To My Dad's Soul

POLITECNICO DI TORINO

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Matematica

Tesi di Laurea Magistrale

Image Classification using Machine Learning Techniques

Relatore:
prof. Paolo Garza

Candidata:
Noura Maanna

Supervisore aziendale
Data Reply srl
dott. Marco Gatta

Anno accademico 2017-2018

Summary

Image classification problem is one of the core problems in Compute Vision, that despite its simplicity, has many different practical applications. Our brains can recognize objects and faces without any effort, but this is challenging for computers. So, the goal of my thesis was to create an algorithm that classifies satellite images indicating the presence or not of a tree/house in the tested image.

Deep Learning and Convolutional Neural Networks are pioneer in image classification because of their performance regarding accuracy and time needed for the network to be trained. For this reason, I created a convolutional neural network architecture to be trained and I used Python as a language program because of its libraries and packages that make working with images easy. So, first I divided the data set into Training, Validation and Testing set, then I labelled the training and the validation set indicating the presence of a tree or a house in each image. Then I trained the network and I tested it using the unlabelled images present in the Testing set.

Acknowledgements

I would first like to thank my supervisor, Prof. Paolo Garza, for the patient guidance and advices that he gave to me throughout this research. His office was always opened whenever I faced a problem or had a question about my thesis or writing.

I owe more than thanks to my Mom Fatima and to Federico, my husband, for their support and unflinching courage throughout my years of study and through the process of preparing and writing my thesis. This accomplishment would not have been possible without them. Thank you!

Noura Maanna

Contents

Summary III

Acknowledgements IV

1 Introduction 1

2 Classification Techniques 7
2.1 Logistic Regression . . . 9
2.2 Support Vector Machines . . . 11
2.3 Bayesian Networks . . . 14
2.3.1 Naive Bayes . . . 15
2.4 Decision Tree . . . 17
2.5 Neural Networks . . . 20
2.5.1 Deep Learning and Convolutional Neural Networks . . . 26

3 Problem Specification 35

4 Results 43

5 Conclusions 51

Bibliography 55

List of Figures

1.1 Machine Learning vs Statistics.
Source: *https://analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/.* 3

2.1 Classification as the task of mapping an input attribute set x into its class label y. 8
2.2 The Sigmoid Function.
Source: *https://www.quora.com/What-does-an-x-axis-represent-in-a-logistic-regression-sigmoid-function* 10
2.3 Simple example of linear SVM. 12
2.4 A directed acyclic graph showing the inter-relationships between attributes.
Source: *https://www.semanticscholar.org/paper/Learning-Bayesian-Belief-Network-Classifiers.* 15
2.5 Spam emails classifier.
Source: *https://software.intel.com/en-us/articles/using-intel-data-analytics-acceleration-library-improve-performance-of-na-ve-bayes.* . . 17
2.6 Decision Tree.
Source: *https://pdfs.semanticscholar.org/db26/7c80b215574f4b0f5244 3b86707194bb0ff7.* 18
2.7 An example of a simple decision tree for image classification with respect to the classification class landscapes (L) 20
2.8 The Brain Structure is made of Neurons and Synapses.
Source: *https://medium.com/@startuphackers /building-a-deep-learning-neural-network-startup-7032932e09c.* 21
2.9 Nonlinear model of a neuron.
Source: *https://www.slideshare.net/MostafaGMMostafa/neural-networks-introducton.* 23
2.10 Feedforward with a single layer neurons. 25
2.11 Neural Network with two hidden layers.
Source: *https://medium.com/@rajatgupta310198/getting-started-with-neural-network-for-regression-and-tensorflow-58ad3bd75223.* 25

2.12 Examples of CIFAR and MINIST image datasets.
Source: *https://www.kaggle.com/c/cifar-10.* 27
2.13 CNNs for image classification.
Source: *https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner.* 30
2.14 CNNs for digits handwriting classification.
Source: *https://www.researchgate.net/ figure /An−Example−CNN−architecture − for − a − handwritten − digit − recognition − task_fig*1_220785200. 30
2.15 Confusion Matrix for binary classification.
Source: *https://docs.wso2.com/display/ ML110/Model+Evaluation+Measures.* 32

3.1 Blocks diagram describing the different steps for a Machine Learning model. 35
3.2 Satellite image taken from comune of Bellagio in the Italian region of Lombardy. 36
3.3 The Model's Architecture. 39

4.1 Applying the same architecture used for CIFAR-10 on my dataset. . . 43
4.2 Cifar-10 architecture with learning rate = 0.0001. 44
4.3 Cifar-10 architecture with ADAM optimizer and learning rate = 0.01. 44
4.4 Cifar-10 architecture with ADAM optimizer and learning rate = 0.0001. 45
4.5 Model's loss and accuracy. 46
4.6 Inception architecture. 48

5.1 AlexNet and VGGNet architectures. Source: Hirokatsu Kataoka, Kenji Iwata and Yutaka Satoh, "Feature Evaluation of Deep Convolutional Neural Networks for Object Recognition and Detection". . . . 53

List of Tables

4.1 Model's Architecture 45
4.2 Confusion Matrix 46
4.3 Results of transfer learning 48
4.4 MobileNet body architecture 49

Chapter 1

Introduction

Learning is a phenomenon, that generates thinking and produces knowledge. Humans do it continuously throughout their lives. They also make others do learning as well, e.g., they make their children, their pets, their colleagues learn new things. After the impact of computing, the Artificial Intelligence was described as a term that includes cognitive abilities inside a computing machine. One of the fundamental parts of this is Machine Learning.

Two important innovations led to the upturn of Machine Learning as the vehicle that drives AI development forward with the speed it currently has.

One of these was the Turing Test. In 1950, Alan Turing, introduced the well-known Turing Test. The Turing test consisted of an interrogator that is given a script of certain questions to be answered by two entities: a human and a machine. The machine is said to be Artificially Intelligent, if it is able to answer those questions in a similar manner (natural language conversion) as the human and the interrogator cannot distinguish between the answers of the machine and those of the human. In the 1970s, scientists started to support and further the practical solutions of AI; as a consequence Jane Robinson and Don Walker founded a Natural Language Processing setup in Stanford university.

The second and the more recent innovation, was the emergence of the internet, and the vast increase in the amount of digital information being generated and stored to be available for analysis. This innovation let engineers to realize that it would be more efficient to train machines and computers how to think like humans and giving them the access to all the information in the world by connecting them to internet. So the idea of Machine Learning was based on the ability to give machines access to data and let them learn for themselves.

Machine Learning and statistics are linked together, we cannot speak about one without mentioning the other. Machine Learning has coexisted with Statistics uneasily, throughout history, since Machine Learning has adopted many of statistical methods, without intending to replace statistics. Computers apply statistical learning techniques to automatically identify patterns in data and to make highly accurate predictions. This computational analysis of machine learning algorithms is a branch of statistics known as computational learning theory. Fundamentally, both ML and Statistics work with data to solve problems. Machine learning may emphasize prediction, and statistics may focus more on estimations, but both focus on using mathematical techniques to answer questions [9].

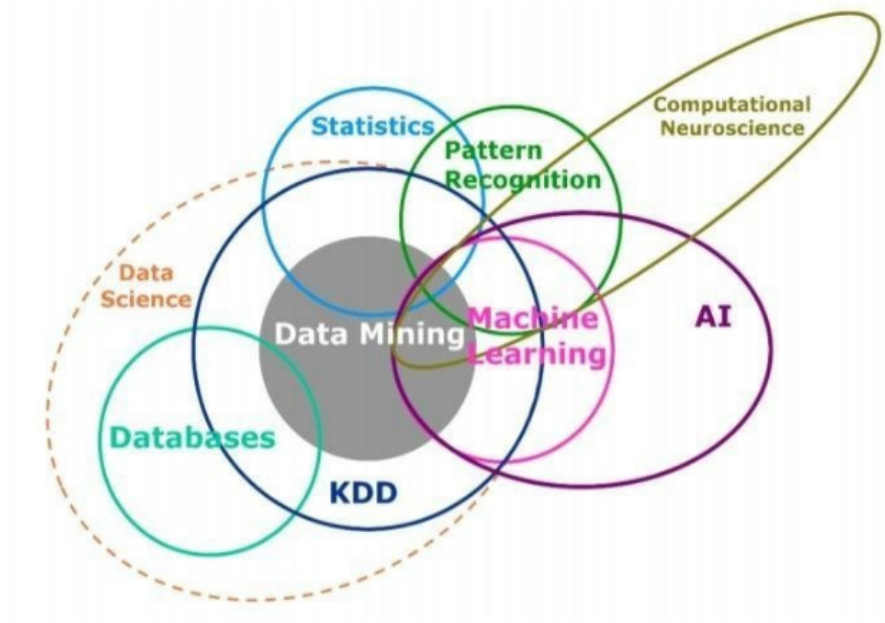


Figure 1.1: Machine Learning vs Statistics.
Source: *https://analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/.*

Machine learning is a method used to construct complex models and algorithms to predict and classify data. There are wide variety of applications of machine learning, some of them are: Image Recognition, Natural Language Processing, Fraud Detection and Stock Market Analysis. There are dozen of algorithms for machine learning, where most of them fall under three macro categories: Supervised Learning, Unsupervised Learning and Reinforcement Learning. Depending on the type of data under study we can choose the algorithm to be used. For example, in image recognition usually the supervised learning is used. For supervised learning, the output should be specified and the input is called training data with a known label. While training the model should learn how to map the correct label to corresponding output. Digit recognition is a common example of supervised learning.

In unsupervised learning, the input data is not labeled and there is no training phase. The correct answers are not given to the model, so the model is prepared by deducing structures and patterns must learn from the input data by itself. Common examples are clustering and association rules where the model will discover similarities from the input data.

Reinforcement learning differs from both supervised and unsupervised learning, it enables an agent to learn in an interactive environment by trail and error using feedback from its own actions, experiences and choices.

Also there are many machine learning algorithms, selecting the right algorithm is an important part of any machine learning program. Since there are many to choose from, understanding their advantages and disadvantages in various business applications is essential. The most common machine learning algorithms are: Decision Trees, Neural Networks, Logistic Regression and K-Means Clustering. In the next chapter we will describe in details the function of some of these algorithms[2].

After this brief introduction to the concept of Machine Learning, I can introduce my work. I used Deep Learning and Convolutional Neural Networks as machine learning algorithms to develop an image recognition program that will be useful in the Insurance domain. Identifying houses between trees is valuable for insurance companies, since it can helps in studying the risk of having a fire beside the house caused by the trees surrounding it, or the risk to have different kind of damages caused by trees falling on the house as a result of a tempest storm, for example. Having human analysts that classify these images is too expensive and time consuming for such companies. So, developing a machine learning algorithm that classifies such kind of images will be less expansive and the results will be much faster. After studying and analyzing these images the insurance companies can price the fire and damage policy in a more efficient way, such that, they can increase the price of such policies depending on the position of the houses and the trees surrounding them.

So, the idea was to create a program using Python that gets as input labelled satellite images of houses surrounded by trees. After training the algorithm using deep learning techniques and different packages of Python like TensorFlow and Keras, the program will be able to classify the images indicating the presence or not of a tree/house in the tested image. We tried different algorithms in order to optimize the program's accuracy.

This work passed by three main steps:

1. Preparation of the dataset.
    - Choosing the images, coloring them, creating there masks and labelling

them.

- Preparing the Training, Testing and Validation sets.

2. Training the algorithm.
3. Testing the program and analyzing different statistical outputs.

In the next chapters we will describe these steps in a more detailed way.

Chapter 2

Classification Techniques

Machine learning is one of the applications of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from training. The learning process begins with the visualization of different examples or observations that helps in making better predictions in the future. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

In machine learning and statistics, classification is specified as supervised learning approach in which the computer program learns from the inout labeled data given to it and then uses this learning to classify unlabeled observations. The data set may be formed by only two classes (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class. Some examples of classification problems are: document classification, biometric identification, handwriting recognition, classifying galaxies based upon their shapes etc [13].

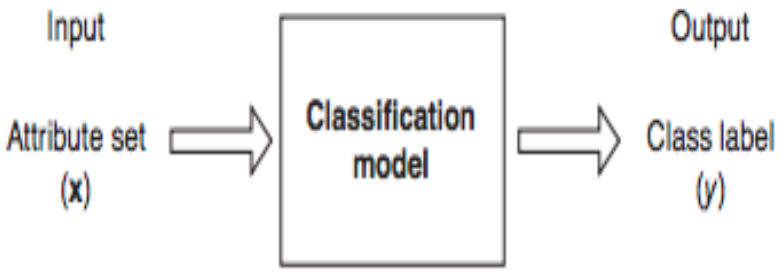


Figure 2.1: Classification as the task of mapping an input attribute set x into its class label y.

The most frequently used algorithms in the area of supervised learning as classification techniques are the following:

1. Logistic Regression
2. Support Vector Machines
3. Bayesian Networks
4. Decision Tree
5. Neural Networks

2.1 Logistic Regression

Logistic regression, which is borrowed from the field of classical statistics, is one of the simpler machine learning algorithms. This machine learning technique is commonly used for binary classification problems, which means that, there are two possible outcomes that are influenced by one or more explanatory variables. The algorithm estimates the probability of an outcome given a set of observed variables.

We can define the logistic regression as the mapping of a family of functions I from $\mathbb{R}^d$ to the interval [0, 1]. $I(t)$ can be considered as the probability that the label of t is 1. The logistic regression is associated to a hypothesis class, that is a composition of the sigmoid function $\phi_{sig} : \mathbb{R} \to [0, 1]$ over the class of linear functions L_d. The sigmoid function used in logistic regression is the logistic function, and it is defined as:

$$\phi_{sig}(z) = \frac{1}{1 + e^{-\alpha^T z}} \tag{2.1}$$

Where the coefficients α are the weights that the model wants to learn and z is the input feature vector. So, the sigmoid function will give us the probability of an output to be 1 under the given input features. In other words, this algorithm calculates the probability of belonging to each class.
The name "sigmoid" means "S-shaped", referring to the plot of this function, shown in the Figure 2.2:

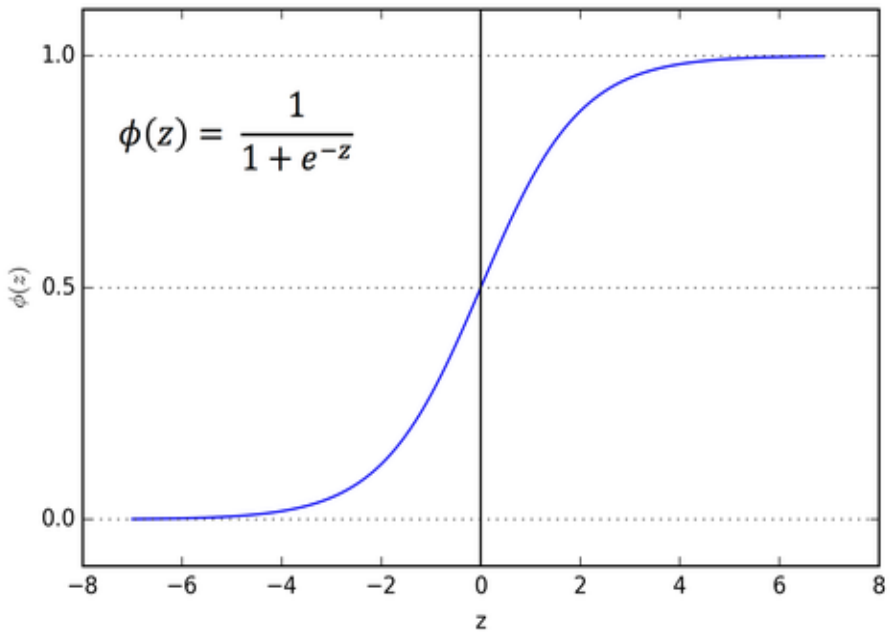


Figure 2.2: The Sigmoid Function.
Source: *https://www.quora.com/What-does-an-x-axis-represent-in-a-logistic-regression-sigmoid-function*

.

Here, is a simple example of applying the logistic regression algorithm. Imagine we want to distinguish between objects belonging to one of two classes based on a set of L training examples. Indicate by y_i the class to which the i-th example belongs, with $y_i = +1$ representing class C_1 and $y_i = -1$ representing class C_2. Logistic regression is a classical approach to this problem, that attempts to estimate the a-posteriori probability of class membership based on a linear combination of the input features,

$$P(C_1|\mathbf{x}) = \frac{1}{1 + e^{-f(x)}} \tag{2.2}$$

where

$$f(x_i) = \sum_{j=1}^{d} a_j x_{ij} + a_0 \tag{2.3}$$

The parameters of the logistic regression model, $\mathbf{a} = (a_0, a_1, ..., a_d)$, can be found by maximizing the likelihood of the training examples.

Where logistic regression differs from other methods is in its interpretability. Since this algorithm is derived from the highly interpretable linear regression algorithm,

the influence of each data feature can be interpreted without much effort. Logistic regression is a well-established classification technique that is widely used in epidemiological studies. It is generally used as a reference, in comparison with other techniques for analyzing medical data[7].

2.2 Support Vector Machines

The Support Vector Machine (SVM) classification was first introduced in 1982 by Vapnik. This method usually used as prediction method in bioinformatics and medical science because of its high accuracy, its flexibility in modeling diverse sources of data and its ability in dealing with high-dimensional data. It can classify highly nonlinear data using kernel functions.

Support vector machines are a combination of Machine learning theory, optimization algorithms and kernel techniques. Given a dataset consisting of features set and their labels, the SVM model predicts the classes for the new test set. If the dataset has 2 labeled classes, then the SVM classifier is called binary.

We have two types of SVM classifiers:

1. **Linear SVM Classifier**: Maximizes the geometric margin of training dataset, that is maximizing the distance from hyperplane to the nearest data point of either class. It predicts a straight hyperplane dividing 2 classes. The drawn hyperplane called the maximum-margin hyperplane. The following is the standard form of a linear binary SVM classifier[11]:

$$F(x) = w^T x + b = \sum_{k=1}^{n} w_k x_k + b, \tag{2.4}$$

where x is the features vector with dimension n. The optimal weight vector $\mathbf{w}$ and bais b can be obtained by maximizing the soft margin, which penalizes each sample by the hinge loss:

$$arg\min_{w,b} \frac{\lambda}{2}\|w\|^2 + \frac{1}{|S|}\sum_{i\in S} \max(0,1 - y_i(w^T x_i + b)), \tag{2.5}$$

where x_i is the $i-th$ feature vector, y_i the corresponding label and S is the set of training samples. The first part of the above optimization problem is a

maximum margin classifier that maximizes the geometric margin $\frac{1}{||\mathbf{w}||}$ which is equivalent to minimizing $||\mathbf{w}||^2$.

The formula in (2.5) is equivalent to a QP problem with a quadratic cost function subject to linear constraints as:

$$\begin{aligned} &arg\min_{w,b} \frac{\lambda}{2}\|w\|^2 + \frac{1}{|S|}\sum_{i\in S}\xi_i \\ &s.t. \forall i \in S : \xi_i \geq 0; \xi_i \geq 1 - y_i(w^T x_i + b). \end{aligned} \tag{2.6}$$

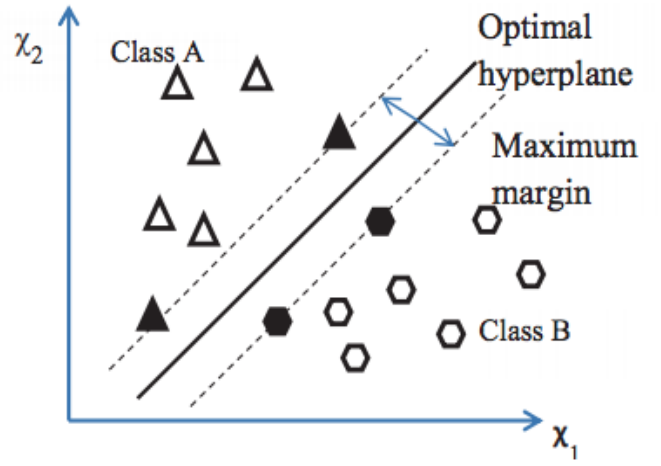


Figure 2.3: Simple example of linear SVM.

2. **Non-Linear SVM Classifier**: This model is used when the training dataset is non-linearly separable, that is, there is no straight hyperplane that can separate the classes. In order to apply SVMs we should use first the nonlinear kernel functions to transform input data to a high-dimensional feature space in which the input data become more separable compared to the original. Then, Maximum-margin hyperplanes are created.

   Let $\phi(\cdot)$ be a nonlinear mapping function from the input space to the high-dimensional feature space and $\mathbf{x}$ be the vector of the n-dimensional space. Then the hyperplane can be defined as follows[17]:

$$\mathbf{w} \cdot \phi(x) - b = 0 \tag{2.7}$$

where $\mathbf{w}$ is the weight vector that maps the training data into the output space. So the weight vector can have the following formula:

$$\mathbf{w} = \sum \alpha_k y_k \phi(x_k) \tag{2.8}$$

where y_i is the label for x_i and α_i is the Lagrange multiplier. Then the classification function can be written as follows:

$$F(x) = \sum_{k}^{n} \alpha_k y_k \phi(x_k) \cdot \phi(x) - b. \tag{2.9}$$

Let's use the kernel trick to replace the inner product, substituting the kernel function $K(u, v) = \phi(u) \cdot \phi(v)$ in (2.9). Thus, the classification function becomes:

$$F(x) = \sum_{k} \alpha_k y_l K(x_k, x) - b \tag{2.10}$$

The followings are popularly used kernel functions with SVMs[18]:

1. Sigmoid kernel: $K(u, v) = tanh(\kappa u \cdot v + c)$. It is used as proxy for neural networks.
2. Polynomial kernel: $K(u, v) = (u \cdot v + 1)^n$, where n is the degree of the polynomial. This kind is usually used in image processing.
3. ANOVA radial basis kernel: $K(u, v) = \sum_{i=1}^{n} exp(-\sigma(u^k - v^k)^2)^d$. It is used in regression problems.
4. Gaussian kernel: $K(u, v) = exp(-\frac{||u-v||^2}{2\sigma^2})$. It is used when there is no prior knowledge about the data.
5. Radial Basis Function (RBF): $K(u, v) = exp(-\gamma||u - v||^2)$. It is also used when there is no prior knowledge about the data.

Some common applications for SVMs are the following:

- Cancer and genes classification in bioinformatics.
- Text and hypertext categorization.
- Face detection in which SVM classifies parts of the image as face and non-face.
- Protien fold and remote homology detection.

2.3 Bayesian Networks

The Bayesian Classification it is listed under the supervised learning category and also it is recognized as statistical method for classification. Bayesian classification assumes an underlying probabilistic model that allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. It can solve diagnostic and predictive problems. This classification is named after Thomas Bayes (1702-1761), who proposed the Bayes Theorem. Naive Bayes classifiers are among the most successful known algorithms for learning to classify text documents and so spam filtering. Naive Bayes classification is used also in identifying credit card fraud and fake Amazon reviews.

Properties of Bayes Classifier:

1. Combines prior knowledge and observed data: prior probability of a hypothesis multiplied with probability of the hypothesis given the training data.

2. Probabilistic hypothesis: outputs not only a classification, but a probability distribution over all classes.

Modeling uncertainty is a challenging topic in Artificial Intelligence applications. For more than 10 years, probabilistic graphical models like Bayesian networks have proved to be well suited and computationally efficient to deal with uncertainties in many practical applications.

Bayesian network was first introduced by Pearl in 1988. It consists of a directed acyclic graph (DAG), that is, all of the edges (or links) are directed in a particular node and there are no cycles. Each node in a Bayesian network represents a domain variable (eg, a dataset attribute) and each link between these nodes represents a probabilistic dependency, quantified using a conditional probability distribution. Given values assigned to other nodes, the Bayesian Network can be used to calculate the conditional probability of one node. Since the Bayesian Network is a complete model for the variables and their relationships, it can be used as a process to calculate the posterior probability distribution of the class node given the values of other attributes. Thanks for this interaction between dataset attributes the Bayesian Networks become more efficient than other types of predictive modelsa. Human experts can easily understand Bayesian Network structures and they can modify them to obtain better predictive models[5].

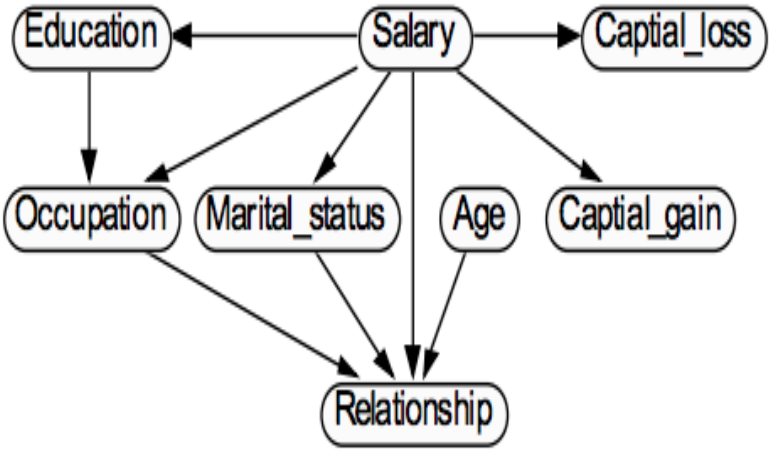


Figure 2.4: A directed acyclic graph showing the inter-relationships between attributes.
Source: *https://www.semanticscholar.org/paper/Learning-Bayesian-Belief-Network-Classifiers.*

2.3.1 Naive Bayes

Naive Bayes is one of the Bayesian Network classifiers. The Naive Bayesian classifier assumes conditional independence among all attributes given the class variable. It learns from training data the conditional probability of each attribute given its class label. This classifier basically learns the class-conditional probabilities $P(X_i = x_i|C = c_l)$ for each attribute X_i given the class label c_l. A new test case $X_1 = x_1$, $X_2 = x_2$, ..., $X_n = x_n$ is then classified by using Bayesian rule to compute the posterior probability of each class c_l given the vector of observed attribute values according to the following equation[4]:

$$\begin{aligned} &P(C = c_l|X_1 = x_1, X_2 = x_2, ..., X_n = x_n) \\ &\quad = \frac{P(C = c_l)P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n)|C = c_l)}{P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n)} \end{aligned} \tag{2.11}$$

The simplifying assumption behind the Naive Bayesian classifier is that the attributes are conditionally independent, given the class label according to the following equation:

$$\begin{aligned} &P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n|C = c_l) \\ &\quad = \prod_{i=1}^{n} P(X_i = x_i|C = c_l) \end{aligned} \tag{2.12}$$

This assumption simplifies the estimation of the class conditional probabilities from the training data. Notice that one does not estimate the denominator in (2.11) since it is independent of the class. Instead, one normalizes the nominator term to 1 over all classes.

Naive Bayesian classifiers have been proven successful in many domains, despite the simplicity of the model and the restrictiveness of the independent assumptions it made. The Naive Bayes algorithm handles only categorical data, but could not reasonably express the probability between two numeric values and preserve the structure of numeric values. Extended Naive Bayesian algorithm is used in data mining as a simple and effective classification algorithm.

Bayesian spam classification is an important example for the use of Naive Bayesian classifier. It calculates the probability of a message being spam based on its contents. Unlike simple content-based filters, Bayesian spam classification learns from spam and from good mail, resulting in a very robust, adapting and efficient anti-spam approach that, best of all, returns hardly any false positives.

We can interpret the term $p(w_1|Spam)$ as the probability of finding word w_1 in a spam email. Using (2.11) we get the following:

$$P(Spam|w_1, ..., w_n) = \frac{P(w_1|Spam)P(w_2|Spam)...P(w_n|Spam)}{P(w_1, ..., w_n)} \tag{2.13}$$

And using the product notation, the above equation can be written as follows:

$$P(Spam|w_1, ..., w_n)) = \frac{P(Spam)\prod_{i=1}^{n} P(w_i|Spam)}{P(w_1, ..., w_n)} \tag{2.14}$$

In other words, this gives the probability if an email is a spam, considering the words in that email. This is different for text classification, because in this case we actually need a label, not a probability, so we simply say that an email is spam if $P(Spam|w_1, ..., w_n)$ is greater than 50%. If not, then it is not spam. That is, based on the higher probability between the two classes we can choose if the email is spam or ham.

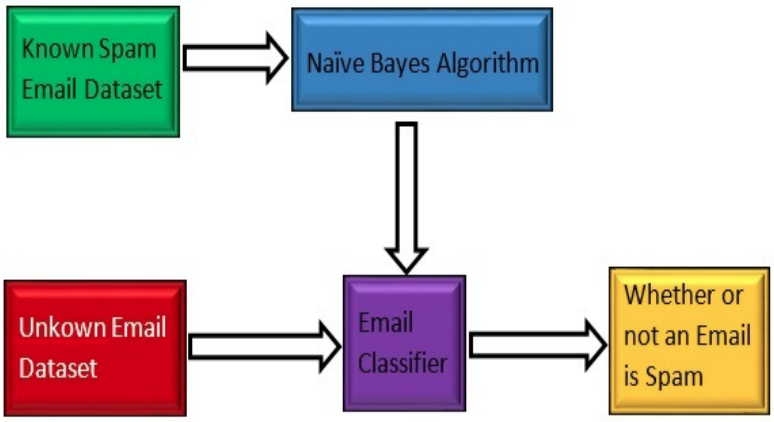


Figure 2.5: Spam emails classifier.
Source: *https://software.intel.com/en-us/articles/using-intel-data-analytics-acceleration-library-improve-performance-of-na-ve-bayes.*

2.4 Decision Tree

Another popular and easy algorithm to understand is decision trees. This algorithm is based on the "divide and conquer" strategy. The idea of using decision trees to identify and classify objects was first reported by Hunt et al. (1996). The construction of decision tree requires supervised training; therefore it is necessary to have a training dataset consisting of response and explanatory variables. The classification structure defined by a decision tree is estimated from training data using a statistical procedure. The "tree" is made of a root node, internal nodes and leaves.Nodes are where trees branch or split the dataset; terminal nodes are called leaves that represent a class label or a decision[13]. Each node in a decision tree represents a feature in an instance to be classified, and each branch has a weight that can be assumed by each node. Instances are classified starting at the root node and sorted based on their feature values. In another word, the feature that best divides the training data into correct classes would be the root node of the tree. The construction of the tree then will be progressed from one node to the next according to the features that best separate the classes in the training data.

Every decision tree should have a training set and a test set that measures the accuracy of the classifier. The generalized method for constructing a decision tree from a dataset T of training cases with classes denoted by $\{C_1, C_2, ..., C_k\}$ is Hunt's method and can be summarized as follows:

1. T contains one or more cases, which all belonging to a single class C_j, then the decision tree for the dataset T is a leaf identifying class C_j.

2. If T contains no cases, the decision tree for T is a leaf.

3. If T contains cases that belong to a mixture of classes, then a test is chosen, based on a single attribute, that has one or more mutually exclusive outcomes $\{O_1, O_2, ..., O_n\}$. T is partitioned into subsets T_1,T_2,...,T_n, where T_i contains all the objects in T that have outcome O_i of the chosen test. The decision tree for the dataset T contains a decision node identifying the test, and one branch for each possible outcome. The same tree building machinery is applied recursively to each subset of training cases[12].

Decision trees have the following two phases:

- Tree construction: In this phase, all the training examples are at the root node and test attributes are selected on a heuristic or a statistical measures.

- Tree pruning: This phase is characterized by the identification and the removal of branches that reflects outliers or noise. One rule is generated for each path in the tree from the root to a terminal node(leaf) that represents a class distribution or label. Each attribute-value pair along a path forms a conjunction.

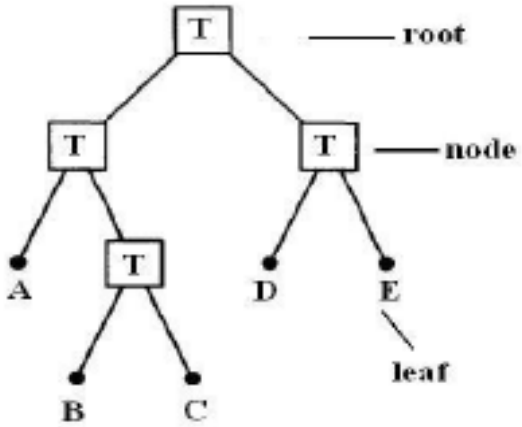


Figure 2.6: Decision Tree.
Source: *https://pdfs.semanticscholar.org/db26/7c80b215574f4b0f5244 3b86707194bb0ff7.*

A Decision Tree is built from a training set, which consists of objects, each of which is completely described by a set of attributes and a class label. Attributes are a collection of properties containing all the information about one object. Unlike class, each attribute may have either ordered (integer or a real value) or unordered values (Boolean value).
There are many decision trees that can be constructed from a given set of attributes. While some of these trees are more accurate than others, finding the optimal tree is computationally unfeasible because of the exponential size of the search space. There are several dimensions along which decision trees might differ:

a. The tests might be multivariate (testing on several features of the input at once) or univariate (testing on only one of the features).

b. The tests might have two outcomes or more than two. (If all of the tests have two outcomes, we have a binary decision tree.)

c. The features or attributes might be categorical or numeric. (Binary-valued ones can be regarded as either.)

d. We might have two classes or more than two. If we have two classes and binary inputs, the tree implements a Boolean function, and is called a Boolean decision tree.

Several methods have been proposed to construct decision trees. These methods generally use the recursive-partitioning algorithm, and its input requires a set of training examples, a splitting rule, and a stopping rule. Partitioning of the tree is determined by the splitting rule and the stopping rule determines if the examples in the training set can be split further. If a split is still possible, the examples in the training set are divided into subsets by performing a set of statistical tests defined by the splitting rule. The test that results in the best split is selected and applied to the training set, which divides the training set into subsets. This procedure is recursively repeated for each subset until no more splitting is possible.

Stopping rules differ from one application to another and multiple stopping rules can be used across different applications. One stopping rule is to test for the purity of a node. For instance, if all the examples in the training set in a node belong to the same class, the node is considered to be pure and no more splitting is performed[1]. Another stopping rule is by looking at the depth of the node, defined by the length of the path from the root to that node. If the splitting of the current node will produce a tree with a depth greater than a pre-defined threshold, no more splitting is allowed. Another common stopping rule is the example size. If the number of examples at a node is below a certain threshold, then splitting is not allowed. Four

widely used splitting rules that segregates data includes: gini, entropy and class probability. The gini index is defined as:

$$gini(t) = \sum p_k(1 - p_k) \tag{2.15}$$

where p_k is the relative frequency of class k at node t. Node t is a node where the split of the data took place. Note that node t can be a parent or child node.

Another measure is the Entropy and is defined as :

$$entropy(t) = -\sum_k p_k log p_k \tag{2.16}$$

where p_k is the ratio of elements with class k at node t. The entropy rule controls how the a decision tree decides to split the data. Class probability is also based on the gini equation but the results are focused on the probability structure of the tree rather than the classification structure or prediction success. The rule attempts to segregate the data based on probabilities of response and uses class probability trees to perform class assignment[1].

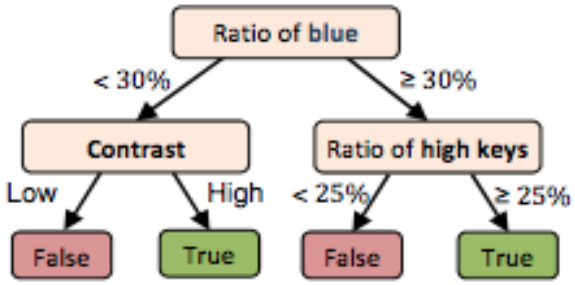


Figure 2.7: An example of a simple decision tree for image classification with respect to the classification class landscapes (L)

2.5 Neural Networks

Brains are recognized as biological computing machines. They can transform a flood of complex and ambiguous sensory information into logic thoughts and actions, allowing an organism to understand and model its environment, synthesize and make

decisions from disparate streams of information, and adapt to a changing environment. Biological brains, consisting of some 100 billion neurons and connected by an estimated 100 trillion synapses, are essentially general purpose complex and efficient problem solving systems that provide living organisms with the trait of intelligence.

The brain is a black box that receives an input in the form of a certain stimulus from the environment and produces a corresponding appropriate output or response. The environmental information is transformed in neural signals in the brain. The brain uses this stored neural information in the form of patterns of neural firing to generate the desired actions that are most likely to accomplish the goal at hand[6].

The goal of artificial neural network machine learning algorithms is to mimic the way the human brain organizes and understands information in order to arrive at various predictions. One of the earliest instantiations of a neural network that could learn was the "perceptron" of Rosenblatt, who proposed a simple arrangement of input and output neurons that could make decisions on the basis of input vectors. In the domain of machine learning, the 1990s saw the development and rise of a variety of machine learning approaches that would supplant the neural network.

(a) The Brain.

(b) Synapses and Neurons.

Figure 2.8: The Brain Structure is made of Neurons and Synapses.
Source: *https://medium.com/@startuphackers /building-a-deep-learning-neural-network-startup-7032932e09c.*

A Neural Network is a computer system that is designed to work and classify information in the same way as a human brain. The Neural Network can be trained to recognize, for example, images, and classify them according to some patterns they contain. A Neural network is formed from nodes, which are similar to our biological neurons in the brain.

Neural networks has the following basic properties:

- *Nonlinearity*: A neural network that is, made by the connection of nonlinear neurons, is considered as nonlinear. Nonlinearity is an important property that it is distributed all over the network, especially If the underlying physical mechanism responsible for generation of the input signal (e.g., speech signal) is inherently nonlinear.
- *Input-Output Mapping*: One of the categories of machine learning algorithms is the so-called Supervised Learning, that takes as input a labeled dataset and the output is known and specified. While training it will modify the synaptic weights of a neural network to minimize the difference between the true label and the target that will be predicted by the model. This phase will be repeated on all the training dataset, until the network will reach the best synaptic weights. This process called the input-output mapping by which the model will learn from the training examples the features needed to predict the desired labels.
- *Adaptivity*: Neural networks have a mechanism to manage their synaptic weights with the surrounding environmental changes. That is, a neural network which is trained to work in a specific environment can be easily retrained to manage minor environmental changes. The benefits of adaptivity can be clearly recognized if the principal time constants of the system long enough for the system to ignore dummy disturbances, and yet short enough to respond to meaningful changes in the environment[10].

Here, we identify 3 main elements of neural model:

(a) Synapses, or connecting links, that connect neurons with each other. Synapses are characterized by their weight. They take the input from the previous layer and multiply it by their weight.

(b) An adder that sums the outputs from all synapses.

(c) An activation function which is a node that is added to the output of the layer for limiting the amplitude of the output of a neuron.The activation function is also used at the classification layer of a neural network model to determine

the predicted output. Most popular activation functions are Relu, Sigmoid, Softmax and Hyperbolic tangent "Tanh".

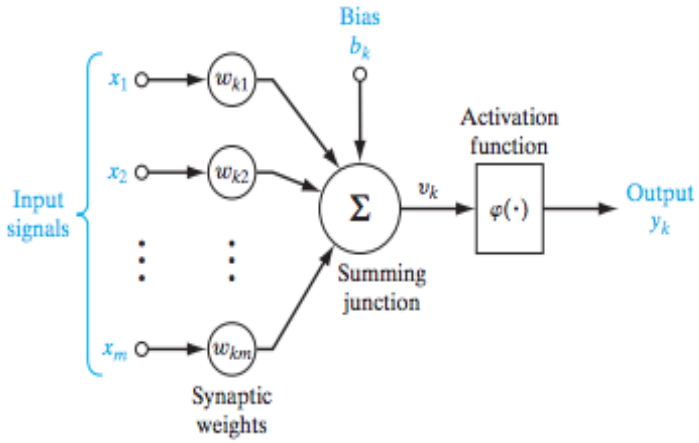


Figure 2.9: Nonlinear model of a neuron.
Source: *https://www.slideshare.net/MostafaGMMostafa/neural-networks-introducton.*

We can notice that an external bias function was applied on the model b_k in order to increase or lower the total activation function output depending on its value(if it is positive or negative). $x_1, \ldots, x_m$ are the input signals; $w_{k1}, \ldots, w_{km}$ are the respective synaptic weights of neuron k and ϕ is the activation function[10].
In mathematical terms, we may describe the neuron k by writing the following two equations:

$$u_k = \sum_{j=1} p_m w_{kj} x_j \tag{2.17}$$

and

$$y_k = \phi(u_k + b_k) \tag{2.18}$$

$$v_k = u_k + b_k \tag{2.19}$$

Where u_k is the linear combiner output and y_k is the output signal of the neuron.

To train neural networks, we should find a decision function that updates the weights of the network, that is, updating the network weights and biases to map correctly arbitrary inputs to outputs. So, first the biases b_k can be initialized to 0, while the weights w_{kj} generally are initialized with small random values belong to the interval $[-c, c]$ with $c = \frac{\sqrt{\epsilon}}{N_k + N_{k-1}}$ where N_k is the size of the hidden layer k. In order to avoid that all the neurons of a hidden layer will have the same behavior, these weights can not be initialized with the same values. Training then progresses through a series of iterations that continuously improve the weights. Each iteration of the complete training set is called an epoch. In each epoch the network adjusts the weights in the direction that reduces the error. As the iterative process of incremental adjustment continues, the weights gradually converge to the locally optimal set of values. Many epochs are usually required before training is completed. The most common error function is the mean squared error (MSE) that is : $MSE = \frac{1}{N} \sum_{k=1}^{N} (\hat{y}_k - y_k)^2$, where $\hat{y}_k$ are the actual outputs and y_k are the desired outputs. In another words, the weight vector is updated along the negative direction of the gradient of MSE as $\mathbf{w} = \mathbf{w} - \eta \frac{\partial MSE}{\partial \mathbf{w}}$, until MSE becomes small enough. Here, the parameter η is called the learning rate, that is used to control how big of a step we are taking each iteration. It is the most important hyper-parameter to tune when training neural networks. If you choose learning rate that is too big the algorithm will diverge, since we are making steps that are too large and so we are skipping the minimum(jump over it). If you choose learning rate that is too small, it might take too much time to converge to some local minima[10].

There exist three fundamentally different classes of neural network architectures:

1. **Single-Layer Feedforward Networks**: The neural networks models are formed from many layers. In the case where the neural network is formed from one input layer, then it called the single feedforward network.

2. **The multilayer Feedforward Networks**: A multilayer is a structure composed by several hidden layers of neurons where the output of a neuron of a layer becomes the input of a neuron of the next layer. We may apply a different activation function on the last layer, called output layer, such as for the hidden layers depending wether the type of the problem we are dealing with is a regression one or a classification problem. The architecture of Multilayers Networks is a basic one, since each unit (or neuron) of a layer is connected to all the units of the next layer but has no connection with the neurons of the same layer. This architecture has the following parameters: the number of hidden layers and of neurons in each layer. A convolutional neural network(CNN) is a special case of multilayer networks, we will speak about CNN

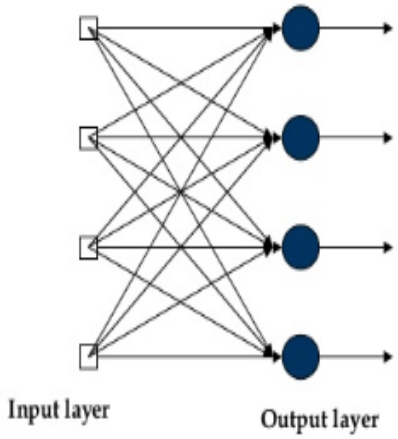


Figure 2.10: Feedforward with a single layer neurons.

in details in the next section[14].

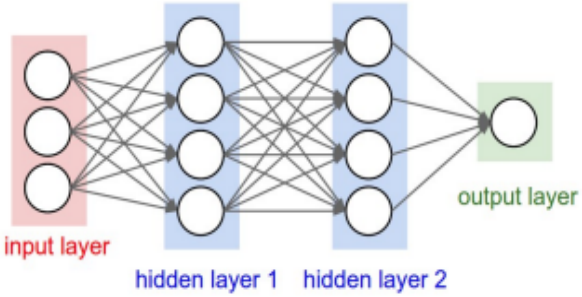


Figure 2.11: Neural Network with two hidden layers.
Source: *https://medium.com/@rajatgupta310198/getting-started-with-neural-network-for-regression-and-tensorflow-58ad3bd75223.*

3. **The recurrent neural networks**: A recurrent neural network it has at least one feedback loop, that makes it differ from a feedforward neural network. This kind of networks usually consist of one single layer of neurons that feed back the inputs of the other neurons.

Neural Networks are a new era of machine learning algorithms that can be applied for many problems, but their training needs huge computational complexity. Neural networks and convolutional deep learning currently provide the best solutions to many problems in image recognition. The Nonlinearities that are represented by convolutional and pooling layers, capable of capturing the characteristic features of images.

2.5.1 Deep Learning and Convolutional Neural Networks

Deep learning is a family of machine learning methods that was based on our knowledge of the human brain, statistics and applied math as it developed over the past several decades. In recent years, deep learning has seen enormous and extraordinary growth in its popularity and usefulness, as the result of more powerful computers, larger datasets and techniques to train deeper networks.

Deep learning is in the intersections among the research areas of neural networks, artificial intelligence, graphical modeling, optimization, pattern recognition, and signal processing. Three important reasons for the popularity of deep learning today are the drastically increased chip processing abilities (e.g., general-purpose graphical processing units or GPGPUs), the significantly lowered cost of computing hardware, and the recent advances in machine learning and signal/information processing research. These advances have enabled the deep learning methods to effectively exploit complex, compositional nonlinear functions, to learn distributed and hierarchical feature representations, and to make effective use of both labeled and unlabeled data[8].

One of the main characteristics of Deep Learning networks that make them different from the single layer net is their depth(i.e., the number of nodes and layers through which data passes in a multistep process of pattern recognition). Based on the previous layer's output, each node is trained on a distinct set of features. While moving forward into the network, the nodes can recognize more complicated features. The importance of a trained deep learning network on labeled data is that, this network can be applied to unstructured data. In order for your model to be more accurate, it should be trained on more data with classes diversity. Usually, the last layer in the Deep earning networks is the classifier layer with a logistic, sigmoid or softmax function that assigns labels to the elements tested set.

One of the principal limitations of traditional artificial neural networks has been that methods for training a system with multiple layers were not straightforward or not available, till the explosion of the concept of Deep learning. In 1986, Rina Dechter introduced the term Deep Learning to the machine learning domain, while in 2000, Igor Aizenberg and colleagues introduced Deep learning to Artificial Neural Networks. The first general, working learning algorithm for supervised, deep, feedforward, multilayer perceptrons was published by Alexey Ivakhnenko and Lapa in 1965. In 1980, Kunihiko Fukushima introduced Neocgnitron as a new architectures

for computer vision[16]. In 1989, Yann LeCun et al. applied the standard back propagation algorithm, which had been around as the reverse mode of automatic differentiation since 1970, to a deep neural network with the purpose of recognizing handwritten ZIP codes on mail. A common evaluation set for image classification is the MNIST database data set. MNIST is composed of handwritten digits and includes 60,000 training images and 10,000 test images. Another example on image classification was the CIFAR-100 dataset, this data set has 100 classes containing 600 image each class with 50000 training images and 10000 testing images. By using different deep learning architectures these two data sets were classified with a very high accuracy[3].

(a) CIFAR dataset.

(b) MINST dataset.

Figure 2.12: Examples of CIFAR and MINIST image datasets.
Source: *https://www.kaggle.com/c/cifar-10.*

Depending on how the architectures and techniques are intended for use, one can broadly categorize most of the work in this area into three major classes:

1. Deep networks for unsupervised or generative learning, this kind of networks can find high-order correlation from the observed examples or they can synthesis purposes when no information about target class labels is available. When used in the generative mode, may also be intended to characterize joint statistical distributions of the visible data and their associated classes when available and being treated as part of the visible data. Usually in this case, Bayes rules are used turning the type of generative networks into a discriminative one for learning.

2. Deep networks for supervised learning, they are characterized by providing a

discriminative power for pattern classification purposes, that is, they give us the posterior distributions of classes conditioned on the observed data. Usually in this case, the target data are labeled. This kind of networks are also called discriminative deep networks.

3. Hybrid deep networks, where the goal is discrimination which is assisted, often in a significant way, with the outcomes of generative or unsupervised deep networks. This can be accomplished by better optimization or/and regularization of the deep networks in the second category. The goal can also be accomplished when discriminative criteria for supervised learning are used to estimate the parameters in any of the deep generative or unsupervised deep networks in the first category above[8].

Deep learning is a crucial component of modern speech recognition systems used at major companies, including Microsoft, IBM and Google. It is also used by Google in its voice and image recognition algorithms, by Netflix and Amazon to decide what you want to watch or buy next.

We can not speak about deep learning without explaining the concept of convolutional neural networks (CNNs). Convolutional neural networks are special type of artificial neural networks. They are pioneer in classifying images, cluster them by finding similar features, and perform object recognition within scenes. These algorithms can be used in identifying tumors, faces, individuals identify faces, individuals and many other aspects of visual data. The efficacy of convolutional networks (CNNs) in image recognition is one of the main reasons why the world has woken up to the efficacy of deep learning. They were introduced in computer vision (CV) and its applications such that: drones, robotics, self-driving cars and security.

The word "convolutional" is derived From the Latin word convolvere, "to convolve", which means to roll together. While in mathematics, a convolution is the integral that measures how much two functions overlap as one passes over the other, that is, is a sort of mixing two functions by multiplying them. Convolutional networks work as follows: they apply many filters over a single image, these filters can be horizontal line filters, vertical or curved... where each filter will memorize different features. This step is needed to create a map of the edges in the image. Convolutional networks take those filters, slices of the image's feature space, and map them one by one; that is, they create a map of each place that feature occurs. Usually after every convolutional layer the input image is passed through a nonlinear transformation function such as tanh or rectified linear unit (RELU), which will map input values into a range between -1 and 1.

The input image for a convolutional network is usually transformed into a rectangular box whose width and height are measured by the number of pixels along those dimensions, with a depth of three layers, one for each color in RGB format. Those depth layers are referred to as channels, for example: we can express them mathematically as matrices of multiple dimensions in this form: 30x30x3 (30x30 are height and width of the images and 3 describes the 3 colors RGB). For each pixel of an image, the intensity of R, G and B will be expressed by a number, and that number will be an element in one of the three, stacked two-dimensional matrices, which together form the image volume.

Convolutional network structure includes the following forms of constraints[10]:

1. **Feature extraction**: To extract local features, each neuron takes its synaptic inputs from a local receptive field in the previous layer. After the extraction of features was taken place, the feature's exact location becomes less important, so long as its position relative to other features is approximately preserved.

2. **Feature mapping**: Feature map is the output of one filter applied to the previous layer. It is formed by a sliding filter that passes over the image to figure out the synaptic weights.

3. **Subsampling(Max pooling)**: This layer is used to reduce the number of parameters and the resolution of the feature map but it retains the important information.

Training plays an important role in the convolutional neural networks, since the network extracts features from every example in the training set and so it will learn how to extract features automatically by itself.

Here an example for convolutional neural network architecture consists of convolutional, max pooling and fully connected layers.

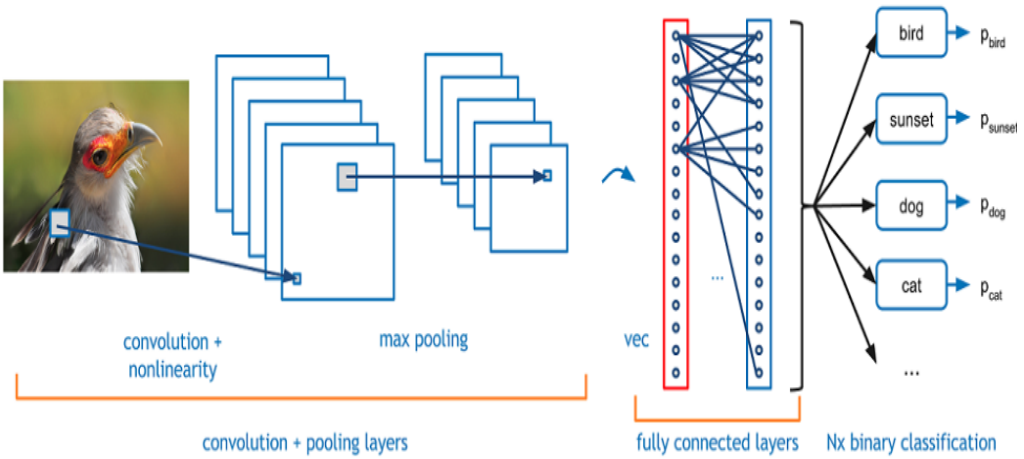


Figure 2.13: CNNs for image classification.
Source: *https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner.*

Let's consider another example from the MINST dataset.

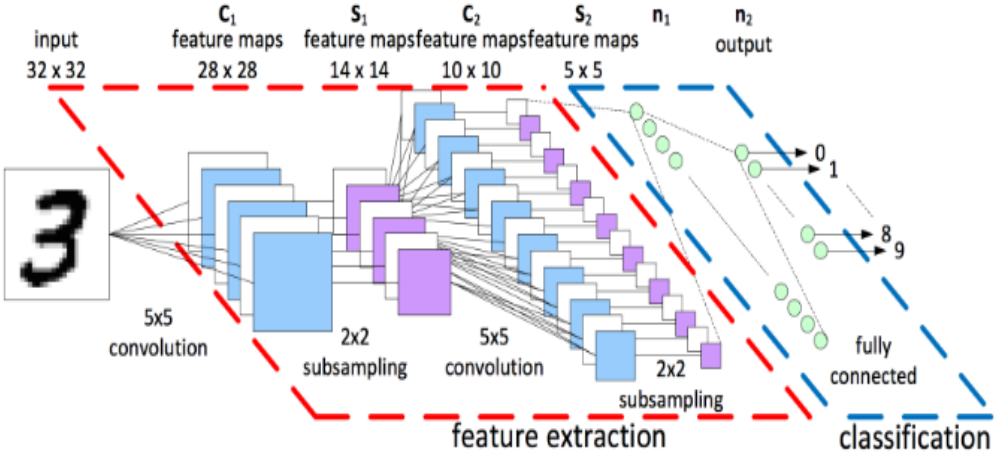


Figure 2.14: CNNs for digits handwriting classification.
Source: *https://www.researchgate.net/ figure /An − Example − CNN − architecture − for − a − handwritten − digit − recognition − task_ fig*1_220785200.

The input image is 32×32 pixels and the computational layouts alternate between convolution and subsampling:

1. The first hidden layer is a convolutional layer, that consists of 6 feature maps,

with each feature map consisting of 28×28 neurons. Each neuron is assigned a receptive field of size 5×5.

2. The second hidden layer performs subsampling(average pooling) and it consists of 6 feature maps, where each is made of 14×14 neurons with a receptive field of size 2×2 with a sigmoid activation function.

3. The third hidden layer performs is another convolutional layer that consists of 16 feature maps, with each feature map consisting of 10×10 neurons.

4. The fourth hidden layer performs a second subsampling(average pooling) that consists of 16 feature maps, but with each feature map consisting of 5×5 neurons.

5. At the end we have the classification part that consists from the fully connected layer and the output layer. The output layer in this case it has 10 neurons, with each neuron assigned to one of 10 possible characters.

Deep convolutional nets are pioneer through image and video processing, whereas recurrent nets were powerful in sequential data such as text and speech.
The choice of which specific learning algorithm we should use is challenging, because we should choose the algorithm that will give us the most accurate predictions. There are methods that help in choosing the best classifier. One method is known as cross-validation and another is the Leave-one-out validation. Using these techniques we can calculate the classifier's accuracy on the set of labeled data and so, we can choose one of the classifiers mentioned above.

- **Cross Validation**: K-fold cross-validation is the most common method for estimating the performance of a learning model, where k is the number of subgroups that a given data set will be splited into. The general procedure is as follows [15]:

  1. Split the dataset randomly into k equal sized folds.
  2. k iterations of training and validation are performed such that within each iteration a unique fold of the data is used for validation while the remaining $k - 1$ folds are used for training.
  3. Retain the evaluation score and discard the model.

  The k results are usually summarized using the sample of model evaluation scores, such as the standard deviation or the standard error. The choosing the value of k is very important, because poor values will give us a wrong or

not clear estimation of our model. A good and fair estimate of the algorithms performance on tested data, is the size of the test partition in which it should be large enough to be a good and reasonable sample of the problem. For datasets with thousands or tens of thousands of records, k values of 3, 5 or 10 are commonly used.

- **Leave-One-Out**: Leave-one-out cross-validation (LOOCV) is a special case of k-fold cross-validation. For a dataset with N examples, we split our dataset into $k = 1$ folds and we iterate N times. For each iteration we use $N - 1$ examples for training and the remaining example for testing. A generalization error estimate is obtained by repeating this procedure for each training examples available.
  LOOCV it is computationally expensive because it generally requires one to construct many models equal to the size of the training set.

The metrics that you choose to evaluate your machine learning model are very important. Choice of metrics influences how the performance of machine learning algorithms is measured and compared. Those metrics are the following:

1. Confusion Matrix: It is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is used for classification problem where the output can be of two or more types of classes.

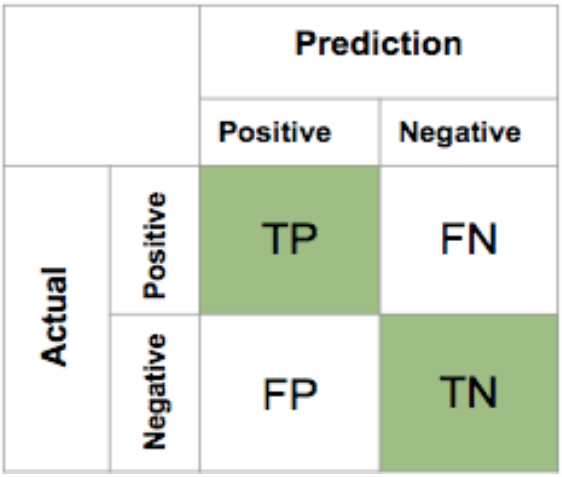


Figure 2.15: Confusion Matrix for binary classification.
Source: *https://docs.wso2.com/display/ ML110/Model+Evaluation+ Measures.*

- True Positives (**TP**): True positives are the cases in which the predicted and the actual label were both 1(True). That is, the algorithm classified truly the un labeled cases.
- True Negatives (**TN**): True negatives are the cases when the actual and the predicted label were both 0(False). That is, the model was able to give the correct label to the tested cases.
- False Positives (**FP**): False positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True). In other words, the classifier was not able to give the correct label to the tested case and instead of assigning a 0, it labeled the case as 1.
- False Negatives (**FN**): False negatives are the cases when the actual label of the tested case was 1(True) and the predicted is 0(False). in this case, the classifier also was not able to classify correctly the tested data.

2. Accuracy: It is the number of labels predicted correctly by the model over all predictions has been made. Accuracy is a good measure when the target variable classes in the data are nearly balanced.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.20}$$

3. Precision of the positive class: It gives us information about its performance with respect to false positives.

$$Precision = \frac{TP}{TP + FP} \tag{2.21}$$

4. Recall or Sensitivity of the positive class: It is clear that recall gives us information about a classifier's performance with respect to false negatives.

$$Recall = \frac{TP}{TP + FN} \tag{2.22}$$

5. F1 Score of the positive class:: It is a measure of a test's accuracy(weighted average of precision and recall). The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.23}$$

6. Area under the Roc curve: The AUC represents a model's ability to distinguish between positive and negative labels. If the area is 1.0, this means that the model predicted all the labels correctly. While an area of 0.5 signifies that the model was good as random.

Chapter 3

Problem Specification

The aim of this study was developing of an experimental case for Data Reply srl using Machine Learning techniques and following the analytical process for the Data Science (Problem definition, Data Discovery, Feature Engineering, Model Building, Test and Tuning) used within Data Reply. More specifically, I created an algorithm for image classification, that can classify images with trees and houses, using Convolutional Neural Networks (CNNs) and Deep Learning as Machine Learning techniques.Python was used as the development language.

Before creating the model, few steps should be done regarding the preparation of the dataset or the so-called Data Preprocessing.

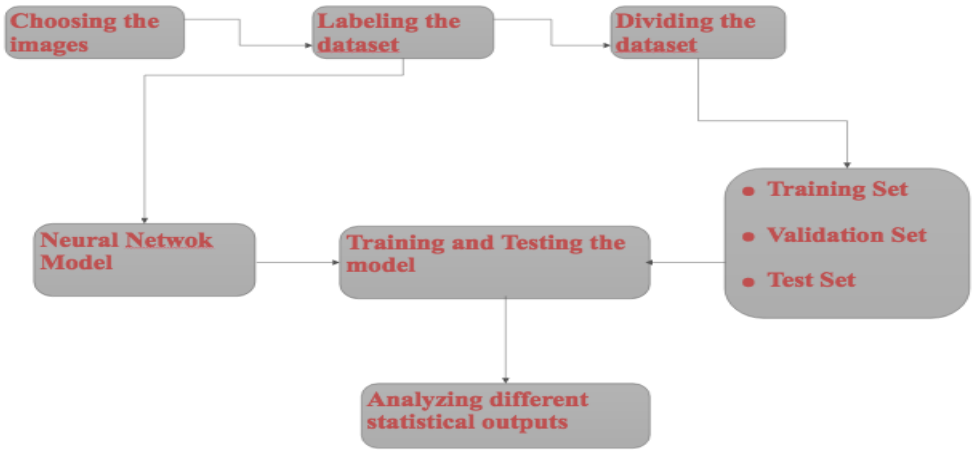


Figure 3.1: Blocks diagram describing the different steps for a Machine Learning model.

Step 1: **Preparation of the dataset**.

The original dataset was created by choosing 24 satellite images(dimension: 700×500 with RGB pixel format) from google maps. Figure 3.2 shows one of the images that were taken from different Italian comune like Bellagio, Stropino, Magreglio, Germignaga, etc.

Figure 3.2: Satellite image taken from comune of Bellagio in the Italian region of Lombardy.

Step 2: **Labelling the dataset**.
In order to assign a label to every image, I did the following steps:

- Coloring the original 24 images manually by 2 different colors: blue for the trees and red for the buildings.
- I generated 40.585 images of size 20×20 from the 24 original images by writing a Python script as follows: I defined a function "crop_offset" that takes as input one of the 24 images the height and the width of the new small images are parameters of this function in this case I chose them to be both 20. I chose an offset of 10 pixel from the left and the top side of the image and I did two nested loops in order to get square tiles of dimension 20×20.
- Create two masks for each image(a mask is a black image with white spots that indicates the position of the trees/houses in the image) by using openCV library(Open Source Computer Vision Library) available

in Python. The goal from creating these masks was to label the data set. So, I created a Python script as follows:

(a) I defined a function FindColor(image, color): That takes an image as input and searches for a specific color inside it and returns true if the color was found. In my case the color that we are searching for is the white color.

(b) I also defined another function ExtractMask(ColoredFolder, OriginalFolder, OutFolderHouse, OutFolderTree, LabeledFolder): This function takes as input the colored images located in the ColoredFolder generate the masks and transform them into black and white scale. Then, it reads the original images from the OriginalFolder and applies the black and white mask on each image creating 2 images for each single image one with only trees in white color and one with only the houses in white color also. I also defined a new variable maskIndex that takes 2 values 0/1 where 0 indicates images of houses and 1 for the images with trees.

(c) Applying the following condition: if maskIndex = 0 and the function FindColor returns "True" then the image's label is "CSi" else "CNo". While if the maskIndex = 1 and the function FindColor returns "True" then the image is labeled as "ASi" else "ANo".

- Every image in the dataset was labeled with one of the following 4 classes by applying the house and the trees masks on every image in the data set:
  - **CSi_ASi**: This class represents the presence of at least one building and one tree in the image.
  - **CNo_ANo**: This class represents the absence of both buildings and trees in a specific image.
  - **CNo_ASi**: This class indicates the presence of a tree and the absence of a building in the image.
  - **CSi_ANo**: This class indicates the presence of a building and the absence of a tree in the image.

Step 3: **Preparing the Training, validation and Test sets**.
I divided the dataset in 3 main sets using random sampling :

(a) **Training Set**: It consists of 10000 images, 2500 for each class.

(b) **Validation Set**: It consists of 3000 images, 750 for each class.

(c) **Test Set**: It consists of 2636 images distributed as follows:

- 685 images having the label CSi_ASi.

- 264 images having the label CNo_ANo.
- 712 images having the label CSi_ANo.
- 957 images having the label CNo_ASi.

Step 4: **Neural Network Model**.
Before creating the model we should prepare the input of the neural network in a specific way. Since my dataset is a set of images with 4 different labels, the input will be an n-dimensional array, that is, multidimensional container of items of the same type and size. In my case, the training n-dimensional array will have the following shape (10000, 20, 20, 3) where 10000 is the number of images in the set, (20, 20) is the height and the width of the every image. The fourth component of the array indicates the depth of color. In my case the depth is 3 which means that every pixel is associated to a vector of 3 elements(RGB). Their positions represent respectively the color levels: red, green and blue.

The labels will be transformed into a vector in which the value of each member can be [1, 0, 0, 0], [0,1, 0, 0], [0, 0,1, 0] or [0, 0, 0, 1], the position of the number 1 indicates the class label, that is, [1, 0, 0, 0] refers to the label CSi_ASi, while [0,1, 0, 0] refers to CNo_ANo, [0, 0,1, 0] indicates the class CSi_ANo and [0, 0, 0, 1] refers to CNo_ASi.

After preparing the training and the validation set in the way described above, I can introduce the Convolutional Neural Network model. This is the final model that I create after many trails. I did cross validation test in order to tune model's parameters, and I chose the ones that gave me best accuracy with no overfitting of the model. The optimizer I used for this model was the "Stochastic Gradient Descent", with learning rate $lr = 0.01$. I also fixed the number of epochs to 100(that is the model will train for 100 epochs) while the batch size to 32, which means that the model will train on mini-batches of 32 examples at each step and this requires less memory. Since you train network using less number of samples the overall training procedure requires less memory. Note that, usually the same architecture is used on the training and validation sets. In the following chapter I will discuss the results of different optimizers I tried.

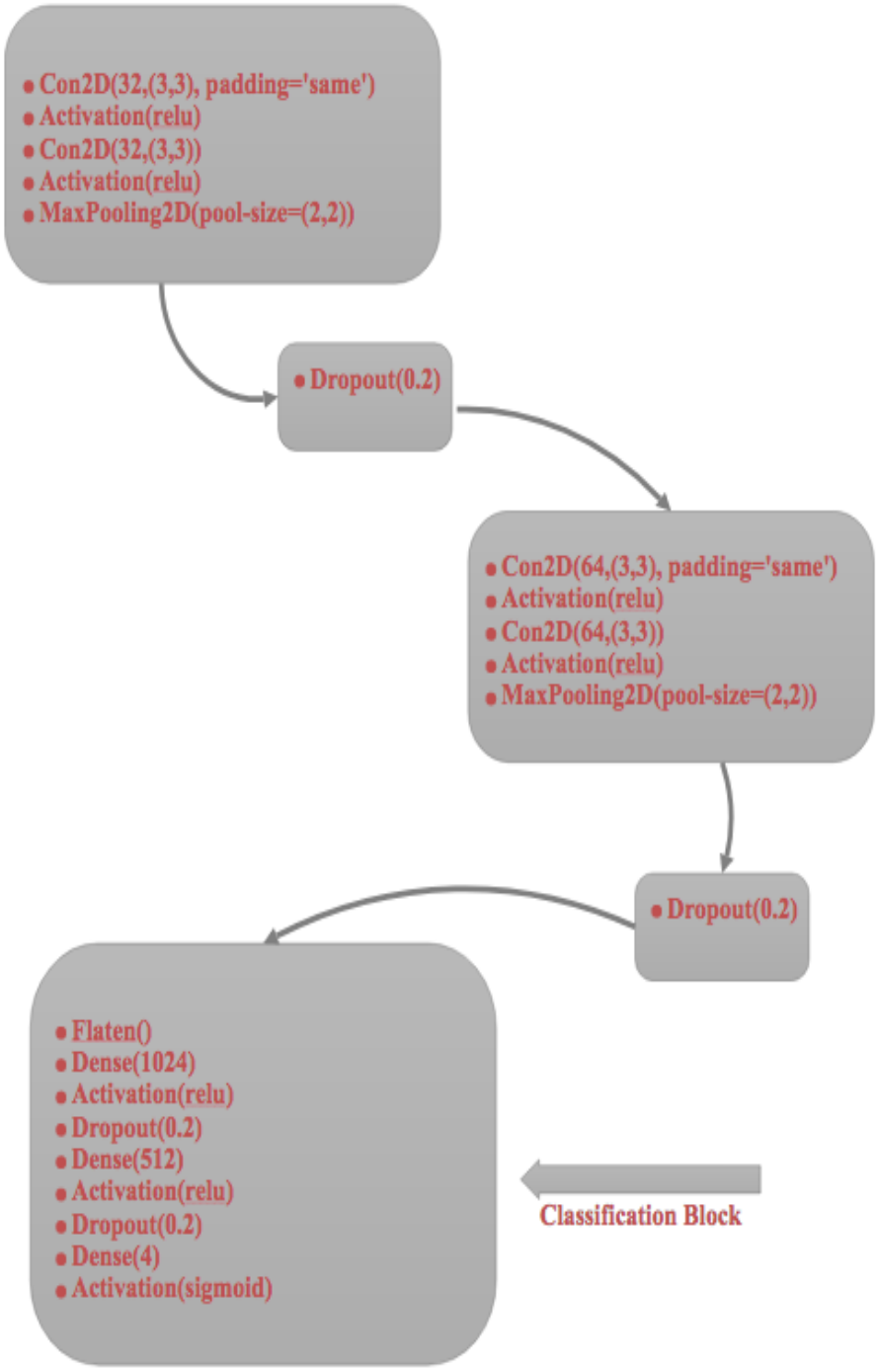


Figure 3.3: The Model's Architecture.

The parameters of the Convolutional Layer are related to number of filters, kernel size and padding. That is, for the first layer Conv2D(32, (3,3), padding= ' same') the number of filters is 32, the kernel size is 3×3 pixel subregions, and to specify that the output tensor should have the same width and height values as the input ones, we set padding= 'same'(which instructs TensorFlow to add 0 values to the edges of the output tensor to preserve width and height of image size). The output tensor produced by conv2d(), has the same dimensions as the input ones, but now with 32 channels holding the output from each of the filters. After each convolutional layer, it is convention to apply a nonlinear layer, specifies the activation function to apply to the output of the convolution that is, the "Relu" function(rectified linear unit). The MaxPooling2D(pool_size=(2,2)) is a filter with dimension 2×2 pixel. Dropout(0.2) it is throws away 20% of the data randomly during training to prevent outfitting. Dense(1024) is the fully connected 1024 node neural network(one-dimensional). Dense(4) is the classification layer with 4 nodes representing the 4 different classes, together with the Activation('sigmoid') layer both layers give us the output vector with the probability of each class in every image.

The motivation that stands behind choosing this architecture was the Convolutional Neural Network for CIFAR-10 dataset. This problem was developed by the Canadian Institute for Advanced Research(CIFAR). The dataset consists of 60,000 image of 10 different classes of objects such as, birds, cats, horses, automobiles, airplanes and so on. All the images were 32 by 32 pixel squares in the RGB scale. One of the architectures that was used for training and validation consists from:

1. Convolutional input layer, 32 feature maps with a size of 3×3, a rectifier activation function .

2. Dropout set to 20%.

3. Convolutional layer, 32 feature maps with a size of 3×3, a rectifier activation function.

4. Max Pool layer with size 2×2.

5. Flatten layer.

6. Fully connected layer with 512 units and a rectifier activation function.

7. Dropout set to 50%.

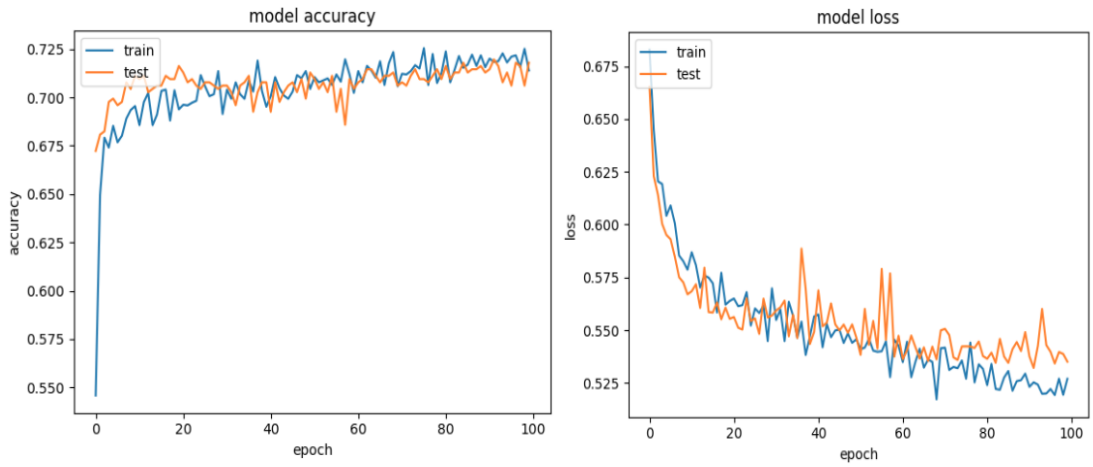
8. Fully connected output layer with 10 units and a softmax activation function.

They used as optimizer the stochastic gradient descent with learning rate 0.01. The accuracy was 70.85%. I started my work trying this architecture but I did not got good results so I modified it by adding another layers and I used cross validation test for tuning the parameters. In the next chapter I will discuss the results of applying this model on my dataset. Also I will show some statistical outputs of my model, and I will compare them with the results of other models with different parameters.

Chapter 4

Results

In the previous chapter I introduced my model's architecture. Now I will discuss the performance of this architecture and I will compare it with the results of different architectures that I tried.
I first applied the same architecture with the same parameters used for the CIFAR-10 dataset, introduced in the pervious chapter, I got a validation accuracy of 72% and after only 100 epochs. We can see that there is a considerable difference between the training and validation loss(see figure 4.1). This indicates that the network has tried to memorize the training data and thus, is able to get better accuracy on it. This is a sign of Overfitting. Usually it is required that the accuracy and the loss of both training and validation sets should behave in the same manner and after a given number of epochs they should behave as asymptotics.



(a) Validation accuracy vs Training accuracy.

(b) Validation loss vs Training loss.

Figure 4.1: Applying the same architecture used for CIFAR-10 on my dataset.

I applied the same architecture but with different learning rate for the Stochastic Gradient Descent optimizer and I got the following result after 50 epochs only:

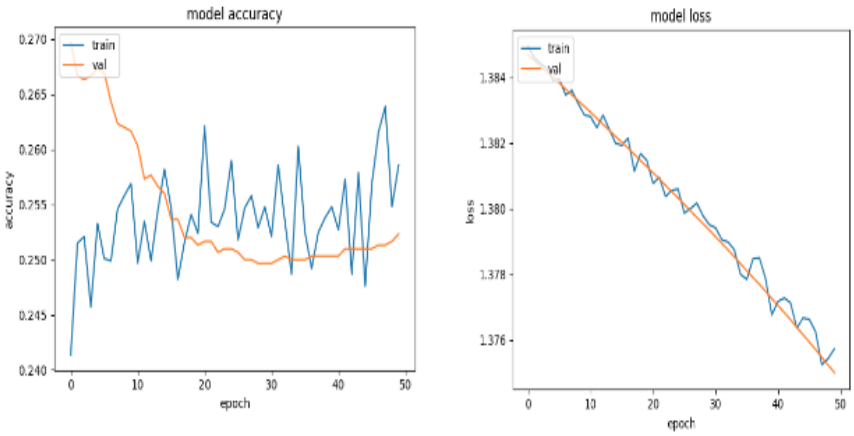


Figure 4.2: Cifar-10 architecture with learning rate = 0.0001.

We can notice the overfitting. I also changed the optimizer and I used "ADAM optimizer" with different learning rates and I got the same problem of overfitting.

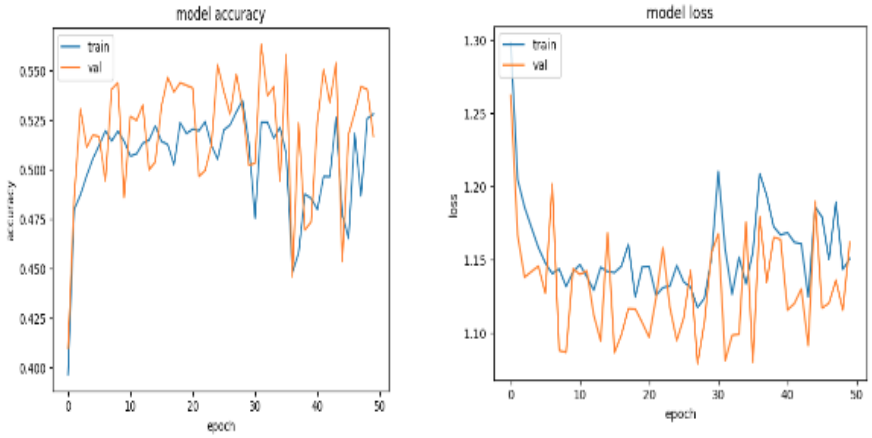


Figure 4.3: Cifar-10 architecture with ADAM optimizer and learning rate = 0.01.

After these trails, I wrote a Python script where I defined the cross validation test to choose the best parameters, architecture and layers. Table 4.1 shows the layers of my model and their parameters with learning rate $lr = 0.01$ and as optimizer I used "Stochastic Gradient Descent":

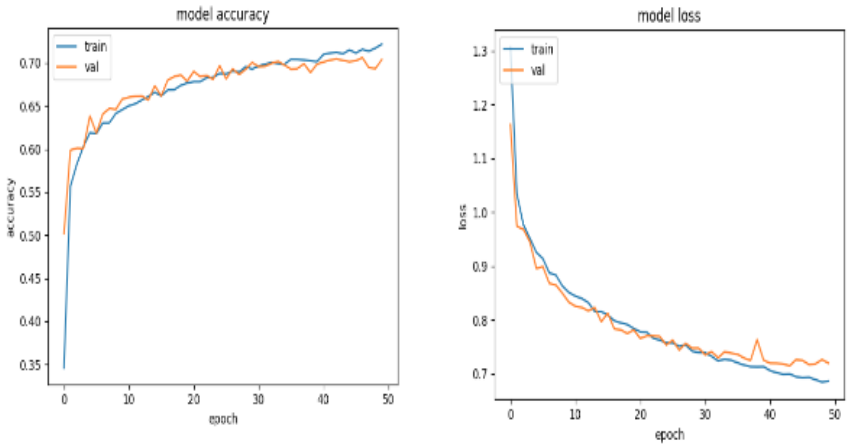


Figure 4.4: Cifar-10 architecture with ADAM optimizer and learning rate = 0.0001.

Layer	**Parameters**
CN_1 (Conv2D)	(32,(3,3))
activation_1	relu
conv2d_2 (Conv2D)	(32,(3,3))
activation_2	relu
max_pooling2d_1 (MaxPooling2D)	(2,2))
dropout_1	(0.2)
conv2d_3 (Conv2D)	(64,(3,3))
activation_3	relu
conv2d_4 (Conv2D)	(64,(3,3))
activation_4	relu
max_pooling2d_2 (MaxPooling2D)	2,2))
dropout_2	0.2
dense_1 (Dense)	1024
activation_4	relu
dropout_3	0.2
dense_2 (Dense)	512
dropout_4	0.2
dense_3 (Dense)	4
activation_5	sigmoid

Table 4.1: Model's Architecture

This model has a validation accuracy of 70% at the last epoch. We can notice from Figure 4.5 how the behavior of both accuracies and both losses are synchronized. So there is no overfitting or under-fitting.

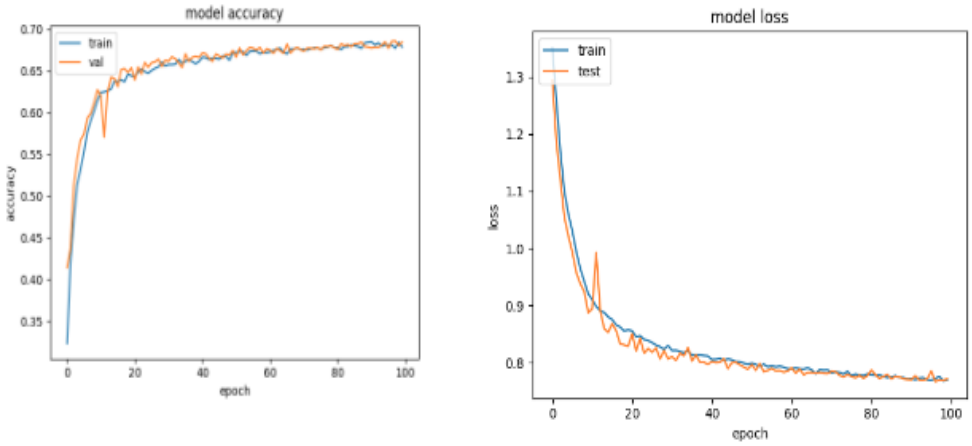


Figure 4.5: Model's loss and accuracy.

Table 4.2 shows the classification report of the classes that were predicted by the model. We can notice that the trees were recognized better than the houses since the precision of the class CNo_ASi(i.e., the presence of a tree and the absence of a house in the image) is 0.88 while the precision of CSi_ANo(i.e., the presence of a house and absence of a tree) is 0.77. Also the model performed good when both houses and trees were present in the same image since the precision of class CSi_ASi is 0.73. Note that the support column refers to the total number of tested images for each class.

Label	**Precision**	**Recall**	**F1-Score**	**Support**
CNo_ANo	0.58	0.55	0.56	264
CSi_ANo	0.77	0.65	0.71	712
CNo_ASi	0.88	0.65	0.71	975
CSi_ASi	0.73	0.39	0.51	685
Total	0.78	0.50	0.60	2636

Table 4.2: Confusion Matrix

Note that in the case of class CNo_ANo and class CSi_ASi the $F1$ score is 0.56 and 0.51 respectively. This can be explained since the F1 score is the weighted average of the precision and recall, where its best value is 1. Where the F1 score can be calculated by the following equation:

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.1}$$

When both Recall and Precision are low or their is a significant difference between

them then the F1 score will be low as a consequence. Also the low value of F1 score in these two cases is related to the fact that the true positive cases for the classes CNo_ANo and CSi_ASi where very few. In other words, the algorithm misclassified the correct labels of these two classes.
By this result my goal was achieved. I was able to create and train a convolutional neural network model that can distinguish between houses and trees with an accuracy of 70%.

Now I will discuss an additional work that I also developed regarding the transfer Learning. Transfer learning is a machine learning method, which is popular in deep learning where the pre-trained models are used as a starting point to train a new model. It is common to perform transfer learning with predictive modeling problems that use image data as input, for this reason I applied transfer learning on my dataset. The purpose from using transfer learning is that training from scratch requires a lot of labeled training data and a lot of computing power. So we can retrain our dataset on a very large dataset for example, ImageNet that contains 1.2 million images with 1000 classes with images that varies in dimensions and resolutions.

Transfer learning works as follows: in the first phase it analyzes all the images on disk and calculates and caches the bottleneck values for each of them. 'Bottleneck' is an informal term that is usually use for the layer just before the final output layer that actually does the classification. This layer has been trained to output a set of values that's good enough for the classifier to use to distinguish between all the classes it's been asked to recognize. The reason that the final layer retraining can work on new classes is that it turns out the kind of information needed to distinguish between all the 1,000 classes. Inception and MobileNet are two architectures that were trained on the dataset ImageNet.

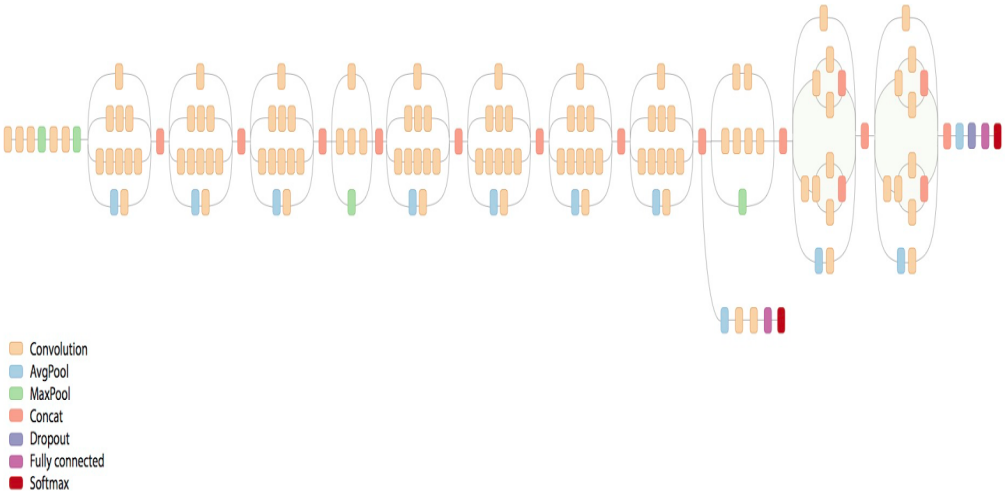


Figure 4.6: Inception architecture.

Applying these 2 pre-trained models on my dataset, I got the following results:

	Training Accuracy	**Validation Accuracy**	**Time**
Inception	65%	60%	2 hr
MobileNet	50%	52%	30 min

Table 4.3: Results of transfer learning

These results for transfer learning were expected since the size of the images in my dataset is too small comparing to those of the ImageNet dataset.

Type / Stride	**Filter Shape**	**Input Size**
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw/ s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5×Conv dw / s1 5×Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
Pc/ s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1 \times 100$

Table 4.4: MobileNet body architecture

Chapter 5

Conclusions

One of the main classes of artificial intelligence is machine learning. The aim of machine learning is enabling machines to perform jobs using intelligent softwares that are usually developed by using statistical learning methods. Machine learning needs a big size of data in order for the algorithm to be trained and tested. It focuses on prediction, based on known properties learned from the training data.
Machine Learning is divided into two main classes: Supervised and Unsupervised Learning. In Supervised learning, the goal is to search and memorize features from the labeled training data that is usually consists from a vector X with a vector Y of labels. It is called Supervised learning, since these labels are often provided by the supervisor, that is humans. Algorithms used for supervised learning can be grouped in two main categories: Regression and Classification. In the second chapter I covered some of these algorithms such as:

1. Logistic Regression.
2. Support Vector Machines.
3. Bayesian Netwroks.
4. Decision Tree.
5. Neural Networks and Deep Learning.

In the case of Unsupervised Learning, all the data that we have is unlabeled data and the idea is to find a hidden pattern from this data. Classical examples include principal components analysis and cluster analysis. Unsupervised learning is very useful in exploratory analysis because it can automatically identify structure in data.

Machine learning and supervised learning algorithms have many applications nowadays. Here are few examples of machine learning that we use everyday without knowing that they are driven from it:

1. Traffic Predictions using GPS navigations services such as google maps.
2. Classification of emails as spam or no.
3. Fraud Detection.
4. Virtual Personal Assistants such as Siri and Google Now.
5. Social Media Services such as Face Recognition and People You May Know by Facebook.

These applications use different machine learning algorithms, as for emails classification usually they use Nayive Bayesian algorithms. Neural Networks and Deep learning they are used usually for image classification task and so on.... In my work I used Neural Networks algorithm and specifically Convolutional Neural Networks in order to classify satellite images of size 20×20 taken from google maps. The goal was to predict the presence or not of a house/tree in the examined image. As I explained in chapter 3, the dataset was divided in 3 sets: training, validation and test. Where the training and validation sets were labeled according to one of the 4 classes: CSi_ANo, CNo_ASi, CSi_ASi and CNo_ANo, such that "Si"/"No" indicates if a house/tree is present in the image. The model's architecture was defined as follows:

- **Block 1**
  1. Convolutional layer(32,(3,3), activation='relu')
  2. Convolutional layer(32,(3,3), activation='relu')
  3. MaxPooling(2,2)
  4. Dropout(0.2)
- **Block 2**
  1. Convolutional layer(64,(3,3), activation='relu')
  2. Convolutional layer(64,(3,3), activation='relu')
  3. MaxPooling(2,2)
  4. Dropout(0.2)
- **Classification Block**
  1. Flaten()
  2. Dense(1024)
  3. Activation (Relu)

4. Dropout(0.2)
5. Dense(512)
6. Activation (Relu)
7. Dropout(0.2)
8. Dense(4)
9. Activation(Sigmoid)

The optimizer I used was the "Stochastic Gradient Descent", with learning rate lr = 0.01 and the model's accuracy was 70%.
The limitation of this work can be summarized in two main points:

- A computer with GPU(Graphics Processing Unit) was not available. Usually, the training process is done using computers with GPU, since the training phase is too much faster.
- The original images were 24 of size 700×500 and in order to have a big dataset I was restricted to generate very small images of size 20×20.

My goals in the future is to use bigger images(for example 100×100) in order to optimize the performance of the algorithm and as a consequence having a higher accuracy. Also, I would like to apply different neural networks architectures as AlexNet and VGGNet because of there good performance in image classification.

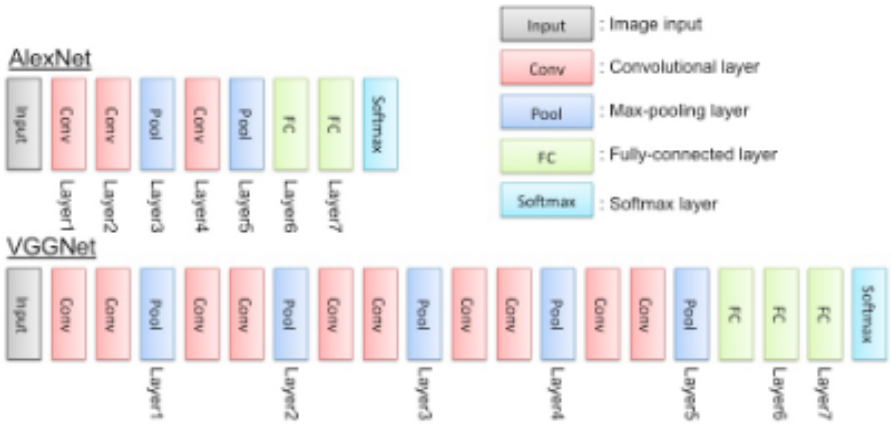


Figure 5.1: AlexNet and VGGNet architectures. Source: Hirokatsu Kataoka, Kenji Iwata and Yutaka Satoh, "Feature Evaluation of Deep Convolutional Neural Networks for Object Recognition and Detection".

Bibliography

[1] E. Alpaydin, *Introduction to Machine Learning*, 2^{nd} Edition, Massachusetts Institute of Technology.

[2] R. Brooks, *Understanding the 3 Categories of Machine Learning ? AI vs. Machine Learning vs. Data Mining 101.* https://guavus.com/ai-vs-machine-learing-vs-data-mining-whats-big-difference-part-2/, October 17, 2017.

[3] C. J. Burges, C. Cortes and Y. LeCun, *The MNIST Database of Handwritten Digits*. http://yann.lecun.com/exdb/mnist/.

[4] K.W. Chang, Chung.C. Hsu, Y.P. Huang, *Extended Naïve Bayes Classifier for Mixed Data*, Elsevier Ltd. 2008.

[5] J. Cheng and R. Greiner, *Learning Bayesian Belief Network Classifiers: Algorithms and System*, Canadian Conference on AI 2001.

[6] D.D. Cox, T. Dean, *Neural Networks and Neuroscience Inspired Computer Vision*, Current Biology, September 22, 2014.

[7] S.B. David and S.S. Shwartz. *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press 2014.

[8] L. Deng and D. Yu, *Deep Learning Methods and Applications*, Foundations and Trends in Signal Processing, 2014.

[9] T. Fawcett and D. Hardin, *Machine Learning vs. Statistics*. https://svds.com/machine-learning-vs-statistics/, August 10th, 2017.

[10] S. Haykin, *Neural Networks and Learning Machines*, 3^{rd} Edition, Pearson Education(US) 2009.

[11] Y. Hwanjo and K. Sungchul, *SVM Tutorial - Classification, Regression and Ranking*, Springer, Berlin, Heidelberg 2012.

[12] J. Jayanth, A.P. Pooja and S. Koliwad, *Classification of Rs Data Using Decision Tree Approach*, International Journal of Computer Applications, June 2011.

[13] S.B. Kotsiants, *Supervised Machine Learning: A Review of Classification Techniques*, Informatica 31 (2007) 249-268, July 16, 2007.

[14] D. Kriesel, *A Brief Introduction to Neural Networks*, http://dkriesel.com , 2007.

[15] Z. Reitermanova, *Data Splitting*, WDS'10 Proceedings of contributed papers, Mathematics and Computer Sciences, 2010.

[16] J. Schmidhuber, *Deep Learning*, Scholarpedia 2015.

[17] T.P. Shah and K.S. Parikh, *Support Vector Machine- a Large Margin Classifier to Diagnose Skin Illnesses*, Elsevier Ltd. 2016.

[18] D.F. Team, *Kernel Functions– Introduction to SVM Kernel & Examples*, https://data-flair.training/blogs/svm-kernel-functions/, August 12, 2017.