الجمهورية الجزائرية الديمقراطية الشعبية
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزارة التعليم العالي والبحث العلمي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جامعة عمار ثليجي الأغواط
UNIVERSITY OF LAGHOUAT

FACULTY OF SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Field : Mathematics and Computer Sciences
Option : Computer Science
Specialization : Information Systems and Decision-Making.

Memoire Submitted in Partial fulfillment of the requirements for the Master degree in Computer Science

Submitted by: **Brahim Yagoubi & Mohamed Taher Kenniche**

THEME

Sub-Saharan Sheep Breeds classification and generation based on CNN and GANs

Jury members:

Mr	Mohamed Elhabib Maicha	MC(B)	(University of Laghouat)	President
Mr	Lakhdar Kamal Ouladdjedid	MC(B)	(University of Laghouat)	Examiner
Mr	Ali Boukhila	MC(B)	(University of Laghouat)	Examiner
Mr	Younes Guellouma	MC(A)	(University of Laghouat)	Advisor

2023

Acknowledgments

First, we would like to express our deepest gratitude to Allah who has always given us the support and guidance that have enabled us to complete this work. All our life Allah was with us and gave me strength and blessings. We are deeply grateful to our advisor, **Mr.**

Younes Guellouma for his unwavering support and guidance throughout our master's program. His expertise and patience have been invaluable to us and have played a crucial role in the success of this thesis.

We are deeply grateful to our friends and family for their love and support throughout this process. Without their encouragement and motivation, we would not have been able to complete this course.

We would also like to thank our colleagues for their support and collaboration during the research. In particular, we would like to thank **Ali Belboul** & **Abdelatif Hamdi**. Finally, we are grateful to everyone who has supported us throughout this process.Without

your help and guidance, this thesis would not have been possible.

Brahim & Mohamed Taher, June 2023

Dedications

“

I offer my thesis dedication to my beloved family and the numerous friends who have supported me throughout my academic journey. I hold a deep and profound sense of gratitude towards my loving parents, who have been instrumental in shaping my academic and personal growth,

My brother and sister Hicham and yasmine who have never ceased to stand by my side, hold a truly special place in my heart,

I dedicate this work and give special thanks To my cherished friends, who have brought joy, laughter, and unwavering support into my life. Specially my friends in the university dorm,

In addition, my friends from the wilaya of Laghouat My deepest thanks go to the friends who have been a great support to me on my way. They selflessly opened their homes and offered me unwavering support whenever I needed it. As I write this dedication, words cannot fully convey the depth of my feelings and tears well up in my eyes. My heart swells with appreciation, but my tongue fails to express it all,

À Thank you all.

”

- Brahim

Dedications

“

I would like to express my heartfelt dedication to my beloved family and the countless friends who have supported me throughout my academic journey. I am deeply grateful to my loving parents and sisters who have played a crucial role in shaping both my academic and personal growth,

I dedicate this work with special thanks to my dear friends, who have brought joy, laughter, and unwavering support into my life. I am especially grateful to my friends from the university dormitory for the memorable moments we shared. Specially with my teammate Brahim ,

I would like to extend my dedication to my beloved teachers, who have not only guided me through my courses but have also treated me with immense care and support, treating me as a son rather than just a student. I am deeply grateful for their mentorship and the invaluable knowledge and wisdom they have imparted to me. This work is dedicated to them as a token of my appreciation and respect,

Thank you all from the bottom of my heart,

”

- Mohamed Taher

ملخص

نبحث في هذه الأطروحة تطبيق تقنيات "التعلم العميق" لتحديد سلالات الأغنام، الطرق التقليدية لتحديد سلالات الأغنام تستغرق وقتا طويلا وهي كذلك معرضة للأخطاء.

لمواجهة هذه المشكلة، قمنا بتطوير نموذج تعلم عميق قوي باستخدام الشبكات العصبية التلافيفية (CNNs) و "زيادة البيانات" وشبكات الخصومة التوليدية. (GANs)

استخدمنا مجموعة بيانات صور الأغنام الخاصة بنا لتدريب النموذج وضبطه.

نقارن أداء إجراء "التعلم العميق" الخاص بنا بالطرق التقليدية، مع مراعاة عوامل مثل جودة الصور، تظهر النتائج قدرة التعلم العميق على التعرف على سلالات الأغنام، والتي لها فوائد لتقنيات الثروة الحيوانية والزراعة.

كلمات مفتاحية :
الشبكات العصبية التلافيفية ،'CNN' التعلم العميق، شبكات الخصومة التوليدية ،'GAN' سلالة الأغنام.

Abstract

This thesis examines the application of "deep learning" techniques for computerized sheep breed identification. Traditional methods for identifying sheep breeds are Time-consuming and prone to errors.

To confront this problem, we develop a robust deep learning model using Convolutional Neural Networks (CNN[1]s), "data-augmentation" and Generative Adversarial Networks (GAN[2]s).

Our dataset of sheep images is used for training and adjusting the models.

We compare the performance of our "deep learning" Procedure to traditional methods, taking into account factors such as image quality. The results show the Capability of deep learning to precisely recognize sheep breeds, which has benefits for livestock and Farming techniques.

Key words : CNN, deep learning, GAN, sheep breed.

---

[1] *Convolutional Neural Networks*
[2] *Generative Adversarial Networks*

Résumé

Cette thèse examine l'application des techniques de « deep learning » pour l'informatique identification de la race ovine. Les méthodes traditionnelles d'identification des races de moutons sont Time- consommatrice et sujette aux erreurs.

Pour faire face à ce problème, nous développons un modèle d'apprentissage profond robuste utilisant la Convolution Réseaux de Neurones (CNN), 'data-augmentation' et les Réseau Antagoniste Génératif (GAN).

Notre ensemble de données d'images de moutons est utilisé pour la formation et l'ajustement des modèles.

Nous comparons les performances de notre procédure « deep learning » aux méthodes traditionnelles, en tenant compte de facteurs tels que la qualité de l'image. Les résultats montrent la Capacité d'apprentissage en profondeur pour reconnaître avec précision les races de moutons, ce qui présente des avantages pour élevage et techniques agricoles.

Keywords : CNN, deep learning, GAN, race ovine

Contents

List of Figures

List of Tables

List of Algorithms

Liste des sigles et acronymes

CNN	*Convolutional Neural Networks*
GAN	*Generative Adversarial Networks*
AI	*Artificial Intelligence*
ANN	*Artificial Neural Network*
SNN	*Simulated Neural Network*
LSTM	*Long Short-Term Memory*
ReLu	*Rectified Linear Unit*
tanh	*Hyperbolic Tangent*
MSE	*Mean Squared Error*
SGD	*Stochastic Gradient Descent*
ANN	*Artificial Neural Network*
AE	*Autoencoders*
RNN	*Recurrent Neural Network*

Introduction

Context

Farmers have historically engaged in sheep farming for their prized products such as wool and meat. However, distinguishing between breeds can be difficult for inexperienced breeders or for younger animals. Accurate breed identification is an essential part of successful breeding programs, record-keeping, and business transactions. Previous methods that relied on visual recognition are prone to inaccuracy due to personal bias or limited knowledge and represent a time-consuming process. Artificial intelligence (AI[3]) offers the opportunity to improve this process by providing a faster and more accurate method for unbiased racial identification provide. Our article explores the potential of artificial intelligence in recognizing different breeds of sheep, presenting both its benefits and potential limitations.

Problematic

Successful identification of sheep breeds is vital for sheep farms worldwide and for successful sheep farming. Today, more than a billion sheep are raised in Algeria, providing essential products such as wool and meat. However, accurate information about the genetics of these creatures requires solid observations to properly manage their breeding populations. The main challenges in using AI to identify sheep breeds in Algeria are related to the availability of the data-set and the complexity involved in sheep crossing. While documenting sheep farms through photography, breed identification becomes a daunting task. Even the farmers themselves may not have extensive knowledge of the breed. However, there are possible solutions to these problems.

[3] *Artificial Intelligence*

Proposition

Problems with traditional identification techniques can be solved by using artificial intelligence to identify sheep breeds. The AI can learn to recognize the distinctive traits of each breed and correctly distinguish them from other breeds by analyzing huge databases of sheep photos and genetic information using machine-learning methods. To overcome the challenges of limited data sets and hybridization, we use different techniques and tools to create an efficient model. We start by using social media platforms like YouTube and Facebook to collect different breed images. We then use tools to enhance and refine the quality of these images and ensure they are clear and well-defined. Despite these efforts, we recognize the need for more data, which has led us to explore techniques to increase the amount of data. However, we recognize that even expansion may not be enough. In response, we take a different approach, using generative adversarial networks (GANs) to generate new synthetic images. This extension strategy allows us to extend the data set and improve the accuracy and robustness of the model.

The remainder of this document is organized as follows. The first chapter introduces Local sheep races with some useful detail. The next chapter is dedicated to necessary machine deep learning techniques. An up-to-date state-of-the-art is presented in Chapter 3. Finally, the deployed model is presented and discussed in Chapter 4.

Chapter 1

Sub-Saharan and north-African sheep races

1.1 Introduction

Among the animal resources that provide meat products in In Algeria, sheep are the most popular. Despite the decline in domestic production, it is also important for its wool. The Algerian Aries is more than an economic and commercial asset, it has accompanied Algerian finances for a long time. Indeed, his image appeared on several coins and banknotes. Unfortunately, it is evident that some breeds, real heirs, are hitting the abyss today. Algeria has a lot of sheep from Sub-Saharan and north-African breeds like Ouled Djellal, Hamra, D'man, Sidaoun, Sardi, Rembi, and others. Figure 1.1 shows the distribution of sheep breeds in Algeria.

1.2 Information about some breeds

1.2.1 Ouled Djellal

History The breed depicted in Figure 1.2 was brought to Algeria in the 11th century by the Beni-Hilal, who migrated from the Hijaz region in Arabia through Upper Egypt during the rule of the Fatimid Khalifa. In the Middle East and Asia, it is widely recognized that all sheep breeds have fat tails. Consequently, according to Trouette, the Ouled Djellal breed, which possesses a thin tail and fine wool, was introduced by the Romans, who had a strong affinity for wool, in the fifth century. They brought this type of sheep from Taranto, Italy, where such sheep still exist today. The presence of this breed is also evident

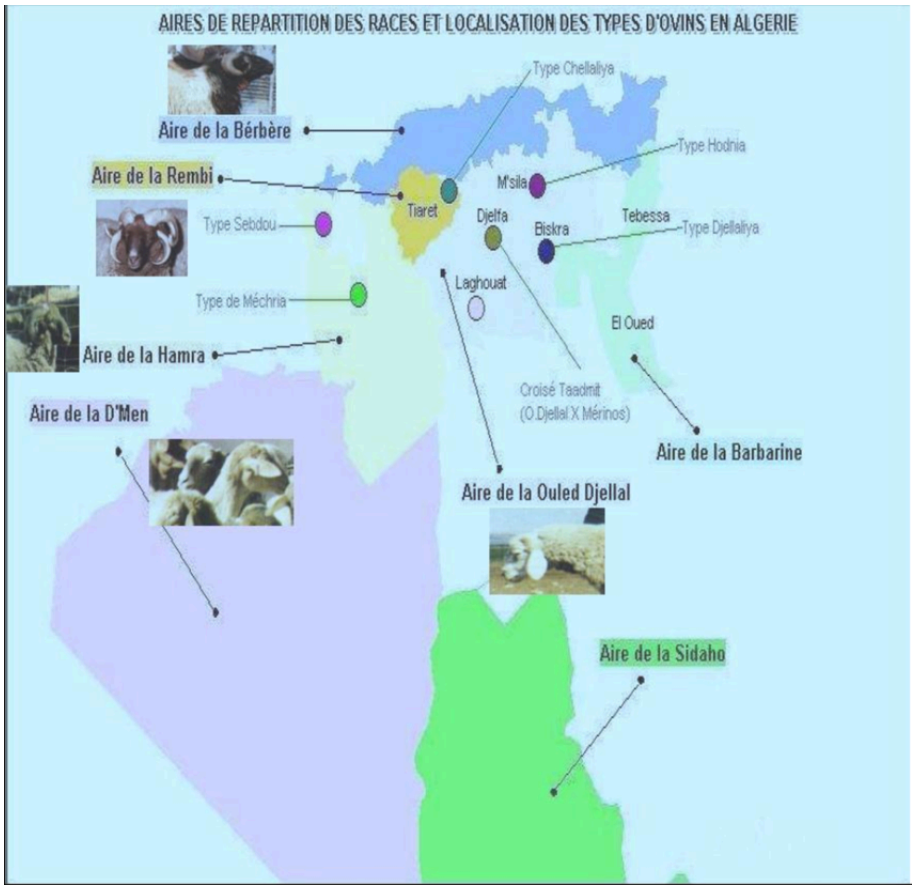

Figure 1.1: The distribution of sheep breed in Algeria

in the funerary steles found in the ruins of Timgad (Batna).

The term "Arabic" refers to the geographical region where the majority of Arabic-speaking shepherds reside and was not introduced by the Arabs, specifically the Béni-Hillal. The sheep population in the steppes can be traced back to the period after the Roman occupation but before the Arab conquest. It is closely associated with the Zenetes invasions and the rise of extensive nomadic practices, which emerged with the arrival of dromedaries in North Africa [12].

Description The Ouled Djellal breed has seen widespread distribution across the tell, steppe, and northern regions of the Sahara. It is also present in Tunisia, where it is known as the "Bergui" or "fine tail of the West" breed. This breed is primarily raised in an extensive farming system. Renowned for its meat quality, it is considered the premier meat breed in Algeria. The wool of Ouled Djellal sheep is white, and their heads are also white in color. Male rams possess medium-sized spiral horns, while ewes typically do not have horns, except for a sub-breed known as the Djellalia variety. The average live weight of these animals ranges from 80 to 140 kg, with males measuring approximately 96.32±8.95cm in height and females measuring around 85.02±5.79cm [12].

Figure 1.2: Ouled Djellal breed

1.2.2 Rembi

History The Rembi breed, also known as "Sagâa" in the Tiaret region, has a rich historical background. It was once prevalent throughout the steppe regions, spanning from the eastern to the western parts of the country. This breed exhibits superior adaptability to both the steppe and mountainous terrains, surpassing the Ouled Djellal breed in terms of hardiness. The Rembi sheep thrives particularly well in the Ouarsenis regions and the Tiaret mountains. It originated from a crossbreeding between the mouflon of Djebel Amour, also referred to as "Laroui," and the Ouled Djellal breed. As a result, it inherits the physical characteristics of the Ouled Djellal, while also possessing the impressive horns and coloration of the mouflon. The Rembi breed demonstrates exceptional resilience and productivity, making it highly recommended for improving the grazing conditions of impoverished mountainous pastures [12].

Description The Rembi sheep (see figure 1.3) shares most of its morphological traits with the Ouled-Djellal breed. However, there are slight differences in its appearance. The Rembi sheep has a dorsal line that is slightly more curved, and its limbs and head exhibit a fawn or slightly greyish coloration. The breed also possesses medium-sized, drooping ears. The wool of Rembi sheep is white and covers the entire body, extending down to the knees and hocks. Male rams feature voluminous, spiraling horns, while ewes may have backward-loping horns [12].

Figure 1.3: Rumbi breed

1.2.3 Hemra (Deghma)

History The Hamra breed called “Deghma” (see figure 1.4) is indigenous to Algeria. However, in Morocco, particularly in the Moroccan High Atlas region, it is known as ”Beni-Ighil” due to its association with the Beni-Ighil tribe. In Algeria, it is commonly recognized as the ”Deghma” breed, primarily owing to its distinctive dark red coloration [12].

Description The Hamra breed is characterized by its medium-sized stature. It bears a resemblance to the Moroccan Béni-Iguil breed, suggesting a common origin. The animals have brown skin, black mucous membranes, black claws, and a blue tongue. Their wool is predominantly white, with spirals often displaying streaks of black. In males, the horns are of medium size, while females may have a lumpy appearance. The Hamra breed possesses an ideal meat sheep conformation and displays remarkable finesse in its frame.

Due to its exceptional organoleptic qualities, the breed gained preference over other breeds in the French market, where it was known as the Oranie sheep. These qualities contribute to the overall sensory experience of the meat, making it particularly appealing [12].

Figure 1.4: Hamra breed

1.2.4 Sardi

History The Sardi breed of sheep, depicted in Figure 1.5, originates from the island of Sardinia, Italy. While it is not a native breed of Morocco, it was introduced to the country for its favorable qualities in meat and milk production. The initial introduction of Sardi breed animals to Morocco took place in the 1970s, primarily with a focus on enhancing meat production. Subsequently, the breed has been crossed with local sheep breeds in order to improve overall performance in terms of meat and milk production [1].

Description The Sardi breed of sheep is known for its distinct features. It has a white head with a black muzzle, along with black spots around the eyes, which has led to its nickname as the ”race with glasses.” Male individuals have a hooked profile, and their broad skulls bear white and robust horns. In contrast, female sheep have an almost straight muzzle, and their heads are hornless. The legs of Sardi breed sheep are white and devoid of wool, with black dots at the ends and the point of the hocks. These sheep are characterized by their large size, emphasizing their robust stature [4].

Figure 1.5: Sardi breed

1.3 Conclusion

In summary, Algeria boasts a diverse array of sheep breeds, each tailored to thrive in specific environmental conditions. These breeds contribute significantly to the country's agricultural sector. With variations in coat color, size, milk production, meat quality, and regional suitability, these breeds offer a wide range of characteristics. In the following chapter, we will delve into valuable explanations related to deep learning, providing useful insights on the topic.

Chapter 2

Deep Learning

2.1 Introduction

As deep learning becomes more popular, researchers and developers are introducing innovative approaches to implementation. Each implementation may have its own interpretation of deep learning, but it generally refers to computer systems that have the ability to think logically and solve problems on their own. With so many implementations available, it can be difficult to keep track of them all. Even within deep learning, there are certain sub-areas that should be given more attention in the future. The goal of this chapter is to provide an overview and breakdown of these important subfields of neural networks and deep learning, as well as the basic concepts needed to actively participate in this field.

Deep learning applications are used in industries from automated driving to medical devices.

- **Automated Driving :** Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.
- **Aerospace and Defense :** Deep learning is used to identify objects from satellites that locate areas of interest and identify safe or unsafe zones for troops.
- **Medical Research :** Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.
- **Industrial Automation :** Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are

within an unsafe distance of machines.

- **Electronics :** Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications

2.2 Machine Learning

Deep learning, as a sub-field of machine learning, encompasses various paradigms and principles that are fundamental to its functioning. In order to gain a better understanding of deep learning, it is essential to explore these key paradigms. The following are some of the important notions in machine learning.

There are three learning process types :

2.2.1 Supervised learning

Supervised learning, a subcategory of machine learning and artificial intelligence, utilizes labeled datasets to train algorithms that accurately classify data or predict outcomes. The process involves feeding input data into the model and adjusting its weights until the model is appropriately fitted. This fitting occurs as part of the cross-validation process, ensuring the algorithm's effectiveness (refer to Figure 2.1) [34].

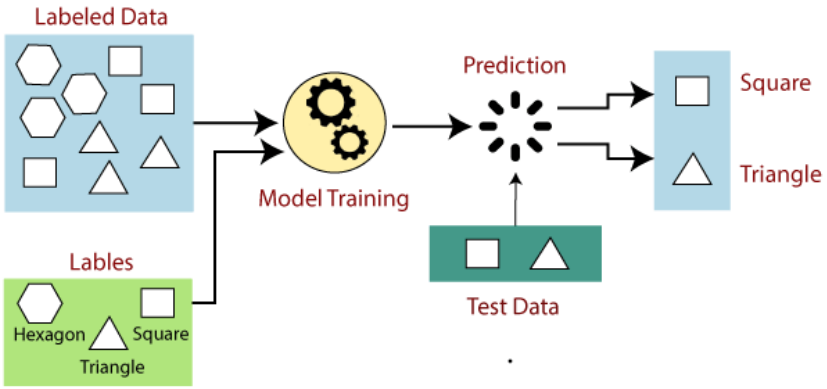

Figure 2.1: Scheme example on supervised learning [25]

Supervised learning is split into two categories of algorithms :

- **Classification** where the output variable is a category, like 'Red' or 'blue', 'human' or 'animal'.

- **Regression** where the output variable is a real value, like 'dollars' or 'grams'. Like any learning process, supervised learning has both advantages and disadvantages :

2.2.2 Unsupervised learning

Unsupervised learning, also referred to as unsupervised machine learning, employs machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms autonomously uncover hidden patterns or groupings within the data, without requiring human intervention (refer to Figure 2.2). Its capability to identify similarities and differences in information makes it particularly useful for tasks such as exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition [25].

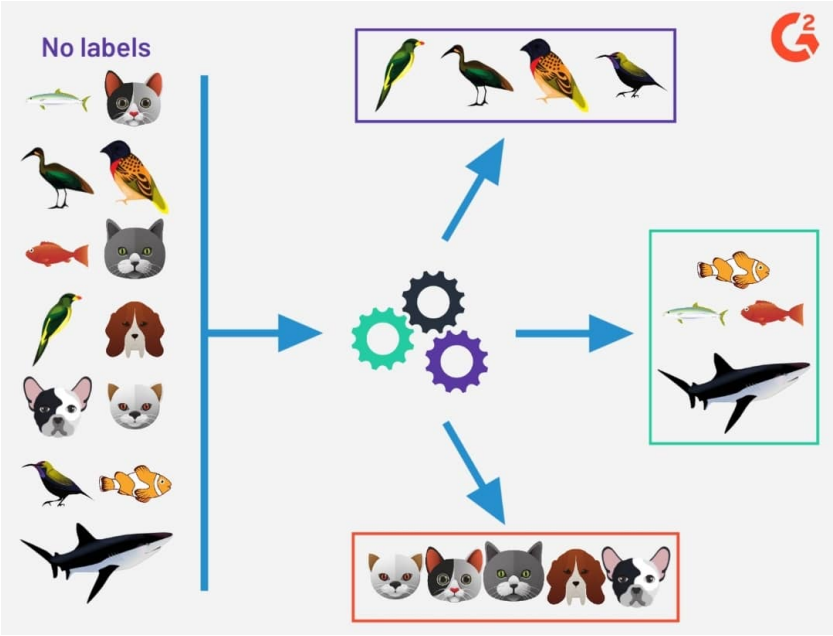

Figure 2.2: Scheme example on unsupervised learning

Supervised learning is split into two categories of algorithms :

- **Clustering :** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association :**

  An association rule learning problem involves the discovery of rules that describe significant relationships within large datasets, such as identifying patterns like "people who purchase X are also likely to buy Y." Unsupervised learning, on the other hand, is commonly applied to massive datasets where labeling the data is challeng-

ing. However, it is important to note that unsupervised learning can be computationally complex and may yield results that are comparatively less accurate than supervised learning methods

2.2.3 Semi-supervised learning

Semi-supervised learning, as depicted in Figure 2.3, combines elements of both unsupervised and supervised learning. This approach leverages a small set of labeled data alongside a large amount of unlabeled data, effectively merging the strengths of both methods. By doing so, it overcomes the challenges associated with obtaining large quantities of labeled data. Semi-supervised learning has proven to be a valuable technique in various applications, providing improved accuracy and efficiency in scenarios where labeled data is scarce or expensive to acquire [25].

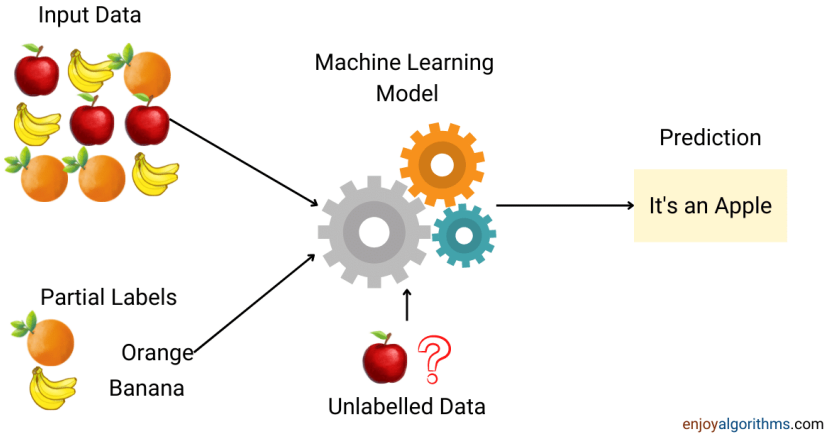

Figure 2.3: Scheme example on semi-supervised learning

2.2.4 Reinforcement Learning

Reinforcement Learning is an integral component of machine learning, where agents undergo self-training based on systems of rewards and punishments. Its focus lies in determining the optimal course of action or pathway that yields the highest rewards while minimizing negative consequences through observations within a given context. This process serves as a feedback mechanism for encouraging positive behaviors and discouraging negative ones (refer to Figure 2.4). Essentially, this involves constructing one or more agents capable of perceiving and comprehending their environment, as well as actively engaging and interacting with it [15].

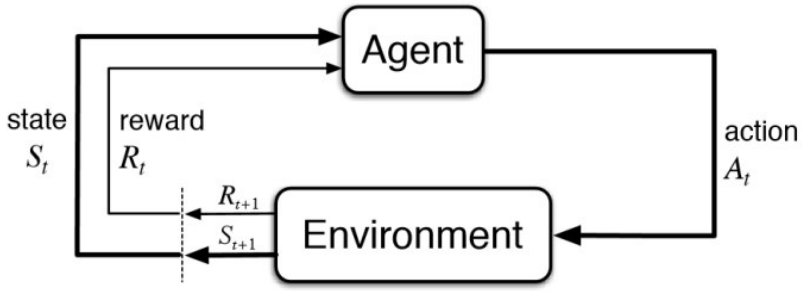

Figure 2.4: Scheme example on Reinforcement learning

2.3 Neural network

Neural networks, also referred to as Artificial Neural Networks (ANN[1]s) or Simple Neural Networks (SNN[2]s), are a subset of machine learning and serve as the foundation for deep learning algorithms. Inspired by the structure and functioning of the human brain, neural networks mimic the communication between biological neurons (refer to Figure 2.6).

The structure of an ANN consists of layers of nodes, including an input layer, one or more hidden layers, and an output layer (see Figure 2.5). Each node, or artificial neuron, is connected to others and possesses a weight and threshold. When the output of a node exceeds its specified threshold, it becomes activated and transmits data to the next layer. Otherwise, no data is forwarded.

Neural networks rely on training data to learn and enhance their accuracy over time. Once these learning algorithms are finely tuned, they become powerful tools in the fields of computer science and artificial intelligence. They enable efficient data classification and clustering, significantly reducing the time required for tasks such as speech recognition or image recognition compared to manual identification by human experts. One notable example of a neural network is Google's search algorithm [30].

[1] *Artificial Neural Network*
[2] *Simulated Neural Network*

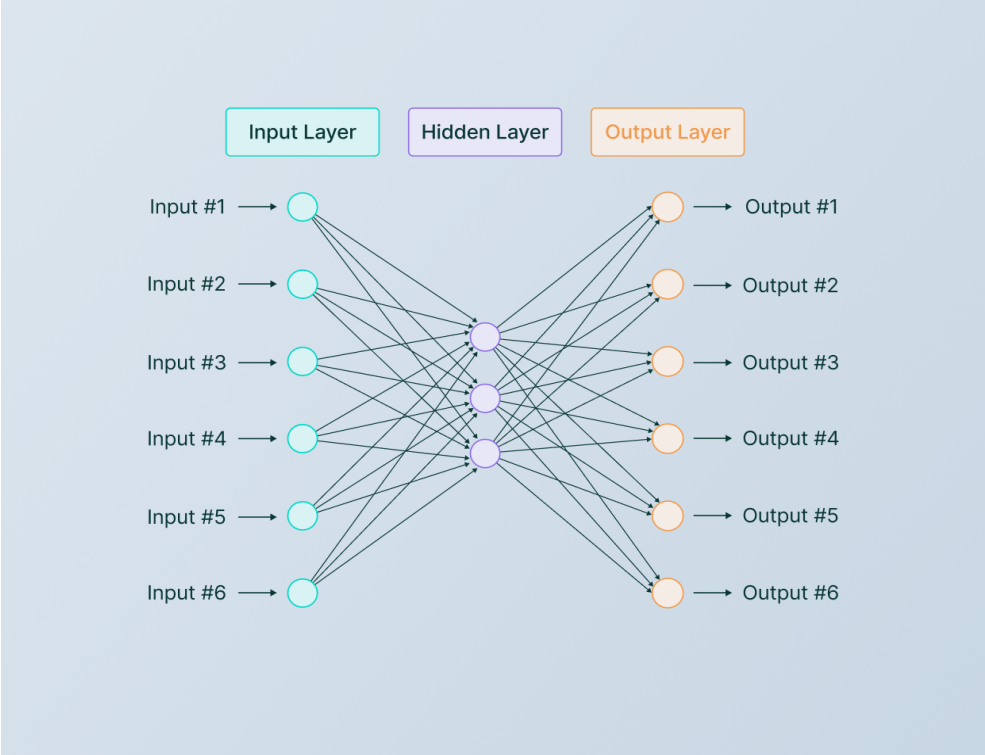

Figure 2.5: Multi-layer neural network [26]

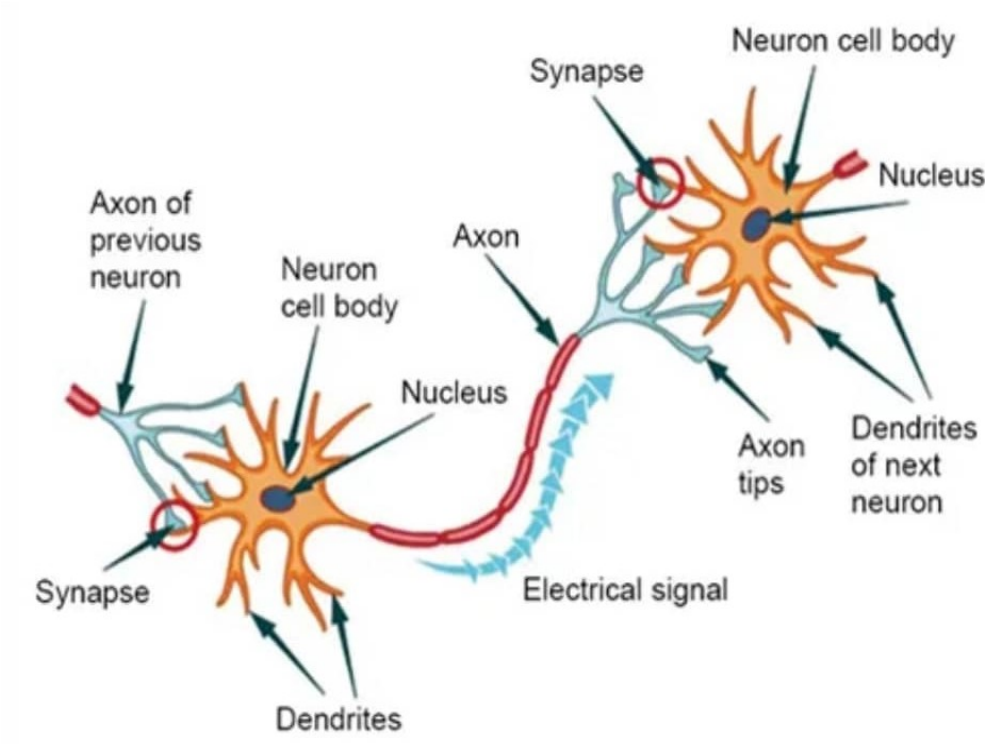

Figure 2.6: Neuron structure in the human brain [31]

2.3.1 History

The origins of neural networks can be traced back to significant milestones throughout history. In 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts published a paper describing the functioning of neurons and created a simple neural network model using electrical circuits.

In 1949, Donald Hebb reinforced the concept of neural pathways strengthening through repeated use in his book, "The Organization of Behavior." This idea laid the foundation for understanding neural networks' learning capabilities.

The 1950s witnessed the first efforts to simulate neural networks, led by Nathanial Rochester at IBM research laboratories. Additionally, the Dartmouth Summer Research Project on Artificial Intelligence (AI) in 1956 boosted research in both AI and neural networks, stimulating interest in emulating neural processing.

In 1957, John von Neumann proposed imitating neuron functions using telegraph relays or vacuum tubes. The following year, Frank Rosenblatt, a neurobiologist, began work on the Perceptron, inspired by the fly's eye's processing mechanisms. The Perceptron, a single-layer neural network, remains in use today for classifying inputs into two classes based on weighted sums and thresholds.

In 1959, Bernard Widrow and Marcian Hoff developed models known as ADALINE and MADALINE, which became the first neural networks applied to real-world problems. MADALINE, an adaptive filter eliminating echoes on phone lines, is still commercially used.

However, neural network research faced challenges when Marvin Minsky and Seymour Papert proved the limitations of the Perceptron in their book, "Perceptrons." This led to a period of slowed growth and reduced funding for neural network research until 1981.

In 1982, John Hopfield presented a paper on useful devices using neural networks, which rejuvenated interest and led to increased funding, including the US-Japan Joint Conference on Cooperative/Competitive Neural Networks.

By the mid-1980s, annual conferences such as the Neural Networks for Computing by the American Institute of Physics and the IEEE International Conference on Neural Networks drew significant attention and participation.

In 1997, the Long Short-Term Memory (**LST!**[3]) framework, a type of recurrent neural network, was proposed by Schmidhuber and Hochreiter. This framework contributed to advancements in sequence modeling and memory retention.

[3] **LST!**

Finally, in 1998, Yann LeCun published a paper titled "Gradient-Based Learning Applied to Document Recognition," which further propelled neural network research [8].

2.3.2 Neural network components

Neural networks are an essential part of deep learning, allowing the processing and analysis of complicated data. They contain several different parts that collaborate to execute tasks such as classification pattern recognition, and prediction.

Weight Weights are essential parameters in neural networks that facilitate the transformation of input data within the hidden layers. When an input enters a node, it is multiplied by a weight value. The resulting output is either observed or passed to the next layer in the neural network. The weights of a neural network are typically situated within the hidden layers, and they play a crucial role in determining the network's overall behavior and performance [29].

Bias In simple terms, neural network bias is a constant value that is added to the product of features and weights within a neural network. It serves to offset or shift the result. The bias helps the models adjust the activation function, enabling them to lean towards either the positive or negative side. By incorporating bias, neural networks gain flexibility in capturing different patterns and making accurate predictions [35].

Activation Function The activation function, also known as the transfer function, plays a crucial role in a neural network by combining input values and producing a single output value. It introduces non-linearity to the computation of the artificial neural network (ANN). Here are some commonly used activation functions:

- **Sigmoid :** A sigmoid function is a mathematical function with a characteristic "S"-shaped curve or sigmoid curve. It transforms any value in the domain $(-\infty, +\infty)$ to a number between 0 and 1 [23].

  it can be represented (see figure 2.7for an input z as [23]:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

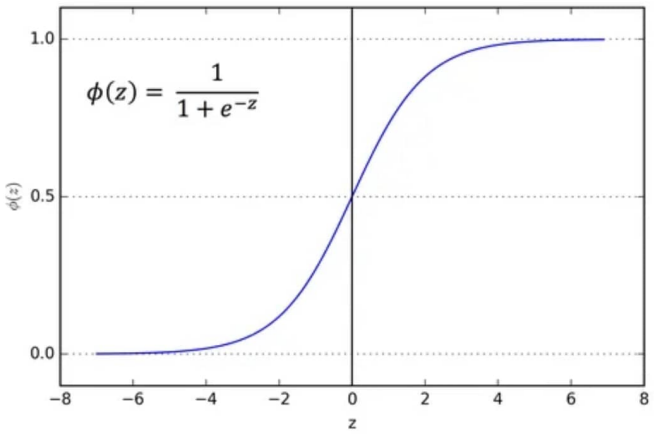

Figure 2.7: Sigmoid function graph

- **Hyperbolic tangent (tanh) :** The hyperbolic tangent function (tanh) is similar to the sigmoid function and shares the same S-shaped curve. It produces output values ranging from -1 to 1. As the input becomes larger and more positive, the output value approaches 1. Conversely, as the input becomes smaller, the output value becomes more negative, approaching -1.

  Due to its output range of -1 to 1, the tanh function is commonly used in the hidden layers of neural networks. This choice helps maintain a mean value close to zero for the hidden layer activations. By centering the activations around zero, the network can effectively learn and capture complex patterns in the data.

  Mathematically, It can be represented as :

$$T(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.2}$$

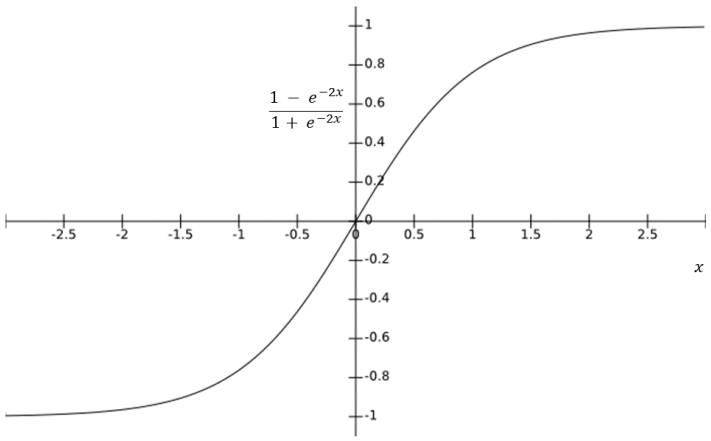

Figure 2.8: Hyperbolic tangent function graph

- **The rectified linear unit (ReLU)** The rectified linear activation function, or ReLU for short, is a piece-wise linear function commonly used in neural networks. It behaves by outputting the input directly if it is positive; otherwise, it returns zero. In other words, ReLU sets negative values to zero while leaving positive values unchanged.

  ReLU has gained popularity as a default activation function in various types of neural networks. It is easier to train and often leads to better performance compared to other activation functions. By allowing the direct propagation of positive values and eliminating the vanishing gradient problem associated with some other activation functions, ReLU promotes efficient learning and helps neural networks effectively capture complex patterns in the data [9].

  it can be represented as :

$$F(x) = \max\{0, z\} \tag{2.3}$$

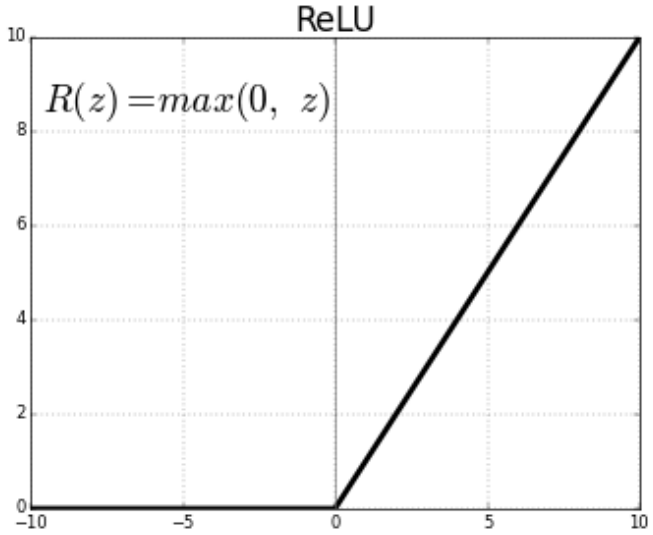

Figure 2.9: The rectified linear unit graph

- **Softmax** The softmax function is a mathematical function that takes a vector of numbers and converts it into a vector of probabilities. The resulting probabilities are proportional to the relative scales of the values in the input vector. In applied machine learning, the softmax function is commonly used as an activation function in neural network models.

  In the context of classification tasks, the neural network is configured to produce N output values, where N represents the number of classes. The softmax function is then applied to normalize these output values, transforming them from weighted

sums to probabilities that add up to one. Each value in the softmax output represents the probability of membership for a specific class.

By utilizing the softmax function, the neural network's outputs can be interpreted as probabilities, allowing for easy interpretation and decision-making in multi-class classification problems. The class with the highest probability can be considered as the predicted class by the model [10]. It can be represented for an input vector $\vec{z}$ by:

$$\sigma(\vec{z}) = \frac{e^{z_i}}{\sum_{k=1}^{k} e^{z_k}}$$

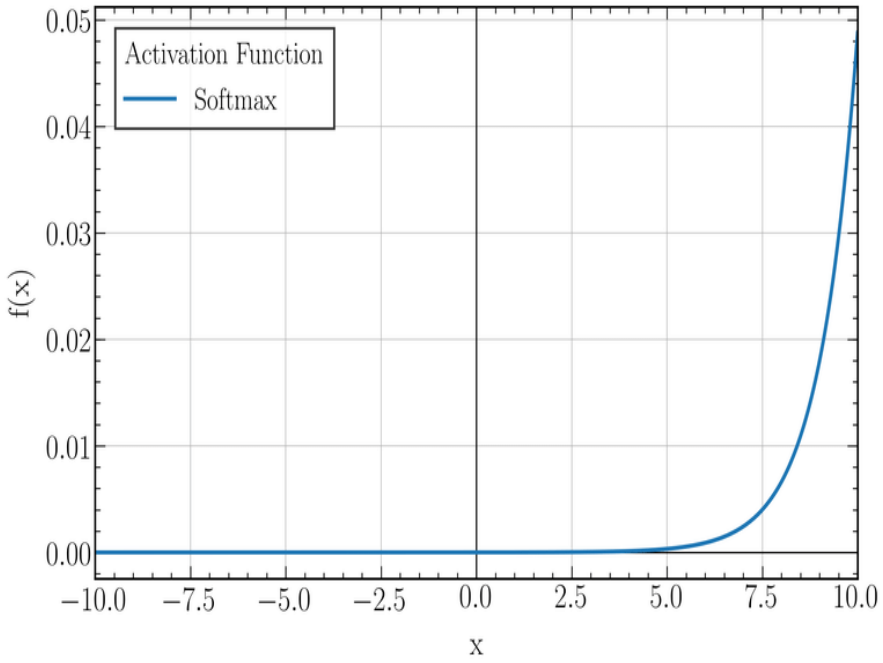

Figure 2.10: SoftMax function graph

Loss Function In a neural network, the loss function quantifies the disparity between the desired output and the actual output produced by the model. It serves as a measure of how well the model is performing. The gradients derived from the loss function are used to update the weights of the neural network during the training process. The cost of the network, also known as the objective or the total loss, is computed as the average of all individual losses across the training dataset.

One commonly used loss function is the Mean Squared Error (MSE). It calculates the average squared difference between the predicted output and the true output. The MSE is often employed in regression tasks, where the goal is to minimize the overall squared

differences between predicted and actual values.

By minimizing the loss function, the neural network strives to improve its predictive accuracy and align its outputs with the expected outcomes. Different types of problems may require the use of specific loss functions tailored to their unique characteristics and objectives [19]. :

$$MSE = \frac{1}{2m} \sum_{i=-1}^{m} (\hat{y} - y)^2$$

The result is simply "loss," and the value computed by the" loss function" is referred to as the "cost function" or" loss function."

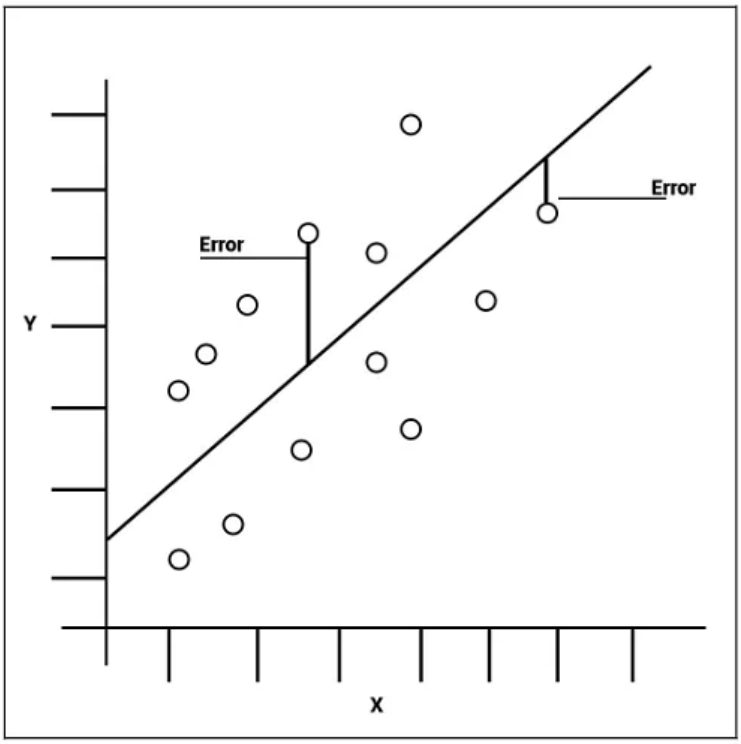

Figure 2.11: Example of a loss function graph

Gradient Descent Gradient descent is an optimization technique utilized to refine the parameters of a machine learning model or train a neural network. Its primary objective is to minimize the degree of error through iterative optimization. The focus of gradient descent is on updating the model's parameters, which encompass regression coefficients in traditional models and neural network weights.

During each iteration, determined by the learning rate specified in the hyperparameters, the cost function in gradient descent serves as a measure of accuracy. It quantifies the disparity between the expected and actual output values. The model aims to minimize the cost function or the discrepancy between the predicted and observed values of y.

By iteratively adjusting the parameters based on the gradients computed from the cost function, the model endeavors to optimize its performance. The process continues until the cost function approaches zero or reaches a sufficiently low level, indicating that the model's predictions align closely with the expected outcomes [11]. Indeed, it uses the gradient descent heuristic to reversely "correct" layers weights using the *Chain Rule:*

Suppose we have an input vector $\mathbf{x}$, and we want to compute the output vector $\mathbf{y}$ by passing the input through a series of layers and activation functions.

Let's denote the output of the i-th layer as $\mathbf{h}^{(i)}$, where $i = 1$ corresponds to the first layer (input layer), and $i = L$ represents the final layer (output layer). The activation function applied in each layer is denoted as $f^{(i)}(\cdot)$.

The output of the i-th layer can be expressed as:

$$\mathbf{h}^{(i)} = f^{(i)}(\mathbf{h}^{(i-1)})$$

where $\mathbf{h}^{(0)} = \mathbf{x}$ represents the input vector.

Given the loss function L with respect to the output of the final layer, $\mathbf{h}^{(L)}$. Let's denote this derivative as $\frac{\partial L}{\partial \mathbf{h}^{(L)}}$.

Then, we can use the chain rule to recursively compute the derivatives of the loss with respect to the outputs of the previous layers. The chain rule states that:

$$\frac{\partial L}{\partial \mathbf{h}^{(i-1)}} = \frac{\partial L}{\partial \mathbf{h}^{(i)}} \cdot \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}}$$

In fact, there are some strategies when applying backpropagation. Some of them are presented in what follows (see Figure 2.12).

- **Batch gradient descent :**Batch gradient descent is an optimization algorithm used in machine learning to update model parameters by calculating the error for each data point in a training set and making updates only after evaluating all examples. This process is known as a training epoch. While batch processing improves computational efficiency, it can result in longer processing times for large datasets due to the requirement of storing all data in memory.

  One advantage of batch gradient descent is its tendency to produce a stable error gradient and converge towards a solution. However, there is a potential drawback: it may converge to a local minimum instead of the global minimum. In other words, the algorithm might not reach the optimal solution for the model, finding a suboptimal point in the error landscape.

Despite this limitation, batch gradient descent remains widely used due to its reliable convergence patterns and effectiveness in many practical scenarios. Researchers and practitioners often explore techniques such as learning rate adjustments and initialization strategies to mitigate the risk of converging to suboptimal solutions [33].

- **Stochastic gradient descent :** Stochastic gradient descent (SGD) is an optimization algorithm that performs a training epoch for each example in a dataset and updates the model's parameters one example at a time. Unlike batch gradient descent, which requires storing the entire dataset in memory, SGD only needs to hold one training example at a time, making it more memory-efficient.

  The advantage of SGD lies in its frequent parameter updates, which can provide more detailed information and faster convergence. However, this frequent updating can lead to computational inefficiency compared to batch gradient descent. Additionally, the individual updates based on single examples can introduce noisy gradients, potentially affecting the stability of the learning process.

  Nevertheless, this noise in the gradients can sometimes be beneficial. It can help the algorithm escape from local minima and converge to the global minimum, resulting in better optimization performance and more accurate models.

  SGD is commonly used in scenarios where computational efficiency is critical, such as large-scale datasets. Despite the noise introduced by the frequent updates, SGD's ability to explore different areas of the optimization landscape can be advantageous in finding better solutions [33].

- **Mini-batch gradient descent :** Mini-batch gradient descent is an optimization algorithm that merges ideas from batch gradient descent and stochastic gradient descent. It partitions the training dataset into smaller batches and updates the model parameters using each of these batches. This method aims to find a compromise between the computational efficiency of batch gradient descent and the swiftness of SGD [33].

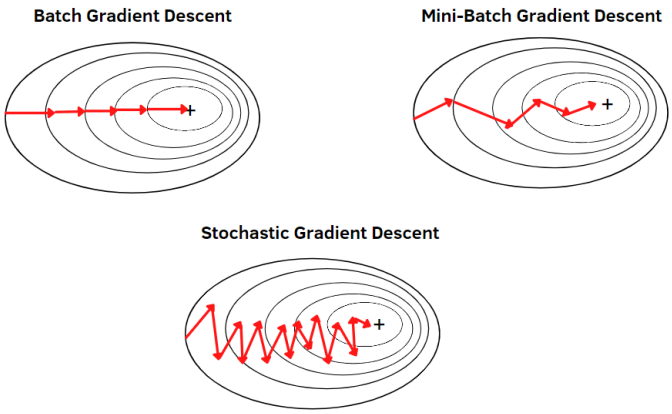

Figure 2.12: Example of a loss function graph

2.3.3 How it works

Once an input layer has been established, weights are assigned to the connections between neurons. These weights, initially set at random, determine the significance of each input variable, with larger weights having a greater impact on the output compared to smaller weights. The calculation of the weighted sum, including the addition of a bias term, is represented by the following formula:

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + \cdots + bias$$

After the weighted sum is calculated, the output is passed through an activation function to determine the final output of the node. The activation function introduces non-linearity into the neural network and helps in capturing complex relationships between inputs and outputs. If the output value surpasses a predefined threshold or meets certain criteria, the node "fires" or activates. The activated output is then transmitted as input to the nodes in the subsequent layer.

This process of passing data from one layer to the next, where the output of one node becomes the input of the following node, characterizes a feed-forward neural network. The data flows in a unidirectional manner, progressing through the network's layers without feedback connections.

$$\text{output} = \begin{cases} 1 \text{ if } \sum w_i x_i + bias \geqslant 0 \\ 0 \text{ if } \sum w_i x_i + bias < 0 \end{cases}$$

During the training process, we employ a cost function, also known as a loss function, to assess the accuracy of the model. The mean squared error (MSE) is one example

of a cost function. The objective is to minimize the cost function, ensuring that each observation is appropriately predicted by the model.

The cost function guides the model towards the point of convergence, often referred to as the local minimum, by adjusting its weights and bias. Gradient descent is the technique used by the algorithm to update these parameters. It enables the model to iteratively refine its approach, reducing errors and progressively converging towards the optimal solution.

As the training progresses, the model adapts its parameters based on each training case, gradually approaching the minimum point of the cost function. This iterative adjustment allows the model to improve its performance and enhance its predictive accuracy.

2.3.4 Types of neural networks

There exists a wide range of artificial neural networks, each with varying levels of complexity. Their shared objective is to imitate the functioning of the human brain in order to tackle intricate problems or tasks. While all types of artificial neural networks incorporate elements resembling neurons and synapses, they differ in terms of complexity, applications, and structure. These disparities encompass the modeling of artificial neurons within each network type, the connections between nodes, the flow of data through the network, and the density of nodes [18]. Some examples illustrating the diversity of artificial neural networks include:

1. **Feedforward artificial neural networks :** A Feedforward artificial neural network functions by facilitating the movement of data in a unidirectional manner, starting from input nodes and progressing towards output nodes. This unidirectional flow of data, as suggested by its name, prevents the data from cycling back through the same layers of nodes. Although Feedforward neural networks can consist of multiple layers and numerous nodes, their simple data flow makes them relatively straightforward to understand. These models are primarily utilized for fundamental classification tasks, surpassing the performance of traditional machine learning models but not reaching the level of abstraction observed in deep learning models [18].

2. **Perceptron and Multilayer Perceptron neural networks :**

   Multilayer Perceptron neural networks enhance complexity and density by incorporating multiple hidden layers between the input and output layers. In this architecture, each node in a given layer establishes connections with every node in

the subsequent layer. This fully connected network structure empowers Multilayer Perceptron models for deep learning applications. They excel at tackling intricate problems and tasks, including complex classification and voice recognition.

Nevertheless, the depth and complexity of these models come with computational costs and time requirements. Processing and maintaining Multilayer Perceptron networks can demand substantial resources and time due to their intricate nature. Despite these considerations, Multilayer Perceptron architectures offer a robust solution for addressing sophisticated challenges within the field of artificial neural networks [18].

3. **Radial basis function artificial neural networks :**

   Radial basis function neural networks typically comprise an input layer, a layer consisting of radial basis function nodes with individual parameters, and an output layer. These networks exhibit versatility and are employed in various domains such as classification, time series analysis regression, and system control. At the core of these networks lies the radial basis function, which determines the absolute value between a center point and a given point. This function assumes a crucial role in defining the behavior and operation of radial basis function neural networks [18].

4. **Recurrent neural networks :** Recurrent neural networks (RNNs) excel at processing sequential data within an artificial neural network (ANN). These models enable data to flow in a forward direction while also looping it back to previous steps, thereby enhancing task performance and prediction accuracy. The intermediate layers in an RNN, situated between the input and output layers, possess a recurrent nature. This means they retain and integrate relevant information by forming loops within the network architecture. The capacity to preserve sequential context makes RNNs particularly effective for tasks involving the processing of sequential data. These models are used for reactive chatbots, translating language, or summarising documents [18].

5. **Modular neural networks :** A Modular ANN is composed of multiple networks or components that collaborate independently to accomplish a given task. By dividing a complex task into smaller components, it becomes more manageable. When applied to data processing or computing, this approach enhances processing speed by enabling smaller components to work together. Each component network performs a distinct subtask, and their collective output completes the overall task. This type of artificial neural network (ANN) offers the advantage of improving the efficiency of complex processes and is applicable in various environments.

6. **Other Types of Neural Networks** Besides Modular ANN, there exist various other types of neural networks, including transformers, auto-encoders, physics-informed networks, vision networks, generative networks, and several others.

2.4 Deep learning

Deep learning is a part of AI that trains neural networks to learn from data and make informed choices or predictions. It's about extracting meaningful insights from complex data like images and texts to efficiently handle problems.

2.4.1 Definition

Deep learning is indeed a subset of machine learning that focuses on the study of computer algorithms enabling self-learning and improvement. While machine learning utilizes simpler concepts, deep learning revolves around artificial neural networks (ANNs) designed to emulate human thinking and learning processes.

In the past, the complexity of neural networks was limited due to constraints in computing power. However, with advancements in big data analytics, larger and more sophisticated neural networks have emerged. This has enabled computers to quickly observe, learn, and respond to complex situations, often surpassing human capabilities. Deep learning has played a pivotal role in various domains, including image classification, language translation, and speech recognition. Through its ability to recognize patterns, deep learning has made it possible to solve problems without the need for human intervention [32].

2.4.2 Deep Learning models

There are several deep learning models, we introduce some of them.

Convolutional neural networks (CNNs)

Let's consider a 2D input image X with dimensions $W \times H$ and C channels, where W represents the width, H represents the height, and C represents the number of channels (e.g., red, green, blue in an RGB image).

In CNNs, the convolution operation involves applying a set of *kernels* (filters) to the input image. Each filter has dimensions $F \times F \times C$.

Let's denote the weights of the k-th filter as W_k. The output feature map (also known as the activation map) for the k-th filter can be computed as follows:

$$Y_k(i,j) = \sigma\left(\sum_{m=1}^{F}\sum_{n=1}^{F}\sum_{c=1}^{C} W_k(m,n,c)\cdot X(i+m-1, j+n-1, c) + b_k\right)$$

where $Y_k(i,j)$ represents the value at position (i,j) in the k-th feature map, $\sigma(\cdot)$ denotes the activation function (e.g., ReLU), $X(i+m-1, j+n-1, c)$ denotes the pixel value at position $(i+m-1, j+n-1)$ of channel c in the input image, b_k denotes the bias term associated with the k-th filter, and i and j represent the spatial indices of the output feature map.

By applying the convolution operation with multiple filters, CNNs can learn how to extract different features and patterns from the input image.

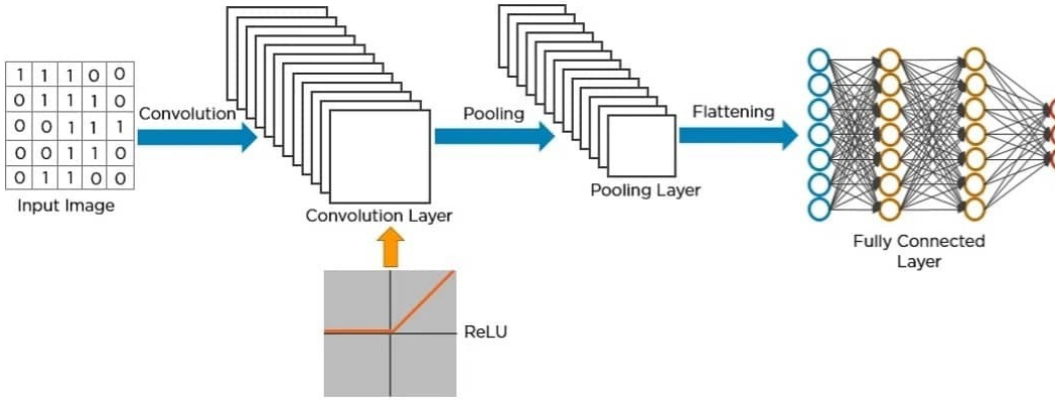

Figure 2.13: Scheme example of a CNN

Convolutional neural networks (CNN) are made up generally of five main types of layers (see Figure 2.18):

1. **Convolution Layer :** The initial stage in extracting meaningful features from an image involves the use of a convolution layer, which contains multiple filters responsible for performing the convolution operation. Each image is treated as a matrix comprising pixel values. Let's consider a specific example: a 5x5 image with pixel values of either 0 or 1. In addition, we have a filter matrix with dimensions of 3x3 (refer to Figure 2.14). To obtain the convolved feature matrix, the filter matrix is slid across the image, and the dot product is computed at each position [14].

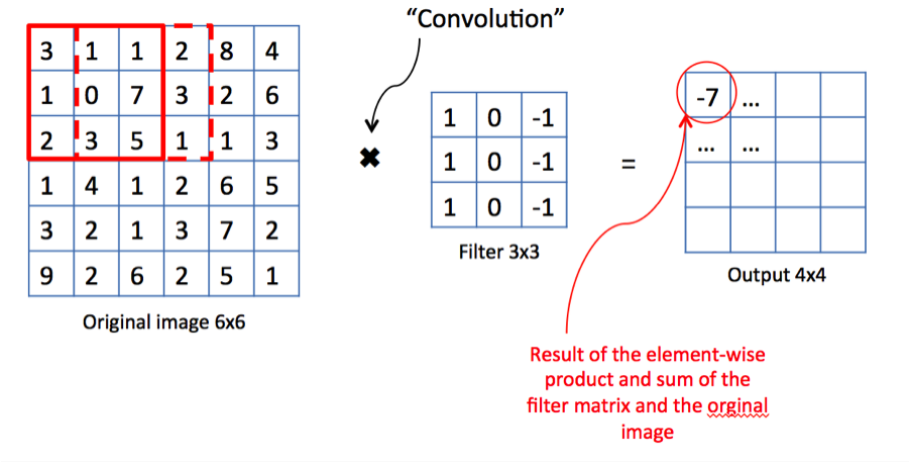

Figure 2.14: Example of a convolutional layer process [2]

2. **ReLU Layer :** ReLu[4] stands for the Rectified Linear Unit. Once the feature maps are extracted, the next step is to move them to a ReLu layer. Relu is commonly used instead of other activation functions. This is because of its simplicity and efficiency, its Mitigation of vanishing gradient, and its sparsity.

3. **Pooling Layer :** This layer is frequently placed between convolution layers, it receives various feature maps and performs the pooling operation on each of them. To reduce computing resources, the pooling operation decreases the size of the photos while keeping their important characteristics (see Figure 2.15). This is achieved by minimizing the connections between layers and operating independently on each feature map. There are numerous sorts of Pooling operations, depending on the method adopted [16]:

   - **Max Pooling :** is the most common type of pooling operation, which extracts patches from input feature maps, outputs the maximum value in each patch, and discards the remaining values. It removes all noisy activations and performs de-noising and dimensionality reduction at the same time [16].
   - **Average Pooling :** is used to compute the average of the elements present in the region of the feature map covered by the filter. It just takes the features from the feature map and averages them. As a noise-suppressing mechanism, it simply performs dimensionality reduction [16].

[4] *Rectified Linear Unit*

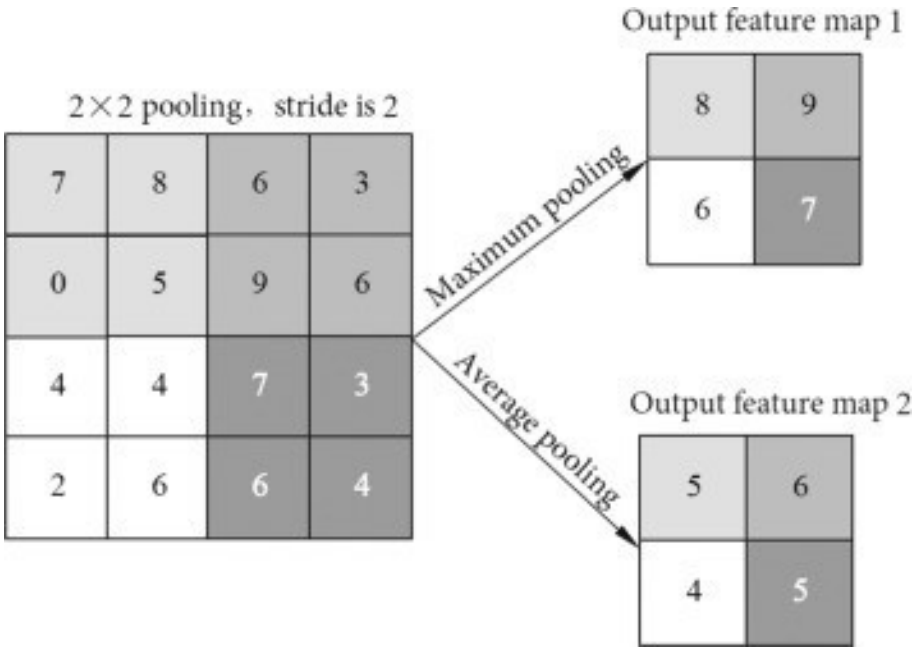

Figure 2.15: Maximum pooling and average pooling [16]

4. **Fully Connected layer :** The prior layers' input image is flattened to a one-dimensional array of numbers or vectors, which is then fed to a neural network with back-propagation applied to each training iteration (see Figure 2.16). The model can distinguish between dominating and certain low-level features in images over a series of epochs and classify them using the last layer activation function [24].

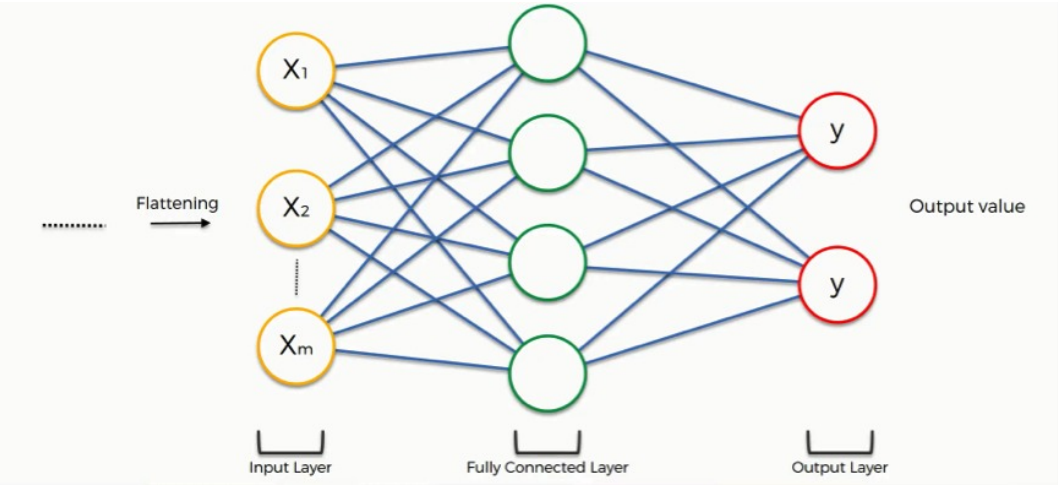

Figure 2.16: Fully-Connected layer diagram [24]

5. **Last layer activation function :** The last fully connected layer's activation function is generally different from the others. Each task requires the selection of suitable activation function. The softmax function The table below (Table 2.1) summarises typical last layer activation function selections for various types of tasks [36].

Table 2.1: Commonly Applied Last Layer Activation Functions [36]

Task	Last Layer Activation Function
Binary classification	Sigmoid
Multiclass single-class classification	Softmax
Multiclass multiclass classification	Softmax
Regression to continuous values	Identity

Generative Adversarial Networks (GANs)

A generative adversarial network (GAN) is an artificial intelligence (AI) framework that utilizes two neural networks engaged in an adversarial game. The objective is to generate computer-generated data that closely resembles real-world data. The GAN consists of two models, namely the Generator and the Discriminator, which work together in a competitive manner to identify, understand, and reproduce patterns within a given dataset (refer to Figure 2.17) [13].

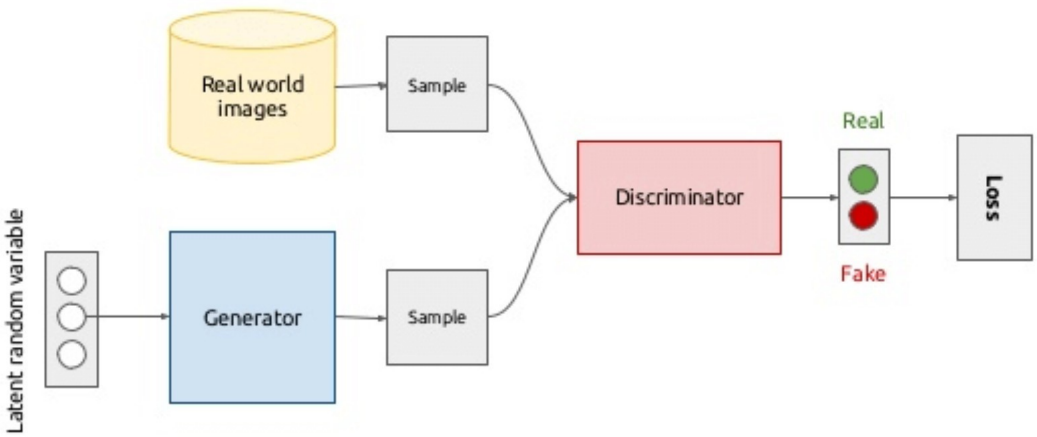

Figure 2.17: Architecture of GAN [28]

1. **Discriminator :** a Discriminator is a neural network consisting of many hidden layers and one output layer that differentiates real data from the Generator's fake data, as shown in the below image [21].

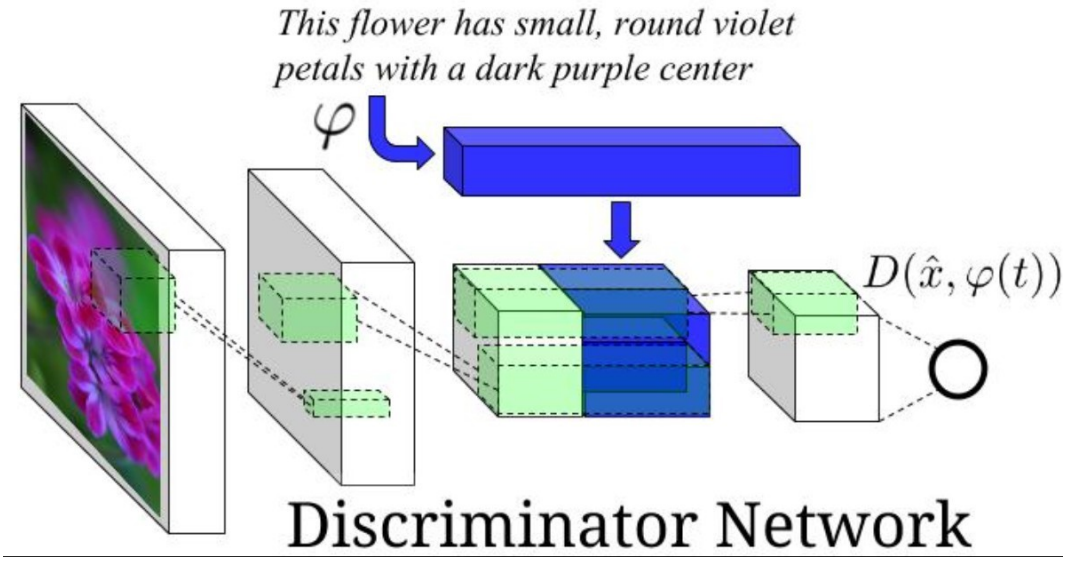

Figure 2.18: Example of a GAN discriminator [21]

The training data for discriminators come from two different sources:

- Real data instances such as real photos used by the discriminator during training as positive samples.
- False Instances of the generator data are used as negative samples in the training phase.

2. **Generator :** a Generator is an Inverse Convolutional Neural Net that generates fake data for the discriminator to be trained on. It learns to use its imagination to create plausible data [21].

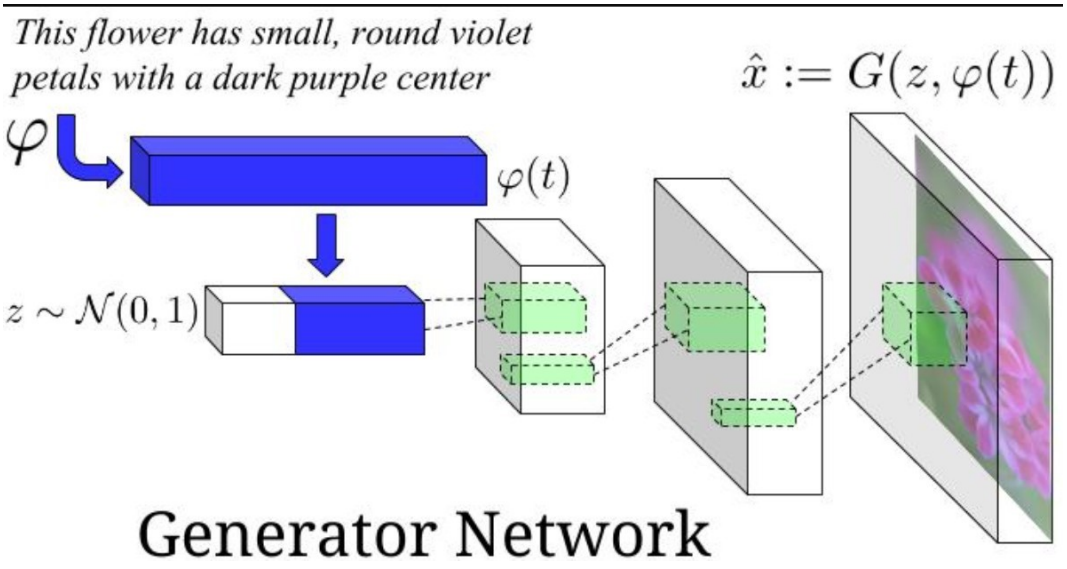

Figure 2.19: Example of a GAN generator

A random value vector is given as input to Inverse-CNN, and after passing through the hidden layers and activation functions, an image is received as the output, as shown in the above image. The discriminator uses the generated examples/instances as negative training examples. It generates a sample from a fixed-length random vector with noise. The GANs are formulated as a minimax game, where the Discriminator is trying to minimize its reward V(D, G) and the Generator is trying to minimize the Discriminator'sreward or in other words, maximize its loss. It can be mathematically described by the formula below [21] :

$$min_G max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))]$$

where:
G = Generator
D = Discriminator
$p_{data}(x)$ = distribution of real data
$p(z)$ = distribution of generator
x = sample from $p_{data}(x)$
z = sample from $p(z)$
$D(x))$ = Discriminator network
$G(z)$ = Generator network

Autoencoders

Autoencoders (AE[5]s) are a specific type of artificial neural network designed to replicate their input data as accurately as possible in their output. They achieve this by compressing the input into a lower-dimensional representation, often referred to as a bottleneck or latent space, and then reconstructing the output from this condensed representation. Autoencoders belong to the category of unsupervised machine learning algorithms. They serve as feature extraction algorithms (refer to Figure 2.20).

Autoencoders can handle various types of input data such as speech, text, images, or videos. Their primary objective is to discover a meaningful representation or code that facilitates useful transformations on the input data [6].

[5] *Autoencoders*

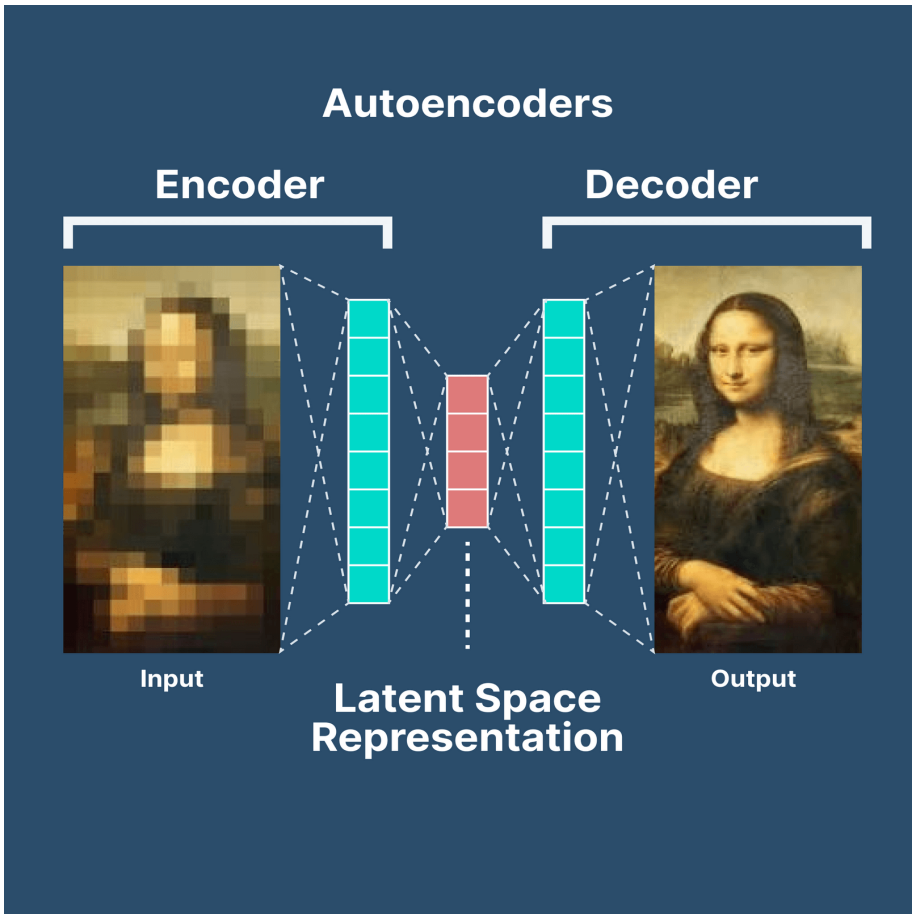

Figure 2.20: Autoencoder Architecture [5]

Autoencoder network is composed of two parts Encoder and Decoder.

- **Encoder :** This part of the network encodes or compresses the input data into a latent-space representation. The compressed data typically looks garbled, nothing like the original data [5].
- **Decoder :** This part of the network decodes or reconstructs the encoded data (latent space representation) back to the original dimension. The decoded data is a lossy reconstruction of the original data [5].

2.4.3 Problems encountered in deep learning

Overfitting and underfitting are two important characteristics of machine learning that are the most frequent cause of a deep learning model's poor performance.

Overfitting : Overfitting is an undesirable machine learning behavior that occurs when the machine-learning model gives accurate predictions for training data but not for new

data. When data scientists use machine-learning models for making predictions, they first train the model on a known data set. Then, based on this information, the model tries to predict outcomes for new data sets. An overfit model can give inaccurate predictions and cannot perform well for all types of new data.

underfitting : Is a scenario in data science where a data model is unable to capture the relationship between the input and output variables accurately, generating a high error rate on both the training set and unseen data. It occurs when a model is too simple, which can be a result of a model needing more training time, more input features, or less regularization. Like overfitting, when a model is underfitted, it cannot establish the dominant trend within the data, resulting in training errors and poor performance of the model. If a model cannot generalize well to new data, then it cannot be leveraged for classification or prediction tasks. The generalization of a model to new data is ultimately what allows us to use machine learning algorithms every day to make predictions and classify data.

Lack of sufficient Data : Profound learning models require a huge sum of labeled information to prepare. Getting high-quality, labeled datasets can be time-consuming, costly, or essentially inaccessible in a few regions.

Vanishing or Exploding Gradients : Like mentioned before, one of the causes of avoiding gradient vanishing in CNNs is the use of Relu. In fact, Both issues happen amid backpropagation the derivatives or slopes get to be great little or too much large. The vanishing gradients occur, as the dependent feature gets smaller and smaller with each slice during backpropagation. And the exploding is the opposite, the derivation gets larger and larger. In vanishing the activation functions (sigmoid, tanh[6]) prone to this problem Little derivations cause moderate weight changes and can disturb your training. Exploding Gradients are caused by high weight values, not the activation function High derivatives result in significant changes to the weights, leading to instability and failure to converge.

Computational Complexity : Deep learning requires significant computer resources, Powerful GPUs, or disturbed computers. To do the training and make efficient predictions. So deep learning training takes a long time and is very expensive.

[6] *Hyperbolic Tangent*

Hyperparameter Tuning : In deep learning, we have a lot of hyperparameters such as learning rate, and batch size, so it's so hard to find the optimal combination of hyperparameters and it requires a lot of experiments.

2.5 Conclusion

Deep learning encompasses a wide range of applications across various sectors. In this context, we specifically concentrated on fundamental concepts and advanced techniques such as Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs). These powerful tools hold immense potential for the future of agriculture and have the capacity to revolutionize the approaches to sheep breeding and husbandry. Leveraging the capabilities of deep learning, we can anticipate substantial advancements and innovations within the agricultural industry, ultimately enhancing productivity, sustainability, and overall optimization

Chapter 3

Related Work

3.1 Introduction

While our work is not the first of its kind in the world, it is pioneering in Algeria. Therefore, we conducted thorough research and identified existing works that can provide valuable insights and support for our own project. In this chapter, we will delve into the relevant studies that have aided us in our research, as well as the tools we have considered for our work on sheep breed recognition using deep learning

3.2 Related Works

During our extensive research on related works, we discovered numerous studies that could potentially aid our thesis. However, after careful consideration, we selected two specific works that we believe will be particularly beneficial to our research objectives :

3.2.1 Sheep Face Recognition Model Based on Deep Learning and Bi-linear Feature Fusion

In [27], the authors propose a deep learning model for sheep facial recognition based on the RepVGG algorithm and bi-linear feature extraction and fusion. The study utilizes training and test datasets containing images of sheep's faces captured from various distances and angles.

The model design involves the creation of a feature extraction channel incorporating an attentional mechanism and RepVGG blocks. The RepVGG block reparameterization technique enables lossless compression of patterns, leading to improved recognition effi-

ciency. Additionally, two feature extraction channels are employed to construct a bilinear feature extraction network, which captures important features from different regions and orientations of the sheep's mouth.

Furthermore, the study combines features from different images at the same scale to enhance feature information, thereby improving recognition capability and network reliability. Experimental results demonstrate that the proposed model effectively mitigates the influence of the sheep's snout position on recognition accuracy. The model achieves recognition rates of up to 95.95

The study includes the organization and utilization of eight models: AlexNet, VGG16, ResNet34, GoogLeNet, EfficientNetV2, DenseNet, RepVGG, and RepB-SheepNet. These models, along with three sheep face datasets, are employed for classification experiments and performance comparison.

Citation: [27]

Results :

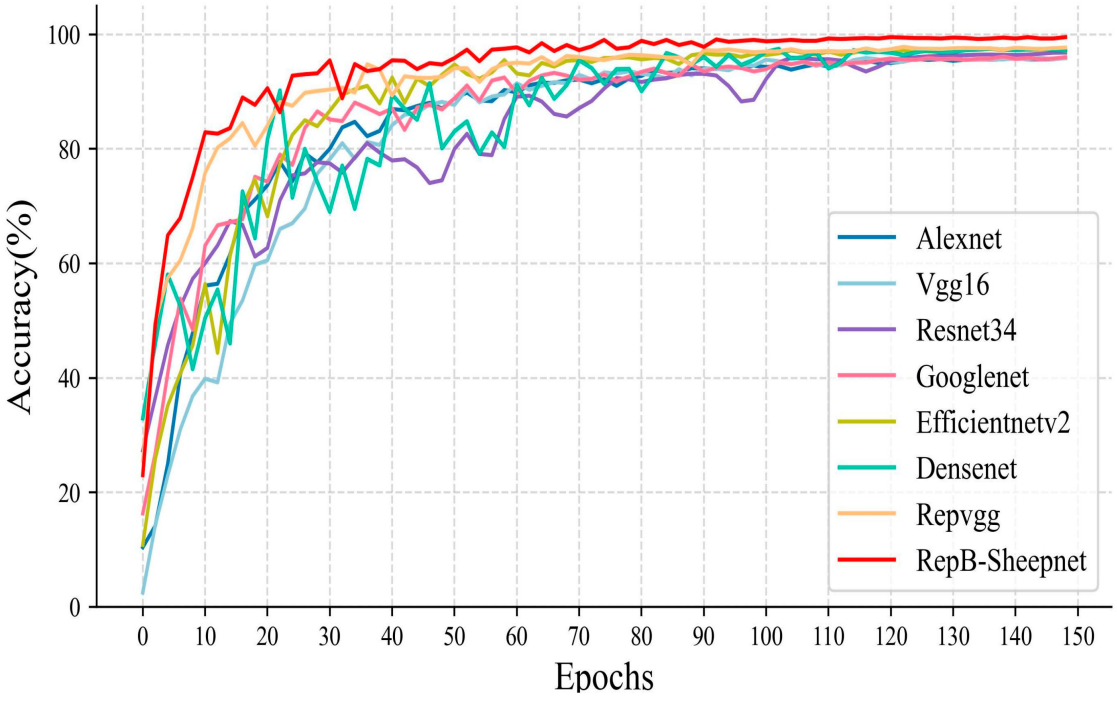

Figure 3.1: Comparison of training accuracy variation of 8 models

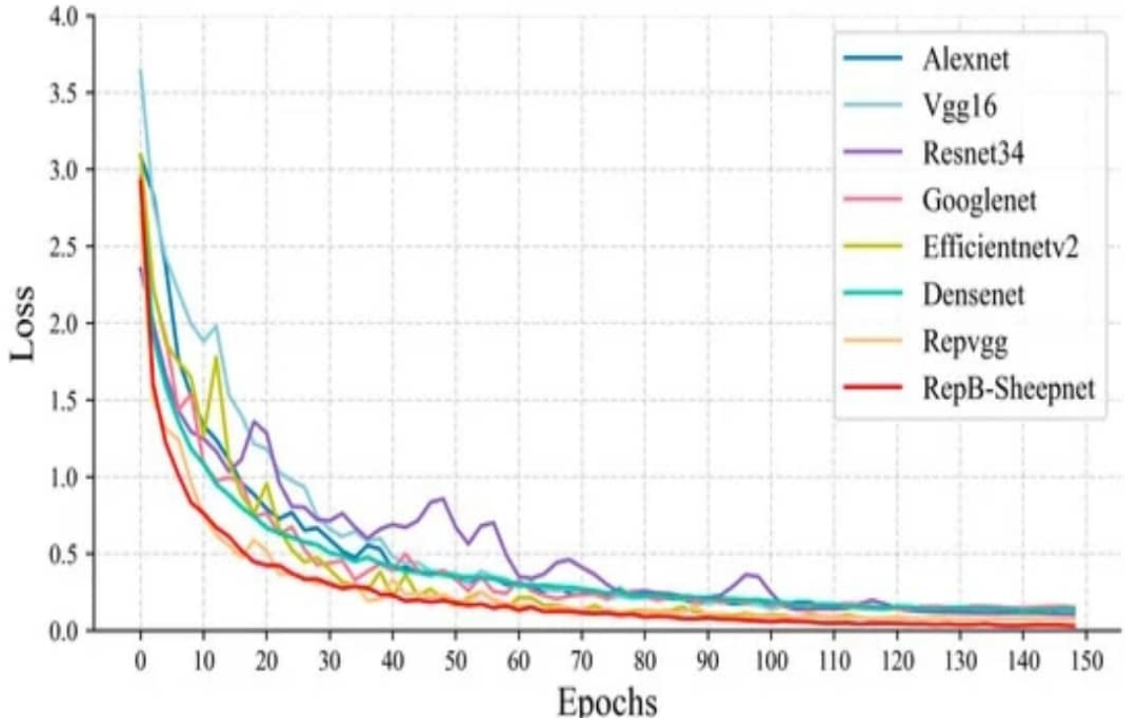

Figure 3.2: Comparison of training loss variation of 8 models

Figure 3.1: shows the training accuracy variation of the eight models, indicating that RepB-Sheepnet exhibits the fastest improvement and convergence, reaching a plateau after 80 epochs. Other models converge gradually after 100 epochs.

Figure 3.2: displays the training loss, with RepB-Sheepnet having the smallest loss and a smooth decreasing curve, while other models show larger oscillations. These findings demonstrate that RepB-Sheepnet possesses characteristics of fast convergence and stability compared to other models.

The model's recognition accuracy is assessed on three datasets (Figure 3.3, Figure 43.4, and Figure 53.5). Results show varying accuracy across datasets, with the model performing best on sheep full-face, followed by front-face, and relatively worse on side-face. Notably, RepB-Sheepnet achieves the highest accuracy of 99.43% on the sheep full-face dataset, outperforming all other models.

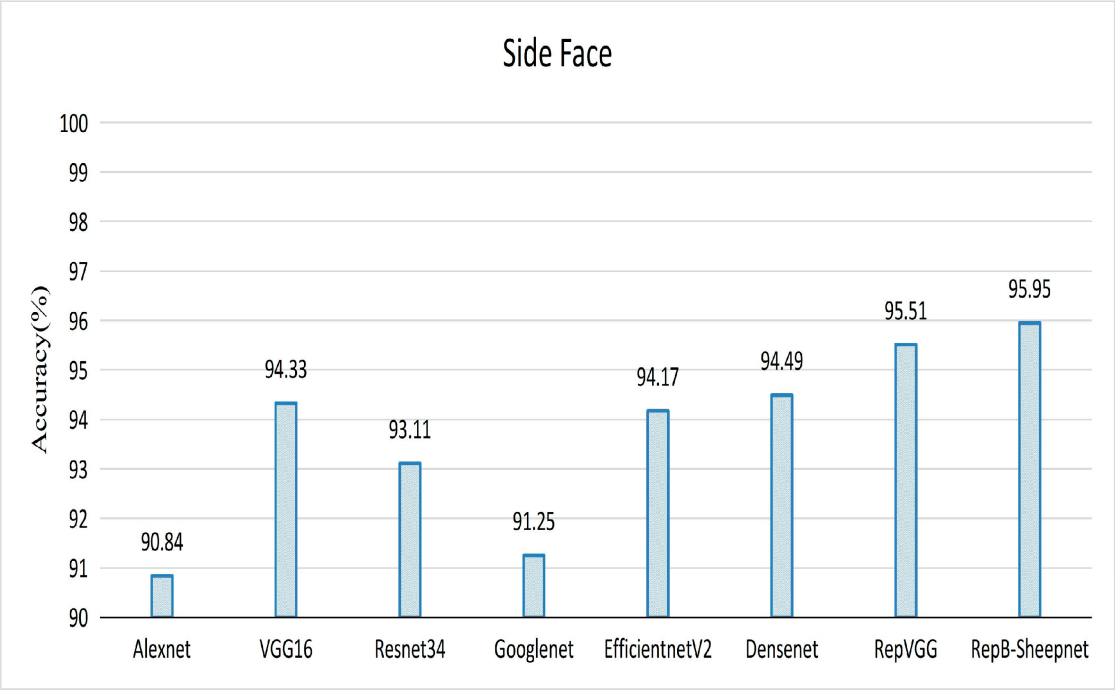

Figure 3.3: Validation accuracy of 8 recognition models on the sheep side-face dataset

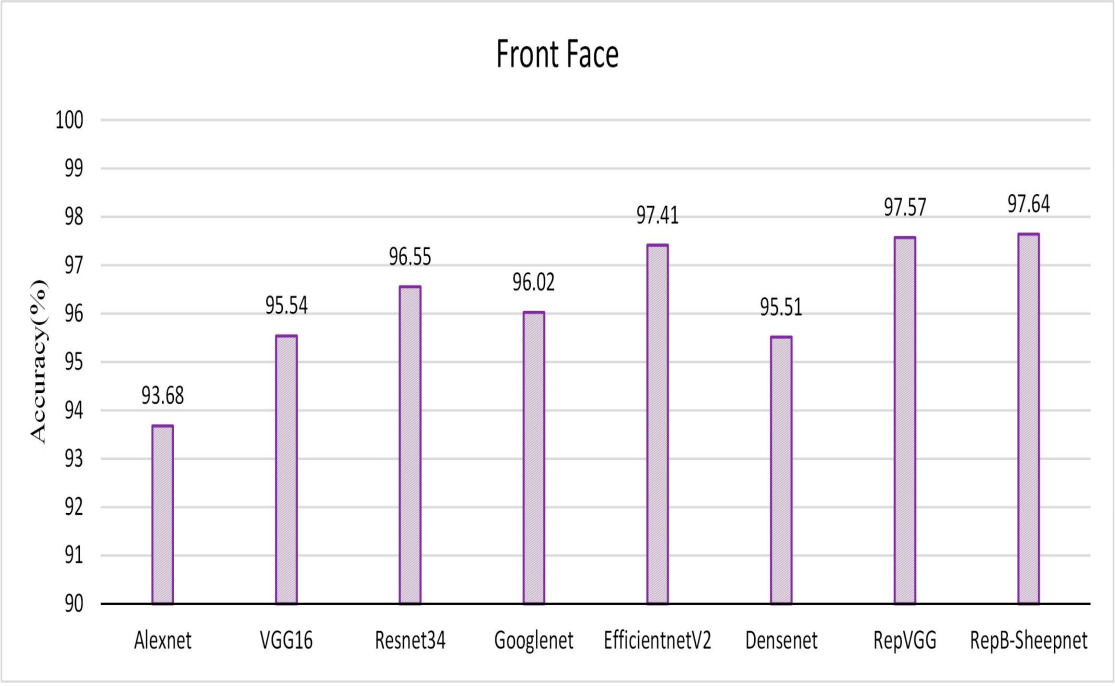

Figure 3.4: Validation accuracy of 8 recognition models on the sheep front-face dataset

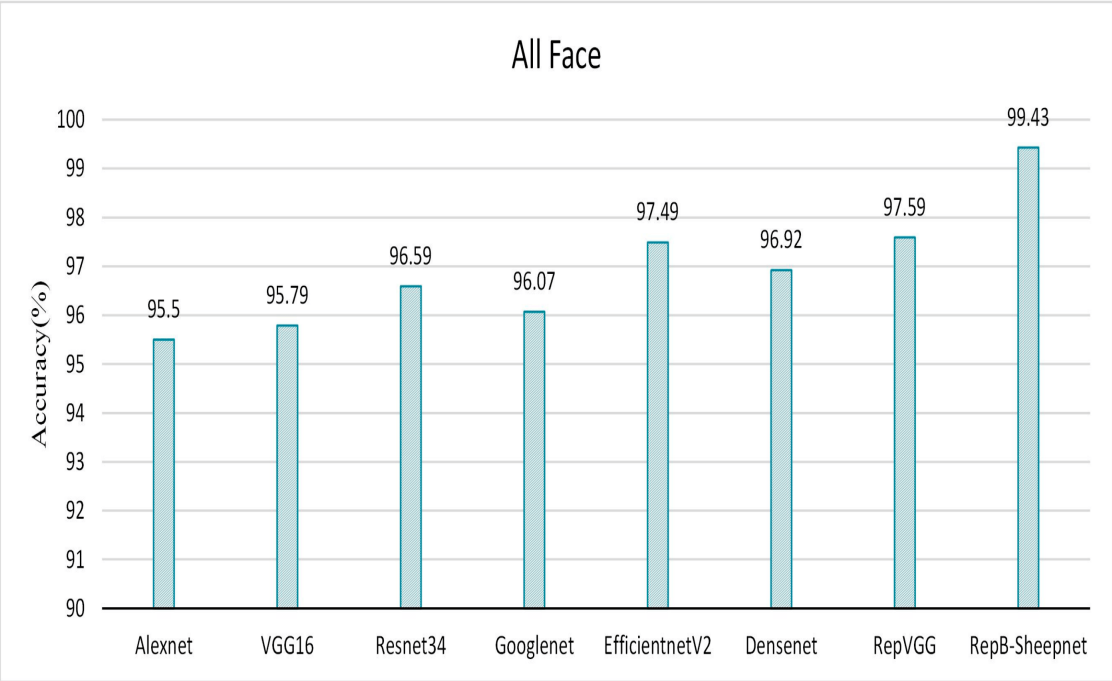

Figure 3.5: Validation accuracy of 8 recognition models on the sheep full-face dataset.

Sheep facial recognition results from Alexnet and RepB-Sheepnet are shown in Figure 3.6. Both models accurately recognize sheep identification information, as shown in Figure 6a. However, in the asymmetric sheep face image (Figure 6b), Alexnet misidentifies the information, while RepB-Sheepnet maintains detection accuracy using a combination of front and side angles.

Figure 3.6: Sheep recognition examples with two images for each sheep. The recognition models are Alexnet and RepB-Sheepnet.

3.2.2 Detection of Bovine Species on Image Using Machine Learning Classifiers

The [22] focuses on classifying cattle species using image processing techniques and mobile applications developed using Flutter and TensorFlow Lite.

To extract features for breed classification, the VGG-16 algorithm was employed. For the actual classification task, the XGBoost and Random Forest algorithms were utilized, and the performance of their combined versions was compared.

In order to address dataset imbalance, the study employed the SMOTE algorithm, which synthesizes new samples, and image augmentation techniques. The performance results of the combined versions of the two classification algorithms were compared, taking into account the utilization of SMOTE and image augmentation.

Results :

After all, the models were trained, it was found that the most suitable model for this problem was VGG16+Random Forest+SMOTE+Augmentation the result is converted

Table 3.1: Comparison of Models

	Accuracy	Precision	Recall	F1 Score
NasNet Mobile	0.84761	0.83888	0.85571	0.85415
Inception V2	0.80125	0.80999	0.79047	0.080974
Inception V3	0.84761	0.83597	0.85326	0.8484167
VGG16+XGBoost	0.85755	0.86043	0.085755	0.85785
VGG16+XGBoost+SMOTE	0.85365	0.85796	0.85365	0.85387
VGG16+XGBoost+Augmentation	0.86330	0.86695	0.86330	0.86310
VGG16+XGBoost+ SMOTE+ Augmentation	0.85796	0.86215	0.85796	0.85812
VGG16+Random Forest	0.87050	0.87283	0.87050	0.87018
VGG16+Random Forest+SMOTE	0.085467	0.85346	0.85467	0.85221
VGG16+Random Forest+Augmentation	0.86474	0.86462	0.86478	0.86327
VGG16+Random Forest+ SMOTE +Augmentation	0.88776	0.88848	0.88776	0.88690

into the mobile application, and give them the result in figure 3.7.

Note : The green means correct and red means wrong.

Figure 3.7: : example of prediction.

3.2.3 Methodology

During our research, we began an in-depth study of sheep breeds, examining their distinctive characteristics and the physical traits that set them apart. This initial phase involved extensive research to gain an in-depth understanding of the various methods and technologies used to identify races, with a particular focus on the role of Artificial Intelligence (AI) in image recognition and classification. Before we start model selection, we perform thorough data preparation. This involved standardizing the format, resolution, and quality of the images collected to create a consistent baseline. In addition, we use data augmentation techniques such as rotation, scaling, and mirroring to increase the diversity and scope of the data set, thereby improving the effectiveness of downstream analysis. The later phase involved a careful evaluation of different AI models suitable for image recognition tasks, with a particular focus on CNN. To leverage existing knowledge and accelerate progress, we consider pre-trained models pre-trained on large image datasets, which provide a strong starting point for our research. Once the model is selected, we begin a rigorous modeling process. The data set has been carefully divided into different sets for training, validation, and testing. Then, using the training data, the selected model was initialized and optimized to ensure optimal performance and accuracy. Leveraging the validation set, we tweak and tune the hyperparameters to further improve the efficiency of the model. Continuous monitoring of relevant metrics such as accuracy, precision, and recall was carefully maintained to track the progress of the model throughout the training phase. By carefully following this methodology, we were able to build a robust AI-based sheep breed detection system that led to the success of our study.

3.3 Conclusion

In this chapter, we present two relevant studies. The first study focuses on sheep recognition based on facial features, while the second study explores the detection of bovine species in images. These works have provided us with valuable insights and information that will greatly contribute to our research. In the following chapter, we will shift our focus to discussing the various sheep breeds found in Algeria.

Chapter 4

Model implementation, results, and discussion

4.1 Introduction

In our work, we focused on creating a deep-learning model that can identify a breed of sheep from photos. To achieve this, we propose a complete model to classify the Sub-saharan Sheep breed and especially the Algerian ones. We introduce a CNN model for this purpose. Then, we enhance the quality of this latter by analyzing its problems and then by using GANs.

This chapter intends to offer an overview of the deployment process, including the methodology, datasets, and experimental setup used. In addition, the results of deep learning models were presented, highlighting the accuracy and efficiency of the classification system in correctly identifying sheep breeds based on visual characteristics.

4.2 The Sub-Saharan sheep classifier

In this section, we propose our CNN classifier for the Algerian sheep breed. We start by dealing with dataset gathering.

4.2.1 Dataset gathering and preparation

In our quest to find a dataset specifically focusing on local breeds, we extensively searched online platforms such as KAGGLE, which is renowned for hosting various datasets, and specialized laboratories that might have conducted research on this topic. However, to

our disappointment, we were unable to locate a suitable dataset that specifically catered to local breeds.

Then, we turned to alternative sources in order to fulfill our data requirements. One of our options was to explore social media platforms, where a vast amount of content is shared by individuals from all walks of life. We scoured through platforms like Instagram, Facebook, and Twitter, searching for posts and images related to local breeds. While this approach was time-consuming and required considerable effort, we were able to gather some relevant information and visuals.

In addition to social media platforms, we also relied on YouTube as a potential source for acquiring data. YouTube is a popular video-sharing platform that offers a wide range of content, including videos featuring various breeds of animals.

The result of image gathering resulted in the acquisition of a dataset consisting of 444 images. While not as comprehensive as we had hoped, this collection of images from social media and YouTube serves as a valuable resource for our study of local breeds.

The gathered dataset is distributed as follows.

- **Dagma :** 149 images.
- **Rambi :** 107 images.
- **Sardy :** 108 images.
- **Olad-djelal :** 80 images.

Moreover, we chose to use full-body images rather than just the face, because with the selected breeds - with the exception of ”Sardi” and ”Dagma” - we cannot decide whether this animal belongs to which breed, only by the face. In addition, it will need hard work and a long time to cut the faces from all pictures of the Dataset, and the model produced will need to prepare inputted images before using it.

4.2.2 CNN Model

We have chosen to deploy a simple CNN model due to many reasons. First, the limited size of our dataset and emphasize that complex CNN architectures typically require a large amount of data to train effectively. Using a simpler architecture can help mitigate the risk of overfitting, where the model learns to memorize the training data instead of generalizing well to unseen examples.

Another reason is that simple CNN architectures generally have fewer layers and parameters compared to more complex models. This results in faster training and inference times, making them more suitable for limited computational resources.

Concerning explainability, simpler CNN architectures often have a more straightforward structure, making it easier to interpret and understand the model's decision-making process.

Finally, this can help us to overcome resource Limitations. In fact, we used free colab cloud machine with the following resources:

- Google colab GPU.
- CPU : Intel Xeon @2.20 GHz.
- GPU : 15GB NVIDIA Tesla T4 GPU.
- RAM : 12.7 GB.
- Storage : 78.2 GB.

The chosen CNN model is shown in figure 4.1. The detailled parameters are presented in table 4.1.

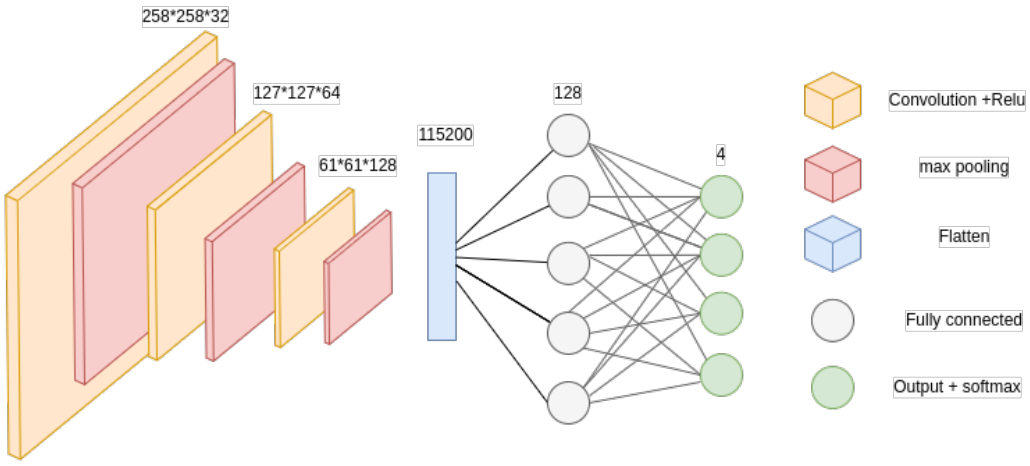

Figure 4.1: Our classifier

The first hyperparameters are :

- IMG_SIZE = 250.
- Training, validation (75%, 25%) ==> (333,111) images.

Table 4.1: CNN model using for classification

Layer (type)	Output Shape	Param #
conv2d	(None, 258, 258, 32)	896
max_pooling2d	(None, 129, 129, 32)	0
conv2d_1	(None, 127, 127, 64)	18,496
max_pooling2d_1	(None, 63, 63, 64)	0
conv2d_2	(None, 61, 61, 128)	73,856
max_pooling2d_2	(None, 30, 30, 128)	0
flatten	(None, 115200)	0
dense	(None, 128)	14,745,728
dense_1	(None, 4)	516

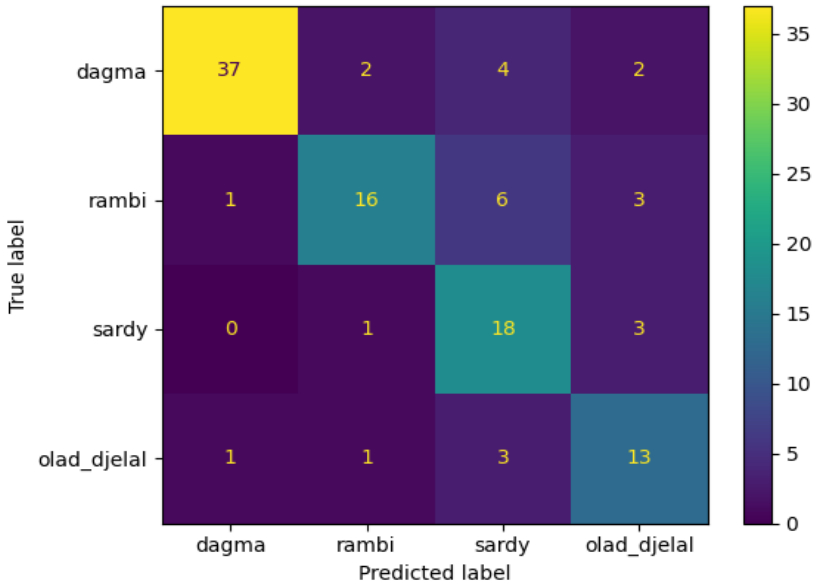

Figure 4.2: Confusion Matrix

- Batch Size = 20.
- Epochs = 90.
- Optimizer= 'Adam'.
- Loss= 'categorical_crossentropy'.

After training and validation, we get the confusion matrix presented in figure 4.2. In fact, we can see that the Dagma race is the one that is recognized the most by the classifier. However, du the relatively unbalanced nature of the dataset, it is more accurate to consider rations. Therefore, the Dagma and the Sardi breeds have more than 80% of recognition.

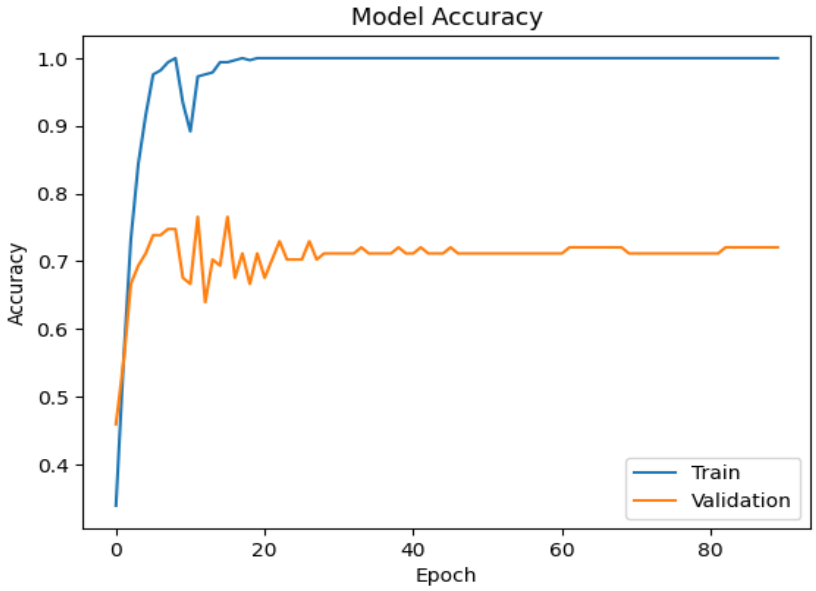

Figure 4.3: Training and validation accuracy curve

We can from figure4.3 that the validation accuracy avoids overfitting. Nevertheless, global accuracy is still not very performant(about 75%). This is due to the small data set at hand. In fact, using deeper CNN in such a dataset can produce underfitting because of the lack of representativity of such a situation.

4.2.3 Hyperparameter Optimization

It is widely acknowledged that finding the optimal hyperparameters for a deep learning approach is a challenging task. With numerous possible combinations, the search for the best hyperparameters can become computationally complex and pose an NP-hard optimization problem. To address this issue, heuristics are employed in hyperparameter optimization to discover satisfactory configurations [37].

In the field of deep learning, various techniques for hyperparameter optimization are commonly employed to refine and optimize the performance of neural networks. Among these techniques are:

Grid Search Grid Search is a hyperparameter optimization technique that entails defining a predetermined set of values for each hyperparameter and exhaustively searching all possible combinations. The model's performance is evaluated for each combination, and the one that yields the best performance is selected. Although Grid Search is straightforward to implement, it can be computationally expensive and inefficient when dealing with large hyperparameter spaces.

Random Search Random Search is a hyperparameter optimization technique that involves randomly sampling values from a predefined distribution for each hyperparameter. It conducts a specified number of iterations and evaluates the model's performance for each sampled set of hyperparameters. Random Search is considered more efficient than Grid Search when dealing with large hyperparameter spaces. Additionally, it has the potential to discover better combinations by chance, as it explores a wider range of hyperparameter values.

Bayesian Optimization Bayesian Optimization employs probabilistic models to represent the objective function whose exact form is unknown. It utilizes these models to intelligently select the next set of hyperparameters for evaluation, taking into account past performance. By constructing a surrogate model, Bayesian Optimization can strategically explore the hyperparameter space, concentrating on areas that are more likely to yield improved results. This approach is known for its efficiency and effectiveness, particularly in comparison to Grid Search and Random Search, when dealing with complex and computationally demanding models.

Genetic Algorithms Genetic Algorithms draw inspiration from natural selection and evolution. Initially, a population of randomly generated hyperparameter sets is created. Through successive iterations, the population evolves via selection, crossover, and mutation. Individuals with superior performance, referred to as the fittest, are favored for selection and contribute to the subsequent generation. Genetic Algorithms excel at efficiently exploring extensive search spaces, but their utilization may demand additional computational resources.

Gradient-based Optimization Gradient-based optimization methods are utilized to optimize hyperparameters by utilizing gradients computed from the model's performance on the training data. Techniques like Gradient Descent, Adam, or RMSprop can be employed to iteratively update the hyperparameters based on their corresponding gradients. This approach assumes that the hyperparameters are differentiable with respect to the model's performance, thereby imposing limitations on its applicability.

In fact, because of our limited computational power, we chose a reduced number of hyper-parameters to be optimized for our experiment. The hyperparameter tuning algorithm iterates over the following values.

- Batch Size : [8, 16, 32, 64, 128, 256].
- Max epochs : [10, 20, 30, 40, 50, 100].

- Optimizer : ["Adam", "sgd", "rmsprop"].
- Random state : [0, 1, 42, 100, 1000].

We choose the Random Search approach to our experiment (see figure 4.4. In fact, random search is a simple and intuitive approach for hyperparameter optimization in deep learning. It works by randomly sampling hyperparameter values from predefined ranges and evaluating the performance of the model with each combination of hyperparameters. Here's how the random search approach typically works:

First, we need to define the hyperparameters and their corresponding ranges that you want to optimize. This is what we already defined above. Next, we decide on the number of iterations or trials to be performed during the search. After that, randomly sample hyperparameter values from the defined search space. Then comes the training and evaluating step. We train the deep learning model using the sampled hyperparameters on a subset of your training data. Then, we evaluate the model's performance on a validation set or through cross-validation. This performance metric could be accuracy, loss, F1 score, or any other appropriate metric for your specific task.

Finally, the algorithm keeps track of the best-performing set of hyperparameters and the corresponding performance metric. This is typically the set of hyperparameters that resulted in the highest validation performance so far. Steps from random sampling to the hyperparameters update are repeated for the specified number of iterations or until the stop condition is fullfiled.

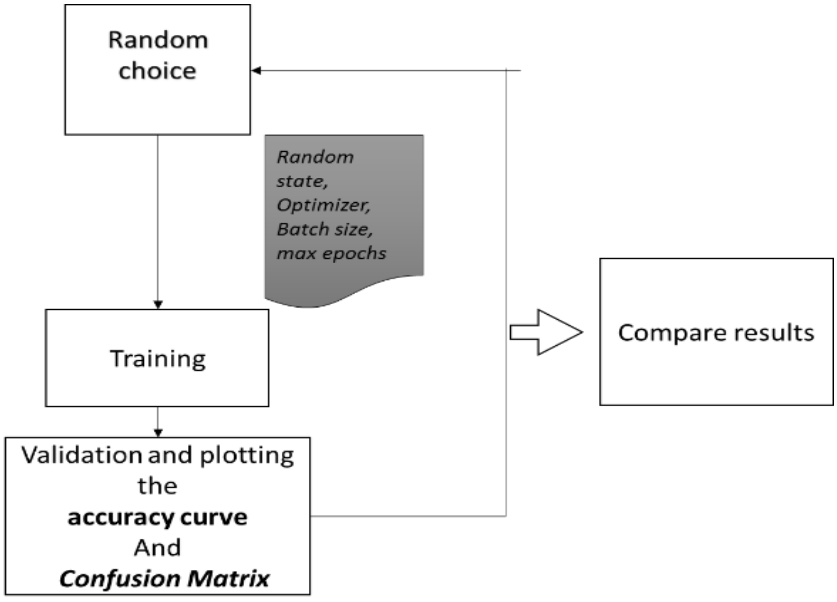

Figure 4.4: Algorithm used in Hyper-Parameters Optimization

After seven iterations, we remark that the hyper-parameter optimization did not outperform the initial configuration.

We believe that this happens because of the nature of the chosen algorithm. The algorithm, which relies on a random search, does not have any inherent mechanism to ensure that each iteration will outperform the previous solution. This is because random search does not incorporate any heuristics, optimization techniques, or autocorrection formulas during its interactions. In other words, the algorithm does not have a systematic approach to guide its search process toward more optimal solutions or make adjustments based on previous iterations. Consequently, the algorithm's reliance on randomness alone makes it unpredictable in terms of performance improvement, leading to a lack of consistent progress. Results are shown in table 4.2.

Table 4.2: Hyper-Parameters sets and results

No.	Batch Size	Max Epochs	Optimizer	Random State	Accuracy
1	16	100	adam	42	81 / 111 = 0.729
2	128	20	adam	1	77 / 111 = 0.693
3	128	30	sgd	42	66 / 111 = 0.594
4	64	100	sgd	100	70 / 111 = 0.630
5	8	10	rmsprop	1000	79 / 111 = 0.711
6	8	10	adam	0	66 / 111 = 0.594
7	128	30	rmsprop	1000	55 / 111 = 0.495

4.3 Genrarive adverserial network for local breed image generation

In this section, we introduce the second part of our approach. We use a generative neural network to enhance our gathered dataset with new synthetic images of local breeds.

4.3.1 Why using GANS

We will see that the major problem of our model is the lack of datasets about Sub-Saharan breeds. We believe that a good alternative will be to use fine-tuning on what is already at hand using some complete CNNs like Yolo and others. Nevertheless, it will be more accurate and beneficial to have a specific dataset for Algerian breeds. The first solution is to use GANs to have synthetic but realistic images. This will have a good impact on future research in this field.

Actually, GANs are commonly used for several reasons. GANs surpasse at generating new data that resembles a given training dataset. They can learn the underlying distribution of the training data and generate new samples that capture the characteristics, patterns, and diversity of the original data. This ability is valuable in situations where obtaining large amounts of labeled or diverse data is challenging or expensive. GANs also enable domain adaptation by transforming or translating data from one domain to another. For example, they can convert images from a source domain (e.g., photos) to a target domain (e.g., paintings) while preserving the content or style. This is useful in scenarios where labeled data is abundant in one domain but scarce in another. In terms of security and privacy, GANs can generate synthetic data that retains the statistical properties of real data without exposing private or sensitive information. This is valuable for privacy protection when sharing or analyzing sensitive datasets.

GANs can be used to augment existing datasets by generating additional samples. Data augmentation is beneficial in machine learning tasks as it increases the diversity and quantity of training data. By introducing variations, GAN-generated samples can improve the generalization and robustness of machine learning models, leading to better performance [3].

4.3.2 Deployed GAN model

There exists an increased number of GAN architectures in the literature. GAN ZOO[1] indexes more than 300 different GAN models. Nevertheless, we chose one of the most popular and useful GANs in real data generation. DCGAN stands for Deep Convolutional Generative Adversarial Network. It is a type of Generative Adversarial Network (GAN) architecture specifically designed for generating realistic and high-quality images. DCGANs leverage deep convolutional neural networks to learn the mapping from random noise to the space of real images [20].

Figure 4.5 shows the generator used by the authors of DCGAN.

[1]https://github.com/hindupuravinash/the-gan-zoo

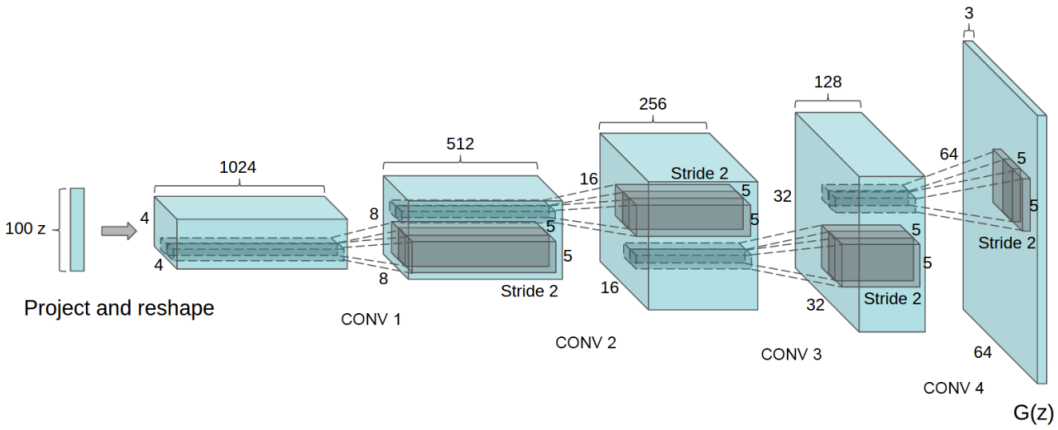

Figure 4.5: DCGAN generator like described in [20]

We give the pseudo-code of the deployed generator and discriminator respectively in

Algorithm 1 and Algorithm 2.

Algorithm 1: Generator Model

```
1  def build_generator(seed_size, channels):
2      model = Sequential()
3      model.add(Dense(4*4*256, activation="relu", input_dim=seed_size))
4      model.add(Reshape((4,4,256)))
5      model.add(UpSampling2D())
6      model.add(Conv2D(256, kernel_size=3, padding="same"))
7      model.add(BatchNormalization(momentum=0.8))
8      model.add(Activation("relu"))
9      model.add(UpSampling2D())
10     model.add(Conv2D(256, kernel_size=3, padding="same"))
11     model.add(BatchNormalization(momentum=0.8))
12     model.add(Activation("relu"))
13     model.add(UpSampling2D())
14     model.add(Conv2D(128, kernel_size=3, padding="same"))
15     model.add(BatchNormalization(momentum=0.8))
16     model.add(Activation("relu"))
17     if GENERATE_RES > 1 then
18         model.add(UpSampling2D(size=(GENERATE_RES,GENERATE_RES)))
19         model.add(Conv2D(128, kernel_size=3, padding="same"))
20         model.add(BatchNormalization(momentum=0.8))
21         model.add(Activation("relu"))
22     end
23     model.add(Conv2D(channels, kernel_size=3, padding="same"))
24     model.add(Activation("tanh"))
25     return model
```

Algorithm 2: Discriminator Model

```
def build_discriminator(image_shape):
    model = Sequential()
    model.add(Conv2D(32, kernel_size=3, strides=2,
     input_shape=image_shape, padding="same"))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.25))
    model.add(Conv2D(64, kernel_size=3, strides=2, padding="same"))
    model.add(ZeroPadding2D(padding=((0,1),(0,1))))
    model.add(BatchNormalization(momentum=0.8))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.25))
    model.add(Conv2D(128, kernel_size=3, strides=2, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.25))
    model.add(Conv2D(256, kernel_size=3, strides=1, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.25))
    model.add(Conv2D(512, kernel_size=3, strides=1, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(1, activation='sigmoid'))
    return model
```

4.3.3 GAN evaluation

As mentioned earlier, hyperparameter optimization is not suitable in our case. In fact, evaluating the quality of a (GAN) can be a challenging task. GANs are unsupervised learning models, which makes the evaluation process inherently subjective. However, there are several commonly used methods and metrics that can help assess the quality of a GAN [7].

Existing evaluation metrics

Visual Inspection One of the simplest ways to evaluate a GAN is to visually inspect its generated samples. Check if the generated images or outputs are realistic, visually appealing, and resemble the desired data distribution. Human judgment plays a crucial role in assessing the quality of the generated samples.

Inception Score (IS) The Inception Score measures the quality and diversity of generated images. It uses the Inception model, a pre-trained image classification network, to compute a score that balances both image quality and diversity. Higher scores indicate better quality and diversity. However, the Inception Score has limitations and may not be suitable for all types of datasets.

The Inception IS can be computed as follows

$$IS = exp(E[KL(p(y|x)||p(y))])$$

where:

- KL represents the Kullback-Leibler divergence.
- $p(y|x)$ is the conditional class distribution for an image x generated by the GAN.
- $p(y)$ is the marginal class distribution, computed by averaging $p(y|x)$ over all generated images.

Frechet Inception Distance (FID) FID measures the similarity between the generated samples and the real data distribution using feature statistics extracted from the Inception model. Lower FID scores indicate higher quality and similarity to the real data. FID is known to correlate well with human judgment.

$$FID = \|\mu_1 - \mu_2\|^2 + Tr(C_1 + C_2 - 2 * (C_1 * C_2)^{1/2})$$

where:

- μ_1 and μ_2 are the mean feature vectors of the real and generated data distributions, respectively.
- C_1 and C_2 are the covariance matrices of the real and generated data distributions, respectively.

- $Tr(.)$ denotes the trace of a matrix.

Precision, Recall, and F1-Score If you have labeled data, you can evaluate the GAN's performance using precision, recall, and F1-score. Train a classifier on the real and generated data and measure the performance of the classifier. This approach is useful when generating data with specific characteristics or classes.

User Studies Conducting user studies can provide valuable insights into the perceived quality of generated samples. You can gather feedback from human evaluators who rate the generated outputs based on predefined criteria or conduct preference tests to compare the GAN's outcomes with other baselines.

Domain-Specific Metrics Depending on the application domain, specific metrics might be more relevant. For instance, if you are generating text, metrics like perplexity, BLEU score, or human evaluation of the generated text's quality can be used.

Evaluation of obtained results

Given the small size of the dataset, it becomes highly improbable that the generated images produced by the model will exhibit a distribution that closely matches the one presented in the original dataset. The size of the dataset presents a challenge for the generative model to capture and replicate the intricate patterns, nuances, and statistical properties encapsulated within the dataset. As a result, the generated images may deviate from the characteristics, diversity, and fine-grained details present in the original dataset [17].

Therefore, we adopt the user studies strategy to evaluate the deployed GAN.

The first impression of our study is that the quality of the generated images does not meet the realistic criteria of GANs. This is an expected ascertainment due to the size of the gathered dataset. Nevertheless, the obtained images were better than we expected. Figure 4.6 presents examples of generated images, we can see that despite the size of Sardi images (108), the result is promising.

Figure 4.6: First Sardi generated images

Hence, we fixed a "subjective" evaluation criteria. It is based on the following formulae.

$$\text{Quality} = \frac{\#\text{ acceptable images}}{\#\text{ generated images}}$$

Based on this ratio, we classify experiment results following these classes.

1. **No progress** if the ratio is null. The experience produces nothing but just noisy images.

2. **Unacceptable** If the ratio is too small, the experience fails to generate any image with "significant" content.

3. **Almost acceptable** if the experience results in good images (based on a subjective opinion).

Instead of using a hyper-parameter optimizer which is not suitable for this case (absence of case-adapted evaluation metrics), we executed the model using four different configurations. Different hyper-parameters chosen for our experiments are shown in Table 4.3.

Table 4.3: Hyper-parameters variations of different experiences with SARDI breed

Experiment	**exp1**	**exp2**	**exp3**	**exp4**
Image Resolution	128px	128px	160px	160px
Batch Size	16	16	64	16
Activation (Generator)	”relu”	LeakyReLU (alpha=0.2)	model.add (Activation (”sigmoid”))	”relu”
Activation (Discriminator)	LeakyReLU (alpha =0.2)	LeakyReLU (alpha=0.2)	model.add (Activation (”sigmoid”))	LeakyReLU (alpha=0.2)
Generator Optimizer	tf.keras. optimizers .Adam(1.5e-4 , 0.5)	Adagrad (learning_rate =0.01)	optimizer = RMSprop (learning_rate =0.001)	tf.keras. optimizers.Adam (1.5e-4, 0.5)
Discriminator Optimizer	tf.keras. optimizers .Adam(1.5e-4, 0.5)	Adagrad (learning_rate =0.01)	optimizer = RMSprop (learning_rate =0.001)	tf.keras. optimizers. Adam(1.5e-4, 0.5)
EPOCHS	many (more than 10,000)	5,000	5,000	6,500
Dataset (Breed/Race)	SARDI	SARDI	SARDI	SARDI
Results	Almost acceptable	Unacceptable	No progress	Unacceptable

The first experience (exp1) reaches the best results compared with the others. Almost all the image was good enough compared to the subjective threshold fixed before executing the process.

Figure 4.7 shows some of the generated images.

Figure 4.7: Sardi images generated by exp1

Experiences 2 and 4 fail to generate distinguishable images. Some of them present a kind of sheep silhouette. But, it cannot be considered an “acceptable” image. Figure 4.8 shows some of them.

Concerning exp3, it produces just noisy images, and the chosen hyperparameters can be abandoned. An example of the generated images is shown in figure 4.9

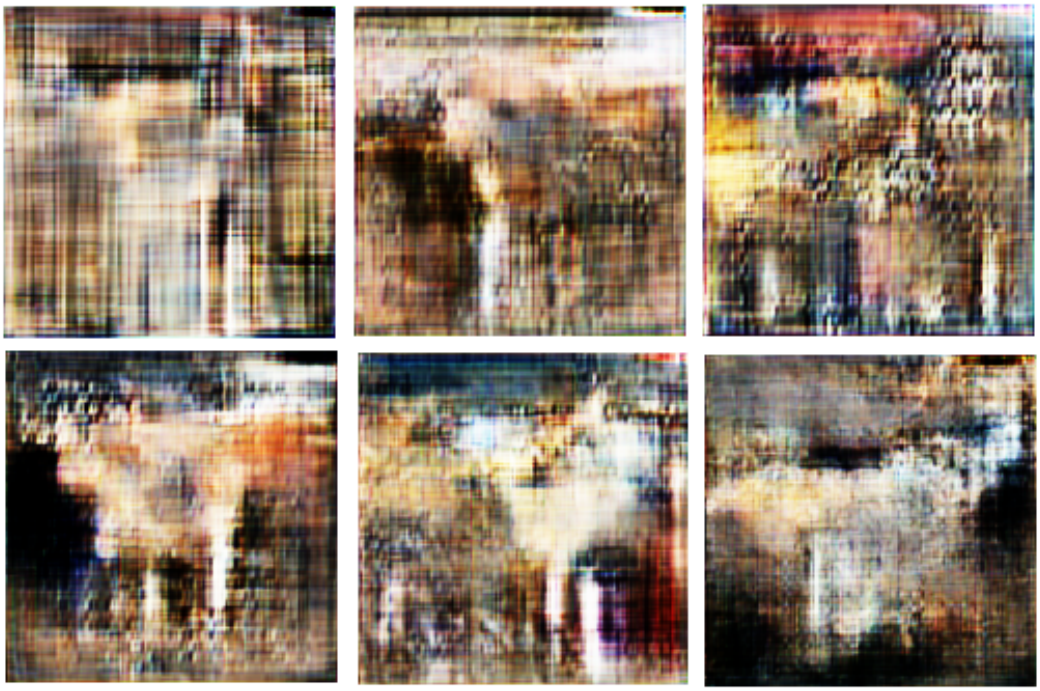

Figure 4.8: Sardi images generated by exp2 and 4

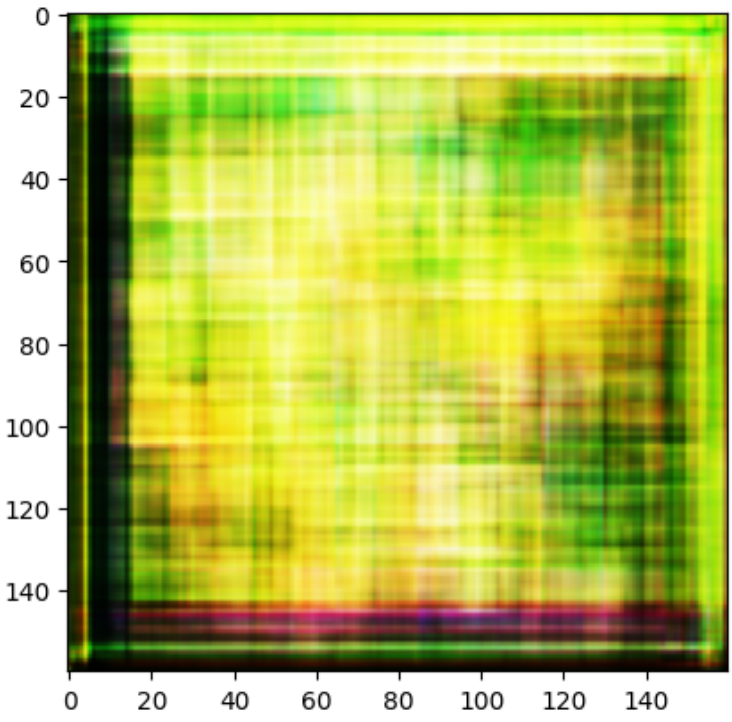

Figure 4.9: Sardi images generated by exp3

In the end, it can be concluded that the obtained results yielded by the suggested Generative Adversarial Network (GAN) are highly encouraging and hold the potential to enhance results significantly, particularly when employed in conjunction with a more robust dataset. By leveraging the power of this advanced machine-learning technique,

researchers and practitioners can harness the strengths of GANs to generate more accurate, realistic, and high-quality outputs. These encouraging results affirm the viability and efficacy of the proposed GAN model, signifying its value in pushing the boundaries of what is achievable in data generation and synthesis.

4.4 Conclusion

In this chapter, we introduced the architecture of the system and discussed in detail the process of preparing and collecting a data set to train the models. Based on the results obtained, it is clear that our models have the potential to revolutionize local sheep breed detection for the better.

Conclusion and future perspectives

A significant aspect of this thesis involves the automated classification of sheep breeds in Algeria. This work contributes to the field of smart farming particularly breed recognition, enhancing our knowledge and capabilities in this domain. To secure the recognition of the sheep breed in, Algeria, This research aims to develop an essential automated detection system to classify the breed. Consequently, we have used supervised and unsupervised deep-learning methods by proposing CNN and GAN-based models that can detect the sheep's breed from pictures The goal of our system is to make it a new or substitute solution to make the recognition of sheep's breed easier with receiving the help of deep learning model that Assist the farmers in the agriculture field. Such systems as an example can make a big difference, they let the farmer's job easier and the breeders will help them to choose the right sheep for the breeding process and avoid the mistakes that came from the familiarity between the sheep of Algeria.

In future work, we hope to consolidate our models with a more accurate and representative dataset. For this reason, we will try to develop a mobile application that helps farmers manage their sheep's livestock. In contrast, these applications will help us improve the local dataset.

In future work, our focus lies in enhancing the accuracy and representativeness of our models by consolidating them with a more precise and comprehensive dataset. To achieve this goal, we are planning to develop a mobile application specifically designed to assist farmers in efficiently managing their sheep's livestock. This application will serve as a valuable tool for farmers, providing them with features and functionalities that streamline various aspects of sheep management, such as tracking health records, monitoring feeding schedules, optimizing grazing patterns, and facilitating timely interventions. By deploying this mobile application in real-world farming scenarios, we anticipate gathering valuable data that will contribute to an improved local dataset. The collected data will encompass a diverse range of sheep-related information, encompassing various breeds, ages, health conditions, and geographical considerations.

This augmented dataset will serve as a vital resource to refine and enhance our models'

capabilities, fostering greater accuracy, robustness, and applicability. We can also propose new models for biometric sheep identification for instance.

Bibliographiy

[4] Chikhi Abdelkader and Ismaïl Boujenane. “Caractérisation zootechnique des ovins de race Sardi au Maroc”. In: *Revue d’Élevage et de Médecine vétérinaire des Pays tropicaux* 56 (Mar. 2003), pp. 187–192.

[7] Ali Borji. “Pros and cons of GAN evaluation measures: New developments”. In: *Computer Vision and Image Understanding* 215 (2022), p. 103329.

[11] Niklas Donges. “Gradient Descent: A Quick, Simple Introduction”. In: *Built In* (July 2021).

[16] Xiaoyao Liang. *Ascend AI Processor Architecture and Programming*. Jan. 2020, pp. 1–40.

[17] Mohammad Motamedi, Nikolay Sakharnykh, and Tim Kaldewey. “A Data-Centric Approach for Training Deep Neural Networks with Less Data”. In: *CoRR* abs/2110.03613 (2021).

[19] Austin Pollard. “What are Neural Networks?” In: *Sensor Review* 10.3 (1990), pp. 115–116.

[20] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016.

[21] Scott Reed et al. “Generative Adversarial Text to Image Synthesis”. In: *33rd International Conference on Machine Learning, ICML 2016*. Vol. 3. May 2016, pp. 1681–1690.

[22] Ali Tezcan SARIZEYBEK and Ali Hakan ISIK. “Detection of Bovine Species on Image Using Machine Learning Classifiers”. In: *Gazi University Journal of Science* (2023).

[24] SuperDataScience Team. “Convolutional Neural Networks (CNN): Step 4 - Full Connection”. In: (Aug. 2018). SuperDataScience — Machine Learning — AI — Data Science Career — Analytics — Success.

[27] Zhuang Wan, Fang Tian, and Cheng Zhang. “Sheep Face Recognition Model Based on Deep Learning and Bilinear Feature Fusion”. In: *Animals* 13.12 (2023).

[28] Zhixiao Wang et al. “Revisiting Recent and Current Anomaly Detection based on Machine Learning in Ad-Hoc Networks”. In: *Journal of Physics: Conference Series*. Vol. 1288. Institute of Physics Publishing, Aug. 2019.

[36] Rikiya Yamashita et al. “Convolutional Neural Networks: An Overview and Application in Radiology”. In: *Insights into Imaging* 9.4 (2018), pp. 611–629.

[37] Aston Zhang et al. “Dive into Deep Learning”. In: *arXiv preprint arXiv:2106.11342* (2021).

Webographiy

[1] https://maferme.ma/elevage-ovins-au-maroc/#La_race_Sardi, 2023.

[2] https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-networ. 2023.

[3] *A Complete Guide to Data Augmentation*. https://www.datacamp.com/tutorial/complete-guide-data-augmentation. 2022.

[5] *Autoencoders in Deep Learning: Tutorial Use Cases [2023]*. https://www.v7labs.com/blog/autoencoders-guide. 2021.

[6] *Basics of Autoencoders*. https://medium.com/@birla.deepak26/autoencoders-76bb49ae6a8f. 2019.

[8] *Brief History of Neural Networks*. https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec. 2019.

[9] Jason Brownlee. *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. 2019.

[10] Jason Brownlee. *Softmax Activation Function with Python*. https://machinelearningmastery.com/softmax-activation-function-with-python/. 2020.

[12] *Elevage ovin au Maroc*. https://www.scala-medi.eu/sheep/,

[13] *Generative Adversarial Network*. https://www.techopedia.com/definition/32515/generative-adversarial-network-gan. 2023.

[14] *Introduction to Convolutional Neural Networks (CNN)*. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-learning. 2023.

[15] *Introduction to Reinforcement Learning for Beginners*. https://www.analyticsvidhya.com/blog/2021/02/introduction-to-reinforcement-learning-for-beginners/.

[18] *Neural Network Models Explained*. https://www.seldon.io/neural-network-models-explained.

[23] *Sigmoid Function*. https://www.learndatasci.com/glossary/sigmoid-function/.

[25] *Supervised Machine Learning*. https://www.javatpoint.com/supervised-machine-learning.

[26] *The Essential Guide to Neural Network Architectures*. https://www.v7labs.com/blog/neural-network-architectures-guide. 2021.

[29] *Weight (Artificial Neural Network)*. https://deepai.org/machine-learning-glossary-and-terms/weight-artificial-neural-network#:~:text=What%20is%20Weight%20(Artificial%20Neural,weight%2C%20and%20a%20bias%20valuee.

[30] *What is a neural network?* https://www.ibm.com/topics/neural-networks.

[31] *What is Artificial Neural Network and How it mimics the Human Brain?* https://medium.com/analytics-vidhya/what-is-artificial-neural-network-and-how-it-mimics-the-human-brain-f92c45564e20. 2019.

[32] *What is Deep Learning and How Does It Works [Explained]*. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-learning.

[33] *What is gradient descent?* https://www.ibm.com/topics/gradient-descent#:~:text=There%20are%20three%20types%20of,and%20mini%2Dbatch%20gradient%20descent.

[34] *What is supervised learning?* https://www.ibm.com/topics/supervised-learning.

[35] *What Is the Necessity of Bias in Neural Networks?* https://www.turing.com/kb/necessity-of-bias-in-neural-networks.