

الجمهورية الجزائرية الديمقراطية الشعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Scientific Research

جامعة غرداية

University of Ghardaia

كلية العلوم و التكنولوجيا

Faculty of Science and Technology

قسم الرياضيات و الإعلام الآلي

Department of Mathematics and Computer Science



## THESIS

Presented for the degree of **Master**

**In:** Computer Science **Specialty:** Intelligence System for Knowledge Extraction

**By:** Abderrahim Mokhtar Sehil

**Theme**

# Image Classification using Deep Learning

Jury members

M. Kerrache Chaker Abdelaziz	Doctor	Univ. Ghardaia	President
M. Bouhani Abdelkader	MAB	Univ. Ghardaia	Examiner
M. Adjila Abderrahmane	MAB	Univ. Ghardaia	Examiner
M. Mahdjoub Youcef	MAA	Univ. Ghardaia	Supervisor

College year : 2018/2019

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ  
مَكْتَبَةُ  
۱۴۲۰



# Dedicated to

## My Dear Mother

The most precious person to me and the purest source of inspiration and positivity, who stood by my side whenever I fell, I really can't express how much I'm really grateful to have you ...

## My Father

The person who taught me trust in Allah and to never give up, I really can't find the proper words to describe my appreciation and gratitude to the support given by you in life and through all my educational career.

## My Sisters AND Brother

Aya, Yacine and Mimi the greatest gift of God.

## My Squad

Hamza , the brother from another Mother and the buddy for life.

Arzzedine , the source of insanity you're also the brother from another mother don't get jealous lol.

Couldn't have a better life without you guys, I really appreciate your presence weird people.

## My Special Friends

Amina , Hadjer and Ibtissem, Amira I'm really glad to have such wonderful persons and beautiful souls around , much appreciated !!

AND

My colleague Abderrahmane, Charaf, Mamel & everyone who wished me the good luck and helped me even with a word.

Cordially  
Rahim.

# Acknowledgment

“Words fly away, writings remain”

In the name of "ALLAH", The most beneficent and merciful who gave us strength and knowledge to complete this thesis.

Firstly, I would like to express my sincere sense of gratitude to my supervisor Mister **Mahdjoub Youcef** who offered his continuous advice and encouragement throughout the course of this thesis. I thank him for the guidance and great effort he puts into training me in the scientific field.

I am deeply grateful to all members of the jury for agreeing to read this manuscript and to participate in the defense of this thesis.

I thank all the teachers who taught me in the five past years for the vast amounts of information.

For all those who participated in the development of this work.

# Abstract

Deep learning (DL) is a sub-domain of machine learning (ML), it consists of artificial neural networks. The word (deep) refers to the number of layers in the network, the more layers there are, the deeper the network. Advanced tools and techniques have significantly improved Deep Learning algorithms to the point where they can outperform humans in classifying images. Among artificial neural network architectures there are convolutional neural networks (CNN). Image classification is a classic problem in the fields of image processing and convolutional neural networks. Our work studies the images classification by using an architecture of a convolutional neural network on the CIFAR-10 Dataset.

## **Keywords :**

Machine Learning (ML), Deep Learning (DL), images classification, convolutional neural networks (CNN), CIFAR-10.

# Résumé

Le Deep Learning (DL) est un sous-domaine de Machine Learning (ML), il est constitué des réseaux de neurones artificiels. Le terme Deep (profond) désigne le nombre de couches dans le réseau, plus il y a des couches, plus le réseau est profond. Des outils et des techniques avancés ont amélioré de façon spectaculaire les algorithmes du Deep Learning, au point où ils peuvent surpasser les humains à classer les images. Parmi les architectures de réseaux de neurones artificiels il y a le réseau de neurones convolutionnels (CNN). La classification d'images est problème classique dans les domaines du traitement d'images et les réseaux de neurones convolutionnels. Ce mémoire étudie la classification d'images en employant le Dataset CIFAR-10 à partir d'une architecture d'un réseau de neurone convolutionnel.

## Mots-clés :

Machine Learning (ML), Deep Learning (DL), classification des images, réseau de neurone convolutionnel (CNN), CIFAR-10.

## ملخص

التعلم العميق ( $DL$ ) ، فئة من التعلم الآلي ( $ML$ ) ، والذي يتضمن شبكات العصبونات الاصطناعية. تشير الكلمة (العميقة) إلى عدد الطبقات في الشبكة ، وكلما زاد عدد الطبقات، زاد عمق الشبكة. لقد حسنت الأدوات والتقنيات المتقدمة خوارزميات التعلم العميق بشكل كبير إلى درجة تمكنهم من التفوق على البشر في تصنيف الصور. من بين أبنية الشبكات العصبية الاصطناعية هناك الشبكات العصبية التلافيفية ( $CNN$ ). يعتبر تصنيف الصور مشكلة كلاسيكية في مجالات معالجة الصور والشبكات العصبية التلافيفية. يدرس عملنا في هذه المذكرة تصنيف الصور باستخدام بنية شبكة عصبية تلافيفية على مجموعة بيانات  $CIFAR - 10$ .

### الكلمات المفتاحية :

التعلم الآلي ( $ML$ ) ، التعلم العميق ( $DL$ ) ، تصنيف الصور ، الشبكات العصبية التلافيفية ( $CNN$ ) ،  $CIFAR - 10$ .



# Contents

<b>Introduction</b>	<b>1</b>
<b>1 Machine learning and Image classification</b>	<b>2</b>
1.1 Introduction	2
1.2 Machine learning	2
1.2.1 What is Machine learning?	2
1.2.2 The importance of machine learning	2
1.2.3 Types of machine learning	3
1.2.3.1 Supervised machine learning	3
1.2.3.2 Unsupervised machine learning	3
1.2.3.3 Semi-supervised machine learning	4
1.2.3.4 Reinforcement machine learning	4
1.3 Image Classification	5
1.3.1 Classification Definition	5
1.3.2 Image classification motivations	5
1.3.3 Types of classification algorithms	6
1.3.3.1 $k$ -Nearest Neighbor	6
1.3.3.2 $K$ -means	7
1.3.3.3 Fuzzy c-means	7
1.3.3.4 Support Vector Machine	8
1.3.3.5 Decision Trees	9
1.4 Image classification and Machine learning	10
1.5 Conclusion	10
<b>2 Deep learning and Convolutional Neural Network</b>	<b>11</b>
2.1 Introduction	11
2.2 Definition of deep learning	11
2.3 Biological neural network	12
2.3.1 Neurons	12
2.3.2 Axons	12
2.3.3 Dendrites	12
2.3.4 Synapses	13
2.3.5 Soma (Cell Body)	13
2.4 Artificial Networks (ANNs)	13
2.5 Different types of Neural Networks	14
2.5.1 Feedforward Neural Network	14
2.5.2 Radial basis function Neural Network	14
2.5.3 Kohonen Self Organizing Neural Network	15

2.5.4	Recurrent Neural Network (RNN)	16
2.5.5	Convolutional Neural Network (CNN)	16
2.6	Convolutional Neural Network Layers	17
2.6.1	Convolutional Layer	17
2.6.2	Pooling Layer	17
2.6.3	Fully Connected Layer	18
2.6.4	Dropout Layer	18
2.7	Convolutional Neural Network Architectures	19
2.7.1	LeNet-5 - LeCun & al	19
2.7.2	AlexNet	20
2.7.3	VGG-16	21
2.7.4	GoogLeNet (Inception)	21
2.7.5	ResNet-Kaiming He & al	22
2.8	Training of an Artificial Neural Network	23
2.8.1	Image Preprocessing	23
2.8.2	Loss Functions	24
2.8.2.1	Cross Entropy	24
2.8.2.2	Binary Cross Entropy	24
2.8.2.3	Categorical Cross Entropy	24
2.8.2.4	Mean Squared Error (MSE)	25
2.8.2.5	Mean Squared Logarithmic Error (MSLE)	25
2.8.2.6	Mean Absolute Error (MAE)	25
2.8.3	Optimizers	26
2.8.3.1	Stochastic Gradient Decent	26
2.8.3.2	Adagrad	27
2.8.3.3	RMSprop	27
2.8.3.4	Adam	27
2.8.4	Activation Functions	27
2.8.4.1	ReLU (Rectified Linear Unit)	27
2.8.4.2	Softmax	28
2.8.4.3	Sigmoid	28
2.8.5	Regularization	29
2.8.5.1	Dataset augmentation	29
2.8.5.2	Early stopping	29
2.8.5.3	Dropout	30
2.8.5.4	Dense-sparse-dense training	30
2.8.5.5	Batch Normalization	31
2.8.6	Transfer Learning	31
2.9	Conclusion	33
<b>3</b>	<b>Implementation</b>	<b>34</b>
3.1	Introduction	34
3.2	Softwares and tools	34
3.2.1	Python	34
3.2.2	Tensorflow	34
3.2.3	Keras	35
3.2.4	Jupyter Notebook	35
3.2.5	Google Colab	36

3.2.6	Hardware . . . . .	36
3.3	Dataset . . . . .	37
3.4	CNN's Architecture . . . . .	38
3.5	Results . . . . .	40
3.6	Conclusion . . . . .	43
	<b>Conclusion</b>	<b>44</b>

# List of Figures

1.1	Machine Learning subcategories . . . . .	3
1.2	Algorithms of supervised machine learning . . . . .	4
1.3	<i>KNN</i> algorithm's application example . . . . .	6
1.4	Fuzzy c-means algorithm's example . . . . .	7
1.5	SVM's application example . . . . .	8
1.6	Decision trees application example . . . . .	9
2.1	The relation between AI, machine learning and deep learning . . . . .	11
2.2	Typical biological neuron . . . . .	12
2.3	A simple artificial neuron network . . . . .	13
2.4	Feedforward neuron network . . . . .	14
2.5	Architecture of an RBF Network . . . . .	15
2.6	Architecture of a SOM network . . . . .	15
2.7	Architecture of a Recurrent Neural Network . . . . .	16
2.8	Convolution Operation . . . . .	17
2.9	Pooling Layer . . . . .	18
2.10	Dropout process . . . . .	18
2.11	LeNet-5's Architecture . . . . .	19
2.12	AlexNt's Architecture . . . . .	20
2.13	A figure that represents the architecture of VGG-16 CNN . . . . .	21
2.14	GoogLeNet's architecture . . . . .	22
2.15	Residual Network's architecture . . . . .	23
2.16	ReLU function . . . . .	27
2.17	Segmoid function . . . . .	28
2.18	Early stopping process while training . . . . .	30
2.19	Dense-sparse-dense technique . . . . .	31
3.1	Jupyter Notebook interface . . . . .	35
3.2	Google Collab interface . . . . .	36
3.3	CIFAR10 dataset sample visualization . . . . .	37
3.4	The model's architevture . . . . .	38
3.5	A random sample from the dataset . . . . .	40
3.6	Test Accuracy and Model Loss plots . . . . .	40
3.7	The confusion matrix of the model . . . . .	41
3.8	The error rate of the model . . . . .	41
3.9	The error rate of the model . . . . .	42
3.10	The effect of the model's layers . . . . .	42
3.11	A sample of predicted images with their labels . . . . .	43

# List of Tables

2.1	A list of pretrained neural network . . . . .	32
3.1	The hardware used to run the tests . . . . .	36
3.2	The architecture of the used model for my test . . . . .	39
3.3	The resume of obtained results . . . . .	43

# Introduction

Today's world is characterized by the availability of enormous amounts of information and data where they are stored, processed, indexed and searched for by artificial intelligence-based systems. The success of these systems is due to the progress and evolution of computing power as well as the availability of international Data-Sets which made the processing of these data a fast and a cheap task.

Machine learning (ML) is a subfield of artificial intelligence (AI), it is science and art that studies how to develop algorithms that can learn from data. Machine learning and its applications are omnipresent in many areas of our daily lives.

In this document, I am interested in supervised learning problems, and more specifically deep learning.

Among the problems encountered when handling large amounts of data is the structuring and research, therefore the use of the classification can help reduce the size of these problems. Classification is a systematic arrangement in groups and categories based on its features. The classification of images allows to classify an image dataset into several classes such as for example (animal class, human class, transport class, etc.), this categorization allows a better exploitation of this dataset.

Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data. In this thesis, I explore the study of image classification using deep learning, focusing mainly on the construction of a convolutional neural network (CNN) model and its different layers, then applying this model on CIFAR-10 image dataset, after that I will try to perform several optimizations by varying the different parameters that constitute the model in order to understand the impact of each of them on the final result.

I have structured my manuscript in three main chapters:

- **The first chapter** begins with a definition of machine learning and its types. Then, it introduces the basics of image classification.
- **The second chapter** is devoted to the description of neural networks, its different types, the architecture of convolutional neural networks, as well as the different properties that characterize the training of the neural network.
- **The third and last chapter** shows the experimental part of my work, it contains the definition of the various tools and software used, the dataset, the architecture of the convolutional neural network model and then the results.

---

# Machine learning and Image classification

---

## 1.1 Introduction

This chapter spotlights and introduces the concepts and the basics of machine learning, I will talk briefly about the importance of Machine Learning (ML) then I will pass through its different types. At the second part of chapter I will highlight Image Classification which is the main subject of this research.

## 1.2 Machine learning

### 1.2.1 What is Machine learning?

**Machine Learning** is a sub field of artificial intelligence, it's a quite vast field that is expanding rapidly, being continually partitioned and sub-partitioned into different sub-specialties and types of machine learning .

There are some basic common threads, however, and the overarching theme is best summed up by this oft-quoted statement made by Arthur Samuel way back in 1959:

“Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.”

“And more recently, in 1997, Tom Mitchell gave a “well-posed ” definition that has proven more useful to engineering types”

“A computer program is said to learn from experience  $E$  with respect to some task  $T$  and some performance measure  $P$ , if its performance on  $T$ , as measured by  $P$ , improves with experience  $E$ .” [1]

### 1.2.2 The importance of machine learning

The machine learning field is constantly evolving. And along with evolution comes a rise in the demand and importance. There is one crucial reason why data scientists need machine learning, and that is: ‘High-value predictions that can guide better decisions and smart actions in real time without human intervention’ [2]

### 1.2.3 Types of machine learning

Machine Learning solves problems that cannot be solved by numerical means alone. Two of the most widely adopted machine learning methods are supervised ML and unsupervised ML, but there are also other methods of machine learning. Here's an overview of the most popular types (check figure 1.1).

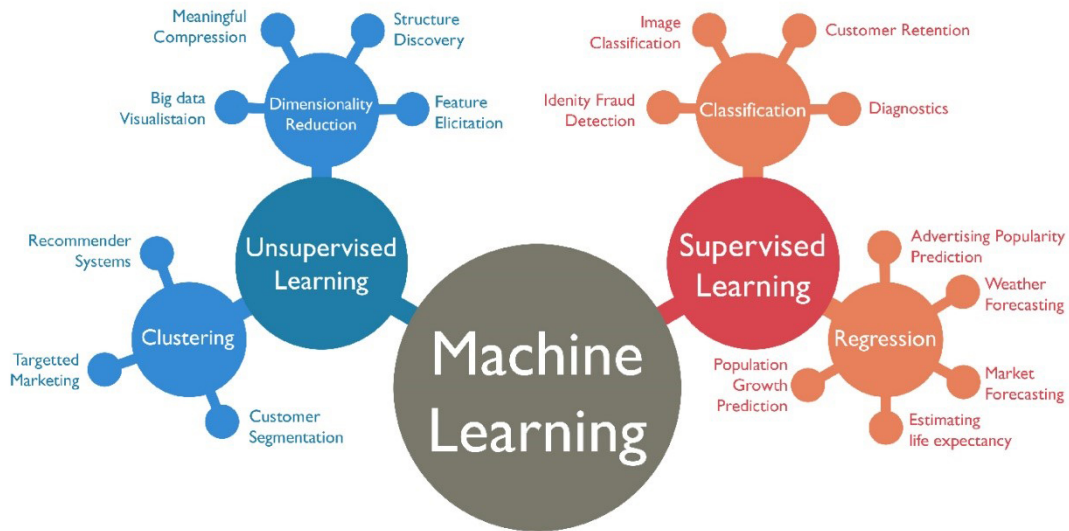


Figure 1.1: Machine Learning subcategories

#### 1.2.3.1 Supervised machine learning

These algorithms are trained using labeled examples. For example, a piece of equipment could have data points labeled either “F” (failed) or “R” (runs). The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with correct outputs to find errors. It then modifies the model accordingly. Through methods like classification, regression and prediction (check figure 1.2), supervised learning uses patterns to predict the values of the label on additional unlabeled data. Supervised learning is commonly used in applications where historical data predicts likely future events.

#### 1.2.3.2 Unsupervised machine learning

This type of machine learning is used against data that has no historical labels. The system is not told the "right answer". The algorithm must figure out what is being shown. The goal is to explore the data and find some structure within. Unsupervised learning works well on transitional data. Popular techniques include self-organizing maps, nearest-neighbor mapping, k-means clustering and singular value decomposition. These algorithms are also used to segment text topics, recommend items and identify data outliers [3].



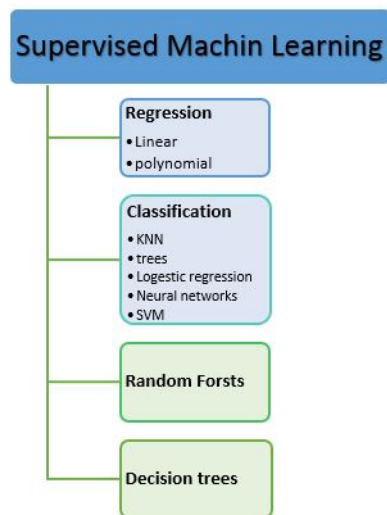


Figure 1.2: Algorithms of supervised machine learning

### 1.2.3.3 Semi-supervised machine learning

This subcategory is used for the same applications as supervised learning. But it uses both labeled and unlabeled data for training, typically a small amount of labeled data with a large amount of unlabeled data (because unlabeled data is less expensive and takes less effort to acquire). This type of learning can be used with methods such as classification, regression and prediction. Semi-supervised machine learning is useful when the cost associated with labeling is too high to allow for a fully labeled training process. Early examples of this include identifying a person's face on a webcam [3].

### 1.2.3.4 Reinforcement machine learning

It's often used for robotics, gaming and navigation. With reinforcement learning, the algorithm discovers through trial and error which actions yield the greatest rewards. This type of learning has three primary components: the agent (the learner or decision maker), the environment (everything the agent interacts with) and actions (what the agent can do). The objective is for the agent to choose actions that maximize the expected reward over a given amount of time. The agent will reach the goal much faster by following a good policy. So, the goal in reinforcement machine learning is to learn the best policy. [3]

In this document I will focus on the supervised machine learning precisely, the classification sub-category, and more precisely on the image classification sub-sub-category.

## 1.3 Image Classification

### 1.3.1 Classification Definition

Classifying an image is a task or a series of methods that a unified theory to use the images for further analysis or for mapping, it is often important to translate the frequency information contained in the images into a thematic information. There is usually a choice between two approaches: supervised and unsupervised classification.

### 1.3.2 Image classification motivations

Image classification consists in distributing images according to previously established classes, classifying an image makes it correspond a class, and marking its relationship with other images.

Generally, recognizing an image is an easy task for a human during his lifetime, he has acquired knowledge that allows him to adapt to variations resulting from different conditions of acquisition. It is for example relatively simple for him to recognize an object in several orientations partially hidden by another from near or far and according to various illuminations [4].

However, technological evolution in terms of image acquisition (cameras, sensors, microscopes) and storage are generating rich databases of information and multiplying the domains of applications, it becomes difficult for the human to analyze the large number of images. However, this is not necessarily easy for a computer program for which an image is a set of numerical value.

The goal of image classification is to develop a system that can assign a class automatically to an image. Thus, this system makes it possible to carry out an expertise task that can be costly to acquire for a human being, particularly because of physical constraints such as concentration, fatigue or the time required by a large volume of image data.

The applications of automatic image classification are numerous and range from document analysis to medicine to the military field. Thus, I find applications in the medical field such as the recognition of cells and tumors, handwriting recognition for checks postal codes. In the field of biometrics such as face recognition, fingerprints, irises.[4]

The common point for all these applications is that they require the establishment of a processing chain from the available images composed of several steps to attend to a decision. Each step of the implementation of such a classification system requires the search for appropriate methods for optimal overall performance, namely the feature extraction phase and the learning phase. Typically, I have image data from which I need to extract relevant information translated in the form of digital vectors. This extraction phase allows us to work in a digital space. It is then necessary to elaborate in the learning phase, from these initial data, a decision function for deciding the membership of a new datum to one of the classes in the presence.[4]

### 1.3.3 Types of classification algorithms

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio image classification, document classification etc.

There is a lot of classification algorithms available now but it is not possible to conclude which one is superior to other. It depends on the application and nature of available data set.

#### 1.3.3.1 $k$ -Nearest Neighbor

The  $k$ -nearest neighbor algorithm is a supervised classification algorithm. It takes a bunch of labeled points and uses them to learn how to label other points. To label a new point, it looks at the labeled points closest to that new point which are its nearest neighbors, and has those neighbors vote. So whichever label, the most of the neighbors have is the label for the new point. Here  $k$  in  $K$ -Nearest Neighbors is the number of neighbors it checks. It is supervised because the objective is to try to classify a point based on the known classification of other points. Usually KNN is robust to noisy data since it is averaging the  $k$ -nearest neighbors.(check figure 1.3)

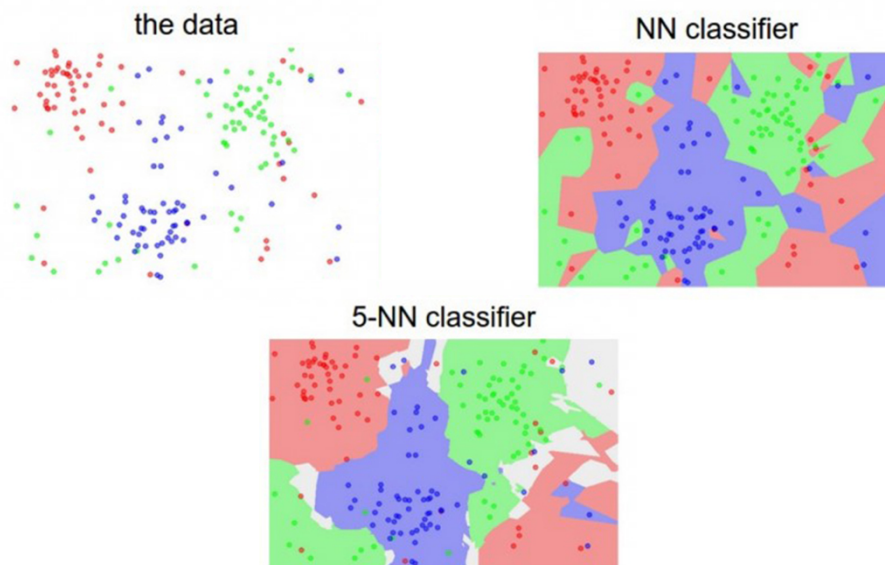


Figure 1.3:  $KNN$  algorithm's application example

In the Figure 1.3 the picture on the left represents points in a 2D plane with three possible types of labeling (red, green, blue). For the 5-NN classifier, the boundaries between each region are quite smooth and regular. As for the N-NN Classifier, I note that the limits are "chaotic" and irregular. The latter comes from the fact that the algorithm tries to get all the blue dots in the blue regions, the red with the red ...it is a case of overfitting. [5]

### 1.3.3.2 *K*-means

The *k*-means algorithm is the best-known unsupervised clustering algorithm because of its simplicity of implementation. It partitions the data of an image into *K* clusters. Unlike other so-called hierarchical methods, which create a "cluster tree" structure to describe groupings, *k*-means creates only one level of clusters. The algorithm returns a partition of the data, in which the objects within each cluster are as close as possible to one another and as far as possible from the objects of the other clusters. Each cluster of the partition is defined by its objects and its centroid. The *k*-means is an iterative algorithm that minimizes the sum of the distances between each object and the centroid of its cluster.[6] The initial position of the centroids determines the final result, so that the centroids must be initially placed as far as possible from each other in order to optimize the algorithm. *K*-means changes cluster objects until the sum can no longer decrease. The result is a set of compact and clearly separated clusters, provided that the correct *K* value of the number of clusters is chosen. The main steps of the *k*-means algorithm are:

1. Random choice of the initial position of *K* clusters.
2. Re-Assign objects to a cluster according to a criterion of distance minimization (usually according to a measure of Euclidean distance).
3. Once all the objects have been placed, recalculate the *K* centroids.
4. Repeat steps 2 and 3 until no more reassignments are made.

*K*-means is a simple algorithm that has been adapted to many problem domains.[7]

### 1.3.3.3 Fuzzy *c*-means

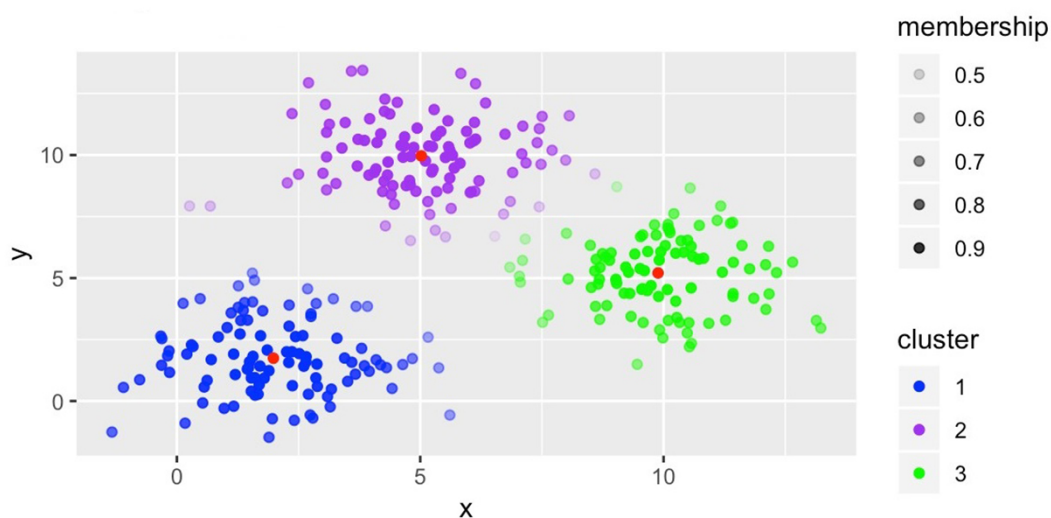


Figure 1.4: Fuzzy *c*-means algorithm's example

Fuzzy C-Means (FCM) is a fuzzy unsupervised classification algorithm. Coming from the C-means algorithm, he introduces the notion of a fuzzy set into the definition of classes: each point in the dataset belongs to each cluster with a certain degree, and all clusters are characterized by their centroid (check figure 1.4). Like other unsupervised classification algorithms, it uses a criterion of minimizing intra-class distances and maximizing interclass distances, but giving a certain degree of membership to each class for each pixel. This algorithm requires prior knowledge of the number of clusters and generates the classes by an iterative process by minimizing an objective function. Thus, it makes it possible to obtain a fuzzy partition of the image by giving each pixel a membership degree (between 0 and 1) to a given class. The cluster with a pixel is associated is the one with the highest degree of membership [8] .

The main steps of the Fuzzy C-means algorithm are:

1. The arbitrary fixation of a membership matrix.
2. The computation of class centroids.
3. The readjustment of the membership matrix according to the position of the centroids.
4. Calculation of the minimization criterion and return to Step 2 if there is no convergence of criteria.

#### 1.3.3.4 Support Vector Machine

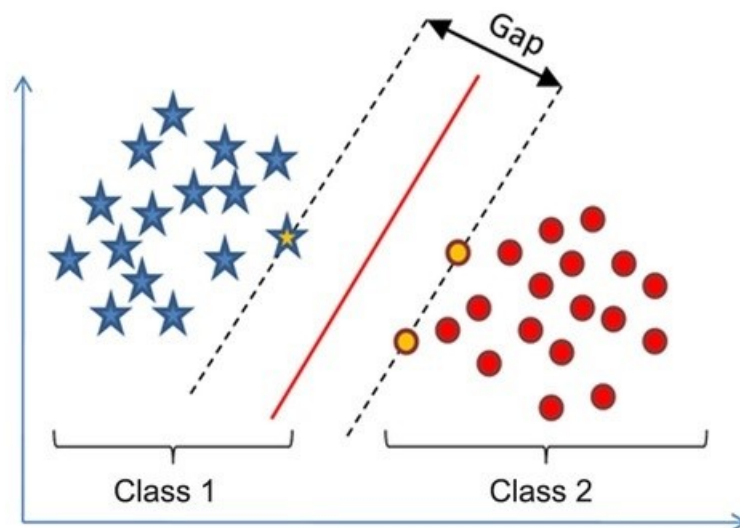


Figure 1.5: SVM's application example

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N the number of features) that distinctly classifies the data points.

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin (check figure 1.5), i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. [9]

### 1.3.3.5 Decision Trees

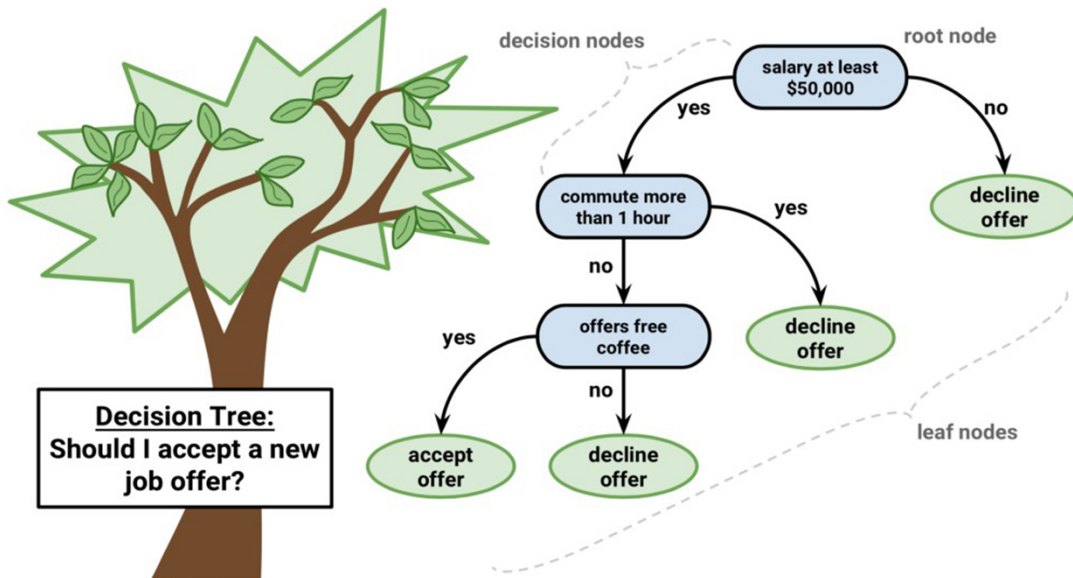


Figure 1.6: Decision trees application example

The tree is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. (check figure 1.6)

A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers. An over-fitted model has a very poor performance on the unseen data even though it gives an impressive performance on training data. This can be avoided by pre-pruning which halts tree construction early or post-pruning which removes branches from the fully-grown tree. [10]

## 1.4 Image classification and Machine learning

Manual methods have proven very difficult to apply for seemingly simple tasks such as image classification, object recognition in images, or voice recognition. The data coming from the real-world samples of a sound or the pixels of an image are complex, variable and tainted with noise. For a machine, an image is an array of numbers indicating the brightness (or color) of each pixel, and a sound signal a sequence of numbers indicating the air pressure at each moment. It is virtually impossible to write a program that will work robustly in all situations. This is where machine learning comes in. It is the learning that drives the systems of all major Internet companies. [11]

They have been using it for a long time to filter desirable content, order responses to a search, make recommendations, or select information of interest to each user. A drivable system can be seen as a black box with an entry, for example an image, sound, or text, and an output that can represent the category of the object in the image, the spoken word, or the subject of which the text speaks. This is called classification systems or pattern recognition. In its most used form, the machine learning is supervised: at the entrance of the machine is shown a picture of an object, for example a car, and it is given the desired exit for a car. Then I show him the picture of a dog with the desired exit for a dog. After each example, the machine adjusts its internal parameters to bring its output closer to the desired output. After showing the machine thousands or millions of examples labeled with their category, the machine becomes able to classify most of them correctly. But what is more interesting is that it can also properly classify car or dog pictures that it has never seen during the learning phase. This is called the generalization capacity. [12]

## 1.5 Conclusion

In this chapter I have presented the common types of machine learning, also I have summarized its the types which are Supervised, Unsupervised, Semi-supervised and Reinforcement machine learning. I have parsed also some Image Classification's algorithms. Finally I set the relation between Machine Learning and Image Classification.

---

# Deep learning and Convolutional Neural Network

---

## 2.1 Introduction

This chapter introduces the concept of Deep Learning (DL), I will talk about the the original idea of its concept which is the Biological Neural Networks, then I will present its projection in Deep Learning which are Artificial Neural Networks and its types. Then, I will focus more on the Convolutional Neural Network and talk about its layers and architectures and how I can train the network and how I can transfer the learning from a network to another.

## 2.2 Definition of deep learning

Deep learning is a sub-field of machine learning dealing with algorithms inspired by the structure and function of the brain called artificial neural networks (check figure 2.1 ). In other words, it mirrors the functioning of our brains. Deep learning algorithms are similar to how nervous system structured where each neuron connected each other and passing information.

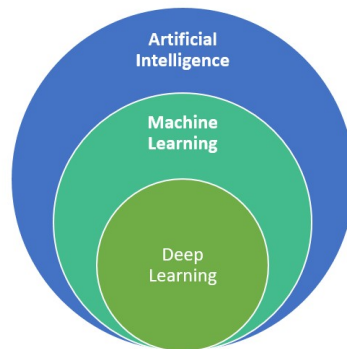


Figure 2.1: The relation between AI, machine learning and deep learning



## 2.3 Biological neural network

### 2.3.1 Neurons

The neuron is not only the basic building block of intelligence but it also forms a message passing and control system within multi-cellular organisms. The basic structure of a neuron can be seen as a body, called the soma, with one or many processes, dendrites or axons, branching out from it. The processes carry messages to and from the soma and terminate in synapses (check figure 2.2).

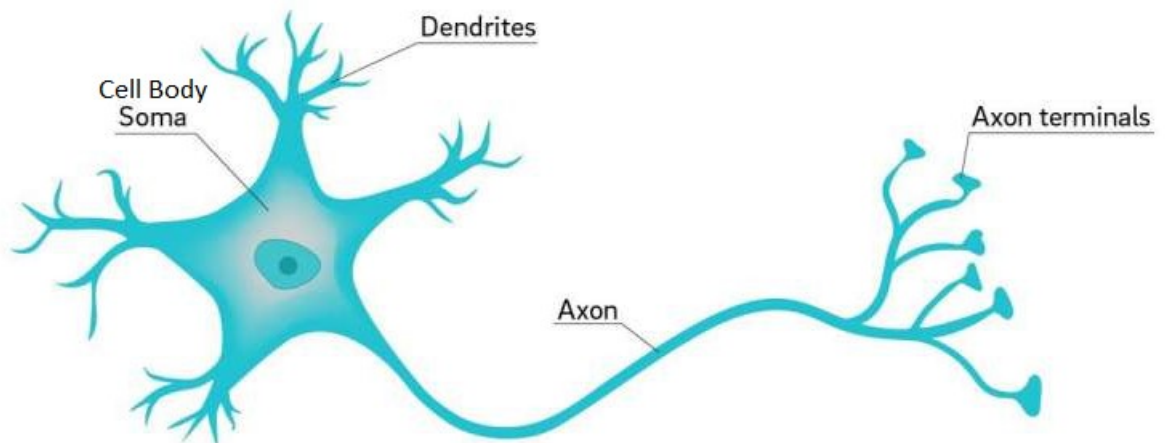


Figure 2.2: Typical biological neuron

The neuron is only one cell, much like any other in the body. It has DNA code and is generated in much the same way most cells are generated; One major difference between neurons and most other cells in the human body is that neurons don't regenerate. [13]

### 2.3.2 Axons

Axon, also called nerve fiber, portion of a nerve cell (neuron) that carries nerve impulses away from the cell body. A neuron typically has one axon that connects it with other neurons or with muscle or gland cells. Some axons may be quite long, reaching, for example, from the spinal cord down to a toe. Most axons of vertebrates are enclosed in a myelin sheath, which increases the speed of impulse transmission; some large axons may transmit impulses at speeds up to 90 meters per second. [14]

### 2.3.3 Dendrites

Dendrites are really quite similar to axons. In vertebrates they have the function of receiving signals from other axons or sensory organs. They are generally unidirectional, transmitting to the soma, but they can also transmit signals as well.

### 2.3.4 Synapses

Synapses are the points where neurons communicate with each other. There are two main types of synapse, chemical and electrical.

- **Chemical Synapses:** commonly link axons to dendrites. They are asymmetric and there is a small gap of approximately 200- to 300- Å between them.
- **Electrical Synapses:** in contrast to chemical synapses, are symmetric. The gap between the cells is much smaller than the gap in chemical synapses. Ions can pass through channels directly from one cell to the other. This makes for much faster inter-cellular signaling. [15]

### 2.3.5 Soma (Cell Body)

Positive signals pass from the dendrites (and axons) to the soma where they increase the neuron's potential. If enough signals reach the soma to excite it (reach its action potential), then it will fire the whole neuron. The rate of signals arriving at the soma is important as the potential is constantly declining. The Soma also has many cell support functions.

## 2.4 Artificial Networks (ANNs)

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites. ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value [16].

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values. The Figure 2.3 . shows a simple ANN

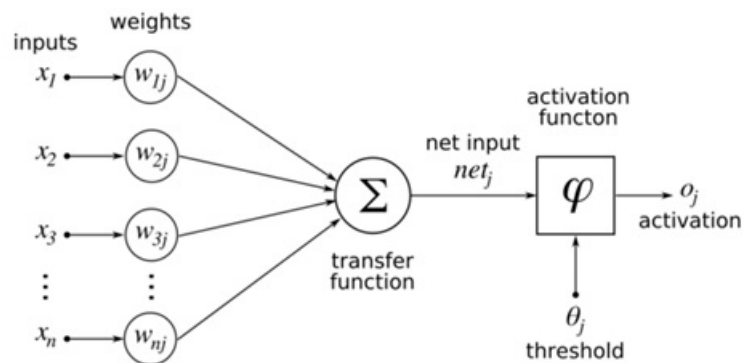


Figure 2.3: A simple artificial neuron network

## 2.5 Different types of Neural Networks

### 2.5.1 Feedforward Neural Network

This neural network is one of the simplest form of ANN, where the data or the input travels in one direction. The data passes through the input nodes and exit on the output nodes. This neural network may or may not have the hidden layers. In simple words, it has a front propagated wave and no back propagation by using a classifying activation function usually (check figure 2.4).

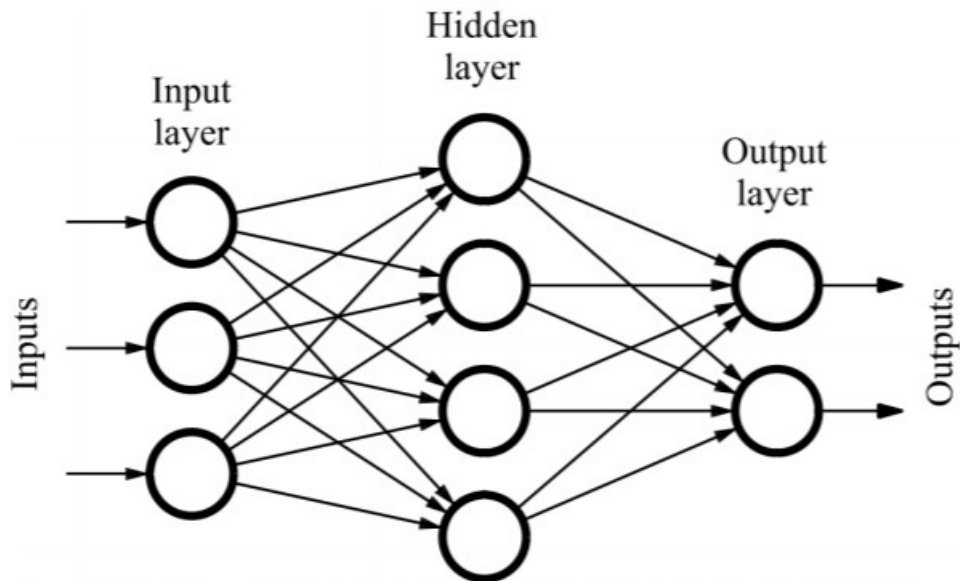


Figure 2.4: Feedforward neuron network

Application of Feed forward neural networks are found in computer vision and speech recognition where classifying the target classes are complicated. This kind of Neural Networks are responsive to noisy data and easy to maintain. [17]

### 2.5.2 Radial basis function Neural Network

The idea of RBFNs is derived from the theory of function approximation. The Euclidean distance is computed from the point being evaluated to the center of each neuron, and a radial basis function (RBF) (also called a kernel function) is applied to the distance to compute the weight (influence) for each neuron. The radial basis function is so named because the radius distance is the argument to the function. In other words, RBFs represent local receptors; its output depends on the distance of the input from a given stored vector. That means, if the distance from the input vector  $\vec{x}$  to the center  $\vec{u}_j$  of each RBF  $\varphi_j$  for example,  $\|\vec{x} - \vec{\mu}_j\|$  is equal to 0 then the contribution of this point is 1, whereas the contribution tends to 0 if the distance  $\|\vec{x} - \vec{\mu}_j\|$  increases [18]. As an example the figure 2.5 illustrate a typical architecture of a radial basis function neural network .

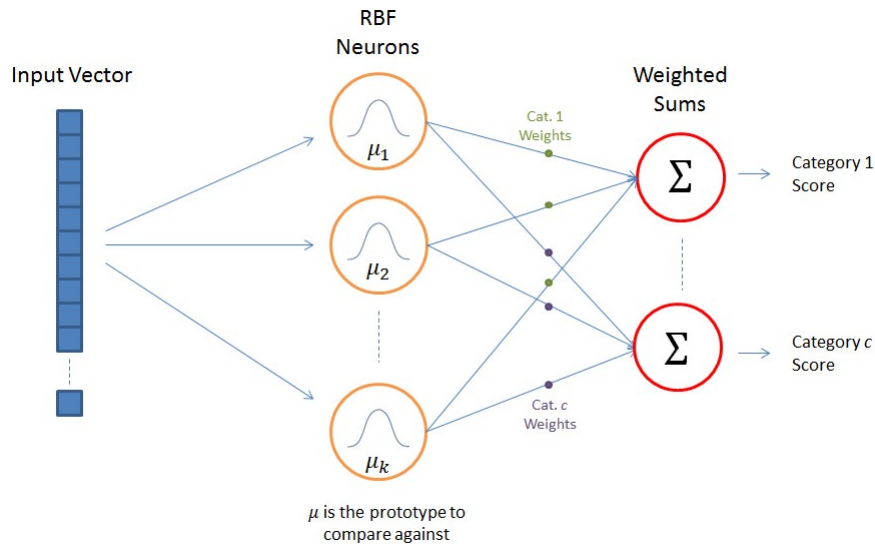


Figure 2.5: Architecture of an RBF Network

### 2.5.3 Kohonen Self Organizing Neural Network

Kohonen Self-Organizing feature Map (SOM) is a neural network which modifies itself in response to input patterns. This property is called self-organization and it is achieved using competitive learning. The basic competitive learning means that a competition process takes place before each cycle of learning. In the competition process a winning processing element is chosen by some criteria. Usually this criteria is to minimize an Euclidean distance between the input vector and the weight vector. After the winning processing element is chosen, its weight vector is adapted according to the learning law used. SOM differs from the basic competitive learning so that instead of adapting only the winning processing element also the neighbors of the winning processing element are adapted. The self-organization property of SOM is based on the use of the neighborhood of the winning processing element [19]. The figure 2.6 shows an example of Kohonn SOM.

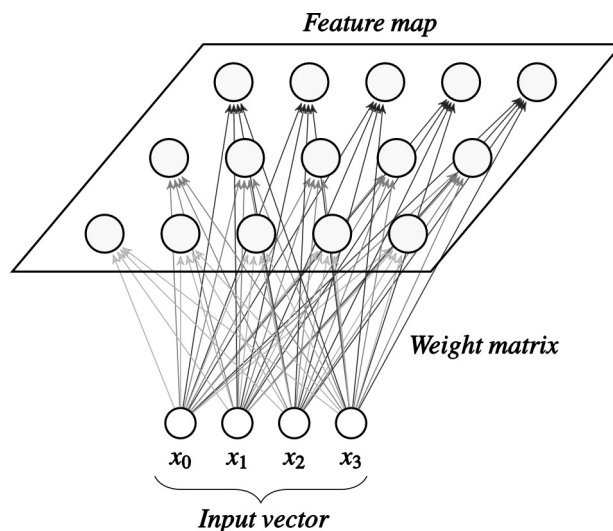


Figure 2.6: Architecture of a SOM network

## 2.5.4 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence (check figure 2.7).

RNN have a “memory” which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks. [20]

RNNs can be used in a lot of different places. For example, at language modelling and generating text, machine translation, speech recognition, generating image descriptions and video tagging.

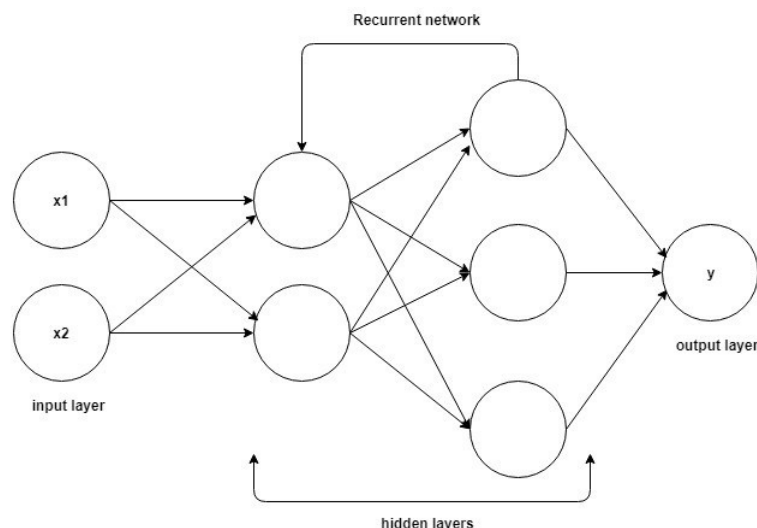


Figure 2.7: Architecture of a Recurrent Neural Network

## 2.5.5 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of specialized neural network for processing data with a grid-like topology. image type data, which can be considered as a 2D grid of pixels. Convolutional networks have had considerable success in practical applications. The name "convolutional neural network" indicates that the network uses a mathematical operation called convolution. Convolution is a special linear operation. Convolutional networks are simply networks of neurons that use convolution instead of matrix multiplication in at least one of their layers. They have wide applications in image and video recognition, recommendation systems and natural language processing.

## 2.6 Convolutional Neural Network Layers

### 2.6.1 Convolutional Layer

The convolution layer is the core building block of the CNN. It carries the main portion of the network’s computational load. This layer performs a dot product between two matrices (check figure 2.8), where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image, but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels. During the forward pass, the kernel slides across the height and width of the image producing the image representation of that receptive region. This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride. [21]

If I have an input of size  $W \times W \times D$  and  $D_{out}$  number of kernels with a spatial size of  $F$  with stride  $S$  and amount of padding  $P$ , then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1 \quad (2.1)$$

This will yield an output volume of size  $W_{out} \times W_{out} \times D_{out}$

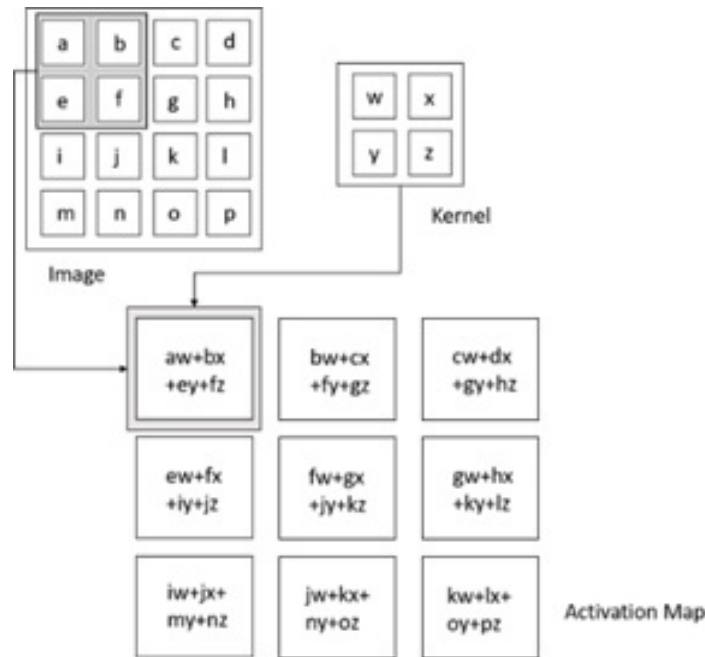


Figure 2.8: Convolution Operation

### 2.6.2 Pooling Layer

The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the

representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

There are several pooling functions such as the average of the rectangular neighborhood, L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel. However, the most popular process is max pooling, which reports the maximum output from the neighborhood.

The figure 2.9 below shows the most common type of pooling the max-pooling layer, which slides a window, like a normal convolution, and get the biggest value on the window as the output. [22]

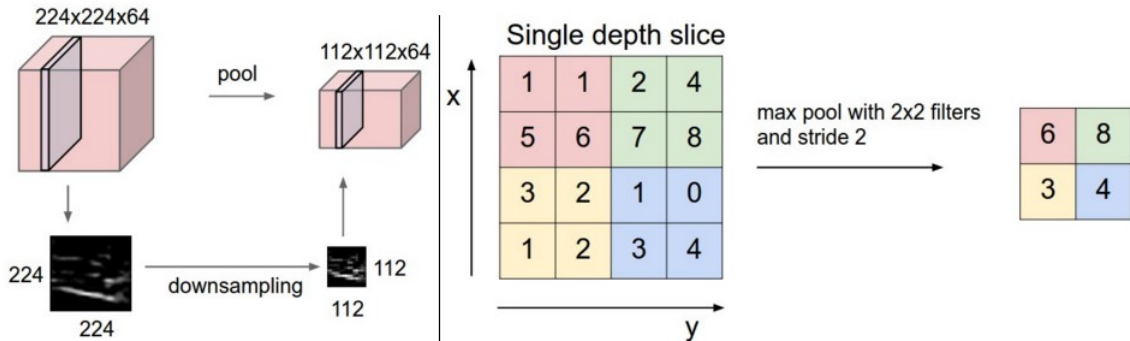


Figure 2.9: Pooling Layer

### 2.6.3 Fully Connected Layer

The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

### 2.6.4 Dropout Layer

Dropout is a form of regularization that randomly drops some proportion of the nodes that feed into a fully connected layer (check figure 2.10). Here, dropping a node means that its contribution to the corresponding activation function is set to 0. Since there is no activation contribution, the gradients for dropped nodes drop to zero as well. [23]

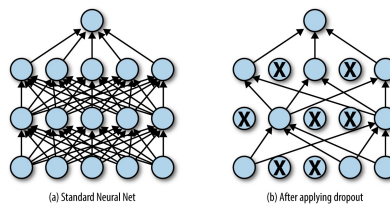


Figure 2.10: Dropout process

## 2.7 Convolutional Neural Network Architectures

Discussing the commonly used architectures for convolutional networks, almost all CNN architectures follow the same general design principles of successively applying convolutional layers to the input, periodically down sampling the spatial dimensions while increasing the number of feature maps.

While the classic network architectures were comprised simply of stacked convolutional layers, modern architectures explore new and innovative ways for constructing convolutional layers in a way which allows for more efficient learning. Almost all of these architectures are based on a repeatable unit which is used throughout the network.

These architectures serve as general design guidelines which machine learning practitioners will then adapt to solve various computer vision tasks. These architectures serve as rich feature extractors which can be used for image classification, object detection, image segmentation, and many other more advanced tasks. I'll talk briefly about some of these architectures later. The most common form of a CNN architecture stacks a few CONV-RELU layers, follows them with POOL layers, and repeats this pattern until the image has been merged spatially to a small size. At some point, it is common to transition to fully-connected layers. The last fully-connected layer holds the output, such as the class scores.

### 2.7.1 LeNet-5 - LeCun & al

LeNet-5, a 7-layer Convolutional Neural Network, was deployed in many banking systems to recognize hand-written numbers on cheques. [12]

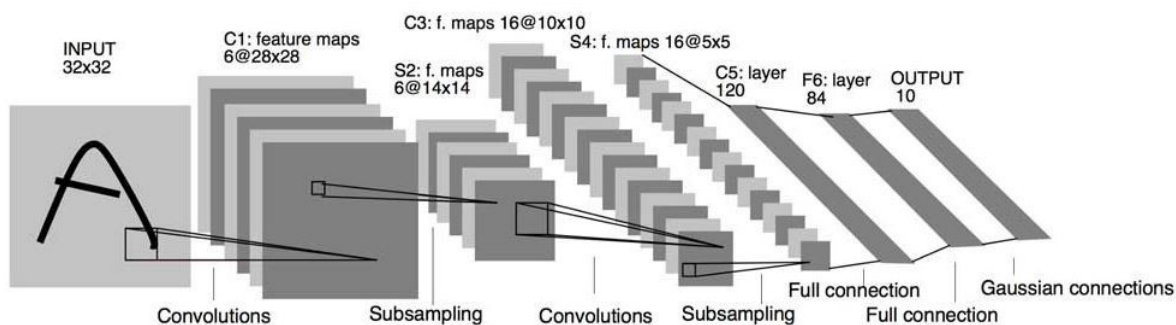


Figure 2.11: LeNet-5's Architecture

The hand-written numbers were digitized into grayscale images of pixel size— $32 \times 32$ . At that time, the computational capacity was limited and hence the technique wasn't scalable to large scale images. The model contained 7 layers excluding the input layer (check figure 2.11). Since it is a relatively small architecture, here are its layers:

1. **Layer 1:** A convolutional layer with kernel size of  $5 \times 5$ , stride of  $1 \times 1$  and 6 kernels in total. So the input image of size  $32 \times 32 \times 1$  gives an output of  $28 \times 28 \times 6$ . Total params in *layer* =  $5 \times 5 \times 6 + 6$  (bias terms)



2. **Layer 2:** A pooling layer with 22 kernel size, stride of  $2 \times 2$  and 6 kernels in total. The input values in the receptive were summed up and then were multiplied to a trainable parameter (1 per filter), the result was finally added to a trainable bias (1 per filter). Finally, sigmoid activation was applied to the output. So, the input from previous layer of size  $28 \times 28 \times 6$  gets sub-sampled to  $14 \times 14 \times 6$ . Total params in *layer* =  $[1(\text{trainableparameter}) + 1(\text{trainablebias})] \times 6 = 12$
3. **Layer 3:** Similar to Layer 1, this layer is a convolutional layer with same configuration except it has 16 filters instead of 6. So the input from previous layer of size  $14 \times 14 \times 6$  gives an output of  $10 \times 10 \times 16$ . Total params in *layer* =  $5 \times 5 \times 16 + 16 = 416$ .
4. **Layer 4:** Again, similar to Layer 2, this layer is a pooling layer with 16 filters this time around. Remember, the outputs are passed through sigmoid activation function. The input of size  $10 \times 10 \times 16$  from previous layer gets sub-sampled to  $5 \times 5 \times 16$ . Total params in *layer* =  $(1 + 1) \times 16 = 32$
5. **Layer 5:** This time around I have a convolutional layer with 55 kernel size and 120 filters. There is no need to even consider strides as the input size is  $5 \times 5 \times 16$  so I will get an output of  $1 \times 1 \times 120$ . Total params in *layer* =  $5 \times 5 \times 120 = 3000$
6. **Layer 6:** This is a dense layer with 84 parameters. So, the input of 120 units is converted to 84 units. Total *params* =  $84 \times 120 + 84 = 10164$ . The activation function used here was rather a unique one. I'll say you can just try out any
7. **Output Layer:** Finally, a dense layer with 10 units is used. Total *params* =  $84 \times 10 + 10 = 924$ .

## 2.7.2 AlexNet

AlexNet was developed by Alex Krizhevsky et al. in 2012 to compete in the ImageNet competition. The general architecture is quite similar to LeNet-5, although this model is considerably larger (check figure 2.12). The success of this model (which took first place in the 2012 **ImageNet competition**) convinced a lot of the computer vision community to take a serious look at deep learning for computer vision tasks. [24]

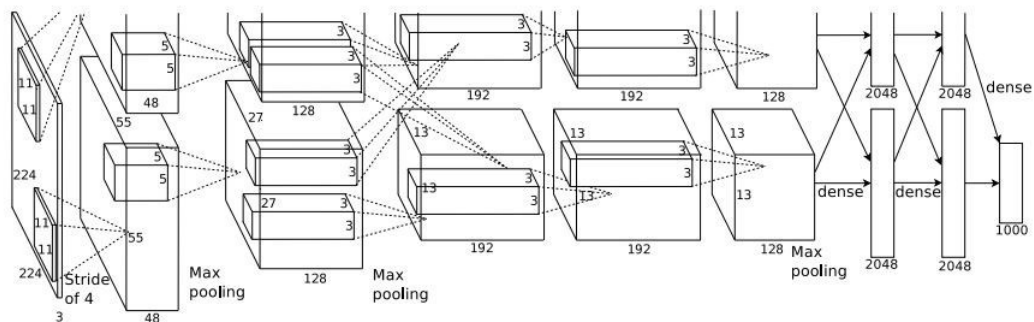


Figure 2.12: AlexNet's Architecture

### 2.7.3 VGG-16

The VGG network, introduced in 2014, offers a deeper yet simpler variant of the convolutional structures discussed above. At the time of its introduction, this model was considered to be very deep (check figure 2.13). [25]

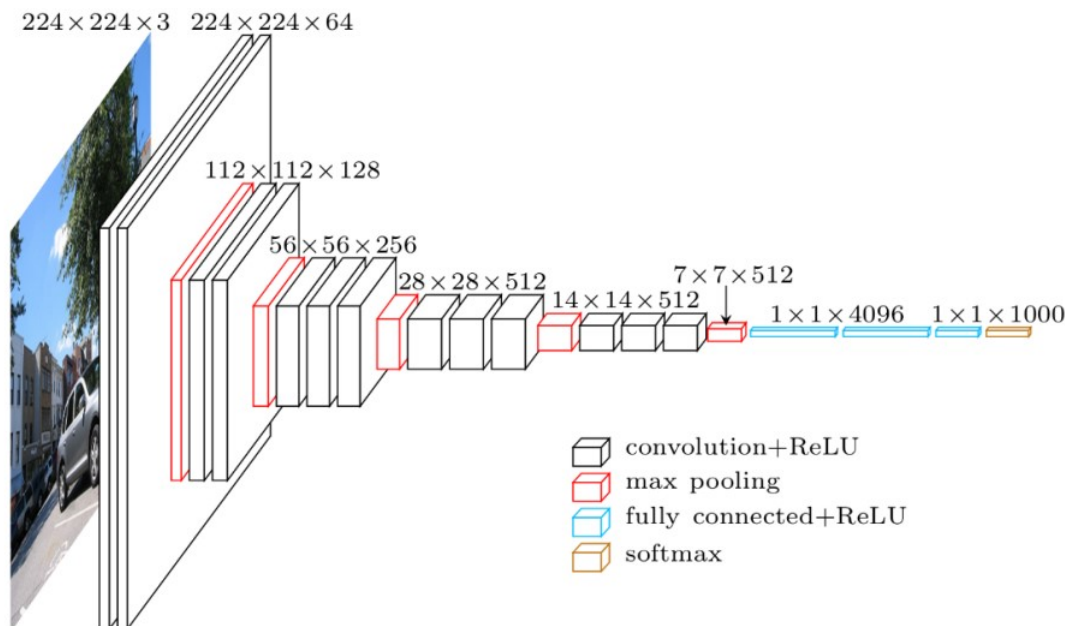


Figure 2.13: A figure that represents the architecture of VGG-16 CNN

### 2.7.4 GoogLeNet (Inception)

In 2014, researchers at Google introduced the Inception network which took first place in the ImageNet competition for classification and detection challenges.

The model is comprised of a basic unit referred to as an "Inception cell" in which I perform a series of convolutions at different scales and subsequently aggregate the results. In order to save computation,  $1 \times 1$  convolutions are used to reduce the input channel depth. For each cell, I learn a set of  $1 \times 1$ ,  $3 \times 3$ , and  $5 \times 5$  filters which can learn to extract features at different scales from the input. Max pooling is also used, albeit with "same" padding to preserve the dimensions so that the output can be properly concatenated (check figure 2.14). [26]

Reasons for using these inception modules:

1. Each layer type extracts different information from input. Information gathered from a  $3 \times 3$  layer will differ from information gathered from a  $5 \times 5$  layer. How do I know which transformation will be the best at a given layer? So, I use them all!
2. Dimensionality reduction using  $1 \times 1$  convolutions! Consider a  $128 \times 128 \times 256$  input. If I pass it through 20 filters of size  $1 \times 1$ , I will get an output of  $128 \times 128 \times 20$ . So,

I apply them before the  $3 \times 3$  or  $5 \times 5$  convolutions to decrease the number of input filters to these layers in the inception block used for dimensionality reduction.

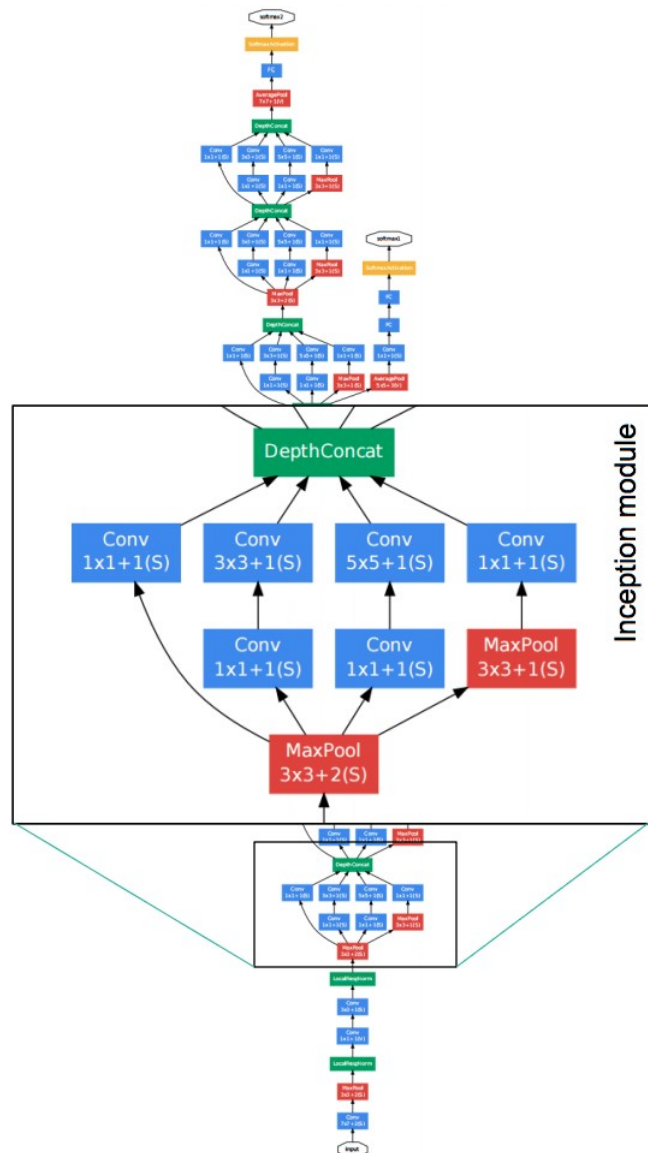


Figure 2.14: GoogLeNet's architecture

### 2.7.5 ResNet-Kaiming He & al

The 2015 **ImageNet** competition brought about a top-5 error rate of 3.57% , which is lower than the human error on top-5. This was due to ResNet (Residual Network) model

used by microsoft at the competition. The network introduced a novel approach called "skip connections".

The idea came out as a solution to an observation—Deep neural networks perform worse as I keep on adding layer. But intuitively speaking, this should not be the case. If a network with  $k$  layers performs as  $y$ , then a network with  $k + 1$  layers should at least perform  $y$ .

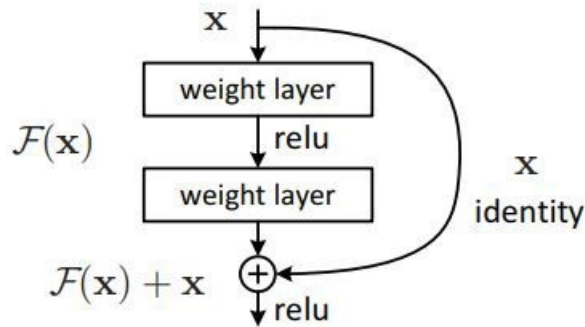


Figure 2.15: Residual Network's architecture

The observation brought about a hypothesis: direct mappings are hard to learn. So instead of learning mapping between output of layer and its input, learn the difference between them—learn the residual.

Say,  $x$  was the input and  $H(x)$  was the learned output. So, I need to learn  $F(x) = H(x) - x$ . I can do this by first making a layer to learn  $F(x)$  and then adding  $x$  to  $F(x)$  hence achieving  $H(x)$ . As a result, I am sending the same  $H(x)$  in next layer. This gave rise to the residual block I saw above.

The results were amazing as the vanishing gradients problem which usually make deep neural networks numb to learning were removed. The skip connections or the shortcuts, as I might say them, gave a shortcut to the gradients to the previous layers, skipping bunch of layers in between. [27]

## 2.8 Training of an Artificial Neural Network

Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins.

### 2.8.1 Image Preprocessing

There are a number of pre-processing steps I might wish to carry out before using this in any Deep Learning project. (I'll be sitting below a list some of the most common steps).

- **Uniform aspect ratio:** One of the first steps is to ensure that the images have the same size and aspect ratio. Most of the NN models assume a square shape input

image, which means that each image needs to be checked if it is a square or not, and cropped appropriately. Cropping can be done to select a square part of the image, as shown. While cropping, I usually care about the part in the center. [28]

- **Image Scaling:** Once that all images are square, it's time to scale each image appropriately. I have decided to have images with width and height of 100 pixels. Starting to scale the width and height of each image by a factor  $x$ . There are a wide variety of up-scaling and down-scaling techniques, usually a library function is used to do this. [28]
- **Dimensional reduction:** by collapsing the RGB channels into a single gray-scale channel. There are often considerations to reduce other dimensions, when the neural network performance is allowed to be invariant to that dimension, or to make the training problem more tractable. [28]

## 2.8.2 Loss Functions

### 2.8.2.1 Cross Entropy

The Cross-Entropy Loss (CE) is defined as:

$$CE = - \sum_i^C t_i \log(s_i) \quad (2.2)$$

Where  $t_i$  and  $s_i$  are the groundtruth and the CNN score for each class  $i$  in  $C$ . As usually an activation function (Sigmoid / Softmax) is applied to the scores before the  $CE$  Loss computation, we write  $f(s_i)$  to refer to the activations. [29]

### 2.8.2.2 Binary Cross Entropy

Binary cross entropy is a loss function used on problems involving yes/no (binary) decisions. For instance, in multi-label problems, where an example can belong to multiple classes at the same time, the model tries to decide for each class whether the example belongs to that class or not.

$$L(y, \hat{y}) = - \frac{1}{N} \sum_{i=0}^N (y \times \log(\hat{y}_i) + (1 - y) \times \log(1 - \hat{y}_i)) \quad (2.3)$$

Binary cross entropy measures how far away from the true value (which is either 0 or 1) the prediction is for each of the classes and then averages these class-wise errors to obtain the final loss. The block before must have a Sigmoid as activation function. [29]

### 2.8.2.3 Categorical Cross Entropy

Categorical cross entropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only.

$$L(y, \hat{y}) = - \sum_{j=0}^M \sum_{i=0}^N (y_{ij} * \log(\hat{y}_{ij})) \quad (2.4)$$

Categorical cross entropy will compare the distribution of the predictions (the activations in the output layer, one for each class) with the true distribution, where the probability of the true class is set to 1 and 0 for the other classes. To put it in a different way, the true class is represented as a one-hot encoded vector, and the closer the model's outputs are to that vector, the lower the loss. [29]

#### 2.8.2.4 Mean Squared Error (MSE)

Mean Squared Error (MSE), or quadratic, loss function is widely used in linear regression as the performance measure, and the method of minimizing MSE is called Ordinary Least Squares (OSL), the basic principle of OSL is that the optimized fitting line should be a line which minimizes the sum of distance of each point to the regression line, i.e., minimizes the quadratic sum. The standard form of MSE loss function is defined as

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2 \quad (2.5)$$

Where  $(y^{(i)} - \hat{y}^{(i)})$  is named as residual, and the target of MSE loss function is to minimize the residual sum of squares. However, if using Sigmoid as the activation function, the quadratic loss function would suffer the problem of slow convergence (learning speed), for other activation functions, it would not have such problem.[30]

#### 2.8.2.5 Mean Squared Logarithmic Error (MSLE)

Mean Squared Logarithmic Error (MSLE) loss function is a variant of MSE, which is defined as

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n \left( \log(y^{(i)} + 1) - \log(\hat{y}^{(i)} + 1) \right)^2 \quad (2.6)$$

MSLE is also used to measure the difference between actual and predicted. By taking the log of the predictions and actual values, what changes is the variance that you are measuring. It is usually used when you do not want to penalize huge differences in the predicted and the actual values when both predicted and true values are huge numbers. Another thing is that MSLE penalizes under-estimates more than over-estimates. [30]

1. If both predicted and actual values are small: MSE and MSLE is same.
2. If either predicted or the actual value is big: MSE > MSLE.
3. If both predicted and actual values are big: MSE > MSLE (MSLE becomes almost negligible).

#### 2.8.2.6 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is a quantity used to measure how close forecasts or predictions are to the eventual outcomes, which is computed by

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n |y^{(i)} - \hat{y}^{(i)}| \quad (2.7)$$

where  $|\cdot|$  denotes the absolute value. Albeit, both MSE and MAE are used in predictive modeling, there are several differences between them. MSE has nice mathematical properties which makes it easier to compute the gradient. However, MAE requires more complicated tools such as linear programming to compute the gradient. Because of the square, large errors have relatively greater influence on MSE than do the smaller error. Therefore, MAE is more robust to outliers since it does not make use of square. On the other hand, MSE is more useful if concerning about large errors whose consequences are much bigger than equivalent smaller ones. MSE also corresponds to maximizing the likelihood of Gaussian random variables. [30]

### 2.8.3 Optimizers

Optimizers help us to minimize (or maximize) an Objective function (another name for Error function)  $E(x)$  which is simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target *values*( $Y$ ) from the set of *predictors*( $X$ ) used in the model.

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. I will talk about the variants of Gradient Decent

#### 2.8.3.1 Stochastic Gradient Decent

SGD (stochastic gradient descent) updates the parameters for each sample in the dataset  $x^{(i)}$  and label  $y^{(i)}$

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i)}; y^{(i)}) \quad (2.8)$$

In Gradient Descent optimization, I compute the cost gradient based on the complete training set; hence, I sometimes also call it batch gradient descent. In case of very large datasets, using Gradient Descent can be quite costly since I are only taking a single step for one pass over the training set – thus, the larger the training set, the slower our algorithm updates the weights and the longer it may take until it converges to the global cost minimum. This method is faster but the updates of too frequent parameters cause the objective function of the swings, these swings on one hand allow to land in potentially better local minima but on the other hand make the convergence more difficult. However, it has been shown that by lowering the learning rate , SGD shows the same convergence as the gradient gradient descent.[31] [32]

**Momentum** Momentum is a method that helps accelerate the optimizer in the relevant direction and dampens oscillations. The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, I gain faster convergence and reduced oscillation.

The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way. If I don't use momentum the ball gets no information on where it was before each discrete calculation step. Without momentum, each new calculation will only be based on the gradient, no history. [33]

In this way, momentum helps the optimizer not to get stuck in local minima.

### 2.8.3.2 Adagrad

Adagrad adapts the learning rate specifically to individual features: that means that some of the weights in the chosen dataset will have different learning rates than others. This works really well for sparse datasets where a lot of input examples are missing. Adagrad has a major issue though: the adaptive learning rate tends to get really small over time. Some other optimizers below seek to eliminate this problem. [34]

### 2.8.3.3 RMSprop

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

### 2.8.3.4 Adam

Adam stands for adaptive moment estimation, and is another way of using past gradients to calculate current gradients. Adam also utilizes the concept of momentum by adding fractions of previous gradients to the current one. This optimizer has become pretty widespread, and is practically accepted for use in training neural nets. [35]

## 2.8.4 Activation Functions

### 2.8.4.1 ReLU (Rectified Linear Unit)

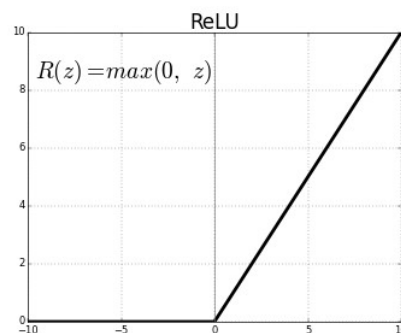


Figure 2.16: ReLU function

The ReLU (Rectified Linear Unit) is the go-to function for many neural networks since it is cheap to compute and still works well enough for many applications. It's defined by

$$f(x) = \max(z, 0) \quad (2.9)$$

. It is a non-linear function that gives the same output as input if the input is above 0, otherwise the output will be 0. that is,

- Output = input, if input is above 0
- Output = 0, if input is below 0

The ReLU function also helps with the problem of vanishing gradients in deep networks by not squashing in both ends. [36]



### 2.8.4.2 Softmax

Softmax function calculates the probabilities distribution of the event over ‘n’ different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs. It’s defined by

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_{k=1}^K e^{x_k}} \quad (2.10)$$

The main advantage of using Softmax is the output probabilities range. The range will be between 0 to 1, and the sum of all the probabilities will be equal to one. If the softmax function used for multi-classification model it returns the probabilities of each class and the target class will have the high probability.

The formula computes the exponential (e-power) of the given input value and the sum of exponential values of all the values in the inputs. Then the ratio of the exponential of the input value and the sum of exponential values is the output of the softmax function. [37]

### 2.8.4.3 Sigmoid

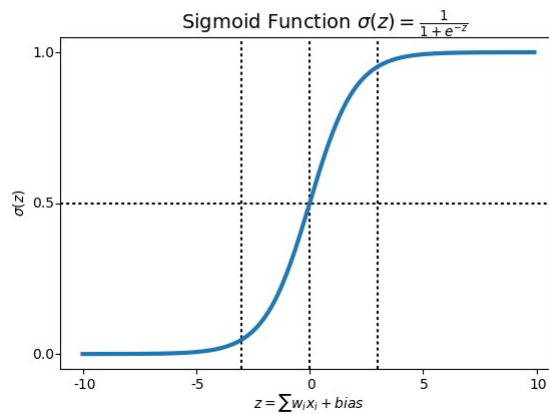


Figure 2.17: Sigmoid function

A sigmoid function or logistic function is defined mathematically as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.11)$$

The value of the function tends to zero when z or independent variable tends to negative infinity and tends to 1 when z tends to infinity (check figure 2.17 ). It needs to be kept in mind that this function represents an approximation of the behavior of the dependent variable and is an assumption. Now the question arises as to why use the sigmoid function as one of the approximation functions [37]. There are certain simple reasons for this :

1. It captures non-linearity in the data. Albeit in an approximated form, but the concept of non-linearity is essential for accurate modeling.
2. The sigmoid function is differentiable throughout and hence can be used with gradient descent and backpropagation approaches for calculating weights of different layers
3. The assumption of a dependent variable to follow a sigmoid function inherently assumes a Gaussian distribution for the independent variable which is a general distribution I see for a lot of randomly occurring events and this is a good generic distribution to start with.

## 2.8.5 Regularization

Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model's performance on the unseen data as well.

### 2.8.5.1 Dataset augmentation

An overfitting model (neural network or any other type of model) can perform better if learning algorithm processes more training data. While an existing dataset might be limited, for some machine learning problems there are relatively easy ways of creating synthetic data. For images some common techniques include translating the picture a few pixels, rotation, scaling. For classification problems it's usually feasible to inject random negatives—e.g. unrelated pictures.[38]

There is no general recipe regarding how the synthetic data should be generated and it varies a lot from problem to problem. The general principle is to expand the dataset by applying operations which reflect real world variations as close as possible. Having better dataset in practice significantly helps quality of the models, independent of the architecture. In keras, I can perform all of these transformations using *ImageDataGenerator*. It has a big list of arguments which can be used to preprocess the training data. [39] Here is a sample code to implement it.

```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(horizontal flip=True)
datagen.fit(train)
```

### 2.8.5.2 Early stopping

Early stopping is a kind of cross-validation strategy where I keep one part of the training set as the validation set. When I see that the performance on the validation set is getting worse, I immediately stop the training on the model. This is known as early stopping (check figure 2.18).



Figure 2.18: Early stopping process while training

In keras, I can apply early stopping using the *callbacks* function. Here is a sample code to implement it.

```
from keras.callbacks import EarlyStopping
EarlyStopping(monitor='val_err', patience=5)
```

Here, *monitor* denotes the quantity that needs to be monitored and '*val\_err*' denotes the validation error. After the dotted line, each epoch will result in a higher value of validation error. Therefore, 5 epochs after the dotted line (since our patience is equal to 5), my model will stop because no further improvement is seen. [38]

### 2.8.5.3 Dropout

This is one of the most interesting types of regularization techniques. It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning. At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections as the figure 2.10 shows. The probability of choosing how many nodes should be dropped is the hyperparameter of the dropout function. In keras, it can be implemented by using the keras core layer.

```
from keras.layers.core import Dropout
model = Sequential([
    Dense(output_dim = hidden1_num_units, input_dim=input_num_units, activation='relu'),
    Dropout(0.25),
    Dense(output_dim=output_num_units, input_dim=hidden5_num_units, activation='softmax'),
])
```

In the previous example I have defined 0.25 as the probability of dropping. I can tune it further for better results using the grid search method. [23]

### 2.8.5.4 Dense-sparse-dense training

The technique consists in 3 steps (check figure 2.19 ):

1. Perform initial regular training, but with the main purpose of seeing which weights are important, not learning the final weight values.

2. Drop the connections where the weights are under a particular threshold. Retrain the sparse network to learn the weights of the important connections.
3. Make the network dense again and retrain it using small learning rate, a step which adds back capacity. [40]

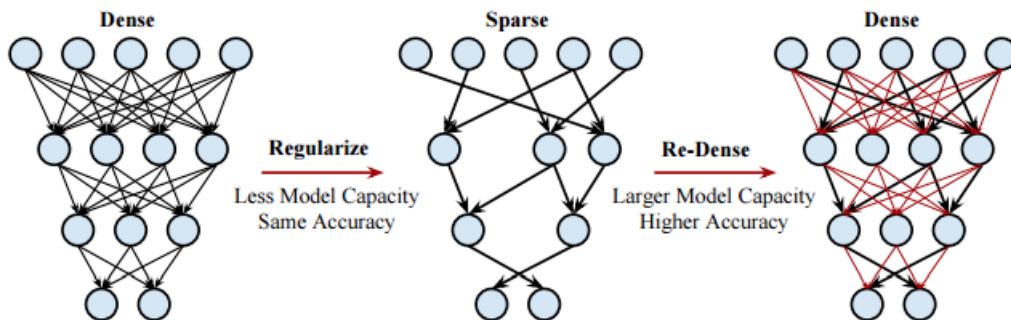


Figure 2.19: Dense-sparse-dense technique

### 2.8.5.5 Batch Normalization

Training Deep Neural Networks is complicated by the fact that the distribution of each layer’s inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. Batch normalization provides an elegant way of reparametrizing almost any deep network. The reparametrization significantly reduces the problem of coordinating updates across many layers.

BatchNorm impacts network training in a fundamental way: it makes the landscape of the corresponding optimization problem be significantly more smooth. This ensures, in particular, that the gradients are more predictive and thus allow for use of larger range of learning rates and faster network convergence. [41]

### 2.8.6 Transfer Learning

Transfer learning is the process of taking a pre-trained model (the weights and parameters of a network that has been trained on a large set by somebody else) and “fine-tuning” the model with your own dataset. The idea is that this pre-trained model will act as a feature extractor. You will remove the last layer of the network and replace it with your own classifier (depending on what your problem space is). You then freeze the weights of all the other layers and train the network normally (Freezing the layers means not changing the weights during gradient descent/optimization). [42] [43]

Transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error. The weights in re-used layers may be used as the starting point for the training process and adapted in response to the new problem. This usage treats transfer learning as a type of weight initialization scheme. This may be useful when the first related problem has a lot more labeled data than the

problem of interest and the similarity in the structure of the problem may be useful in both contexts. The use of a pre-trained model is about how model may be downloaded and used as it is into an application and used to classify new images. [44]

The pre-trained model can be used as a separate feature extraction program, in which case input can be pre-processed by the model or portion of the model to a given an output (e.g. vector of numbers) for each input image, that can then use as input when training a new model. And the pre-trained model or desired portion of the model can be integrated directly into a new neural network model. In this usage, the weights of the pre-trained can be frozen so that they are not updated as the new model is trained. Alternately, the weights may be updated during the training of the new model, perhaps with a lower learning rate, allowing the pre-trained model to act like a weight initialization scheme when training the new model.

this approach can be effective and save significant time in developing and training a deep convolutional neural network model.

The following table 2.1 lists some pretrained networks trained on ImageNet and some of their properties. The network depth is defined as the largest number of sequential convolutional or fully connected layers on a path from the input layer to the output layer. The inputs to all networks are RGB images

Network	Depth	Size	Parameters (Millions)	Image Input Size
alexnet	8	227 MB	61.0	227-by-227
vgg16	16	515 MB	138	224-by-224
vgg19	19	535 MB	144	224-by-224
squeezenet	18	4.6 MB	1.24	227-by-227
googlenet	22	27 MB	7.0	224-by-224
inceptionv3	48	89 MB	23.9	299-by-299
densenet201	201	77 MB	20.0	224-by-224
mobilenetv2	53	13 MB	3.5	224-by-224
resnet18	18	44 MB	11.7	224-by-224
resnet50	50	96 MB	25.6	224-by-224
resnet101	101	167 MB	44.6	224-by-224
xception	71	85 MB	22.9	299-by-299
shufflenet	50	6.3 MB	1.4	224-by-224
nasnetmobile	*	20 MB	5.3	224-by-224
nasnetlarge	*	360 MB	88.9	331-by-331

Table 2.1: A list of pretrained neural network

\* The NASNet-Mobile and NASNet-Large networks do not consist of a linear sequence of modules. [45]

## 2.9 Conclusion

In this chapter, I have discussed the origin idea of Deep Learning and its concept, its different types, and I have detailed the Convolutional Neural Network (CNN), I also have summarized its conception and cited some of architectures, the training part of the CNN is also has an important weight when talking about it, I discussed the image pre-processing, loss functions, some of the important optimizers, activation functions and I cited also regularization. Finally, I closed the chapter by talking about Transfer learning and I have referenced some of the well known pretrained models.

---

# Implementation

---

## 3.1 Introduction

In this chapter, I will define the architecture of the model I have created and then I will apply this model on the image dataset CIFAR 10. To achieve this goal, I will use some libraries, Tensorflow and Keras for learning and classification and to improve the performance of the model I will use some simple and effective techniques such as dropout.

## 3.2 Softwares and tools

### 3.2.1 Python

Python <sup>1</sup> is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

### 3.2.2 Tensorflow

TensorFlow <sup>2</sup> is an open source software library for numerical computation using data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture enables you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. TensorFlow also includes TensorBoard, a data visualization toolkit.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization for the purposes of conducting machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well. TensorFlow provides

---

<sup>1</sup><https://www.python.org/doc/essays/blurb/>

<sup>2</sup><https://www.analyticsindiamag.com/10-popular-machine-learning-projects-github/>

stable Python and C APIs as well as non-guaranteed backwards compatible APIs for C++, Go, Java, JavaScript, and Swift.

### 3.2.3 Keras

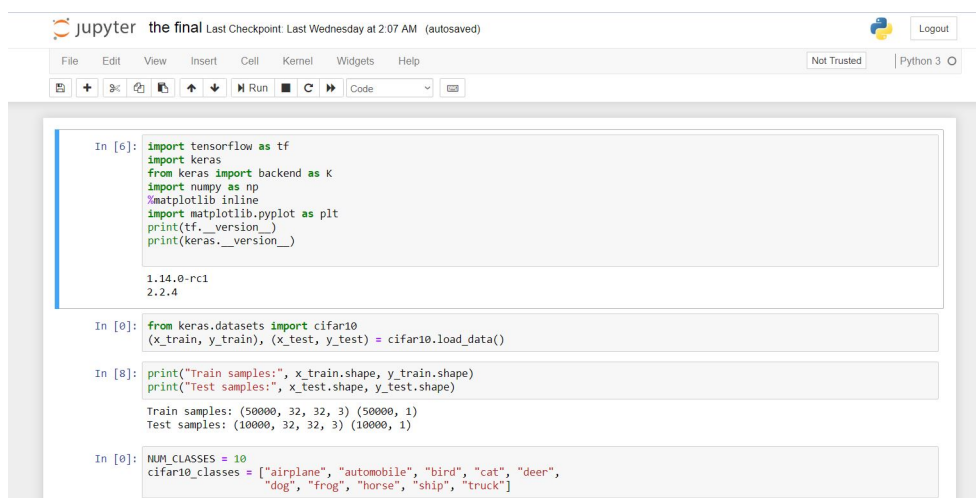
Keras<sup>3</sup> is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Keras was created to be user friendly, modular, easy to extend, and to work with Python. The API was "designed for human beings, not machines," and "follows best practices for reducing cognitive load." Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

### 3.2.4 Jupyter Notebook

The Jupyter Notebook<sup>4</sup> is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. It allows editing and running notebook documents via a web browser (check figure 3.1). The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.



```

jupyter the final Last Checkpoint: Last Wednesday at 2:07 AM (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 O
+ ↻ ↵ ⏮ ⏪ ⏩ ⏭ Run ⏸ ⏹ Code
In [6]: import tensorflow as tf
import keras
from keras import backend as K
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
print(tf.__version__)
print(keras.__version__)

1.14.0-rc1
2.2.4

In [0]: from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

In [8]: print("Train samples:", x_train.shape, y_train.shape)
print("Test samples:", x_test.shape, y_test.shape)

Train samples: (50000, 32, 32, 3) (50000, 1)
Test samples: (10000, 32, 32, 3) (10000, 1)

In [0]: NUM_CLASSES = 10
cifar10_classes = ["airplane", "automobile", "bird", "cat", "deer",
"dog", "frog", "horse", "ship", "truck"]

```

Figure 3.1: Jupyter Notebook interface

---

<sup>3</sup><http://keras.io/>

<sup>4</sup><https://jupyter.org/>



### 3.2.5 Google Colab

Google Colab <sup>5</sup> is a free cloud service and now it supports free GPU, the user can improve his Python programming language coding skills, develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV, it can be related to google drive to store all projects, and the most important feature that distinguishes Colab from other free cloud services is that Colab provides GPU and is totally free.

```

[ ] import tensorflow as tf
import keras
from keras import backend as K
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
print(tf.__version__)
print(keras.__version__)

[ ] 1.14.0-rc1
2.2.4
Using TensorFlow backend.

[ ] from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

[ ] print("Train samples:", x_train.shape, y_train.shape)
print("Test samples:", x_test.shape, y_test.shape)

[ ] Train samples: (50000, 32, 32, 3) (50000, 1)
Test samples: (10000, 32, 32, 3) (10000, 1)

[ ] NUM_CLASSES = 10
cifar10_classes = ["airplane", "automobile", "bird", "cat", "deer",
                  "dog", "frog", "horse", "ship", "truck"]

```

Figure 3.2: Google Collab interface

### 3.2.6 Hardware

The hardware used during the execution of the whole project can be defined in the table 3.1.

CPU	i5-4200H (2.8Ghz)
GPU	Nvidia GTX850M 4 Gb
Ram	8GB
OS	Windows 8.1 Pro

Table 3.1: The hardware used to run the tests

<sup>5</sup><https://colab.research.google.com/>

### 3.3 Dataset

The CIFAR-10 dataset <sup>6</sup> consists of 60000 32x32 color images in 10 classes (check figure 3.3), with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

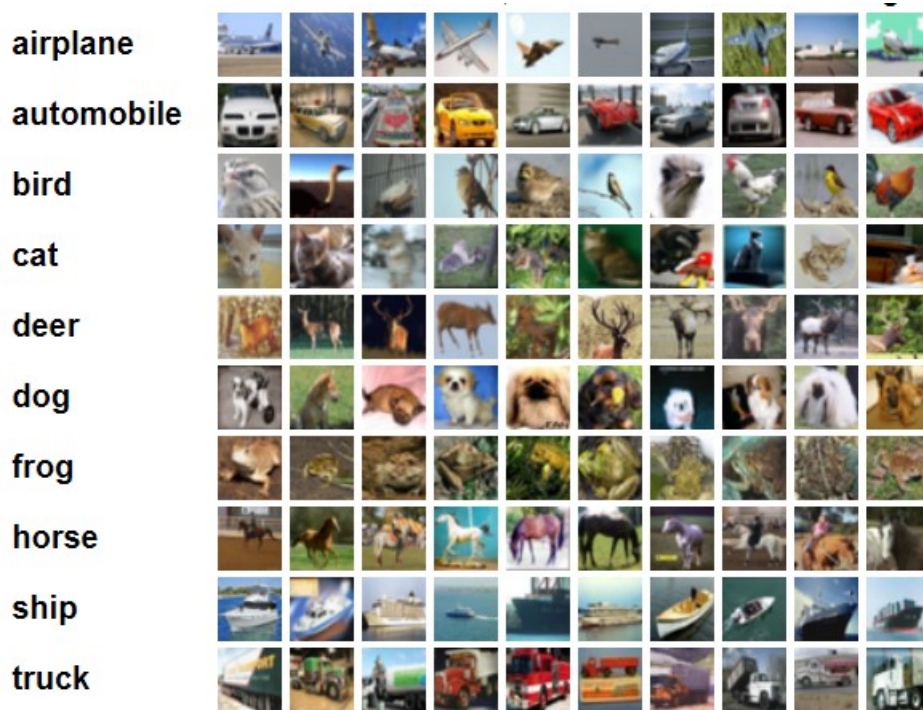


Figure 3.3: CIFAR10 dataset sample visualization

<sup>6</sup><https://www.cs.toronto.edu/~kriz/cifar.html>

### 3.4 CNN's Architecture

The studied model is composed of 18 layers (6 convolutions , 3 maxpooling, 4 batch normalization , 2 dropouts , 1 flatten ,2 dense).

The input image sized  $32 \times 32$ , the image passes through these hidden layers in order to get classified into one of the 10 classes of the dataset. here is the architecture (check figure 3.4)

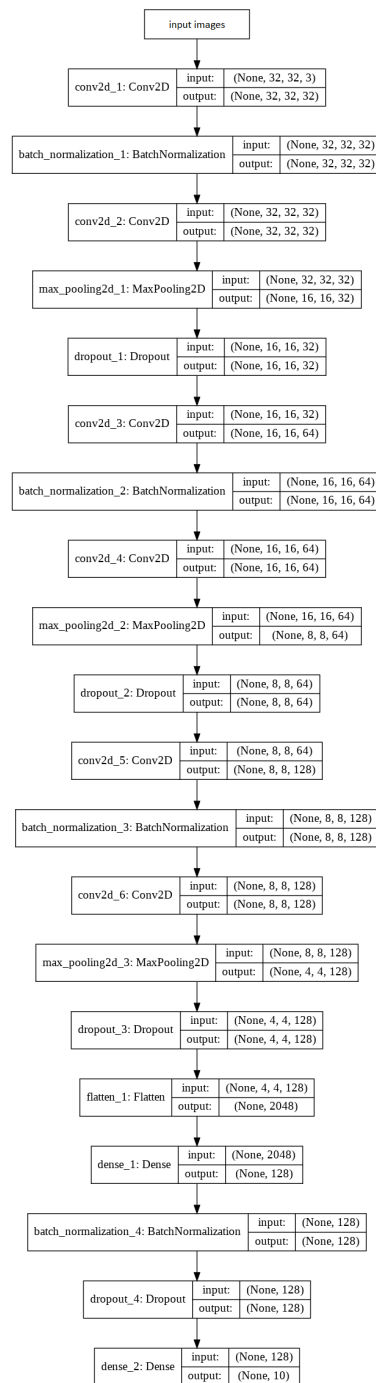


Figure 3.4: The model's architecture

This table (the table 3.2) shows the detailed architecture (summary) of the model.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	896
batch_normalization_1 (Batch Normalization)	(None, 32, 32, 32)	128
conv2d_2 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_3 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_4 (Conv2D)	(None, 16, 16, 64)	36928
batch_normalization_2 (Batch Normalization)	(None, 16, 16, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_5 (Conv2D)	(None, 8, 8, 128)	73856
conv2d_6 (Conv2D)	(None, 4, 4, 128)	147584
batch_normalization_3 (Batch Normalization)	(None, 8, 8, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_1 (Dropout)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 128)	262272
batch_normalization_4 (Batch Normalization)	(None, 128)	512
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 10)	1290
<b>Total params: 551,978</b>		
<b>Trainable params: 551,274</b>		
<b>Non-trainable params: 704</b>		

Table 3.2: The architecture of the used model for my test

### 3.5 Results

At the beginning, I started my tests by viewing a random sample from the dataset (check figure 3.5).



Figure 3.5: A random sample from the dataset

By executing the tests I have reached some results. It can be resumed by the plots in figure 3.6 which represents the model training and the validation accuracy.

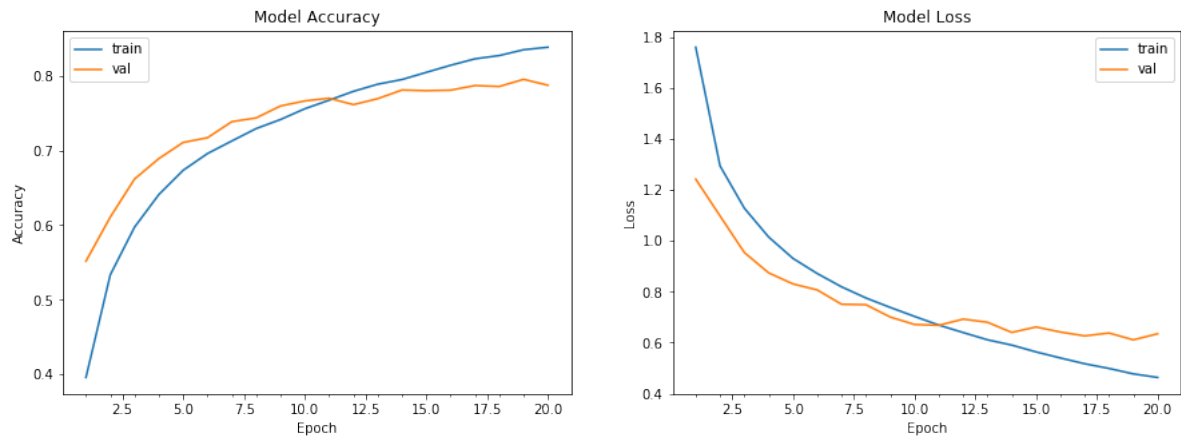


Figure 3.6: Test Accuracy and Model Loss plots

From figure 3.6 at the "Model Accuracy" part , The training increases rapidly than validation with the number of epochs. Therefore the training rate decreases rapidly than validation at the Model loss. this reflects that at each time the model learns more information.

The confusion matrix allows us to evaluate the performance of our model, since it reflects the metrics of True positive, True negative, False positive and False negative. the figure 3.7 closely illustrates the position of these metrics for each class. As an example, the model classified the images bird, cat and truck and misclassified the images of dog, frog and automobile

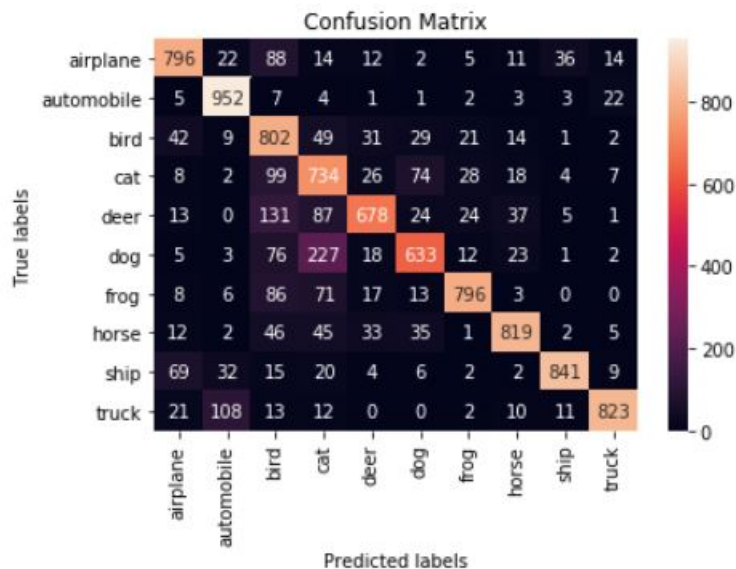


Figure 3.7: The confusion matrix of the model

the figure 3.8 illustrates the rate of the error of classified and misclassified picture by the trained model. by observation I can say that the model got an acceptable accuracy since the misclassification isn't more than the 1/4 of the whole data (check figure 3.9).

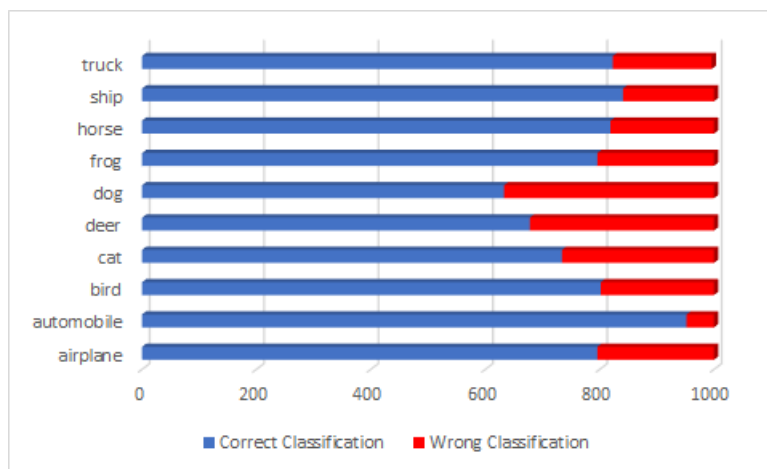


Figure 3.8: The error rate of the model

From Figure (check figure 3.9). I notice that all misclassified images are 2122 images, an error rate of 21.22% and the totality of the well classified images is 7874 a precision rate of 78.74%.

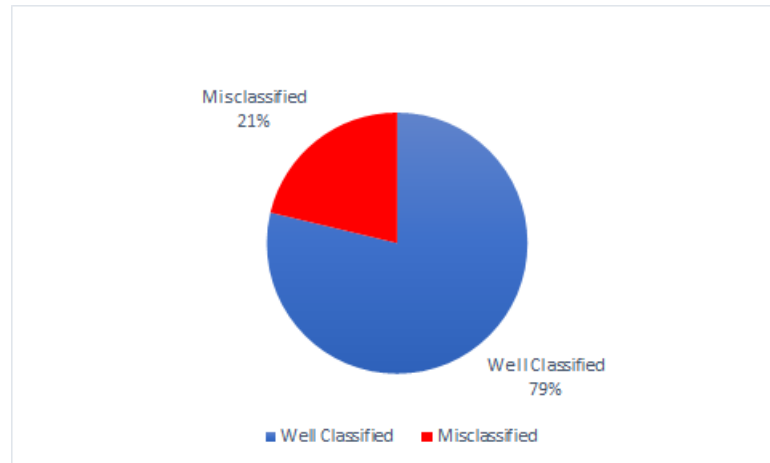


Figure 3.9: The error rate of the model

In addition, the figure 3.10 shows how a sample of images had been effected by the previous layers of the model, more precisely at the last dense layer.



Figure 3.10: The effect of the model's layers

At the end of the tests, the model was able to classify images with a Test Accuracy equals to 0.7874. the figure 3.11 shows a sample of images with their predicted class and validation accuracy and the true class.



Figure 3.11: A sample of predicted images with their labels

The table 3.3 shows the execution time as well as the number of epochs. The results obtained are expressed in terms of test accuracy and execution time. The execution time is kind of expensive. In general, a large and deep convolutional neuron network gives good results and the performance of the model used in this study is acceptable. **Batch normalization** operation improved and reduced the execution time alot. I noticed a huge gap in the execution time comparing when it is used and when it is not.

	Personal Laptop	Google Colab
Execution time	5502 s (91.7 minute)	660 s (11 minute)
epochs	20	20
Test Accuracy	78,74%	77,91%

Table 3.3: The resume of obtained results

By executing the same model with the same epochs in Google Colab, the table 3.3 shows the results. Knowing that I have runned the tests usig the GPU provided by this cloud service (ram: 12Gb and storage: 358Gb)

### 3.6 Conclusion

I have presented in this chapter an Image Classification approach based on Convolutional Neural Networks, for which I used a model with a 18 layers architecture and I have showed the different results obtained in terms of test accuracy.

According to the result found, I have noticed that there is two antagonist options, either I increase the number of epoch and the depth of the network to give a good result but it will cause an increase of the training time.



# Conclusion

I have presented in this work the basics of convolutional neural networks and their use in image classification. I have also introduced the different types of layers used in the classification: the convolutional layer, the pooling layer and the fully connected layer. Subsequently, I presented the best-known convolutional architectures and training parameters of a CNN model such as optimizers, loss functions, activation functions and regularizations.

To save computational time and to avoid overfitting problem, the hyper parameters of the network have been studied like dropout that removes some nodes randomly at every iteration.

The training time was too expensive because of the large size of the dataset which requires the use of the graphics processor GPU in addition to CPU.

The learning dataset is also a critical element in convolutional neural networks, so I need to have a large dataset to achieve better results.

In order to achieve this work, I spent a lot of time reading and studying the documents to see what is best about classification and to design our own model. This work allowed us to put our knowledge of neural networks into practice and to acquire other knowledge, and the time spent reading articles served as a good introduction to research.

As perspectives I am planning to study my experiments on other datasets and use the notion of transfer learning on some pretrained models.

# Bibliography

- [1] Tom M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.
- [2] Daniel Faggella. What is machine learning? <https://emerj.com/ai-glossary-terms/what-is-machine-learning/>, Mars 2019.
- [3] Thomas H. Davenport. Machine learning. [https://www.sas.com/en\\_us/insights/analytics/machine-learning.html](https://www.sas.com/en_us/insights/analytics/machine-learning.html).
- [4] Desir C. and L. H. *Classification automatique d'images, application à l'imagerie du poumon profond*. 2013.
- [5] Klaus Hechenbichler and Klaus Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. 2004.
- [6] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*, volume 20. Siam, 2007.
- [7] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
- [8] Weina Wang, Yunjie Zhang, Yi Li, and Xiaona Zhang. The global fuzzy c-means clustering algorithm. In *2006 6th World Congress on Intelligent Control and Automation*, volume 1, pages 3604–3607. IEEE, 2006.
- [9] Rohith Gandhi. Support vector machine—introduction to machine learning algorithms. <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>.
- [10] Sidath Asiri. Machine learning classifiers. <https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623>.
- [11] Neil D. Lawrence. Machine learning motivation.
- [12] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [13] John Bryden. Biologically inspired computing: The neural network.
- [14] Yamini Chauhan. Axon anatomy. <https://www.britannica.com/science/axon>.
- [15] B Rudy. Diversity and ubiquity of k channels. *Neuroscience*, 25(3):729–749, 1988.

- [16] Tutorials Point. Artificial intelligence neural networks. [https://www.tutorialspoint.com/artificial\\_intelligence/artificial\\_intelligence\\_neural\\_networks.htm](https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm), May 2019.
- [17] KISHAN MALADKAR. Types of artificial neural networks. <https://www.analyticsindiamag.com/6-types-of-artificial-neural-networks-currently-being-used-in-todays-technology/>, JAN 2019.
- [18] Ch Sanjeev Kumar Dash, Ajit Kumar Behera, Satchidananda Dehuri, and Sung-Bae Cho. Radial basis function neural networks: a topical state-of-the-art survey. *Open Computer Science*, 6(1), 2016.
- [19] Markus Törmä. Kohonen self-organizing feature map in pattern recognition. *Photogramm. J. Finland*, 15:1, 1995.
- [20] Aish Warya. Introduction to recurrent neural network. <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>.
- [21] : Mayank Mishra. Convolutional neural networks. <https://www.datascience.com/blog/convolutional-neural-network>.
- [22] Patduc Jacque. Cnn (convolution neural network). <https://patducjacquet.wordpress.com/2017/07/04/cnn-convolution-neural-network-une-introduction/>, May 2019.
- [23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [28] Nikhil B. Image data pre-processing for neural networks. <https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258>, June 2019.
- [29] Raul Gomez. Understanding categorical cross-entropy loss - binary cross-entropy loss - softmax loss and logistic loss. [https://gombru.github.io/2018/05/23/cross\\_entropy\\_loss/](https://gombru.github.io/2018/05/23/cross_entropy_loss/), May 2019.

- [30] Isaac Changhau. Loss functions in neural networks. [https://isaacchanghau.github.io/post/loss\\_functions/](https://isaacchanghau.github.io/post/loss_functions/), June 2019.
- [31] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.
- [32] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.
- [33] Sebastian Ruder. An overview of gradient descent optimization algorithms. <http://ruder.io/optimizing-gradient-descent/>, June 2019.
- [34] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
- [35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [36] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [37] Saimadhu Polamuri. Difference between softmax function and sigmoid function. <http://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/>, June 2019.
- [38] SHUBHAM JAIN. An overview of regularization techniques in deep learning. <https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/>, June 2019.
- [39] Bharath Raj. Data augmentation : How to use deep learning when you have limited data. <https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced>.
- [40] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Shijian Tang, Erich Elsen, Bryan Catanzaro, John Tran, and William J Dally. Dsd: regularizing deep neural networks with dense-sparse-dense training flow. *arXiv preprint arXiv:1607.04381*, 3(6), 2016.
- [41] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.
- [42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.
- [43] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

- [44] Jason Brownlee. How to use transfer learning when developing convolutional neural network models. <https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/>, May 2019.
- [45] Matlab. Pretrained deep neural networks. <https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html>, 2019.
- [46] Arnaud De Myttenaere and Golden. Mean absolute percentage error for regression models. *Neurocomputing*, 192:38–48, 2016.
- [47] Ching-Pei Lee and Chih-Jen Lin. A study on l2-loss (squared hinge-loss) multiclass svm. *Neural computation*, 25(5):1302–1323, 2013.