, biodiversity, environment, and space (territory) general management.

The main work methodology is based on Agile ("We are discovering how to better develop software through practice and by helping others to do so.")

These experiences have led Natural Solutions to value: [3]

- Individuals and their interactions more than processes and tools
- Operational software more than comprehensive documentation
- Collaboration with customers more than contract negotiation
- Adapting to change more than following up on a plan

## Technologies:

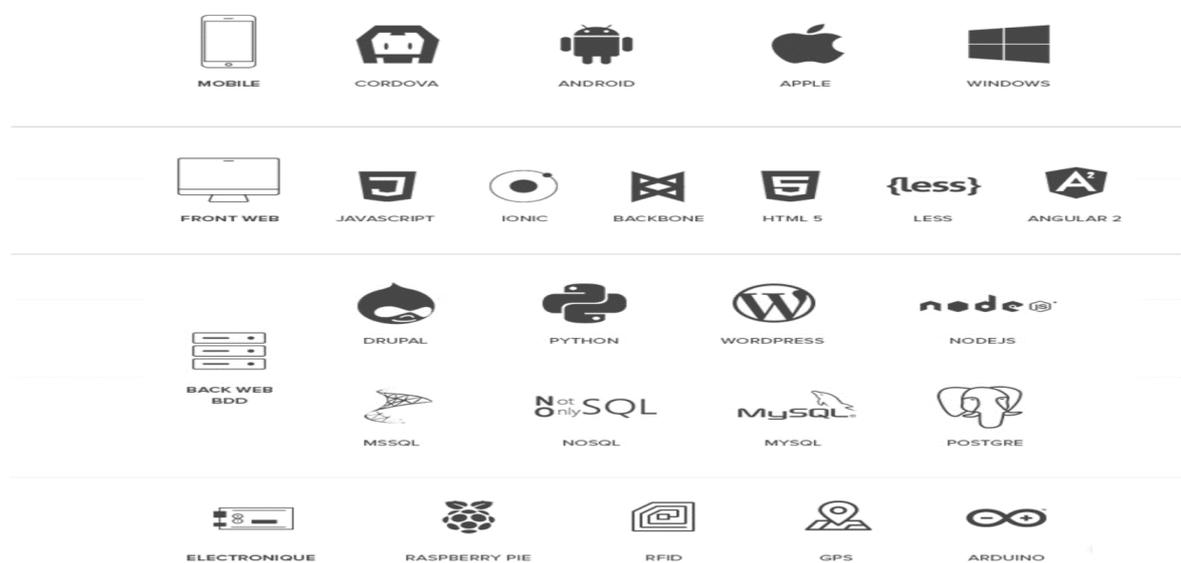


Figure 2: Natural Solutions used and provided technologies [3]

## Research and R&D Programs:

Biodiversity protection and more specifically the threats related to human activity on the environment lead to new scientific and technical challenges and issues.

The "biodiversity informatics" or "eco-informatics" is a recent discipline at the crossroads between the environmental sciences and computer science. Described as "ecological and digital revolution" by the Ministry of Environment, Energy and the Sea in France, it requires the development of new digital services for scientists, professionals and citizens.

It is in this context that Natural Solutions was founded in 2008, with the aim of developing innovative solutions for collecting and managing environmental data.

- **Program #Human4Life:** Human Machine Communication, Ergonomics, Learning Environments, Human Machine Interaction
- **Program #IA4Life:** Artificial Intelligence Learning, Knowledge Engineering, Planning, Meta-Heuristics, Multi Agent Systems
- **Program #IOT4Life:** Communicating Objects, Internet of Things
- **Program #DATA4Life:** Information system, databases, DBMS, data management in the cloud, data mining, semantic web

### **Achievements [3]:**

Our solutions have been awarded several times for their innovative character:

- Winner of the Eco-Industries Call for Projects, 2009
- Best Mobility Project 2010 in the PACA region
- Best Terrain Capture Tool (TDWG 2010)
- Créa13 Award, 2011 and 2012
- Eurasia Wings Trophies 2012
- Winner of the EOL Challenge (Encyclopedia Of Life) Education Innovation 2012
- eTourism Innovation Award (2014)
- Finalist of the company prize and biodiversity (2015)
- Finalist Smart city app hack (2015)
- OpenData Paca Winner (2015)

## **1.2 Laboratoire d'Informatique et Systèmes**

The LIS Laboratory of Computer Science and Systems is a Joint Research Unit (UMR) under the supervision of the National Center for Scientific Research (CNRS) attached to the Institute of Information Sciences and their interactions (INS2I), the University of Aix-Marseille (AMU) and the University of Toulon (UTLN). Ecole Centrale de Marseille is also a main partner of the LIS. Its premises are located on the campuses of Saint-Jérôme and Luminy in Marseille and on the campus of the University of Toulon. This laboratory groups together research activities mainly covered by sections 06 and 07 of the CNRS and sections 27 and 61 of the CNU. The LIS brings together more than 375 members including 190 permanent researchers and research professors and 20 IT / IATSS.

The LIS conducts fundamental and applied research in the fields of computer science, automation, signal and image. It is made up of 20 research teams and structured in 4 areas: the Computing Pole, the Data Sciences Pole, the Systems Analysis and Control Pole and the Signal and Image Pole.



*Figure 3: Laboratoire d'informatique et Systèmes Logo [8]*

The conducted research and R&D at LIS usually find a finalization in different application areas such as transport, health, energy, environment, defense, etc. The LIS thus has a strong link with the socio-economic world and a significant contractual activity. These numerous valorization activities allow it to be involved in several competitiveness clusters (the Marine division, the SCS Secure Communicating Solutions Cluster, the Risk Cluster, the Euro biomed cluster and the OPTITEC cluster) and to be a member of the Carnot STAR Institute. One of the notable characteristics of the LIS is the multidisciplinary of the competencies it brings together. [8]

In fact, this range of complementary skills allows the LIS unit to be involved in several national and local structuring actions such as the ILCB convergence institutes "Language, Communication and Brain Institute" and Centuri "Turing Center for Living Systems", as well as at the 'Archimède' institute of the Midex Initiative of Excellence, bringing together research activities in Mathematics, Computer Science and Interactions at the Aix-Marseille and Toulon sites.

The researchers, PHD students, R&D engineers and professors of the LIS unit are involved in various courses at the University of Aix-Marseille and the University of Toulon (IUT, Licenses, Masters, Polytech Marseille engineering school, SeaTech engineering school Toulon) as well as the Ecole Centrale de Marseille (ECM) in the fields of computer science, electrical engineering and automation.

### 1.3 Ecole Centrale de Marseille

The École Centrale de Marseille is a leading graduate engineering school (part of the “Grandes Ecoles Françaises”, the biggest and most known engineering schools in France) located in Marseille, the second largest city in France and the capital of the south.

The École Centrale de Marseille was mainly created in 2006 by the merge of a variety of engineering institutions and has its origins from the “École d'ingénieurs de Marseille” founded in 1890. It is one of the Centrale Graduate Schools (Paris, Lyon, Lille, Nantes, Marseille and Beijing) and a member of the TIME (Top Industrial Managers for Europe) network.

Therefore, Centrale Marseille is a multidisciplinary school, where the great majority of local and international students have endured two or three years of intensive Mathematics and Physics training (known as preparatory in France or the Pre-Engineering Curriculum):

- Mechanical engineering
- Chemical engineering
- Physics, optics and electrical engineering
- Business Administration and Finance
- Mathematics and computer science



*Figure 4: Centrale Marseille Logo*

The students can also complete their last year in one of the other French Centrale Graduate Schools or be part of an exchange program (inspired by the diversity of partnerships, corporations and collaborations with different Universities, schools, laboratories and engineering firms).

There are three-years PhD programs available in all the aforementioned domains of research and Central Marseille provides an excellent environment for Research and Development projects and theses.

In a parallel way to my work at NS, I had to go several times at Central Marseille not only for the experimental research work but also for the resources (Access to premium PaaS to run the heavy neural architectures...), expert teams and work mentoring. My host AI Unit is called QARMA which provided the precious supervising of Mr. Hachem Kadri and Mr. Thierry Artieres.

## **2. Project's Context and perspectives**

### **2.1. Technical background and problem statement**

The graduation project provides a solution for the firm's need to set and deploy an Image Recognition model based on several neural architectures from the state-of-the-art of deep learning. In fact, the need for implementing and deploying AI models and solutions is on the radar of natural solutions in a major part of its provided solutions to customers.

In fact, the idea of letting the machine systematically identify and classify vegetal or animal species from an input photographed customer's photo is the main goal to achieve.

Throughout building the model, the project is accordingly a comparative study of the top trending nowadays neural architectures providing the intersections and the differences of each and every algorithm.

Moreover, this study will lead to the ultimate choice of the best-fit algorithm to our problem statement and desired accuracies and metrics.

For instance, interpretability and explainability are the perspectives post-graduation project offering a complete detailed study of these concepts illustrating increasingly their crucial importance on the AI market nowadays building the need trust between the machine and the human being.

The technical background of the project is a set of skills in the multi-disciplinary AI Engineering field, specifically in terms of deep learning, Convolutional neural networks, deep neural networks, Data science and augmentation, PaaS implementations (Google Colab, Amazon Web Services and Kaggle) and python coding language and frameworks.

The need for more complete, modern and sophisticated version of the App and the Website requires the ultimate need of being open to the beautiful universe of AI, Deep Learning and Computer Vision. The main idea is to let the machine determine systematically -thanks to an increasingly efficient State-of-the-art model- the identity of the picture (that the users took instantly while walking and discovering). Therefore, the user will get the name of the animal or vegetal specie and a complete description of its history, forms, territories, timelines...

Thanks to the explicability and interpretability concepts, the EcoBalade Developers team the user will be more comfortable and secure with the predictions given by the machine. The goal to achieve is obviously providing automatically the right explanations for a given result. Furthermore, NS are thinking about designing a Chabot in order to assure an appropriate discussion between the user and his device, making the customer feel more comfortable and convinced with the returned classifications, predictions and related explanations.

The work is divided into 2 major parts done through two different workspaces. The laboratory work is closely a set of experiences, manipulations and documentations. On the other hand, the firm provides an excellent team of designers, developers and extremely motivated trainees and supervisors, making the perfect environment for building, processing and developing the State-of-the-art model and its related highly large exclusive Dataset.

The cooperation between LIS and NS illustrates the need for a bigger Research and Development investment around the engineering world. Regarding the specificities of each workspace and its considerably important added value on project, the engineering Market nowadays is increasingly more and more open to R&D offering to engineers, researchers, PHD students and trainees a tremendous set of opportunities, perspectives and paths for a better professional development.

## 2.2. EcoBalade

EcoBalade is a mobile application that offers a discovery of fauna and flora, during walks or hikes.

In fact, the application is free downloadable and offline usable. EcoBalade transforms walks into new outdoor activity, where biodiversity and heritage are within the reach of all.

In addition, it aims to promote and develop the territories, through the discovery of their biodiversity and their nature.

For families, schoolchildren, EcoBalade is deeply a teaching tool. It makes it possible to approach the themes of the biodiversity, the protection of the environment, eco-citizenship...



Figure 5: EcoBalade Logo [2]

## EcoBalade, the App

Available for free download on Play Store (Android) and Apple Store, the EcoBalade application allows you to select and record one or more rides on your smartphone anticipating the courses you want to perform. Once downloaded, they are then searchable in total autonomy, even without an internet connection which explains the need to deploy an image recognition model with full access on Offline mode. This will be one of the most important metrics required for the firm to evaluate the model. [2]

The application proposes, for each walk, a list of potentially visible species (birds, plants ...) in its immediate environment. Each species is illustrated with photos and info graphics to enrich and deepen his knowledge. A determination key facilitates the recognition of plant species. Trees or flowers observed, are easily identifiable in a few clicks. The observations collected during the walks can finally be recorded in a field notebook, for later consultation.



Figure 6: Screenshots from the EcoBalade Application [2]

## EcoBalade, the website:

Ecobalade.fr is the main entry point for the discovery of the EcoBalade service. On the site, the visitor can select and prepare a ride by sorting by type of circuit, difficulty, distance to travel or geographical location. It mainly has access to not only walks but also to all species sheets (more than 1500). Users can also post comments or photos of their walks in order to keep both sharing and interacting. [2]

## EcoBalade, the paper book:

EcoBalade propose to customers the edition of a paper guide as a prolongation of the Internet website and the application. Moreover, it presents and describes the walks and informs the user or the reader about the emblematic species likely to be encountered during the ride, thanks to very complete detailed sheets. A system of QR codes returns to the website [www.ecobalade.fr](http://www.ecobalade.fr) to obtain additional information and go further, before, during or after the ride.

### 2.3. Objectives and perspectives

The 3 months graduation project is a set of objectives related to the needs of the firm Natural Solutions and the laboratory at Centrale Marseille. The main context is a research and development work in which I try to reach a variety of goals:

- Design build and test an Image recognition model for the EcoBalade application at Natural Solutions. In fact, the idea is that the customer should be able to take an instant photo of a flower or an animal while walking around somewhere in the beautiful nature of France and the machine systematically predict the name and a little detailed description of the desired fauna and flora taken from the Data Base of the application. In fact, Image recognition is targeted by the firm as one of the future feature implementations in their apps and provided services to customers.
- Scrap and build a giant Data Set holding the all the covered species on the website of the application. Therefore, depending on the given neural architecture, every class is pre-processed throw some operations of filtering, re-sizing, re-sampling, rotating, zooming... In order to get the adequate Data Augmentation.
- State-of-the-art neural networks comparative study in order to get the intersections and the differences of every built model. The main goal to achieve is to choose the most adequate algorithm regarding our application and the firm's needs. To crop it all, the focus is on the ratio between the quality of service (mainly precision, accuracy and time delay) and the given resources (hardware architecture of the user's device and its performances in terms of processing and memory)
- For Centrale Marseille, the comparative study will also be an opportunity to get familiar with the state-of-the-art deep learning algorithms so that we can not only compare the architectures themselves but also compare the experimental work with the theory and the technical background regarding every proposed model (SqueeZNet, MobileNet, Inception Resnet, Inception v3, VGG...)
- The interpretability is a deep concept being a trending research field in AI and Deep Learning. Reasonably, it's impossible to get a concrete result for just one and half month of work on this concept. The explainability is a set of perspectives for my graduation project. Therefore, my mission is to **start** designing and building a simple trustworthy model being able to defend itself and to argue with the user concerning its choice and prediction.

NB: The interpretability objectives are accordingly the perspectives of this project (it's a post-PFE offered opportunity)

- ✓ Central Marseille offers a "Cifre" R&D PhD in a partnership with NS to design and build a trustworthy model in which predictions are defended through the features extracted and to study and compare different interpretability approaches and algorithms.
- ✓ Natural Solutions "post-PFE" main perspective consists on building a Chabot to assure an automatic trustworthy communication between the user and the machine. This Chabot will provide the right explanations to the machine's predictions regarding the questions of the user.
- ✓ One of the perspectives is obviously to implement the relatively "best" chosen Model on their Application (NS started the work right after my internship). We also had the idea to add a quiz little gaming allowing the user to participate in predictions (for some specific eco-walks) and to let the machine systematically evaluate the user's knowledge about fauna and flora. Sounds fun to let people learn while having fun and getting more familiar with our beautiful mother nature.
- ✓ We should never forget that the work already started for an analogue version of those models working on classifying the animal's species. The idea from EcoBalade and the main spirit of Natural Solutions is to provide the best applications for both Fauna and Flora.

To crop it all, the whole internship took just 3 months in France. Technically, it was impossible to get to work on explicability. In fact, building a model from the very beginning for each and every architecture separately and providing a continuous work of ameliorating, validating and testing in order to optimize and regulate the architectures takes more than that. The main idea is to assure the continuity of my work through the 4 perspectives I mentioned below.

### **3. Internship's Progression**

#### **3.1. Software Environment, packages and Framework**

For Machine/Deep Learning projects, there are more than thirty known and recognized frameworks for programming neural networks for deep learning. In order to simplify this list, we are going to make a first sort selecting from "Open Source" frameworks, which obviously work on Linux, Mac and Windows platforms.

In addition, there are only 14 potential frameworks that use the programming language C++ (8 frameworks) and python (7 frameworks). I made the choice to use the python programming language. Later in this chapter, I will justify the Python and Tensor Flow

choices. Finally, we have the choice among the following 5 open source frameworks, all running on Windows, Mac and Linux:

- **Torch** Facebook AI Research, Twitter, Google Deep Mind
- **Theano** University of Montreal, 2009
- **Caffe** Berkley Vision and Learning Center (BVLC), 2013
- **TensorFlow** Google Brain, 2015
- **Keras** François Chollet, 2015

**Torch :** 

Torch is a scientific computing framework with wide support for machine/deep learning algorithms and models that puts GPUs first. It is easy to use and efficient, thanks to an easy and extremely fast scripting language, LuaJIT, and an underlying C/CUDA implementation.

## theano

Theano is a Python library that allows users to define, design, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano features:

- NumPy tight integration– `numpy.ndarray` in Theano-compiled functions.
- GPU transparent use– Perform data-intensive computations much faster than on a CPU.
- Efficient symbolic differentiation – Theano mainly does derivatives for functions with one or many inputs.
- Optimizations in terms of speed and stability – Get the right answer for  $\log(1+x)$  even when  $x$  is really tiny.
- Dynamic C code generation – Faster evaluations.
- Extensive unit-testing and self-verification – Detect and diagnose many types of exceptions and errors.

## Caffe

Caffe is a deep learning Python framework made with expression, modularity and speed. It was developed by Berkeley AI Research (BAIR) and community contributors. Yangqing Jia created the project during his PhD at UC Berkeley. In fact, Caffe is released under the BSD 2-Clause license.

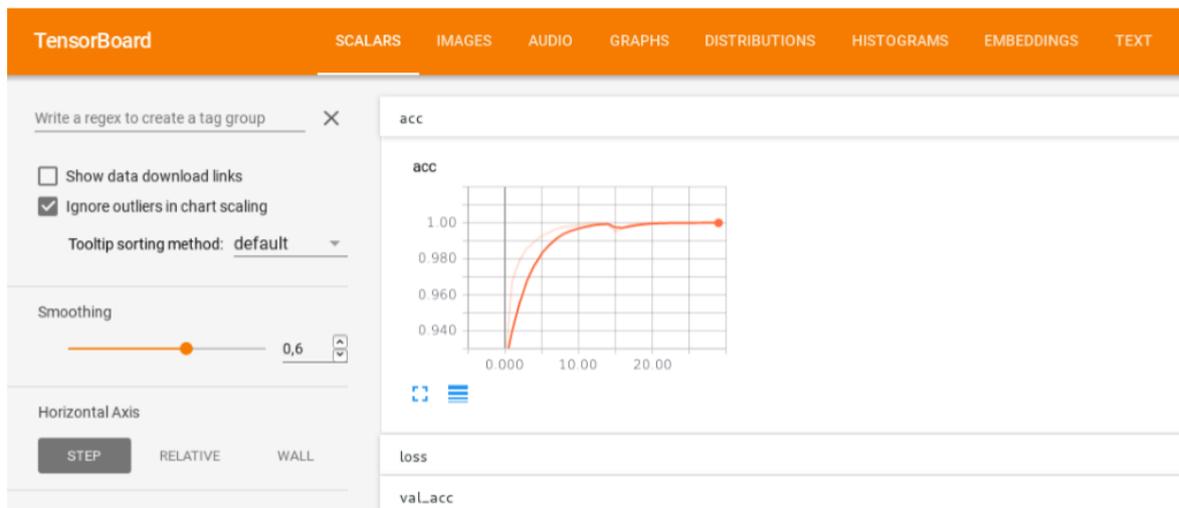
There are many types of learning architectures with a strong focus on image classification and image segmentation. It is an excellent convolutional model; it is one of the most popular tools in the field of computer vision. (“Caffe: Convolutional Architecture for Fast Feature Embedding”) [Ref]

## Tensor Flow:

Let’s first notice that Google developed a first version of its machine/deep learning system called DistBelief a few years ago. The latter has continued to be improved in order to simplify the source code to make it a faster and more robust and efficient library. This is how Distbelief step by step becomes TensorFlow. Moreover, Tensor Flow becomes open source in 2015. The new version 1.0 of TensorFlow includes many python APIs like Keras (see 2.3.4). It also integrates "TensorBoard" which is a very popular visualization tool for network modeling and performance. Therefore, TensorBoard allows the user to monitor and visualize a variety of parameters while training a neural network. In fact, this tool allows us to visualize results via a complete and simple graphical interface.

For example, we can visualize as illustrates the figures below:

- the precision curve of our learning model during training.
- the structure of our neural network.



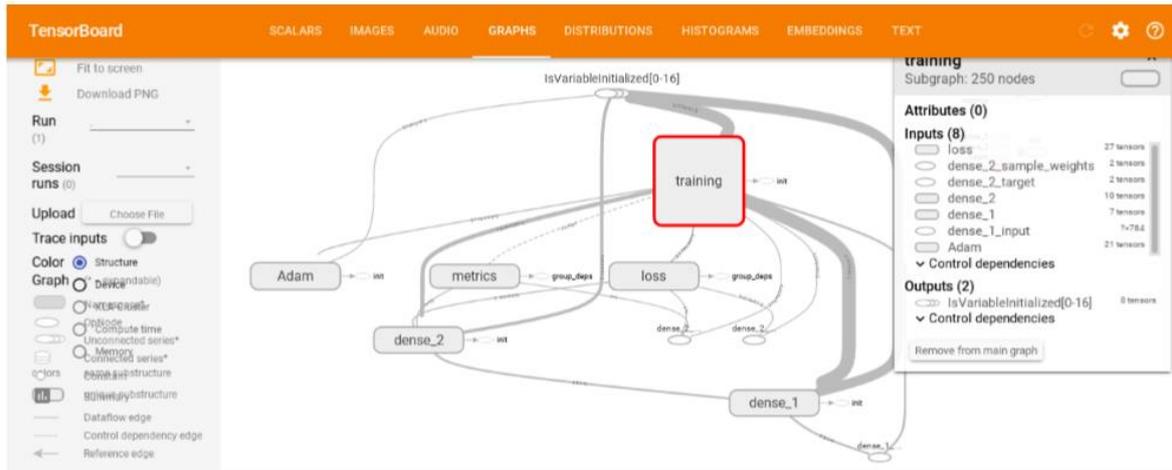


Figure 7: Tensor Board GUI set-up and visualizations [1]

In parallel way, in Tensor Flow, nodes are called Tensors and we've got two types of tensors: placeholders and variables. Therefore, Tensor Flow runs within sessions and we can analogously think that placeholders are buckets into which our data is placed once we build that session. In addition, let's notice that Tensor Flow uses the Gradient Descent Optimizer.

## Keras:

Keras is a library made for deep learning in neural networks. Written in python, it is easy to use and wraps both Tensor Flow and Theano. In fact, it has many pre-trained networks and uses a simple structure adapted to deep learning which provides a very easy and efficient tool for most Data Scientists and ML/DL engineers. [11]

## Python

Python is a general-purpose interpreted, high-level, programming language introducing itself as one of the increasingly most popular and used programming languages in the universe of ICT Engineering and in AI multidisciplinary fields and engineering domains. Created by Guido van Rossum who received the 2001 Award for the Advancement of Free Software from the Free Software Foundation (FSF) for his work on Python.

Python was first released in 1991. As a matter of fact, Python's design philosophy mainly emphasizes code readability with its notable use of significant whitespace (which probably is the biggest innovation contributing step by step to the worldwide shining of Python). Moreover, its language constructs and object-oriented approach aim to help programmers write clear, simple, logical code for small and large-scale projects

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming and is usually described as a "batteries included" language due to its comprehensive standard library.

Python was used throughout the whole project. The choice of Python is deeply justified with the huge number of frameworks, libraries, open source environments, increasingly large community worldwide. For our deep learning project, Python provides a diversity of tools and importations assuring its quality and simplicity especially in Artificial Intelligence, Data Science, Machine/deep Learning, Big Data, IoT and Cloud Computing. Python is everywhere! And that probably proves that nowadays ICT engineers should get more familiar with it and never miss the opportunity to get cooperate and continuously learn about new updates and features.

### Justifications:

Regarding the increasingly growing popularity of Tensor Flow, probably thanks to its ease of use and power, we strongly believe it is extremely interesting to use this programming framework to anticipate future needs. Indeed, Tensor Flow is continuously gaining popularity. As shown in the graph below, it is today the ultimate most used framework for deep learning worldwide.

In addition, this framework includes very functional APIs, extremely good documentations provided with many examples, help features and tutorials. Besides, it supports CPU (processor) and GPU (graphics processor) calculations and it has very short compilation times. Finally, TF is multi-platform (MAC OS, Linux, Windows or even Android and IOS).

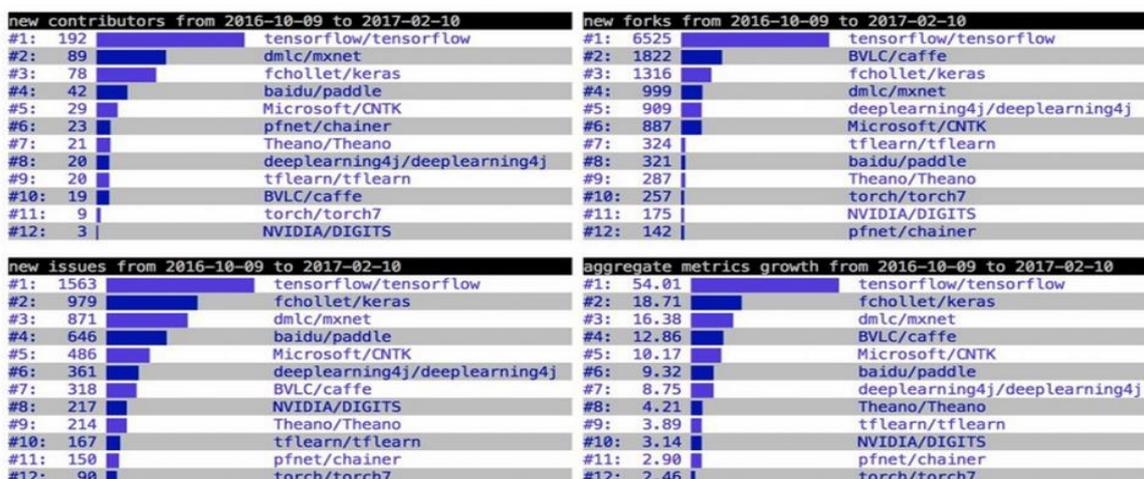


Figure 8: tensor flow respectful results compared to concurrent frameworks and packages [1]

### 3.2. AGIL Methodology and Scrum

**Agile software development** is an approach or a methodology under which requirements and solutions are based on the cooperative effort of self-organizing and cross-functional teams and their customer. In fact, it advocates continuous improvement, adaptive planning, evolutionary development, early delivery and it encourages rapid and flexible response to change.

The term agile was popularized by the Manifesto for Agile Software Development. The values and principles espoused in this methodology were derived from and underpin Scrum, Kanban and a large broad range of software development frameworks

While there is much anecdotal evidence that adopting agile practices and values improves the agility of software professionals, teams and organizations, some empirical, consultant and experts' studies have disputed that evidence.

AGIL Software development is based on several values such as:

- Individuals and Interactions over structured processes and tools
- Working Software over comprehensive references and documentations
- Continuous collaboration and negotiation with the customer
- Tools and processes are extremely important but it is more advantageous to have competent right people working together effectively.
- Good documentation is mainly useful in order to help people understanding how the software is built and how to use it, but the real main point from development is to create, design and build the software, not documentation.
- A contract is important but is no substitute for working closely with customers to discover what they need.
- A project plan is interesting, but it must not be too rigid to accommodate changes in technology or the environment, stakeholders' priorities, and people's understanding of the problematic and its provided or designed solution.

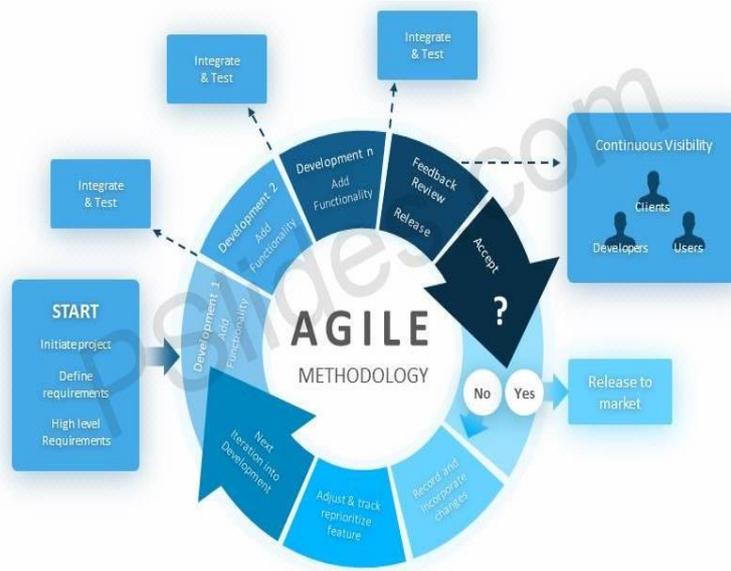


Figure 9: Graphic simplified representation of the AGIL Methodology

Compared to traditional software engineering and classic development methodologies, agile philosophy targets complex systems and product development with non-linear, dynamic and non-deterministic characteristics. Accurate estimates, stable plans, and predictions are often hard to get in early stages, far from being trustworthy in the beginning. Agile practitioners increasingly seek reducing the leap-of-faith that is needed before any evidence of value can be obtained.

### Scrum:

Scrum is an agile process framework for handling and managing complex knowledge projects, with a deep initial emphasis on software development, although it has been used in other disciplines and fields and is increasingly starting to be used and explored for other complex work, research, R&D and advanced technologies. It is designed for teams of ten or fewer members, who handle their work through goals that can be completed within time boxed repetitive iterations, called sprints, generally no longer than one month and most commonly two weeks, then track progress and briefly discuss that in 15-minute time-boxed stand-up meetings, called daily scrums. The re-planning is discussed through the retrospective weekly session.

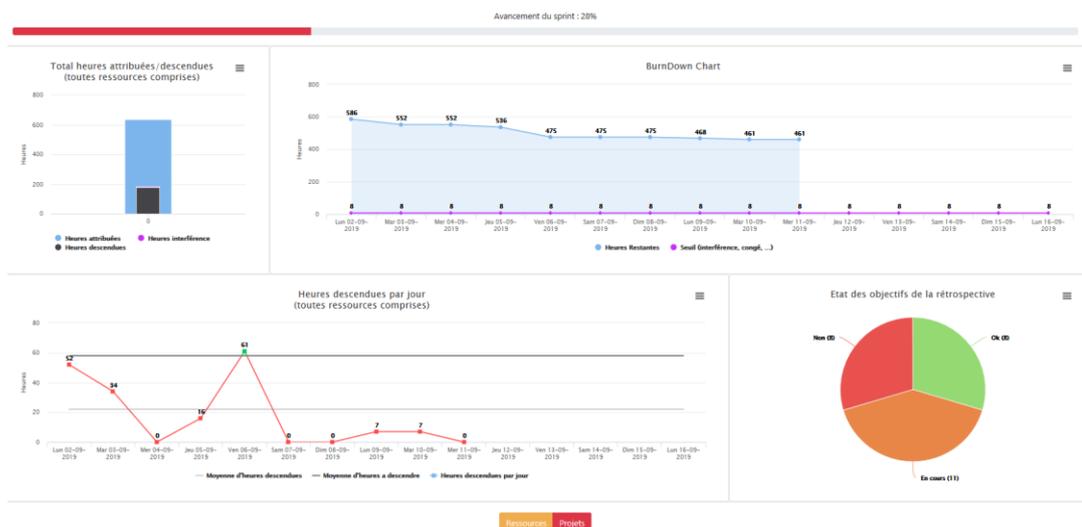


Figure 10: Scrum Manager Sprints and Data Visualizations

At Natural solutions, throughout different steps of the graduation project, getting familiar with Agile and Scrum gave me the ability to understand the concept of planning, estimating time and resources while being open to repetitive set ups and updates depending on the workload different continuous needs. Besides, I set up the plan for the next Sprint after discussion it with my supervisor; we trace the goals for an average of two weeks for every sprint, giving every step of the project the most optimum amount of time. Therefore, I am found of working in such a corrective and adaptive environment based on sprints, planning, daily scrum, reviews all systematically leading to the firm general Retro perspectives and grooming (called the Backlog refinement which is the ongoing continuous process of reviewing product backlog items and double checking that they are appropriately prepared and ordered in a way that makes them clear and executable for teams once they enter sprints via the general sprint planning activity.)

The next chapter provides a bibliographic overview for the project's keywords and highlights. It is mainly a preliminary study of the algorithms behind the state-of-the-art neural architectures.

# II. Project's preliminary study and technical overview

## Introduction

Getting a bibliographic study for a deep learning project is a primordial phase before designing any model. AI, deep learning, computer vision and python programming are all about documentations and bibliographic research.

Regarding the problem statement, this chapter provides a brief overview of the paradigms, algorithms and approaches accordingly related to our context, work objectives and perspectives.

### 1. AI and Deep Learning

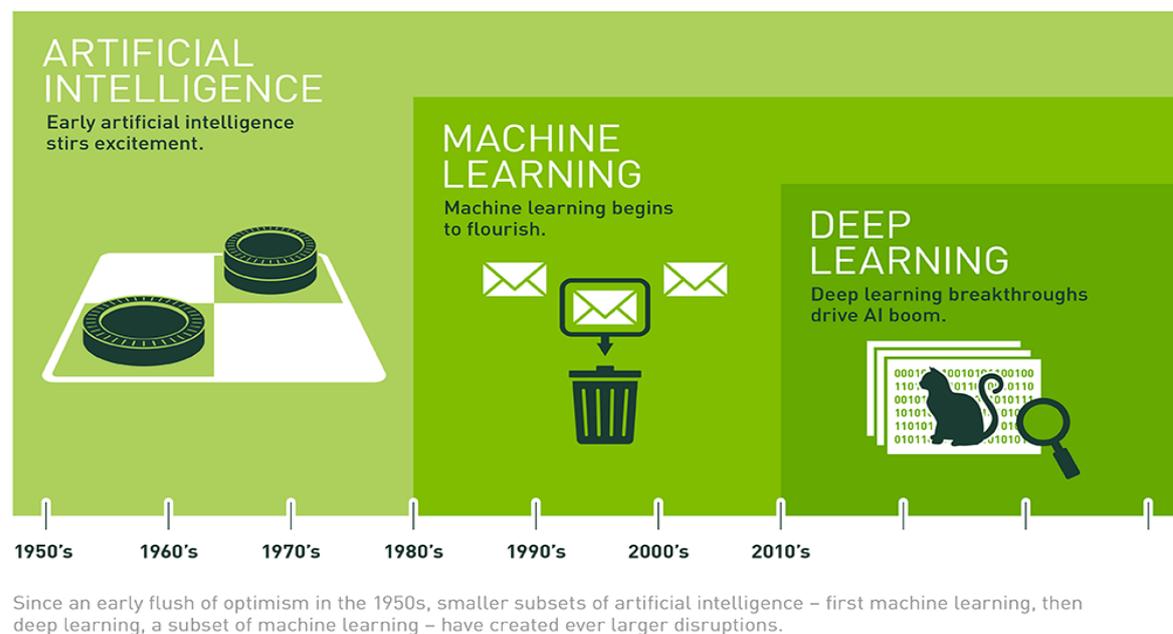
In terms of computer science and ICT, artificial intelligence (AI) or machine intelligence (MI) is the concept of intelligence demonstrated by algorithms through devices and machines, in contrast to the natural intelligence displayed by humans. Colloquially, the term "artificial intelligence" is usually used to illustrate computers that mimic "cognitive" smart functions that humans systematically associate with the human mind, such as "learning", "environment auto-understanding" and "problem solving".

Therefore, AI is always defined as the study of "intelligent agents". As a matter of fact, any device or machine that perceives its surrounded environment and automatically takes actions that maximize its chance of successfully achieving its goals through a set of models and Algorithms. A more elaborate modern definition characterizes AI as the system's ability to correctly interpret external data, to learn from it (through Supervised, non-supervised or semi-supervised different approaches in terms of learning) and to use those trainings, validations and tests in order to achieve specific goals and tasks through flexible adaptation.

On the other hand, Deep learning is a structured hierarchical learning increasingly being part of a broader family of machine learning algorithms mainly based on artificial neural networks.

Deep learning architectures such as deep neural networks (DNN), deep belief networks (DBN), recurrent neural networks (RNN) and convolutional neural networks (CNN) have been applied to applications reaching higher performances and accuracies than humans:

- computer vision (which is our case, Image Recognition is one of the most trending computer vision problems nowadays.)
- speech and audio recognition
- natural language processing (NLS)
- social network filtering
- machine translation
- bioinformatics, drug design, medical image analysis,
- Smart tracking, material inspection and board game programs



*Figure 11: The historical timeline of AI [11]*

Over the past few years, AI has exploded since 2015. Much of that has to do with the wide availability of GPUs that make parallel processing ever faster, cheaper, and more powerful.

In fact, the AI mathematical built models and algorithms didn't find the required performances and calculations to run them on a computer. The AI community worldwide waited for decades in order to increasingly and continuously innovates to create the hardware and software appropriate climate to adopt and develop AI, machine learning and Deep learning.

Besides, Machine learning often defined as the practice of using algorithms to parse, create and augment data, train and learn from it, and then make a determination or prediction about it. So, it is the ultimate solution that avoids traditional hand-coding software routines with a specific set of instructions to accomplish a very particular mission.

Machine learning came directly from minds of the early AI community, and the algorithmic approaches over the years included

- decision tree learning
- inductive logic programming
- clustering
- reinforcement learning
- Bayesian networks...

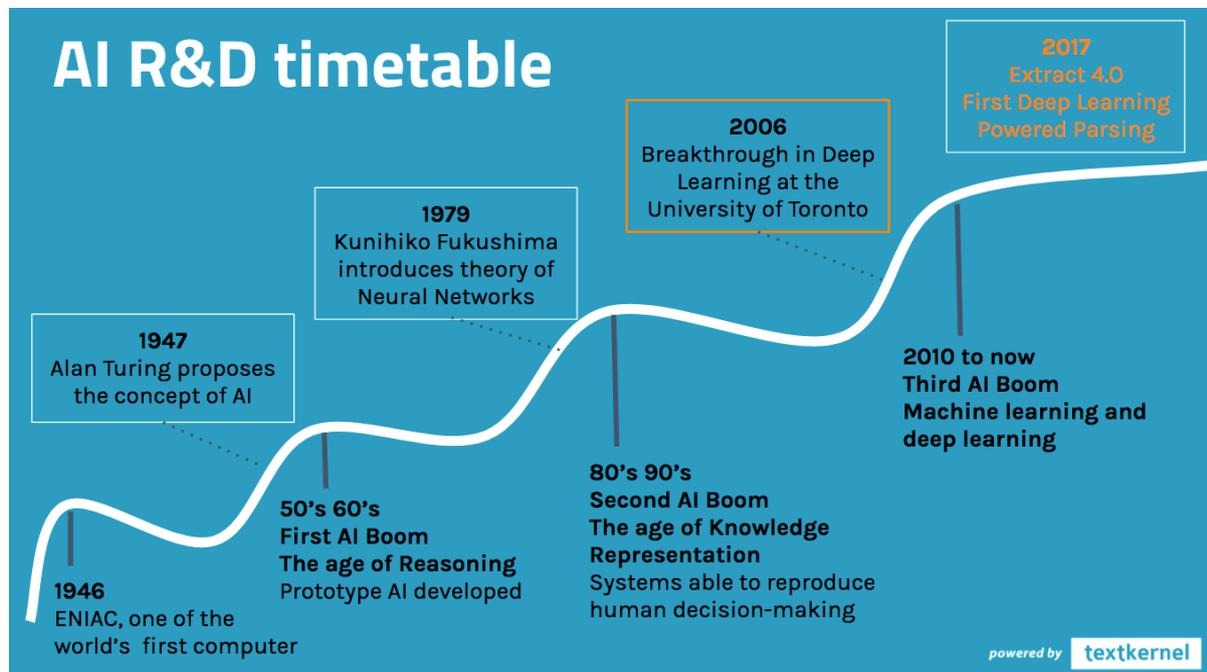


Figure 12: AI Research and Development Timetable [4]

In addition, another algorithmic approach from the early machine-learning community, artificial neural networks, came and mostly went over the decades.

The main idea is that neural networks are inspired from our understanding of our brain's biology referring deeply to all those interconnections between the neurons. On the other hand, unlike a biological brain where any neuron can naturally connect to any other neuron within a certain physical distance, these artificial neural networks have discrete directions of data propagations through layers and neurons.

To crop it all, engineers and scientists deeply think that thanks to deep learning, AI has a bigger, better and brighter future. In fact, Deep learning has enabled many practical applications of machine learning and by continuous extension the overall field of AI. Deep learning breaks down tasks in ways that makes all kinds of machine assists seem possible and reasonable. Driverless autonomous cars, better digital marketing content filtering and better preventive healthcare are among the top trending research, development and investment worldwide. AI is both the present and the future thanks to the huge contribution of Deep Learning. "AI may even get to that science fiction state we've so long

imagined”, Michael Copeland, NVIDIA blogger and tech-journalist on explaining the fundamentals of deep learning.

## 2. Computer Vision: Image Recognition

Image Recognition is increasingly one of the most trending computer vision provided solutions. [9] Therefore, our main focus is on Image Recognition. For that reason, we need to understand the preliminary background required in order to be more efficient while applying this paradigm on our EcoBalade context.

Image recognition is a combination between camera capture and AI algorithms so that the machine efficiently recognizes and tracks images (and even videos, considered technically as images sequences throughout divided time stamps.)

Moreover, evaluating those Image recognition model passes through a variety of steps and mathematical interpretations. Throughout those steps, the developer needs to get a feedback from learning curves, validation, cross-validation, confusion matrix... in order to control the model and to get a deeper clearer idea about how the machine behaves regarding your specific problem statement and data set. All the concepts and approaches explained below will be used and explored throughout different phases of models building and deploying.

### Validation and cross-validation

Building a model mainly converge to a primordial validation.[11] You can call it test phase, but never confuse with the final testing, which is providing unseen new photos to the machine and evaluating how the model predicts. Validation, on the other hand, is a test through “déjà-vu” photos right after training. For instance, we have two main approaches:

- Data set segmentation through 3 sub sets called train, validate and test. Generally, the commonly used segmentation is 80% train, 10% validation and 10% test (which was the segmentation adopted in my project). This technic is considered traditional and classic providing simplicity and transparency. Exploiting just 90% of your data set is the main drawback. For that reason, AI engineers proposed a state-of-the-art cross-validation approach in order to access to the totality of your Data.
- Cross-validation is deeply a segmentation of the Data set to K sub-datasets. The K-1 subsets will be provided for train and validation traditionally. The K remaining dataset is used for testing. Then, the developer codes a little loop to iterate K times and passes the whole dataset through all different combinations for different purposes (train, validate and test). To crop it all, the output metric (accuracy in our case), will be a calculated average of the result of each and every iteration. It is mainly more accurate and precise approach but requires more space (exploring the whole data set) time (every iteration is a traditional validation) and processing.

## Confusion Matrix

It is a mathematical tool used to measure the quality of a classification system. In fact, it is a  $n$ -dim matrix where  $n$  is the number of classes to predict. The horizontal axis represents the true labels of every class while the vertical axis illustrates the predicted labels. The intersection between axis is the true positive items. Getting a diagonal plot shows that the model, after training and adapting to the specific dataset, is able to recognize accurately the labels and features it learned.

Confusion Matrix doesn't depend on the Classification algorithm (decision tree, Support Vector Machine, Logistic regression, Naïve Bayes, Stochastic Gradient Descent, K-Nearest neighbors, Random Forest...). All of those models need to provide a diagonal matrix to indicate to the developer that the training phase went good (features and labels well extracted and results well predicted at the classification layer). Right after that, you can move to Fine Tuning or Bottleneck technic before final validations and tests.

In the next chapter, we will notice that all the confusion matrixes were diagonal with no troubles or problems predicting on the train dataset.

## Learning Curves

It is a graphic representation of the genral progression of learning throughout learning steps in order to gain the required skills. It deeply illustrates the "experience" of the model increasing its accuracies and reducing its errors. Learning curves indicates to the developer if there is a problem of bias or variance explained in the next paragraph.

Therefore, we have different ways of plotting the learning curves where 2 approaches are very popular and commonly used:

- The horizontal axis represents the number of provided input images increasingly. The vertical axis illustrates the metric required (generally cross-validation accuracy and test accuracy in the same plot obviously). The curve, in theory, should be increasing throughout the increase of the number of images. State-of-the-art pre-trained networks fastly and efficiently reach spectacular accuracies from the very first provided images. (That is the main point of using the Transfer Learning concept). In addition, if we notice a bias problem which is kind of a gap of accuracy between the train curve and the validation (or cross-validation) curve related to the first provided photos. The variance problem is related to divergence of the same curves generally noticed with a big number of provided photos.
- On the other hand, it is more interesting for state-of-the-art neural architectures to manipulate and interpret those curves the following way; the horizontal axis provides the epochs (learning iterations where every iteration illustrates the whole back and forward propagation of the whole Dataset on the totality of the neural networks layers.) The vertical axis remains the same. However, the train and validation plots give as a deep understanding of the model's increasing experience at every epoch,

mainly through reducing the errors and increasing the precision. (The main problem for those pre-trained architectures is that you don't have access to what is going on at every epoch of learning. Plotting those curves is a challenging task requires respectful understanding to the mathematical algorithms behind the "black" boxes of every architecture diagram.)

If we handle a "personal" fastly built CNN, we can access to each and every iteration as long as we set up the number of epochs, the batch size and mainly the majority of required configurations. We will get precise information about the calculated cost at every epoch. By the way, we can complementarily plot the errors instead of the accuracy. We will obviously get the same curves but in a complementary way. Increasing accuracy systematically means decreasing error and vice-versa.

### **Hierchical Feature Representation**

To be brief and simple, the hierarchy on features can be explained this way; features computed by the first layers are mainly general and regarding the transfer learning approach, they can be re-used in different problem statements and for a variety of use cases. However, the last layers are more specific; they are deeply related to the problem statement itself.

For that reason, developers generally remove the old classifier and adapt the model to the specific task through 3 different process:

- Train the entire model. This is a traditional typic technic
- Train some layers and leave some others frozen. This can be related to a specific used of Bottleneck and can mainly contribute to reduce or avoid over-fitting.
- Freeze the convolutional base block. This is used to gain time and memory only if your specific problem statement is very similar to the original generalized one.

In our models, the second process was used and related to the Bottleneck technic. For MobilNet, we performed Fine Tuning differently. Further details and explanations on the next chapter.

## **3. Convolutional Neural Networks**

ConvNet or CNN is the class of deep neural networks generally applied for computer vision and image recognition. In fact, they are regularized versions of multilayer perceptron. Multilayer perceptron often means fully connected networks. For that reason, each neuron in one layer is connected to all neurons in the next layer.

The "fully-connectedness" definition criteria of these networks makes them prone and robust facing over fitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. On the other hand, CNNs provides a different approach and method regarding regularization: they take main advantage of the

hierarchical pattern in data and assemble more complex patterns (often in the Dense Layers) using smaller and simpler patterns (extracted from the convolutional Layers).

It all started in 1943 when McCulloch and Pitts defined the first mathematical and computer science model of a biological neuron

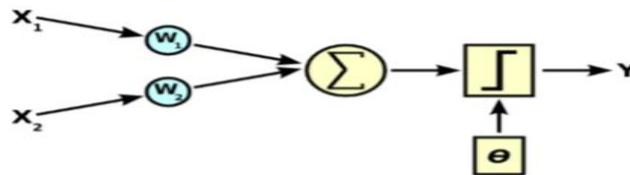


Figure 13: Simplified representation of a binary neuron [9]

It is a binary neuron. In fact, the neuron makes a weighted sum of its inputs and with a threshold activation function; the output of the neuron is worth 0 or 1 (in function of the output value relative to the threshold).

In order to understand how it works, let's focus on the behavior of a single neuron. In fact, a neuron aims to separate a set of data. The simplest case is to separate a data set by a straight line characterized by an equation.

The points in the data set that verify this equation will be on this line. The points for which the equation is negative will be below this line while the points for which the equation is strictly positive will remain above it.

This is absolutely what an artificial neuron does. It separates a data set into two parts. If we use a single neuron then it will be able to separate linearly the data set provided. Data can have several parameters providing the size or dimension of the space and thus the separator.

For single-dimensional data, the neuron will separate the data by a straight line. Accordingly, if the data has two dimensions, the neuron will use a "plan" often called hyper plan in high dimensions.

The name "convolutional neural network" indicates reasonably that the network employs a mathematical operation called convolution which is a specialized kind of linear operation and matrix multiplications.

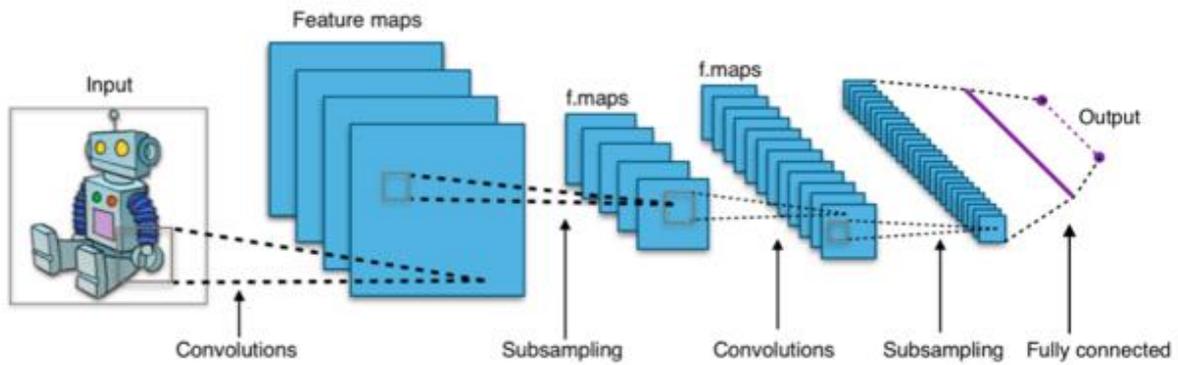


Figure 14: CNN Layers and functions simplified representation [11]

A CNN consists of an input and an output layer as well as multiple in-between hidden layers referred to as hidden layers simply because their inputs and outputs are masked by the activation function (commonly a ReLU layer subsequently followed by additional convolutions such as normalization layers, fully connected layers, pooling and final convolution that usually involves backpropagation).

While designing a CNN, each convolutional layer has these attributes:

- Input is a tensor with shape (number of images) x (image width) x (image height) x (image depth).
- Convolutional kernels (mainly a set of learnable kernels and filters) whose width and height are hyper-parameters, and whose depth must be equal to that of the image. The process consists on the fact that convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a biologic natural neuron in the visual cortex to a specific stimulus which shows the analogy between the artificial and the natural system.

### Pooling:

CNNs may include global or local pooling layers to streamline the underlying computation. In fact, pooling layers tend to reduce data dimensions by combining and connecting the cluster neurons outputs at one layer into a single neuron right in the next layer. However, local pooling combines small clusters, typically 2 x 2 while global pooling acts on the totality of the convolutional layers' neurons. Besides, pooling may compute a max (at the prior layer, it calculates the maximum value from each of a cluster) or an average (the average value from each of a cluster of neurons at the prior layer) depending on the application need.

### Biases and Weights:

The concept of learning for a DL or ML developer is mainly a work of iterative adjustments to biases and weights. In fact, each neuron in a CNN calculates an output value through the application of a specific function to the input values received from the previous layer's receptive field (the input area of every neuron).

### **Fully Connected:**

Inspired from the traditional multi-layer perceptron neural network (MLP), fully connected links all the outputs of the previous layer to one and only neuron, often at the last layer. The idea is that flattened matrix mainly goes through a fully connected layer to classify the images.

### **3D neurons:**

CNN layers have neurons arranged in 3 dimensions: width, height and depth. The main Principle is that the neurons inside a layer are connected to only a very small region of the previous layer, called a receptive field just like mentioned below.

### **Local connectivity:**

CNNs exploit spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers just like the concept of receptive fields. However, the architecture ensures that the learned kernels or filters produce the strongest response to a spatially local input pattern. In fact, in order to let the network, create representations of small parts of the input, it assembles representations of larger areas.

### **Shared weights:**

Inspired by the concept of translation invariance, replicating units allow the detection of the features regardless of their locations in the visual field so that all the neurons in a specific given convolutional layer respond to the same feature with their specific response field.

### **The depth hyper parameter:**

The depth of the output volume controls the number of neurons in a layer that connect to the same region of the input volume. These neurons learn to activate for different features in the input. For example, if the first convolutional layer takes the raw image as input, then different neurons along the depth dimension may activate in the presence of various oriented edges, or blobs of color.

### **The Stride hyper parameter:**

It controls how depth columns around the spatial dimensions (width and height) are allocated. For example, if we want to move the filter for one pixel at a time, we should set it Stride = 1. Thus, when the stride is 2 then the filters jump 2 pixels at a time as they slide around.

### **The Pad hyper parameter:**

The padding is to fulfill the input with zeros on the border of the input volume providing control of the output volume spatial size.

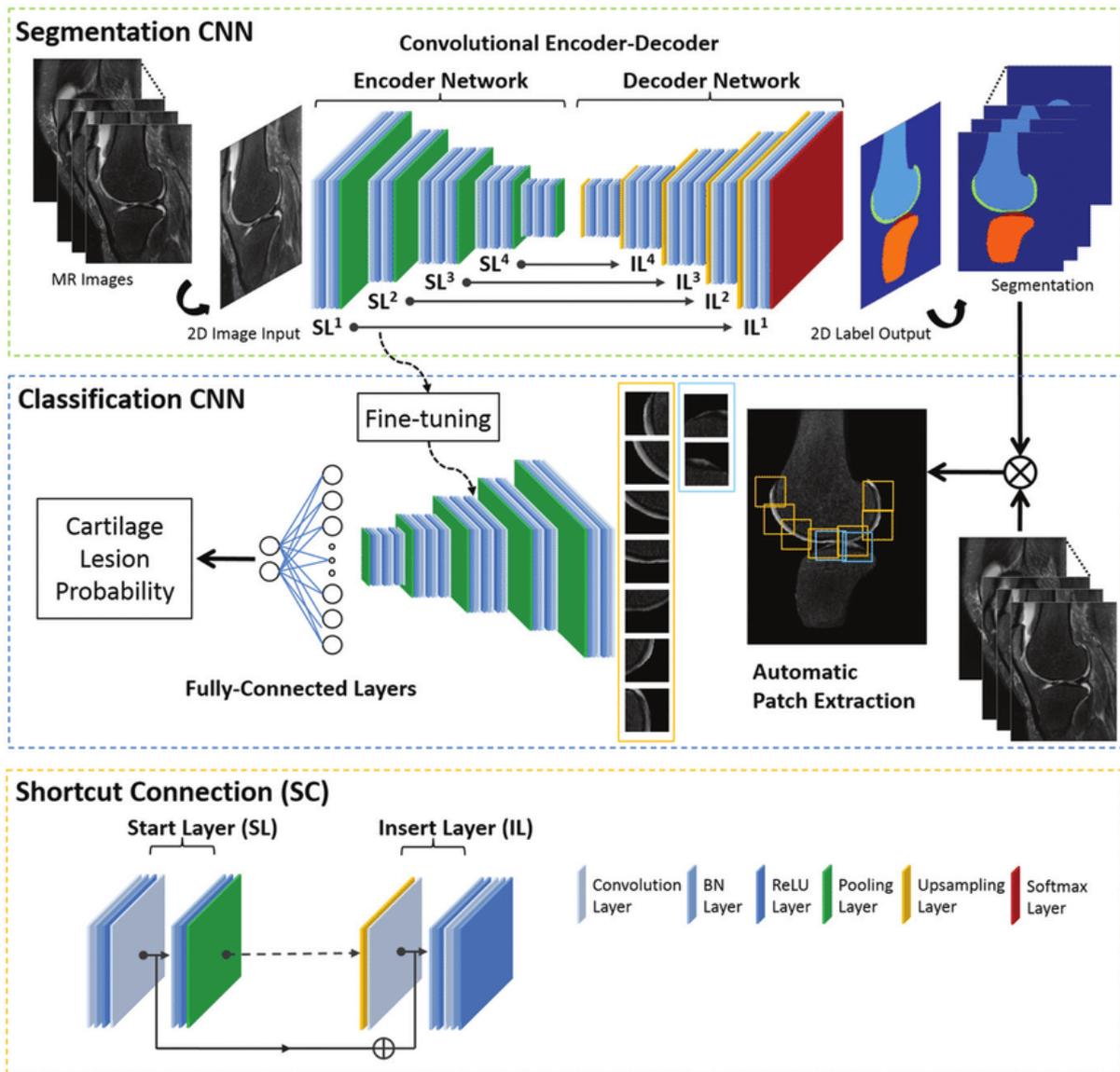


Figure 15: Different CNN architectures, layers and functions [11]

### ReLU Layer:

ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function  $f(x) = \max(0, x)$ . As a matter of fact, ReLU removes the negative from an activation map setting them to zeros. It mainly increases the nonlinear properties of the decision function and of the overall network without impacting the receptive fields of the convolution layer. Let's notice that we can use the saturating hyperbolic tangent or the sigmoid function for the same purpose (increasing nonlinearity). However, ReLU is more popular for its ability to train several times faster than any other mathematical function.

### **Loss Layer:**

At the final layer of a CNN, it specifies how training penalizes the deviation between the predicted (output) and true labels. Various loss functions appropriate for different tasks may be used:

- Softmax loss is used for predicting a single class of K mutually exclusive classes.
- Sigmoid cross-entropy loss is used for predicting K independent probability values in [0, 1]
- Euclidean loss is used for regressing to real-valued labels ( $-\infty, +\infty$ )

### **Empirical Regularization:**

Drop Connect, stochastic pooling, Drop Out are used to fix over fitting. Probably, the most popular technic is Drop Out which is the fact that we remove (drop out) the individual nodes with  $1 - p$  probability and keep those with  $p$  so that we can reduce the network; incoming and outgoing edges to a dropped-out node are also removed. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights.

### **Explicit Regularization:**

Early stopping (stop the training if over fitting is being noticed), limit number of parameters, weight decay (adds to the error at the output of each node an additional error, proportional to the sum of weights (L1 norm) or squared magnitude (L2 norm) of the weight vector) and max norm constraints are probably the most popular Explicit Regularization.

### **Fine Tuning:**

An extremely popular technique when we have got a small dataset is to train the network on a larger data set from a related domain. In fact, an additional training step is performed using the in-domain data to fine-tune the network weights once the network parameters have converged.

## **4. State-of-the-art neural networks preliminary study**

### **4.1. MobileNet**

MobileNet is a neural network mainly known for its obvious contribution to the family of efficient models for mobile applications and embedded vision devices. Inspired by the concept of depth wise separable convolutions, MobileNet is introduced to the world of deep

learning as a lightweight incredibly fast architecture reaching accordingly respectful accuracies and performances on varieties of image recognition problems and Data Sets.

In fact, regular convolutions try to convert each and every input item into one and only pixel while depth wise architecture tend to separate each input channel and performs it separately through setting a respective set of weights to each one. [6]

Besides, performing from one layer to another requires a set of filters. For MobileNet, the most performed filters are color filters and edge and features detectors.

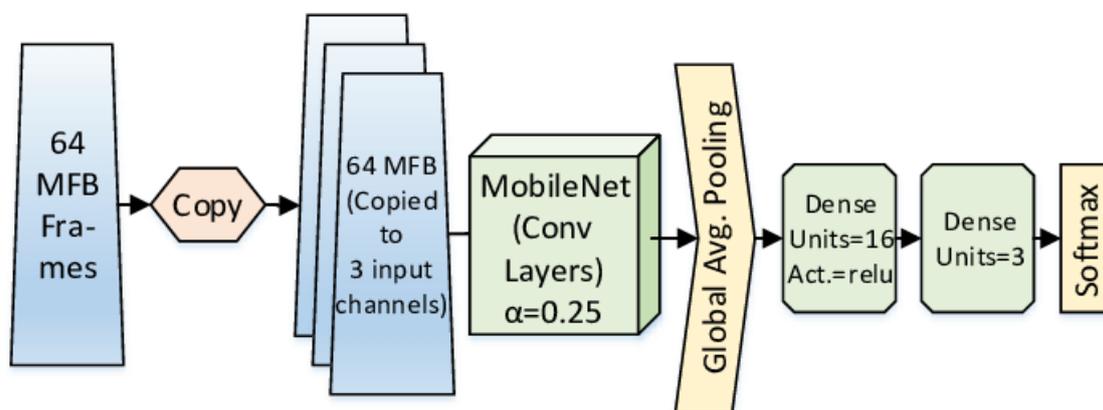


Figure 16: MobileNet architecture simplified diagram [6]

Therefore, processing the input passes by a set of operations, Mobile blocs and convolutions. The Mobile blocs contain 4 kinds of layers: 2 depth wise layers and 2 pointwise layers all followed by a set of 5 consecutive convolutions before getting processed into another mobile bloc. The output will cross into an Average pooling layer and a classification layer (Fully Connected). Another interesting fact about MobileNet architectures is that every convolutional layer is followed by a Batch Normalization and ReLU6 activation function. [6]

MobileNet had several updates and changes in the “black” boxes mainly throw setting up different parameters and changing the purpose of the pointwise from keeping the number of channel identical (or double it) to compressing it. In fact, this reduction of data which flows further in the network is known as the Bottleneck technic (that I used while designing the Flower Recognition Model, explained in the next chapter). The projection layer is accordingly called a Bottleneck layer.

The image being processed in the building block expands in several channels then gets squeezed into smaller channels (through the pointwise layers obviously). We should notice that the same spirit will be illustrating the Inception and Resnet architectures. One more huge intersection with Inception is the idea of the computational replacement of expensive convolutional filters into cheaper ones. (Prices in terms of processing requirements)

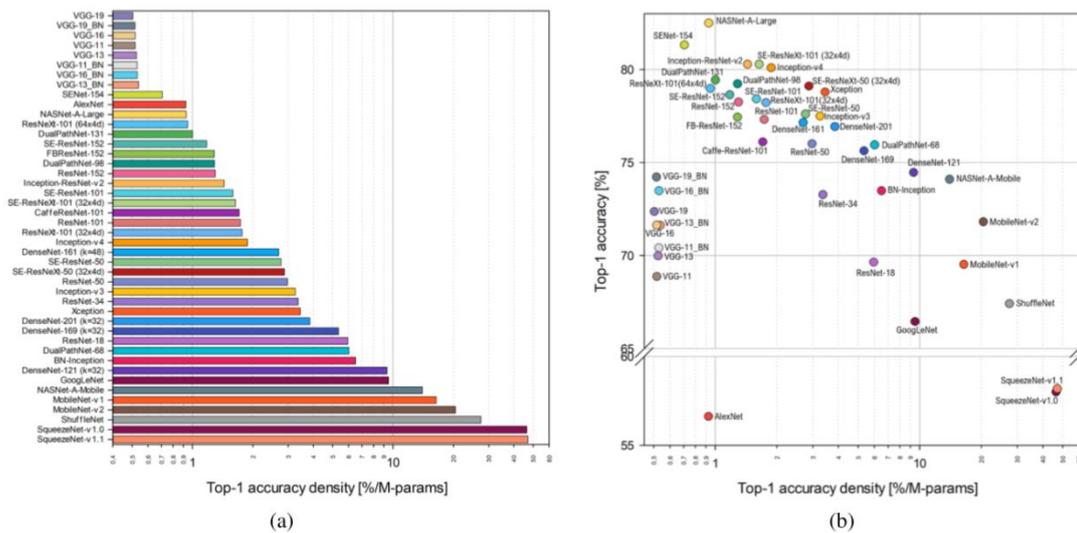


Figure 17: State-of-the-art neural architectures Accuracy-Density comparison [11]

As we can notice in the Figure 15, Accuracy-Density comparisons measure how efficiently each model uses its parameters. In fact, SqueezeNet and MobileNet lead the competition providing the best performances. Let's notice that the competition was about training and testing on a sub ImageNet Data set. In the next chapter, we will get the model summary for the 3 used architectures and we will notice the importance of getting brilliant results and respectful accuracies using the lowest possible number of parameters.

To crop it all, MobileNet main goal is letting the model be able to auto compress and decompress the processed data at every stage of learning in the network.

Last but not least, MobileNet version 2 (being actually used in my Flower Recognition Model) contains 3.5 million parameters outperforms the first version (4.5 million parameters) in each and every scoring method.

## 4.2. Inception

Inception was introduced to the world of deep learning in 2014 by a Google Brain research unit illustrating the architecture as “strong convictions” deep network inspired by the concept of Auxiliary Classifiers (the idea of applying Softmax to the output of the 2 consecutive Inception blocks in order to calculate an auxiliary loss over the same labels.) Auxiliary classifiers provide an elegant solution to the vanishing gradient problems. The auxiliary loss is accordingly added to the total traditional calculated loss with a weight of 0.3

Auxiliary processing is only used in training for Inception architecture. One of the most interesting blocks of the whole network is the Global Average Pooling block by the final layers. GAP is considered as a great innovation in terms of Data Reductions being able to extract a  $h*w*d$  dimensional tensor to  $1*1*d$ .

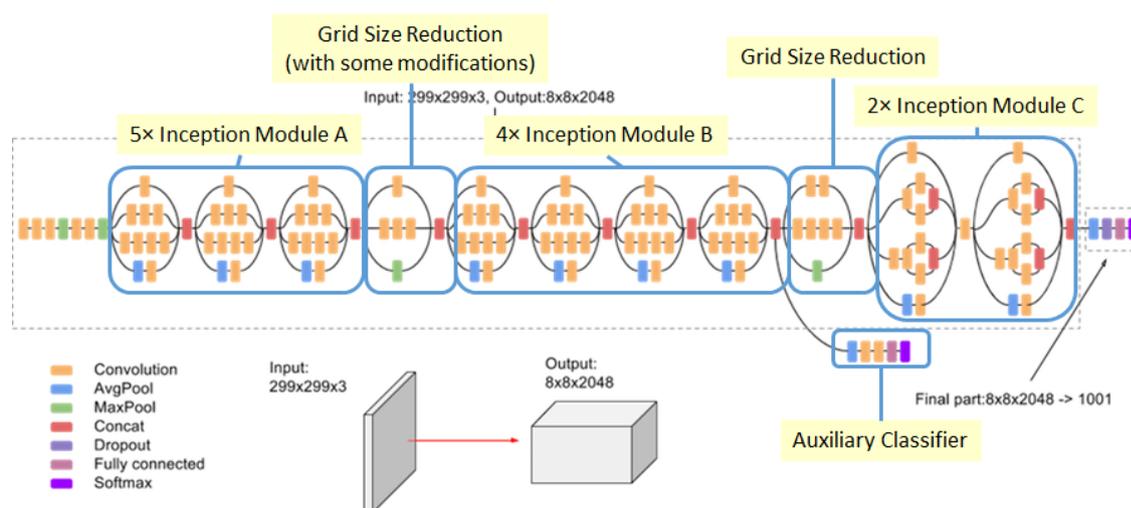


Figure 18.1: Inception v3 simplified diagram [7]

Inception final releases were mainly inspired by Google brain’s introducing the concept of Batch Normalization to the Inception traditional architecture providing an accurate solution to the phenomena of the Internal Covariate Shift (“the distribution of each layer’s inputs changes during training, as the parameters of the previous layers change.”) [7] [11]

Batch Normalization offers the ability to resemble distributions at each training step by normalizing the mean and variance of each of the features. The Factorization of convolutions with larger filter sizes is obviously another important innovation in the Inception latest

releases by Google (version 2 and 3). In fact, every 5\*5 filter will be decomposed into 2 consecutives 3\*3 filters scoring a 27.8% computational gain.  $(3*3) * 2 / (5*5)$

### 4.3. Resnet

Resnet is the Deep Residual Network (from which came the abbreviation), introduced to the world of deep learning through the ILSVRC image classification contest in 2012. Alex Network is probably one of the Resnet-like architectures inspired by the same mathematical algorithm based on a huge number of layers (does a better model mean more layers? That was the main approach behind designing and building Resnet, AlexNet and several DNNs).

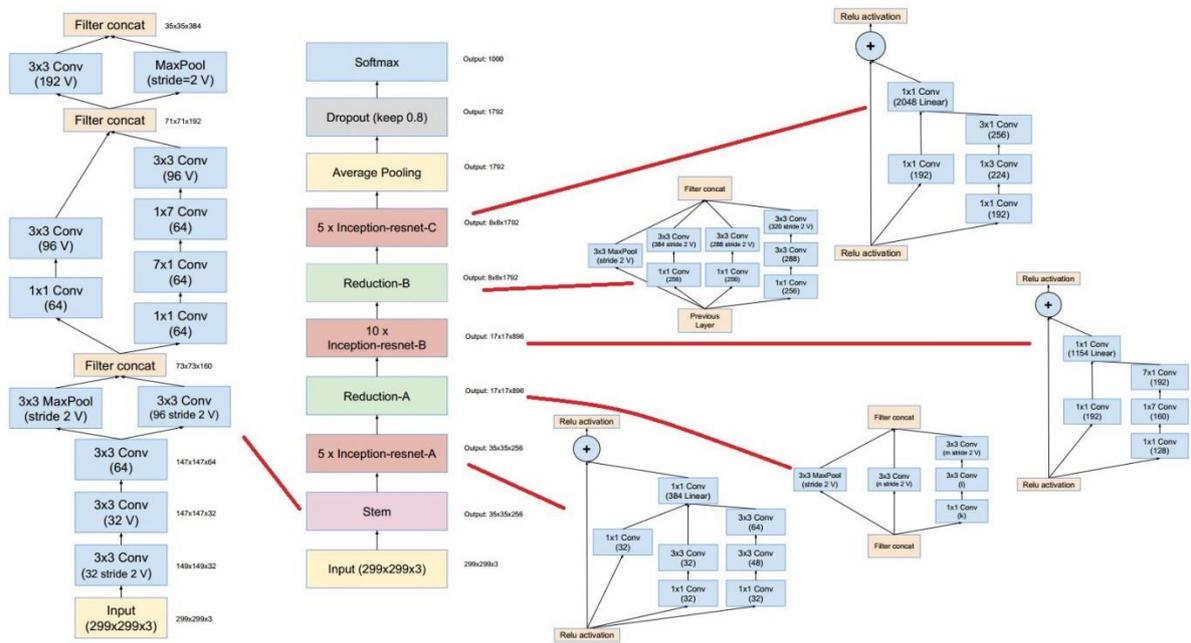


Figure 18.2: Inception ResNet v2 simplified diagram [7]

Training deep models is always linked with the Degradation problem through two main phases: accuracy's saturation and accuracy's degradation. First layers are based on 1\*1 convolution (a little optimization making the model gain some extra computational resources for the heavy convolutions in the deepest blocks of the architecture).

Resnet computational blocks are perfectly similar to the Inception ones:

Input (299\*299\*3) >> Inception/Resnet blocks >> Reductions >> Average Pooling >> Drop Out (keep 0.8) >> SoftMax [7] [11]

Thus, the main difference is that Resnet is all about going deeper while inception is all about going wider. Both architectures use the same techniques and mathematical approaches to perform recognition on images but with different designs and different purposes.





# III. Image Recognition Model: design, deploy, validations and tests

## Introduction

Building an Image Recognition Model is a multi-task and multi-disciplinary objective. It is an intersection between Data Science, AI and ML technical background, python skills and a complete adaptation to the problem statement, to the firm expectations and the customer's needs. The experimental protocol used is divided into a variety of phases:

- ✓ Scraping to grab and browse the required input photos (about 100 photos per class)
- ✓ Filtering those photos (delete odd species, delete repetitive photos...)
- ✓ Data Augmentation (from 100, we expand 550 photos per class adding a variety of filter such as horizontal and vertical shift augmentation, horizontal and vertical flip augmentation, random rotations, brightness and zooms and photometric distortions.
- ✓ Extracting the features related to each and every class. (Output fixtures and labels size is important: a lightweight size is always recommended.
- ✓ Plot the confusion Matrix through the Logistic Regression classifier
- ✓ Bottleneck technic or Fine Tuning regarding the Transfer Learning AI concept.
- ✓ Traditional validation (Data set segmentation: 80% train, 10% validation and 10% test)
- ✓ Final tests on unseen new samples (plot the learning curves if required)
- ✓ Summarize the results and make comparisons and interpretations regarding the fixed metric (recall and precision) and the required CPU/GPU specifications.

This chapter provides a technical description of the different steps of flower recognition models using MobileNet, Inception and Resnet.

### 1. MobileNet

#### 1.1. Preprocessing: Data Set and Data Augmentation

Before getting to training, validating and testing, Image Recognition models are extremely related to the quality of the provided data set. It is not about the fastest or the most

accurate algorithm, it is all about the Images we provide to the model to train it to the right predictions.

Our Data is mainly a set of 55 vegetal species (taken from the EcoBalade application covering the majority of fauna and flora in several ecosystems in France).

In order to achieve 500 photos per class, our pre-processing illustrates 3 major steps:

- Web Scraping
- Filtering
- Data Augmentation

In fact, scraping is a popular technique for extracting web content via a script or program in order to use it in another different context.

An accurate model requires having a giant database of a large variety of images that we want to recognize systematically. For that reason, employees at this firm went through lots of walks proposed by the EcoBalade application, in order to take many pictures of the different species of flowers they wanted to be able to recognize.

Filtering is all about deleting the odd scrapped photos. Wrong species or simply low-quality useless photos are mainly filtered manually.

Data Augmentation is generally related to 2 objectives:

- Enrich the Data S1 et with filtered photos (can probably illustrate a situation in which the EcoBalade app user can take zoomed or rotated photos. The machine will learn to recognize through those photos too.
- Adapt the data to the corresponding architecture. In fact, each algorithm requires specific processing on our input.

For MobileNet, the adaptation consists in:

- sampling
- generating image translations and horizontal reflections
- horizontal and vertical shifts
- Random Rotations (randomized angle in the range [-30,30])
- Random Brightness and zoom
- Fixed output (Size = (224,224), RGB)

## 1.2. Features Extractions and Confusion Matrix

MobileNet is known as an insanely small and fast architecture. Features were extracted locally with the Natural Solution's PC CPU and GPU specifications

- Intel Core TM i7 1.8 GHz, 4 Cores CPU, 8 Go RAM
- Intel UHD Graphics 620, 3.9 Go, 1 GPU Core

Those modest specifications are enough to extract features. (Include\_top layers were disabled which means that we won't process through the final dense layers).

The convolutional layers perform the image as they identify a series of patterns (each layer will identify more elaborate patterns by seeing patterns of patterns in a repetitive iterative long process).

The dense layers are capable of interpreting the found patterns in order to classify the image.

In addition, the weights are fixed ('image net' weights). They are in fact the size of kernels\*filters (3\*3) kernel of 10 filters in our case) and they are totally independent from the input (224, 224, 3) size.

2 main parameters are systematically involved:

- Width multiplier for a thinner network.
- Resolution multiplier to reduce the internal representation at every layer

The output from this operation is two precious files: labels and features.

Those features and labels will be provided as an input while building the model. Their total size for MobileNet case is 5 Go (further in this chapter, we'll notice the importance of the features size)

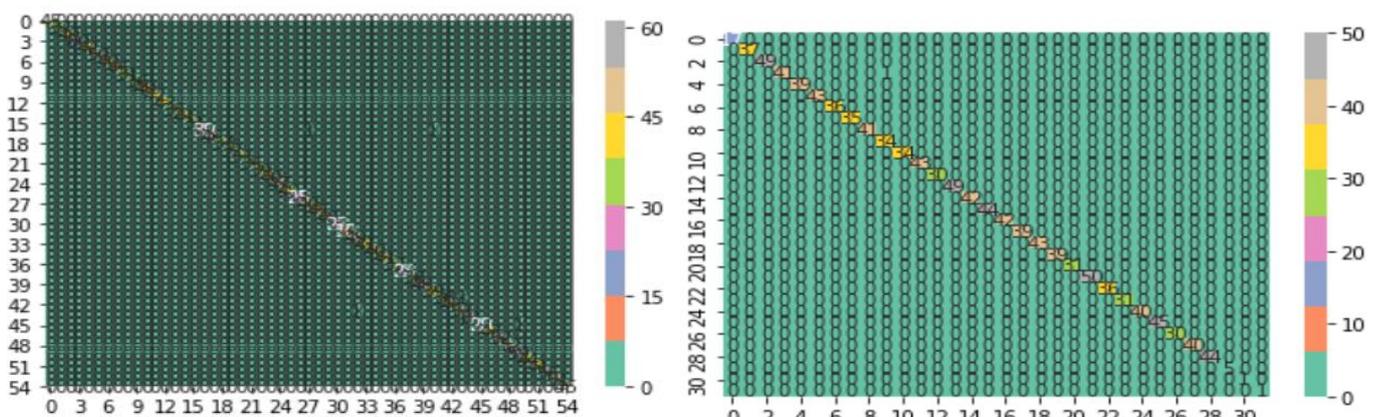


Figure 21: MobileNet Model diagonal Confusion Matrix

Figure 21 represents two plots:

- The first one (on the left) is the model's confusion matrix trained on 55 classes. It shows the predicted labels per true labels through the input images.
- The second one is plotted after 33 class training. It's a clearer illustration of the different coefficients of the Confusion Matrix.

Getting a diagonal Matrix proves that the model building is in the right way. The algorithm shows good predictions (predicted labels = true labels). Diagonal matrix provides a mathematical interpretation and plot to the true positive items. Getting a diagonal plot guarantees that the model performs well and accurate on training data. For instance, the model throw Logistic Regression Classifier is learning efficiently. The next step is the Fine Tuning inspired from the Transfer Learning AI Approach.

### 1.3. Fine Tuning and Transfer learning

Transfer learning concept is one of the most popular AI technological paradigms. Exploring a pre-trained neural architecture starts to be a crucial step while building a deep learning model. In fact, Visual Geometry Group, a powerful CNN designed and deployed by Oxford, pre-trained on Image Net large open source Data Set containing more than 14 million images, was the key element of our MobileNet Model fine tuning. Therefore, we loaded the same pre-trained weights in order to re-train our Flower Recognition Data Set.

Besides, Fine Tuning had its real added value on the model performances. I had a little comparison throughout a set of validations and tests on 30 class Data Set regarding two models: the first one was fined tuned by VGG and had a 5% increase on the accuracies compared to a non-fine-tuned model.

### 1.4. Validation and Tests

MobileNet architecture is one of the most accurate CNNs reaching very positive results in a variety of image classification competitions providing insanely fast time delays reasonably related to their light weights and tuning easiness. Therefore, for our Flower Recognition Model, MobileNet model reaches 99.67% as training accuracy, 95.34% as validation accuracy (4.66 % of total validation cost calculated through the cross-entropy statistical function) and about 85% of test accuracy (tests made on a variety of species and different photos that the machine never "saw" before. I also tried to provide some unclear photos or filtered samples to test the first and the second prediction).

```
orpin
406
orpin
407
orpin
taux d'erreures = 4.66 %
précision à la validation = 95.34 %
```

Figure 22: MobileNet Model's Validation accuracy

For instance, another interesting fact about MobileNet is that in case of wrong prediction, the second proposed prediction is often a wright answer. (The first and the second prediction are simply the highest calculated probabilities in the last fully connected layer through the logistic regression classifier). We will get further explanations and screenshots in the next chapter. In fact, this is most likely the case for every Logistic regression model.

```
raw prediction : [[6.86792886e-09 1.61207085e-11 1.10919559e-10 2.81204575e-13
2.41384133e-13 6.77688746e-05 2.39208700e-06 7.29702897e-11
2.47588078e-11 1.70502868e-08 4.97529464e-07 5.02266959e-07
3.50315120e-12 2.73722133e-12 8.09372173e-14 3.49555729e-11
8.00516510e-09 4.88376895e-11 2.50556458e-10 6.25545420e-11
6.24361648e-12 3.41936601e-08 9.99654255e-01 1.09247535e-10
3.37949631e-12 7.14981546e-09 3.23897815e-09 1.25932960e-09
1.42039350e-07 1.99785710e-06 2.87824554e-11 6.25666703e-07
3.71573171e-10 6.38401936e-12 5.40151777e-10 1.68318369e-09
1.02587899e-09 3.81672501e-06 1.49131788e-07 2.03034526e-04
1.66900812e-07 9.65618076e-11 2.26113630e-08 1.47925985e-09
9.55020512e-12 1.60633245e-08 6.39199455e-05 6.20058146e-08
1.58140601e-09 9.84589023e-10 1.05863134e-07 8.02115245e-08
3.07478451e-11 8.62655090e-09 3.49657215e-07]] coquelicot 0.9996542550597877
```

Figure 23: Screenshot of MobileNet Model's raw prediction while testing

Figure 20 illustrates the items composing the last “classification” layer. It is in fact a NumPy array holding the calculated probabilities for each and every class (vegetal species). This explains the first and the second given predictions in the following test screenshots (Figure 24) which are accordingly related to the highest and the second highest calculated probabilities. Several tests were made on completely unknown samples. (Test photos should not exist in the train Data)



Figure 24: Screenshot of MobileNet Model's Tests

Therefore, working on Data augmentation and amelioration can probably increase the reached accuracies. In fact, adding more filters related to the different situation in which the customer can be put while taking his shots can easily bring more efficiency and realism to the Data which obviously leads to very respectful results.

In a parallel way, the learning curve is another popular mathematical plot to help the developer and the designer understand the general behavior of the model. Our estimator is the Logistic Regression, our learning rate is 0.001, the cost function is the cross-entropy and the score is a wiser statistical formula based on recall and precision.

(Recall is the sum of true-positive per sum of true-positive and false-negative. On the other hand, average precision is the sum of true positive per all the positives (false and true).

$$\text{The Score} = 2 * (\text{recall} * \text{precision}) / (\text{recall} + \text{precision})$$

## 2. Inception v3

### 2.1. Pre-processing: Data Set and Data Augmentation

Regarding the fact that Data Set pre-processing is one of the most important keys to succeed while building an accurate trustworthy model, we accorded two weeks for this phase. In fact, the pre-processing is almost the same for all the neural architectures. Thus, we can notice some differences. For Inception, Data preparation consists on coding the input data to RGB color convention. In addition, we generate some image translations and horizontal reflections. One of the specificities of train-time inception augmentation is adding “photometric distortions”.

On the other hand, test-time data augmentation consists on resampling each and every image at 4 scales where the shorter dimension (height or width) is 256,299,320,352...

Right after that, the algorithm takes the left, center and right square of the resized images in addition to traditional random rotations, brightness and zooms.

Successive cropping, interpolation methods and photometric distortions characterize the majority of the inception pre-processing steps.

For Inception and Resnet, I worked through two different data sets. The first one is a light weight data containing just 10 species (classes) with a reduced number of pre-processing operations. The first one was just helping me getting familiar with those Deep architectures in order to facilitate the processing and the calculations. However, the second dataset is our giant 55 class-Data.

### 2.2. PaaS: Colab, Amazon Web Services and Kaggle

The very first initiatives of running Inception and Resnet on local NS pc all fell down. In every step, we got memory allocation exceptions and memory saturation errors. It was completely impossible to run on local which is mainly reasonable regarding the huge heavy weight of those state-of-the-art neural networks.

The first move was mainly to the free PaaS offered by Google called “Colab”. An interesting notice about Colab is that it offers TPU: Tensor Processing Unit, an alternative to GPU based on the N-dim matrix calculations to process rapidly and efficiently graphic data. Google Colab is one of the most popular PaaS for AI engineers and Image Classification Model designers, developers and project managers. The platform offers a direct and fluid mapping to your Google Drive where we stored our Data.

Google Colab free dedicated Cloud Computing was not able to resolve the problem. Inception and Resnet both require more computing resources and we needed to move some premium offers.

Besides, the best options were offered by Amazon Web Services and Google Cloud. Both PaaS were insanely close and similar in terms of offered calculations and services. However, the payment for AWS is per used instances or modules. However, for Google, it's more like a traditional choice between some premium packs. For instance, we used PaaS as a research student (not as a company).

Therefore, Kaggle is a free PaaS "specialized" in open source Data Set, Kernels, notebooks and image recognition competitions. It is completely free and systematically linked and mapped to your Google cloud.

Kaggle is used for the rest of model building different steps and through the provided specifications; we were able to run not only Inception but also ResNet (further details in the next summarizing paragraphs).

**Kaggle CPU specifications:**

- 4 CPU Cores
- 17 GB RAM
- 5 GB Auto-saved HDD
- 16 GB Temporary Scratchpad

**Kaggle GPU specifications:**

- 2 GPU Cores
- 14 GB RAM
- Additional dedicated NVIDIA Tesla K80

### **2.3. Features extraction and Confusion Matrix**

The features and the corresponding labels extraction were performed on AWS. The output we downloaded is 9 Go sized (compared to just 5 Go for MobileNet and 12 Go for Resnet). In fact, the processing on 55 classes took more than 5 hours although AWS has provided 2 GPU Cores and 17 Gigabytes RAM. We also added a single NVIDIA Tesla K80 to our Kernel.

The confusion Matrix is a perfect Diagonal illustrating that the model building is good. In fact, the predicted labels accurately illustrate the true labels for the whole pack of input images.



to train every time you compile the model which provides optimal time management, crucial point when we are handling heavy architectures with large datasets.

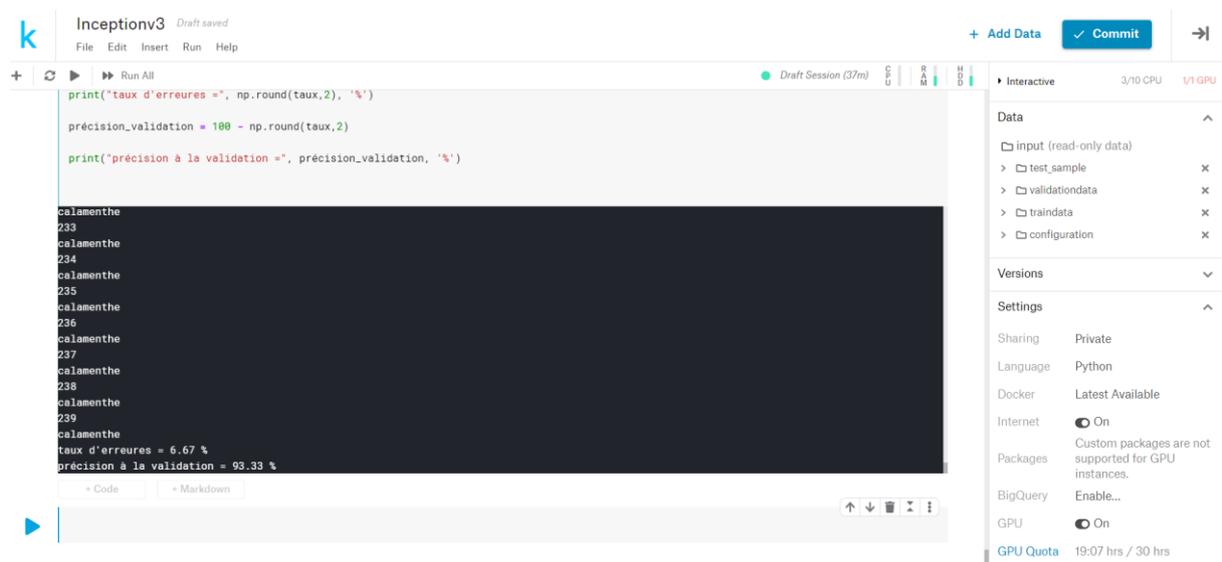
The model performed Bottleneck on PaaS, on Cloud: let's notice that Google provides a free tesla k80 GPU dedicated specifically for Cloud Fine Tuning or Cloud Bottleneck implementations. On the other hand, Kaggle offers Fine Tuning built-in functions, same for AWS or IBM Watson in order to reduce the number of features maps and parameters.

To crop it all, Fine Tuning or Bottleneck approach helps the model combat over-fitting and systematically leads to better accuracies on validations and tests. The analogous neural medical approach to Transfer Learning is called the Cognitive Psychology. Data Scientists, Researchers, developers and AI engineers inspired the artificial work from that Analogy. Fine Tuning remains now a days a primordial step throughout the model design and deploy.

## 2.5. Validation and tests

Inception v3 generally reaches the 90% accuracy in several Image Recognition Competitions worldwide. Google Brain provided a heavy and deep CNN but with tremendous ability to get high precisions and scores on different datasets and problem statements.

For our Flower Recognition Model, Inception v3 reaches 92,92% of accuracy in Validation (7,08% validation error percentage calculated through the cross-entropy statistical function).



```
print("taux d'erreures =", np.round(taux,2), '%')
precision_validation = 100 - np.round(taux,2)
print("precision à la validation =", precision_validation, '%')
```

calamenthe  
233  
calamenthe  
234  
calamenthe  
235  
calamenthe  
236  
calamenthe  
237  
calamenthe  
238  
calamenthe  
239  
calamenthe  
taux d'erreures = 6.67 %  
precision à la validation = 93.33 %

Figure 26: Screenshot of Inception Model's Validation Accuracy

For training, the accuracy is equal to 99,78%. On the other hand, several tests on the model illustrate 87% accuracy.

```

raw prediction : [[3.67242325e-11 1.91718377e-11 4.15387258e-07 9.61564539e-10
3.59963956e-12 2.31394193e-10 1.69717337e-11 9.99992843e-01
2.15725921e-12 5.28034226e-10 6.51484946e-07 3.50064983e-08
9.75970435e-09 4.34959487e-09 4.90473536e-09 3.10305809e-08
1.84119409e-08 5.27380771e-08 1.35604999e-09 3.10107826e-09
1.21748431e-08 2.53242747e-09 3.20544030e-06 1.77329655e-10
9.31470024e-13 1.61709112e-13 3.38659042e-10 2.06178523e-07
2.10748763e-09 3.95424063e-10 2.75589253e-10 2.41518916e-10
8.82053719e-12 2.36775812e-08 2.26647163e-10 5.13285280e-10
2.30563688e-09 5.00657202e-08 1.22213638e-09 2.31032551e-12
5.37824853e-10 2.78846471e-09 9.80320877e-10 1.52499078e-08
5.20656967e-09 1.44150311e-11 4.35656795e-10 1.77980139e-12
9.44964008e-12 5.07137595e-10 6.73827424e-10 1.87927049e-11
1.08578016e-08 2.38188515e-06 2.00217149e-10]] buis 0.9999928434172763

```

Figure 27: Screenshot of Inception Model's raw prediction while testing

Similarly as MobileNet, several tests were made on Inception v3 Model. The same test sample is provided to all the compared neural architectures. Figure 22 illustrates some screenshots from Inception Tests.



Figure 28: Screenshot of Inception Model's Tests

In a parallel way, the learning curve also illustrates that the model works fine: the cost curve decreases throughout different steps of learning. Accordingly, the cross-validation curve increasingly moves up to approximately 0.96 score which by the way is related to the same previous statistical formula:

$$\text{The Score} = 2 * (\text{recall} * \text{precision}) / (\text{recall} + \text{precision})$$

### 3. Inception Resnet v2

#### 3.1. Pre-processing: Data Set and Data Augmentation

Same as Inception and MobileNet, Resnet requires some particularities in terms of building the most adequate Dataset and provide the wright Data Augmentation operations. In fact, classic operation such as cropping (ratio =  $\frac{3}{4}$ ), Gaussian filters (mean = 0 and deviation = 0.1), PCA Filtering (altering the intensities of the RBG channels), multi-scale operations, horizontal flip augmentations, random rotations (angle = [-30,30]), random brightness and zoom.

The input size of Inception Resnet v2 neural network is (299,299). The images were then cropped and resized accordingly.

To crop it all, Data preparation, Train-Time Augmentation and Test-Time Augmentation are mainly the same operations and filters for the 3 compared architectures. The little differences are related to some specific exigency related to each and every architecture (The input size on MobileNet is (244,244)).

#### 3.2. PaaS: Colab, Amazon Web Services and Kaggle

Same as Inception Model, ResNet algorithm runs under the same cloud computing specifications. In fact, features extractions were performed on Amazon Web Services. AWS provides the “Sage Maker”; Machine Learning and Deep Learning train/test end point. It creates an AWS instance with no full control of desired server, security protocols, network and sub-network set up, data-encryption...

On the other hand, EC2 provides the AMI: Amazon Machine Image giving the developer the full control of everything. We stick to the first option regarding our needs.

Let’s notice that the Data had 4 copies; first on Local (NS PC Hard Disk), than on Google Drive, than on S3 AWS cloud Drive and finally on Kaggle. At every cloud computing, it is easier to get your data set imported on the PaaS in which you will perform features

extraction or training or even validating and testing. Getting your Data in just one emplacement can provide some importation errors and exceptions and leads to bugs and misunderstanding in paths and configurations.

Accordingly, the environment was set up (packages importations, Tensor flow back end configuration, versions compatibilities...) in all the PaaS through the same specifications and requirements.

Kaggle freely offered to us the GPU that performed ResNet on training, validating and testing with no processing troubles our memory allocations errors.

#### **Kaggle CPU specifications:**

- 4 CPU Cores
- 17 GB RAM
- 5 GB Auto-saved HDD
- 16 GB Temporary Scratchpad

#### **Kaggle GPU specifications:**

- 2 GPU Cores
- 14 GB RAM
- Additional dedicated NVIDIA Tesla K80

### **3.3. Features Extraction and Confusion Matrix**

Features were extracted via AWS and are sized 10 Go (it's 12 Go in Inception and just 5 Go for MobileNet). Getting a light-weighted model is one of Natural Solutions priorities. For that reason, output continuous size comparison is a must.

On the other hand, the confusion matrix is properly diagonal matrix illustrating the fact the model building and the features extractions went good with no remarkable troubles.

```
[INFO] creating model...
[INFO] evaluating model...
[INFO] saving model...
[INFO] confusion matrix
```

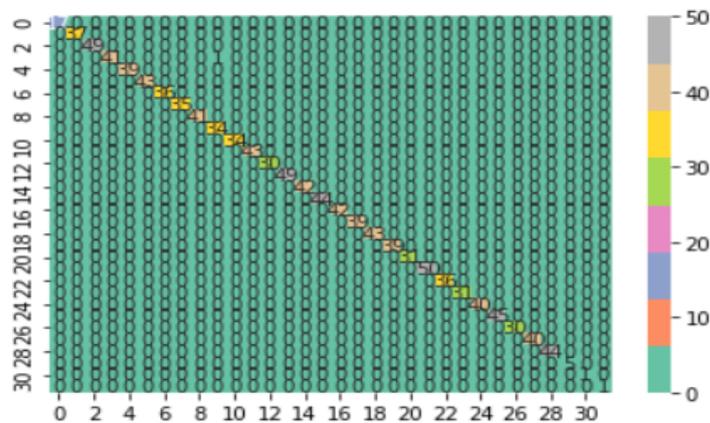


Figure 29: ResNet Model's diagonal confusion Matrix

### 3.4. Bottleneck approach and Transfer Learning

Same as inception, we used the bottleneck technic not only to get better results in terms of precision but also to extract additional extremely useful information with less parameters and less features maps.

In fact, we don't update the frozen convolution layers. We use it to circulate our images outside in order to extract more features which we instantly train at the dense layers to prepare the classification. Number of epochs is 10 and the batch size is 256 images. The used architecture is obviously VGG (the whole network, the whole layers even the densest layers were used) and same as Inception (check the previous chapter), the extracted additional information are stored in a NumPy array.

VGG was pre-trained somewhere (ImageNet...) to recognize some classes on some general extremely large dataset (millions of photos). Inspired from the transfer learning paradigm and the Fine-Tuning approach, the Bottleneck is, through VGG, in terms of diagram and architecture, an additional block for training and extracting information situated between the Convolutions block and the Dense Layers.

Bottleneck and Fine Tuning are both Transfer learning applications. They have lot in common technically but they are two different Deep Learning approaches. Generally, both of them provided a beautiful added value on the model increasing its accuracy and probably helps combatting over-fitting.

### 3.5. Validations and Tests

ResNet architectures generally reach very spectacular accuracies and scores in several Image Recognition Competitions. ResNet CNNs are able to provide trust-worthy solutions for mission-critical IoT applications or for any AI problem statement requiring very respectful accuracies and precisions. If you can provide the large computations and memory requirements, ResNet is one of the top used architectures worldwide in lots of engineering fields.

For our Flower Recognition Model, Inception ResNet v2 reaches 93,75% of accuracy in Validation (6,25% validation error percentage calculated through the cross-entropy statistical function).



```
InceptionResnetV2 Draft saved
File Edit Insert Run Help
+ Add Data Commit
Draft Session (1108m)
Run All
err = 1
elif ((prediction == 'camomille_sauvage') and ((i < 217) or (i > 240))):
err = 1
print (prediction)
taux = (err * 100) / 240
print("taux d'erreures =", np.round(taux,2), "%")
precision_validation = 100 - np.round(taux,2)
print("precision à la validation =", precision_validation, "%")
camomille_sauvage
233
camomille_sauvage
234
camomille_sauvage
235
camomille_sauvage
236
camomille_sauvage
237
camomille_sauvage
238
camomille_sauvage
239
camomille_sauvage
taux d'erreures = 6.25 %
precision à la validation = 93.75 %
```

Figure 30: Screenshot of ResNet Model's Validation Accuracy

For training, the accuracy is equal to 99,96%. On the other hand, several samples were tested on the model illustrating 88% accuracy.

```
raw prediction : [[7.85049738e-10 3.49505052e-08 2.39238841e-09 4.29207647e-06
2.22862690e-10 1.93600690e-12 7.09620458e-08 1.17590896e-10
2.35087623e-11 2.48893045e-09 4.42015977e-08 2.10220355e-10
1.44591618e-09 2.39098031e-10 3.16402631e-09 5.71726436e-12
9.99642451e-01 2.15387363e-10 9.71461102e-09 2.47074595e-09
1.50719801e-12 4.60704772e-07 9.16822179e-08 4.69380735e-07
1.38618419e-05 2.40070895e-09 1.74186057e-11 1.11287213e-11
5.62886261e-10 2.46930442e-08 2.18865623e-08 7.51237383e-08
1.91121458e-11 2.00979352e-11 1.32411927e-08 2.68178544e-06
2.49357958e-08 7.48340126e-11 3.55028568e-11 3.90699810e-09
2.36382433e-11 1.58591016e-09 1.12712058e-09 4.25569683e-09
1.22213392e-09 5.36708858e-09 4.71926699e-09 4.81988659e-11
2.66677047e-04 1.02343628e-09 6.39016616e-07 5.63656076e-07
6.71805945e-05 3.14586067e-13 2.71129014e-07]] catananche_bleue 0.9996424511698695
```

Figure 31: Screenshot of ResNet Model's raw prediction while testing

Figure 27 illustrates the calculated probabilities at the classification last layer through the ResNet architecture similarly to MobileNet and Inception (see previous chapters).



Figure 32: Screenshot of ResNet Model's Tests

## 4. State-of-the-art neural architectures Comparison

### 4.1. Summarizing results

Figure 32 is mainly the output of our comparison. It is a summarizing detailed table containing all the comparison criteria and the variety of steps we crossed while designing, deploying, validating and testing our model.

	Mobile Net	Inception v3	Resnet v2
Validation Error	4.66%	6.67%	6.25%
Validation Accuracy	95.34%	93.33%	93.75%

Test Accuracy	85%	87%	88%
Time Delay (average)	90 ms	110 ms	120 ms
Input Size	(224,224)	(299, 299)	(299,299)
Fine Tuning	VGG v19	VGG v19	VGG V19
Output Size (features)	5 Go	12 Go	10 Go
CPU Requirements and Specifications	Intel Core i7 (6 <sup>th</sup> gen) 1.8 GHz 4 CPU Cores 8 GB RAM	4 CPU Cores 17 GB RAM 5 GB auto saved Disk 16 GB of temporary Scratchpad	4 CPU Cores 17 GB RAM 5 GB auto saved Disk 16 GB of temporary Scratchpad
GPU Requirements and Specifications	Intel UHD Graphics 620 3.9 Gb 1 GPU Core	2 GPU Cores + additional NVIDIA Tesla K80 14 GB RAM	2 GPU Cores + additional NVIDIA Tesla K80 14 GB RAM
Weights	'imagenet'	'imagenet'	'imagenet'
Seed	9	9	9
Memory Allocations	dtype32	dtype64	dtype64
Data Set Segmentation	10% test 10% validation 80% train	10% test 10% validation 80% train	10% test 10% validation 80% train
Include Top	False	False	False
Features Extractions (Cloud or Local)	Local	Amazon WS + Google Collab	Amazon WS + Google Collab
Validation and Tests	Local	Kaggle	Kaggle
Confusion Matrix	Diagonal Matrix	Diagonal Matrix	Diagonal Matrix
Offline Availability	True (Embedded Mobilnet)	False (Only if we're able to provide high computational performances)	False (Only if we're able to provide high computational performances)
Learning Rate	0.0001	0.0001	0.0001
Data Set (Vegetal species)	55	55	55
Back end Framework	Tensor Flow	Tensor Flow	Tensor Flow
Library/Package	Keras	Keras	Keras
Open Source	Google released 16 pre-trained checkpoints	No (*)	The 1000+ layer net is open source
Classifier Model	Logistic Regression	Logistic Regression	Logistic Regression

Figure 33: Comparison summarizing Table

#### 4.2. Intersections and Differences

The compared models are mainly Convolutional neural networks illustrating the state-of-the-art of deep learning. Inception and ResNet have a lot in common; their heavy weight, good accuracies and millions of parameters designing the algorithm. They are deeply used in mission-critical application in which we provide huge computational resources in order to

get the best possible predictions. Besides, both of them share the fact that they have been developed and introduced to the world by Google Brain. Those architectures while treating huge datasets strongly require a good understanding to the different blocs making the neural network and great cloud computing skills. Building a model via PaaS is such a magnificent discovery for me.

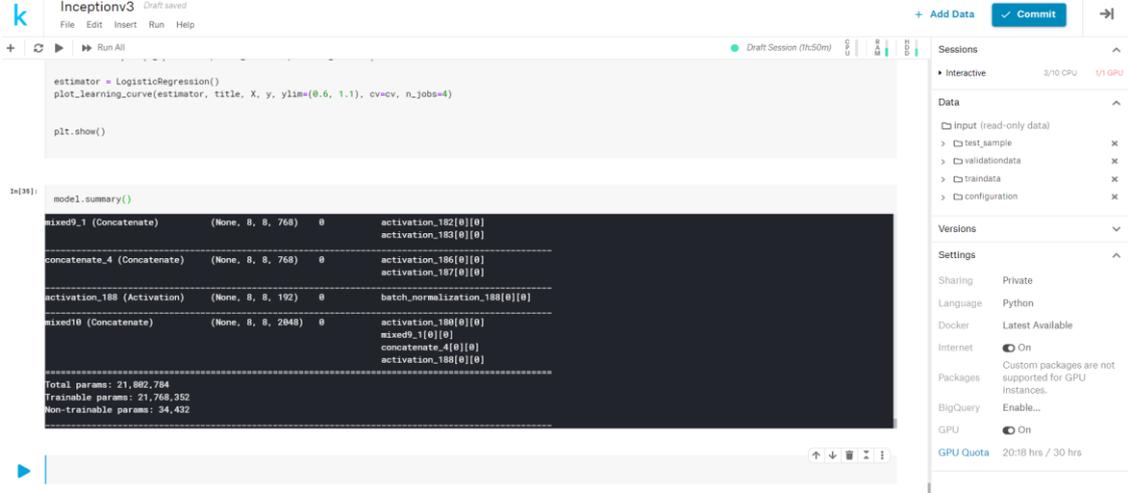


Figure 34: Inception model summary

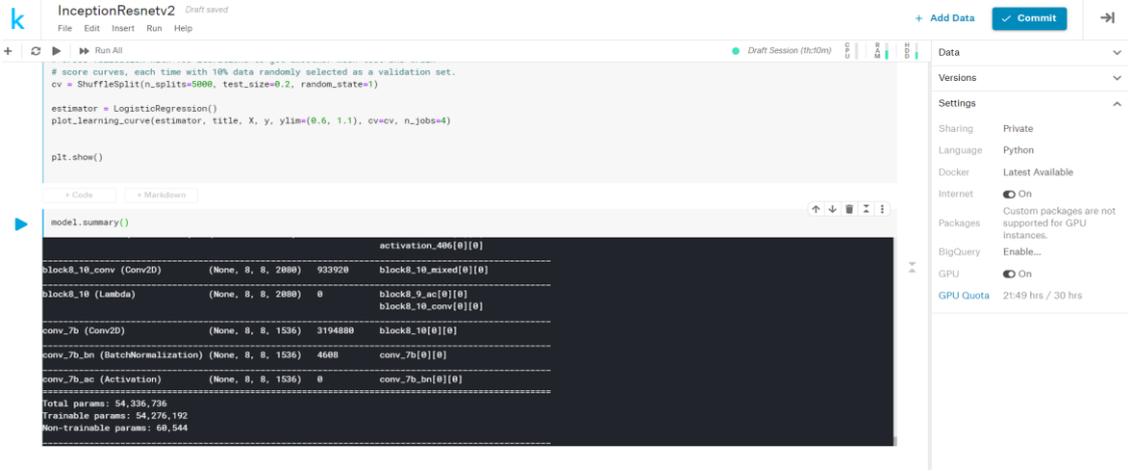


Figure 35: Resnet model summary

Figures 34 and 35 provide the model summary for Inception and ResNet. In fact, ResNet has 54,336,736 total parameters (54,276,192 trainable and 60,544 non-trainable.) Inception has 21,802,784 total parameters (21,768,352 trainable and 34,432 non trainable).

On the other hand, MobileNet is different. The light-weight architecture is mainly designed in order to get embedded on Mobile Systems and offers a respectful CNN with high accuracies, fast predictions and so much less parameters to tune and handle. MobileNet and SqueezeNet are trending architectures being used and deployed in several application ICT domains. Figure 36 is the MobileNet model summary (just 3,228,864 total parameters through 3,206,976 trainable and 21,888 non-trainable)

```
Entrée [14]: model.summary()
conv_pw_12_relu (ReLU) (None, 7, 7, 1024) 0
conv_dw_13 (DepthwiseConv2D) (None, 7, 7, 1024) 9216
conv_dw_13_bn (BatchNormaliz (None, 7, 7, 1024) 4096
conv_dw_13_relu (ReLU) (None, 7, 7, 1024) 0
conv_pw_13 (Conv2D) (None, 7, 7, 1024) 1048576
conv_pw_13_bn (BatchNormaliz (None, 7, 7, 1024) 4096
conv_pw_13_relu (ReLU) (None, 7, 7, 1024) 0
=====
Total params: 3,228,864
Trainable params: 3,206,976
Non-trainable params: 21,888
```

Figure 36: MobileNet model summary

Let's notice that non-trainable parameters are quite a broad subject. A straightforward example is to consider the case of any specific NN model and its architecture. Say we have already setup your network definition in Keras, and your architecture is something like 256->500->500->1. Based on this definition, we seem to have a Regression Model (one output) with two hidden layers (500 nodes each) and an input of 256.

One non-trainable parameters of your model is as a matter of fact the number of hidden layers itself. Other could be the nodes on each hidden layer (500 in this case), or even the nodes on each individual layer, giving you one parameter per layer plus the number of layers itself.

These parameters are "non-trainable" because you can't optimize its value with your training data. Training algorithms (like back-propagation) will optimize and update the weights of your network, which are the actual trainable parameters here (usually several thousands, depending on your connections). Your training data as it is can't help you determine those non-trainable parameters.

To crop it all, non-trainable parameters of a model are those that you will not be updating and optimized during training, and that have to be defined *a priori*, or passed as inputs. In some cases, we tune the model in order to train the non-trainable parameters of the hidden layers accordingly.

### 4.3. Summary and perspectives

To crop it all, Natural Solutions sticks with the MobileNet option. Getting a light-weight architecture with spectacular scores is the main objective of my internship. In fact, the project-s perspectives are:

- Getting the MobileNet model deployed and implemented in the EcoBalade Application (embedded on Mobile System).
- Build a Chatbot and gets the trustworthy exchange with the customer or the app user. The Chatbot explains the predicted results and provides a detailed description of the concerned vegetal species.
- The Chatbot is also the greatest opportunity to get familiar with the concept of explainability and interpretability which is the main objective regarding the Laboratory research.

- Explicability is the major part of the perspectives. Experimental protocols, validations and tests, bibliographic studies and laboratory group-work optimization and regularization research are the next tasks provided by QARMA deep learning of Centrale Marseille.

## **Conclusion**

This chapter resumes the technical work with detailed steps for the state-of-the-art neural network deep learning architectures comparison. Therefore, Natural Solutions choice was to stick with MobilNet model. In fact, it provides spectacular accuracies on our dataset regarding our needs with lightweight processing in terms of GPU and CPU specifications. The firm will start working on deploying the model on an embedded system. On the other hand, MobileNet provides a beautiful ground to the perspectives of the project soliciting interpretability and explicability deep learning state-of-the-art concepts.

As a matter of fact, building a deep learning model passes through Data construction and Augmentation, Software Environment preparations, features extractions, cloud computing if required, Fine Tuning through the transfer learning concept, validation and finally testing and deploying.

# Conclusion and perspectives

Designing and deploying image recognition models is one of the top trending computer vision problems nowadays providing spectacular results in terms of rapidity, reliability and accuracy. Deep learning neural architectures are the ultimate option to get precise predictions related to your specific problem statement. For instance, AI transfer learning concept is a brilliant key towards exploring the knowledge of a generalized model pre-trained on huge dataset with millions of photos which is rapidly and easily adapted to our application and needs.

Natural Solutions illustrates the ultimate need for a lightweight architecture, offering the beautiful ability to be embedded on a mobile system and to provide services to customers even through offline mode. This project mainly compares the state-of-the-art neural networks and highlights the focus on the quality of service per CPU and GPU requirements ratio. Regarding the respectful results we reached, MobileNet seems to be the best choice accordingly.

Besides, building the model, getting familiar with Tensor Flow backend, designing your strategies for learning, regularizing and optimizing the CNN is mainly a continuous work based on a set of validations, tests and mathematical interpretations through matrixes and curves. Cloud computing for heavy-weight architectures is increasingly required nowadays. PaaS provides spectacular CPU and GPU specifications offering the developer more processing and memories for greedy models.

However, the most important step in every AI or Machine Learning project is the Data analysis and treatment. In fact, it is not about who got the best algorithm, it is deeply about who got better Data set. Pre-processing is a major crucial phase which explains the continuous increasing need for specialists and data scientists in the AI international market.

On the other hand, interpretability and explainability are AI paradigms offering a trustworthy relationship between the machine and the human been. It is accordingly one of the top trending fields of investment, research and innovation worldwide. For that reason, explicability is a beautiful perspective post-project offering more confidence to the customers and more reliability in the provided EcoBalade Application.

# References

- [1] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017
- [2] EcoBalade official website, <http://www.ecobalade.fr/>, last accessed 18 October 2019.
- [3] Natural Solutions official website, <https://www.natural-solutions.eu/>, last accessed 15 October 2019
- [4] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *ICML*, 2018.
- [5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.
- [7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Conference on Neural Information Processing Systems*, 2012.
- [8] Laboratoire d'Informatique et Systèmes, <https://www.lis-lab.fr/en/>, last accessed 20 Aout 2019
- [9] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *ICLR*, 2018.
- [10] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," *arXiv preprint arXiv: 1807.11626*, 2018.
- [11] Computer Vision, an overview of Image Classification Architectures, <https://medium.com/overture-ai/4-chapters>, last accessed 26 September 2019
- [12] Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin, "Why Should I Trust You?" *Explaining the Predictions of Any Classifier*, *KDD* 2016.

# Annex A

## Explainability and Interpretability

### A.1 Justify the need for interpretability:

In mathematical logic, interpretability is a bidirectional relation between formal theories and approaches that expresses the possibility of translating one into the other. Cathy O’Neil says: “Models are opinions embedded in Mathematics”. Regarding the strong relationship between math theories and AI Models, in Information Technologies IT and Information and Communication Technologies ICT, interpretability is the concept of giving the machine or the artificial intelligence the ability to explain its own decision or prediction without the intervention of the developer or the designer. In fact, it is considered as one of the top trending research and innovation subjects in the world of Machine Learning.

Analyzing some real problem statements provides a concrete illustration justifying the need for this concept. For social media digital targeting, the AI should systematically refer its targeting choices with the user’s Data environment around him based on its different interactions and topics of interest. While in health care, it is obvious that a doctor explanation is wiser if the model provides intelligible explanation. In this case, an explanation is a set of symptoms with relative weights. The symptoms are either contributing to the prediction or illustrating evidence against it. The most detailed article that can illustrate this concept is titled “Why Should I trust you?” from the IEEE Explore written by Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin [12].

On the other hand, in case of Police crime predictions, Police officers have prior knowledge about the application domain which they can use to accept or reject crime prediction if they understand the reasoning behind it. This can be generalized to all the problem statements requiring the concept of interpretability.

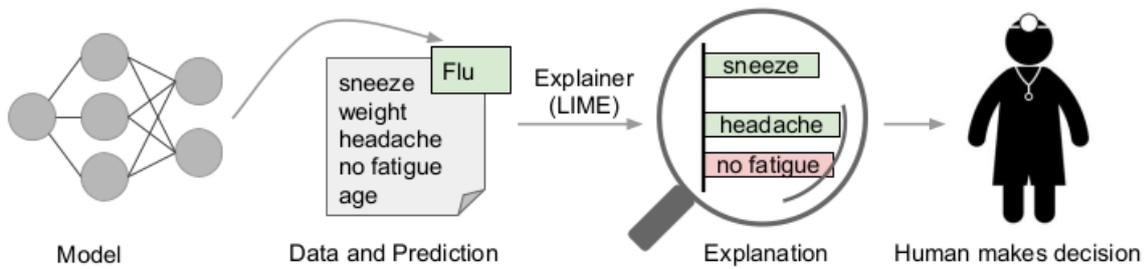


Figure 1: Explaining individual predictions simplified illustration [12]

A model predicts that a patient has the flu, and LIME highlights the symptoms in the patient’s history that led to the prediction. Sneeze and headache are portrayed as contributing to the “flu” prediction, while sneeze “no fatigue” is evidence against it. With these, a doctor can make an informed decision about whether to trust the model’s prediction. [12]

## A.2 Technical background and Python Development tools:

Interpretability is technically outlining a number of desired characteristics from explanation algorithmic specific methods. In fact, an essential criterion for explanations is that they must be interpretable. For instance, it should provide qualitative understanding between the input variables and the response. We note that interpretability must take into account and consideration the user’s limitations.

Some models are easy to interpret such as Linear/Logistic Regression or Single Decision Tree. On the other hand, others are harder such as random forest, boosting.. Because it is hard to understand the role of each feature and doesn’t really tell the developer if feature affects decision positively or negatively. Deep Neural Networks are characterized by no straightforward way to relate output to input layer in addition to the block boxes inside the network.

Therefore, there are two main interpretability types:

- Local: explain how and why a specific prediction was made (“why did our model predict that patient X has disease Y”). In fact, in python we code this instruction “eli5.show\_prediction(model, observation)” which will systematically plot the corresponding Fully Connected layer’s probabilities to each class in case of classification problem statement.
- Global: explain how our model works overall; feature importance is a global interpretation with amplitude only, no direction (never changes the outcome). For instance, the python script code “eli5.show\_weights(model)” will plot the weights of our model. It is extremely important not to confuse those weights with those related to the neural architectures. Interpretability weights are related to the features. In fact, the machine decides to give more importance to a feature than another one.

In my project for example, the machine can eventually consider that the color feature is more important than the size feature to predict the corresponding class of the flower and explain it accordingly.

For Python developers, we've got the ELI5 which is a Python implementation that can be used to interpret sklearn-like models, white-box models and provides permutation importance for black-box models through the "perm = PermutationImportance(model))" function.

In addition, two approaches are involved:

- LIME: Local Interpretable Model-agnostic Explanation
- SHAP: Shapley Additive Explanation

Let's get my initiative to enroll "Interpretability\_eli5-skeleton", the process passes through different phases:

- 1- Flat: after Categorical Hot Encoding in which we try to get the features and the names corresponding to each and every category, we list them through a specific index or "key" representing the name of the class.
- 2- Check it pd.DataFrame for a beautiful plot and a clearer data presentation.
- 3- In order to explain a specific prediction, we enter i=4 as the index of the corresponding class and we write X\_test.iloc(i) to plot the column of features explaining the decision.
- 4- eli5.show\_predictions provides the set of probabilities for each characteristics. In fact, it can show as if the model is good at predicting based on feature1 more than feature5. (For every feature, there will be a corresponding contribution score. We can get negative scores! Which mainly shows that the feature is far from being helpful to predict for that specific class at a very specific problem statement.)
- 5- eli5.show\_weights plots the corresponding weight to each feature. At a balanced Data Set case, we logically get the same weight for all the features. This depends on the use case; we can get an imbalanced Data Set or simply imbalance a balanced one. In fact, we can manually try to put the focus on some features more than others if we think that they might get better contributions for more accurate predictions. In addition, for Decision Tree algorithm, this " eli5.show\_weights" function will systematically plot a detailed graphic representation to the whole tree; that is mainly a graphic precise explanation to the results.

However, what can we do when it's about a black boxed model? Eli5 won't have access to the features. Here is when LIME Method is extremely useful. Let's understand every acronym:

- Local: Explains why a single Data point was predicted in that specific way.
- Model-agnostic: mainly refers to black boxed models in which we don't know how the Algorithm makes predictions.

Let's focus on Figure1 illustrating the following steps: Model > data, tune, fit, predict.. > explain > human decides.

How does it work? The algorithm generates some data through random sampling. As long as we are in a non-linear case, this sample distribution will be our "linear" Local space of work. The algorithm fits through Linear Regression at that specific local region.

Two potential problems can take place:

- Data Leakage which is mainly the presence of an obvious correlation between training data and validation data.
- Data set Shift which is related to the fact that training data is very different from test data.

For example, a possible interpretable representation for text classification is a binary vector indicating the presence or absence of a word, even though the classifier may use more complex (and incomprehensible) features such as word embedding.

Likewise for image classification, an interpretable representation may be a binary vector indicating the “presence” or “absence” of a contiguous patch of similar pixels (a super-pixel), while the classifier may represent the image as a tensor with three color channels per pixel. When using sparse linear explanations for image classifiers, one may wish to just highlight the super-pixels with positive weight towards a specific class, as they give intuition as to why the model would think that class may be presented.

### A.3 Image Classification and Detection through Convolution Neural Networks and Dense Neural Networks:

This chapter illustrates 5 primordial steps in which interpretability can become a key performance dimension of responsible AI within organizations. It helps to support the other dimensions: bias, fairness, robustness and security, while being underpinned by a foundation of end-to-end governance, and enable ethically sound decisions as well as in compliance with regulations.

Making AI interpretable can foster trust, enable control and make the results produced by machine learning more actionable.



Figure 2: The 5 practical steps to make AI interpretable

For our specific problem statement, the goal is to plot the features (color of plant, dimension, size, nature of leaves...) that could probably explain the prediction. That is mainly what we can call: performing interpretability in our EcoBalade Application. In addition, the level of rigour could depend on the user itself. For a “normal” citizen, with a mediocre ecology background, the basic features are enough. “Coquelicot” is predicted simply because the color is red and the specific form of leaves in the plant itself. For an ecologist or an environmentalist, features should be specialized and more details should be given. Therefore, the level of rigour increases.

That’s how interpretability mainly depends on understanding the stakeholder and regulatory requirements. The specific need for explainability remains obvious and was explained several times in the report and the annex. In fact, building a trust-worthy conversation between the EcoBalade

App user and the machine is the real long-term challenge. For that reason, the main strategy is based on the technical background and the Python Development tools mentioned in the previous chapter.

## A.4 Building the Application's Chat bot

One of the most challenging innovations towards interpretability is building a trustworthy communication and systematic exchange between the application's user and the machine itself.

As an initiative for the "post-graduation-project" perspectives, the firm proposed the idea of a chat bot for the EcoBalade Application. A typical python script through high-level built-in functions provides the chat bot. The long-term idea behind the chat bot design is to link it (data mapping) with the interpretability provided by the machine systematically explaining its predictions. I just started the design, and this part of the project remains one of the most challenging perspectives for the after-graduation work.

This chapter of the Annex A is an overview for the built chat bot, inspired from the medium.com article: "Building a chatbot in python using NLTK"

In fact, a chatbot is an artificial intelligence-powered piece of software in a device (Siri, Alexa, Google Assistant etc), application, website or other networks that try to gauge consumer's needs and then assist them to perform a particular task like a commercial transaction, hotel booking, form submission.. Today almost every company has a chatbot deployed to engage with the users. Some of the ways in which companies are using chatbots are:

- To deliver flight information and to justify predictions
- to connect customers and their finances in order to build a trustworthy communication
- As customer or app user's support

For instance, there's a variety of chatbots for different problem statements. However, there are mainly two approaches that can define their functionalities:

- In a Rule-based approach, a bot answers questions based on some rules on which it is trained on. The rules defined can be very simple to very complex. The bots can handle simple queries but fail to manage complex ones.
- Self-learning bots are the ones that use some Machine Learning-based approaches and are definitely more efficient than rule-based bots. These bots can be of further two types: Retrieval Based or Generative.

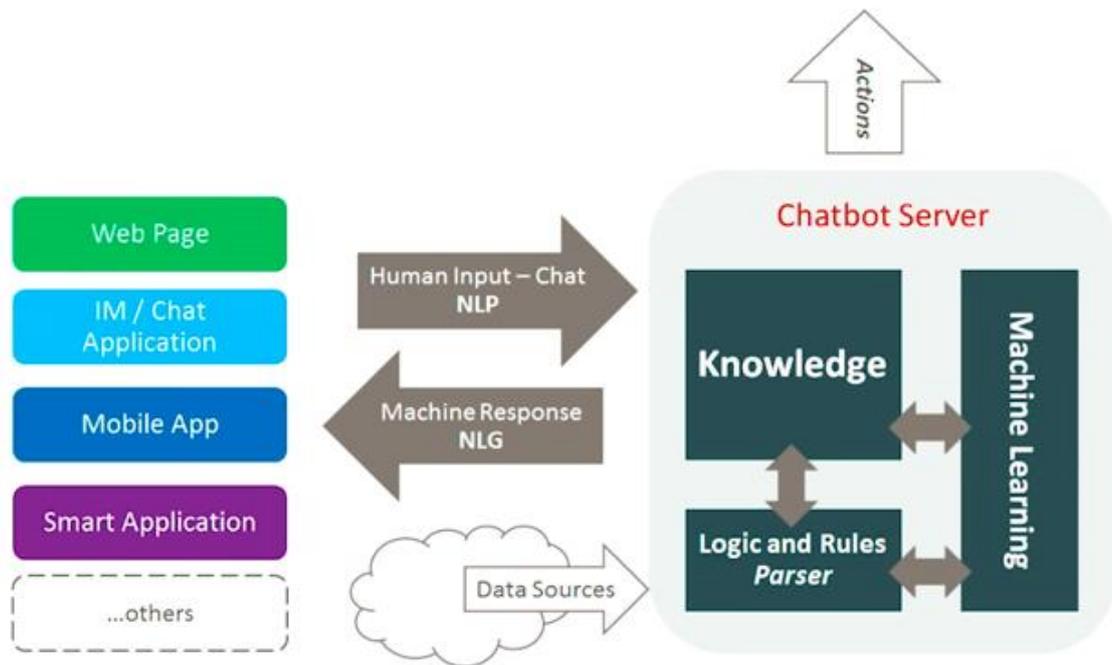


Figure 3: Simplified Diagram illustrating the Anatomy of a Chatbot

Our chatbot is built through the NLTK. The Natural Language Toolkit is a leading platform for building Python programs to work with human language data. Therefore, it provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries. By the way, NLTK has been called “a wonderful tool for teaching and working in, computational linguistics using Python,” and “an amazing library to play with natural language.” [11]

Building the chatbot passes through a variety of steps:

- Installing the NLTK packages and preparing the software environment (check the compatibility of versions)
- Pre-process the text through NLTK package: convert the whole text to uppercase or lowercase characters and process the Tokenization (convert the strings into lists of tokens.)
- Removing noise and stop words
- Import the dictionary often called the “bag of words” before converting the words into binary vectors.
- Perform Cosine Similarity: measure of similarity between two non-zero vectors.
- Read an input file called a Corpus.
- Pre-process the raw data from the input file by defining a function called LemTokens which will take as input the tokens and return normalized tokens
- Keywords matching
- The response we will be generating is mainly based on the Similarity of documents; defining a response function search the user’s utterance for one or more known keywords and returns one of several possible responses. If it doesn’t find the input matching any of the keywords, it returns a response:” I am sorry! I don’t understand you”

- Last but not least, we should enter the expressions of greeting in order to program the chatbot on how to begin and end a conversation

Building a chatbot is one of the basics of python programming. Lots of technics, approaches and high-level programming packages provides very simple ways to build different types of Chatbots.

The most challenging part is the mapping between the chatbot itself and the machine's interpretability. The chatbot should provide a good conversation with the user, understand the keywords and be able not only to answer but also to justify the machine's prediction. Therefore, this work could illustrate one of the biggest perspectives of the graduation project.

To crop it all, this report annex provides an overview about my work on the interpretability and explainability. The first paragraph justifies the need for such a technological concept. Moreover, the technical background resulting from a bibliographic research study is primordial to get familiar with the trending innovations around interpretability. The third paragraph is mainly a specialized explainability related to our problem statement: CNN, Computer Vision and Image Classification. Finally, building the chatbot remains just one basic step before getting the mapping between the interpretability and the machine-user trust-worthy conversation which illustrates a continuity to the whole graduation project work and open a variety of further perspectives such us getting more animal and vegetal species (BigData) to predict while justifying and debating each and every classification (Interpretability).