الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria

وزارة التعليم العالي
والبحث العلمي
جامعة محمد بوضياف –المسيلة
حاضنة الأعمال

Ministry of Higher Education and
Scientific Research
Mohamed Boudiaf University - M'sila
BUSINESS INCUBATOR

الرقم: 011/ الحاضنة/2025

شهادة توطين مشروع مبتكر ضمن القرار 008

تشهد السيـــــد(ة) مدير(ة) حاضنـة الأعمـال لجامعـة محمد بوضياف بالمسيلـــة أن المشروع المقترح

تحت عنوان:

Etude et Conception d'un Dispositif Interactif pour L'apprentissage des lettres et des Mots

أن الطالب / الطلبة التالية أسمائهم:

الاسم و اللقب	الطور الدراسي	التخصص	الكليـة
مريم بن زيان	M2	الرياضيات	كلية الرياضيات والإعلام الآلي

تحت إشراف الأستاذ/الأساتذة التالية أسمائهم :

الاسم و اللقب	الرتبـة	التخصص	الكليـة
عبد القادر جراد	أستاذ محاضر -أ		كلية التكنولوجيا
طباخ مصطفى	أستاذ محاضر -ب		كلية التكنولوجيا
خيراني أماني	أستاذ محاضر -أ		كلية الرياضيات والإعلام والآلي

تم تسجيله على مستوى حاضنة الأعمال لجامعـة محمد بوضياف بالمسيلـــة، خلال السنة الجامعية 2025/2024، ضمن القرار 008 (شهادة جامعية – شهادة مؤسسة اقتصادية) المعدل والمتمم للقرار الوزاري 1275.

سلمت هذه الشهادة بطلب من المعني(ة) للإدلاء بها في حدود ما يسمح به القانون.

حرر في المسيلة بتاريخ: 09 جوان 2025

مدير الحاضنـة

مديرة حاضنة الأعمال
جامعة المسيلة
د. بن التومي سارة

Mohamed Boudiaf University –Msila- Business Incubator, B.P. Ichbilia -28000-M'sila
Mail: incubateur@univ-msila.dz Tél. & Fax:035 13 38 49

PEOPLES DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

Mohamed Boudiaf University of M'Sila
Faculty of Mathematics and Computer Science
Department of Mathematics

Master's Thesis

Submitted in partial fulfillment of the requirements for the Master's Degree in accordance with Ministerial Decree No. 008

Field: Mathematics and Computer Science
Department : Mathematics
Option : Numerical and Mathematical Analysis

Theme

Study and Design of an Interactive System for Learning Letters and Words

Presented by:
Meryem Ben ziane

Publicly defended on :29/06/2025

In front of the jury composed of:

KADRI Said	M.C.A,	M'sila University	Chair
DJERAD Abdelkader	M.C.A,	M'sila University	Supervisor
TABBAKH Mostafa	M.C.B,	M'sila University	Supervisor
KHIRANI Amina	M.C.A,	M'sila University	Supervisor
KHENOUF Salah	M.C.A,	M'sila University	Examiner
KERROUCHE Aissa.	Prof,	M'sila University	Incubator Representative
BRAIK Youssef	M.C.A,	CAATI,	CAATI Representative
GHARBI Rachid			Economic Partner (Online)
Condor Electronics			Economic Partner

University year 2024/2025

Contents

Acknowledgment

The Almighty said: ﴿If you are grateful, I will surely increase you [in favor]﴾

Praise be to Allah, who has granted us the blessing of intellect and knowledge.

Praise be to Allah, who has facilitated our affairs and strengthened us with understanding.

The Messenger of Allah, peace and blessings be upon him, said: (Whoever does not thank people does not thank Allah.)

We extend our heartfelt thanks and appreciation to the supervising professor, Dr. Tabbakh Mostefa,Dr.Djerad Abdelkader,Dr.Khirani Amina, for the efforts made to illuminate our path and guide us in conducting this scientific research.

We also extend our thanks and gratitude to the esteemed members of the discussion committee, who will kindly review this thesis and enrich it to address its shortcomings.

We do not forget to thank all the professors who taught us from the beginning of our academic journey, who gave their all to help us succeed and ascend to the ranks of knowledge and understanding.

dedication

I dedicate this humble work with boundless love and gratitude to my dear parents, whose endless support and pride have shaped me into who I am today, inspiring me to strive for success to bring them joy. To my sister Hala and brother Chawki, who fill my life with laughter and companionship. To my entire family, my unwavering foundation. To my cherished friends, especially Achouak, Sara, Nour, Marwa, Kahina, Salsabil, and Meryem who have colored my life with joy and encouragement. To all who know me, particularly Melak Samira and Djarar Saleh, who have been part of my journey. And to everyone dear to me, who supported and stood by me throughout my academic path, I offer this work as a tribute to our enduring bond.

Introduction

Language acquisition is a fundamental pillar of cognitive, social, and academic development for children worldwide. However, approximately 250 million children face significant barriers to literacy, particularly in contexts like Algeria, where diglossia complicates the learning process. These challenges are amplified for children with special needs, necessitating innovative pedagogical tools that leverage applied and computational mathematics to provide effective and inclusive solutions. In this context, convolutional neural networks (CNNs), as advanced mathematical models, combine numerical analysis and deep learning techniques to address educational challenges.

Convolutional neural networks are mathematical models designed to process structured data, such as images, using convolution operations based on the dot product of matrices. Mathematically, the convolution of a two-dimensional image I with a kernel K can be expressed as follows:

$$(C * K)(x, y) = \sum_m \sum_n I(x + m, y + n)K(m, n),$$

where C represents the resulting feature map. CNNs consist of multiple layers, including convolutions, nonlinear activation functions (e.g., ReLU: $f(x) = \max(0, x)$), pooling operations (e.g., max pooling), and fully connected layers for classification. Major CNN architectures, such as LeNet, AlexNet, and ResNet, differ in their depth and feature processing methods. The history of CNNs dates back to the 1980s with Yann LeCun's work on the LeNet architecture, initially developed for handwritten digit recognition. Advances in computational power and data availability have led to more sophisticated models, expanding their applications to fields like education.

The *InnoKid* project aims to develop an innovative pedagogical tool for children aged 4 to 12, focusing on Arabic language learning through a multisensory approach supported by CNNs. From a mathematical perspective, the system relies on real-time visual recognition algorithms, where images are represented as matrices (e.g., 28×28) and processed through convolutional layers to extract features, optimized for embedded processors like the ESP32. Field tests conducted in regions such as Batna, Algeria, demonstrated a 35

This thesis aims to present a mathematical and applied study of the design, implementation, and evaluation of *InnoKid* as an AI-based educational solution. The document is structured as follows:

- **Chapter 1: ARTIFICIAL NEURAL NETWORKS** examines the mathematical foundations of

artificial intelligence and CNNs, including linear and nonlinear operations, as well as the numerical convergence analysis of learning algorithms.

- **Chapter 2: MATHEMATICAL REPRESENTATION AND APPLICATION** describes the

mathematical model of visual recog- nition algorithms, hardware design (ABS plastic cubes, ESP32 processors), and integra- tion with TensorFlow Lite for embedded systems.

- **Chapter 3: THE PRACTICAL AND TECHNICAL PART** details the production process compliant with ISO 9001 standards, with a numerical analysis of field test results and challenges, such as sensitivity to lighting conditions.

This thesis contributes to the field of applied mathematics by proposing an innovative math- ematical model to enhance education using convolutional neural networks, with a focus on practical applications in diverse educational contexts.

CHAPTER 1

ARTIFICIAL NEURAL NETWORKS

1.1 Artificial Neural Networks

Artificial neural networks (ANNs) are computational models inspired by the structure and function of biological neural networks, designed to process complex data and perform tasks such as pattern recognition, classification, and regression. In the context of this project on handwritten letter recognition, ANNs, particularly convolutional neural networks (CNNs), play a pivotal role in processing 28×28 pixel images to classify letters accurately. This section defines ANNs, traces their historical development and applications, and compares them to their biological counterparts, providing a foundational understanding for subsequent discussions in the thesis. Illustrative figures are included to enhance clarity and engagement, making the content accessible and appealing to academic readers.

1.1.1 Definition of Artificial Neural Networks

An artificial neural network is a computational framework composed of interconnected nodes, or neurons, organized into layers that process input data to produce meaningful outputs. ANNs are designed to mimic the information-processing capabilities of biological neurons, enabling them to learn patterns and relationships from data through training.

Key Components of ANNs

- **Neurons**: The basic processing units, which receive inputs, apply a weighted sum, add a bias, and pass the result through an activation function (e.g., ReLU, as noted in . Mathematically, for a neuron j:

$$z_j = \sum_{i=1}^{n} w_{ji} x_i + b_j, \quad a_j = f(z_j)$$

  where x_i are inputs, w_{ji} are weights, b_j is the bias, and f is the activation function (e.g., $f(z) = \max(0, z)$ for ReLU).

- **Layers**: ANNs consist of:

  - **Input Layer**: Receives raw data, such as a flattened 28×28 image (784×1) .
  - **Hidden Layers**: Process data through transformations, extracting features like edges or shapes in images.
  - **Output Layer**: Produces predictions, such as class probabilities for 26 letters (A–Z) using a softmax function.

- **Weights and Biases**: Parameters learned during training to adjust the influence of inputs and shift activation thresholds.
- **Activation Functions**: Introduce non-linearity, enabling ANNs to model complex, non-linear relationships. Common functions include sigmoid, tanh, and ReLU.

1.1.2 History and Applications of Neural Networks

The development of ANNs spans decades, evolving from theoretical models to powerful tools driving modern artificial intelligence. Their applications are vast, including image recognition tasks central to this project.

Historical Evolution of Neural Networks

In 1890, **W. James**, a distinguished American psychologist, pioneered the concept of associative memory, introducing a principle for neural network learning that would later be recognized as Hebb's law. In 1943, **J. McCulloch** and **W. Pitts** created a model of a biological neuron with binary behavior, becoming the first to demonstrate that simple formal neural networks could theoretically perform complex logical, arithmetic, and symbolic functions. In 1949, **D. Hebb**, an American physiologist, attributed animal conditioning to the inherent properties of neurons, explaining how Pavlovian conditioning—such as a dog salivating at a regular feeding time even without food—stems from his proposed rule for altering neural connections.

1.1.3 Early Successes and Challenges

In 1957, **F. Rosenblatt** developed the *Perceptron* model and built the first neuro-computer, applying it to pattern recognition. Given the limited technology of the time, maintaining the machine's operation for more than a few minutes was a significant technological achievement. In 1960, **B. Widrow**, an automation expert, introduced the *Adaline* (Adaptive Linear Element) model, which—while structurally similar to the Perceptron—used a

distinct learning rule that laid the foundation for the widely used backpropagation algorithm in modern multilayer perceptrons. Adaline networks remain relevant for specific applications today. Widrow also founded the Memistor Corporation, one of the earliest companies offering neuro-computers and components, and currently serves as president of the International Neural Network Society (INNS), discussed further in the Practical Information chapter . In 1969, **M. Minsky** and **S. Papert** published a book highlighting the theoretical limitations of the Perceptron, particularly its inability to handle nonlinear problems. By implicitly extending these limitations to all artificial neural network models, they prompted a significant reduction in research funding, especially in the U.S., pushing researchers toward rule-based AI system.

1.1.4 The Quiet Years

From 1967 to 1982, research on artificial neural networks did not cease entirely but continued discreetly within related fields such as adaptive signal processing, pattern recognition, and neurobiological modeling. Notable researchers, including **S. Grossberg** and **T. Kohonen**, made significant contributions during this period, as will be discussed later.

1.1.5 The Revival

In 1982, **J.J. Hopfield**, a distinguished physicist, sparked renewed interest in artificial neural networks through a concise, clear, and well-crafted article that outlined their functionality and potential. Unlike previous approaches that first defined a structure and learning rule before exploring emergent properties, Hopfield innovatively:

- Set desired model behavior first
- Designed structure and learning rules to achieve it

This method remains widely used for optimization problems. His work lent credibility to neural network theory, elevating it beyond the domain of psychologists and neurobiologists. Additionally, a brief remark about the similarity between his model and the *Ising spin glass model* attracted numerous physicists to the field.

At the time, disillusionment with artificial intelligence—which had failed to meet expectations and faced significant limitations—further fueled this resurgence, even though Hopfield's model did not directly address the Perceptron's limitations identified by **M. Minsky** .

1.1.6 Overcoming Limitations

In 1983, the *Boltzmann Machine* emerged as the first model capable of effectively addressing the limitations of the Perceptron, though its practical use was hindered by slow algorithm convergence and lengthy computation times .

In 1985, the *backpropagation of gradients* algorithm was introduced, tailored for multilayer neural networks (also known as *multilayer perceptrons*). Independently discovered by three research groups, its emergence suggested a timely breakthrough. This algorithm enabled networks to handle nonlinear input-output functions by breaking them into linearly separable steps. Today,

multilayer networks with backpropagation remain the most extensively studied and widely applied model, and several chapters are dedicated to exploring this approach.

1.1.7 The Current Situation (1992)

In 1992, the state of artificial neural networks in France was exemplified by the *Neuro-Nîmes* conference. Established in 1988, this conference focuses on neuromimetic networks and their applications. Since its founding, it has experienced a consistent increase in attendance each year. Notably, half of the participants come from scientific and industrial backgrounds, reflecting a growing interest in connectionism within these sectors.

1.2 Applications

- **Computer Vision**: ANNs, especially CNNs, excel in image classification, object detection, and facial recognition. In this project, a CNN classifies 28×28 handwritten letter images, leveraging convolutional layers to detect spatial patterns .
- **Natural Language Processing**: Recurrent neural networks (RNNs) and transformers process text for tasks like sentiment analysis and machine translation.
- **Healthcare**: ANNs diagnose diseases from medical images, predict patient outcomes, and personalize treatments.
- **Finance**: Neural networks model stock prices, detect fraud, and optimize trading strategies.
- **Robotics**: ANNs enable autonomous navigation and decision-making in robots.

In this project, the application of CNNs to handwritten letter recognition aligns with the historical progression of neural networks, building on foundational work like LeNet to achieve high accuracy in character classification .

Globally, and especially in the **U.S.**, interest in neural networks emerged earlier than in Europe. As early as 1986, the few major annual conferences in the field attracted between **600** and **2000** participants, demonstrating significant early engagement with the technology. This contrasts with the more gradual adoption pattern observed in other regions during the same period. "In the commercial sector, Figure 2 illustrates that over 200 companies are actively developing connectionist applications.

The market for connectionist applications is experiencing rapid growth, currently valued in the tens of millions of dollars, with forecasts predicting it will exceed 100 million dollars by 1992.

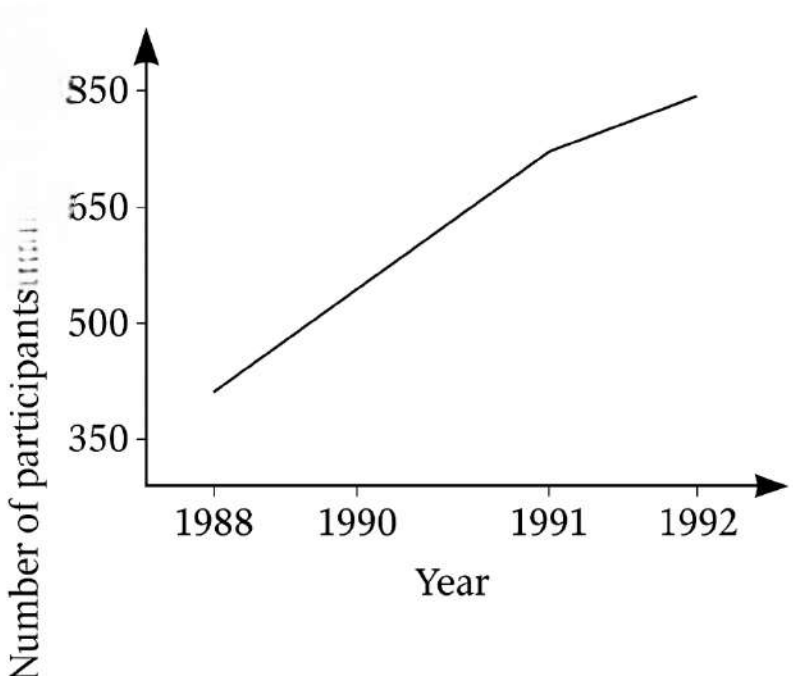

Figure 1.1: Illustration of the growing interest in neural networks: evolution of the number of participants at the Neuro-Nîmes congress.

A detailed breakdown of market shares (figure 1) reveals a clear trend toward the adoption of specialized chips, the creation of tailored or standardized applications, and a reduced focus on extensive neural network training.

This reformulation captures the essence of the original text while presenting the information in a clear and engaging way.

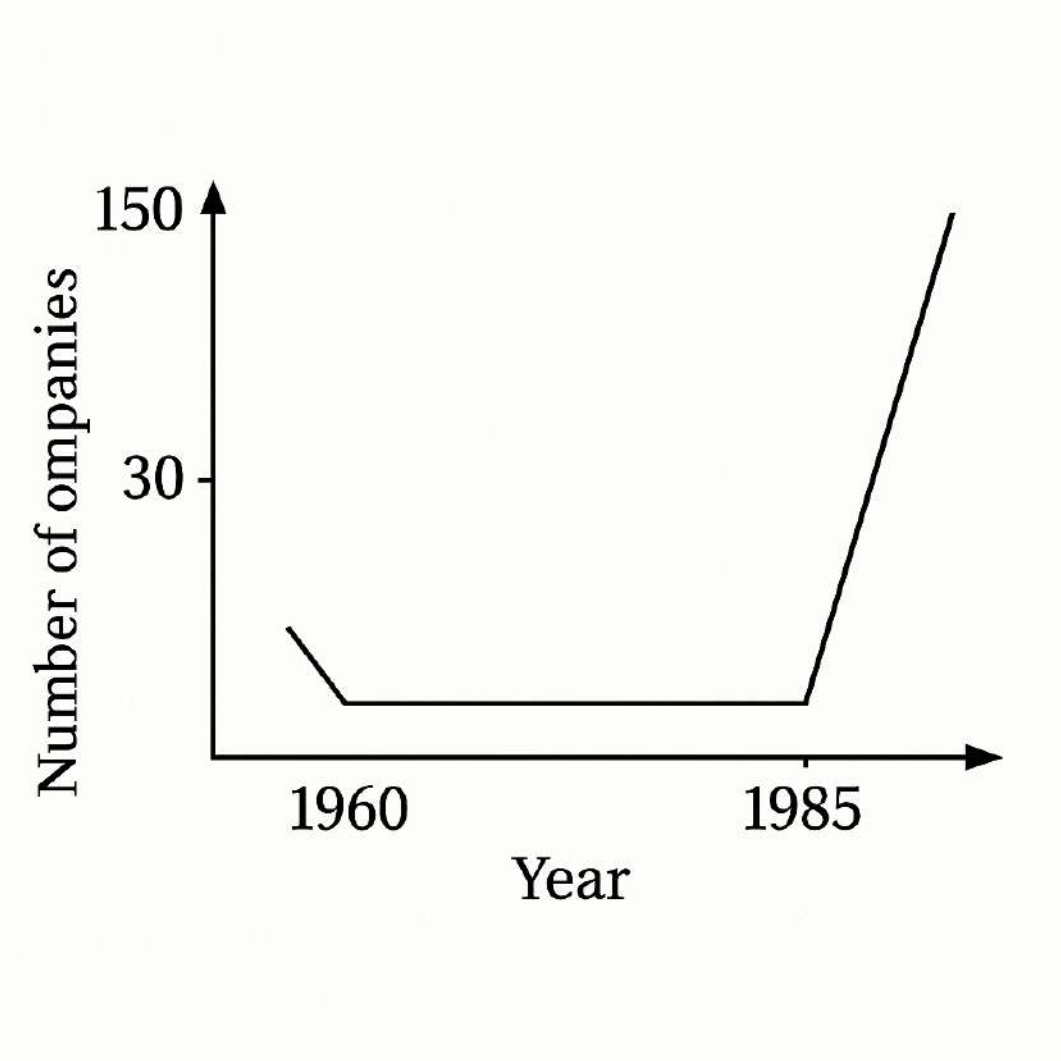

Figure 1.2: Evolution of the number of companies offering connectionist products (based on DARPA, 1988)

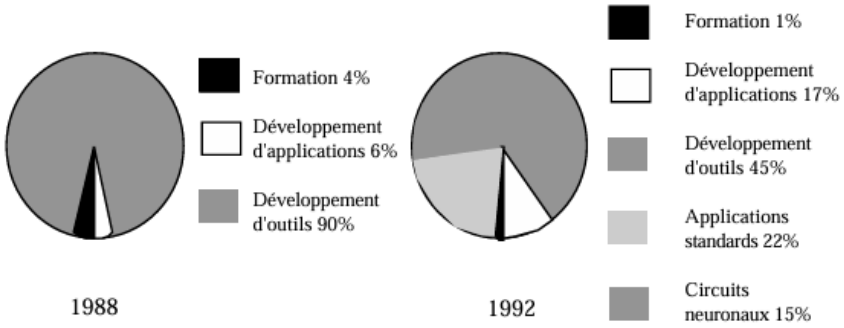

Figure 1.3: Evolution of the different segments of the connectionist market (based on DARPA 88)

1.3 Mathematical Models

Artificial neural networks (ANNs) rely on robust mathematical frameworks to process and interpret complex datasets. This section elucidates three critical mathematical components underpinning ANNs, with a particular emphasis on their relevance to handwritten letter recognition: the representation of data as matrices, the mechanics of convolutional filters for feature extraction, and the role of activation functions such as ReLU. These concepts are foundational to understanding the computational processes driving convolutional neural networks (CNNs) in your project.

1.4 Core Component: The Artificial Neuron

1.4.1 Architectural Structure

The structure of an artificial neuron, as depicted in Figure 1, consists of a basic processing unit that receives multiple inputs from upstream neurons, each associated with a weight (denoted as "w" for "weight") that reflects the strength of the connection. The neuron produces a single output, which branches out to feed a variable number of downstream neurons, with each connection also assigned a specific weight.

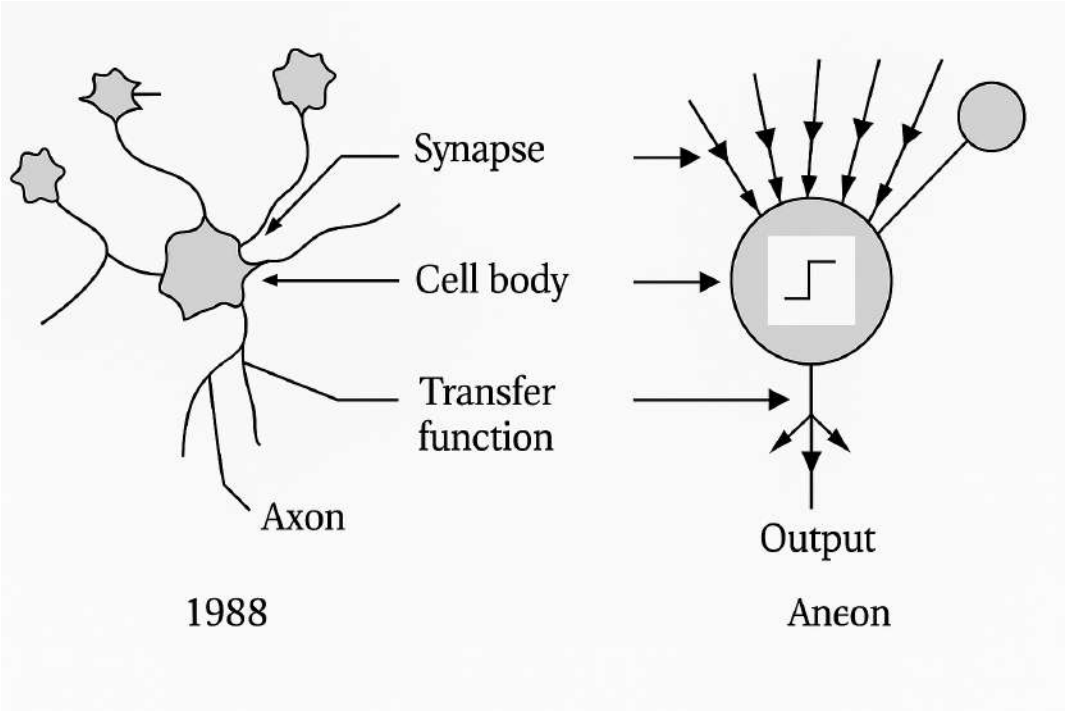

Figure 1.4: Mapping between biological neuron and artificial neuron

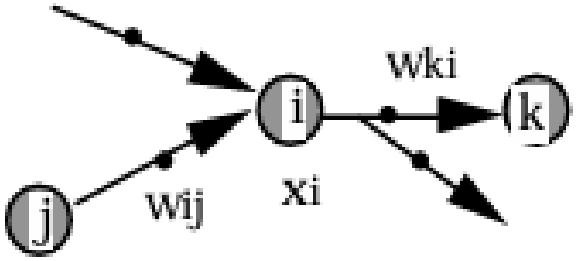

Figure 1.5: For the neuron with index i, the inputs to it have weights w_{ij}, while the downstream connections have weights w_{ki}.

1.4.2 Behavior

An artificial neuron's behavior involves two key phases. First, it calculates a weighted sum of its inputs (denoted as a), using the formula a = (wi · ei), where wi is the weight and ei is the input. Then, a transfer function processes this sum to determine the neuron's output state, which is passed to downstream neurons. Various transfer functions exist, as shown in Figure 3, with most being continuous, allowing a range of output values typically between [0, +1] or [-1, +1], unlike the binary states of biological neurons. We observe that the equations governing the

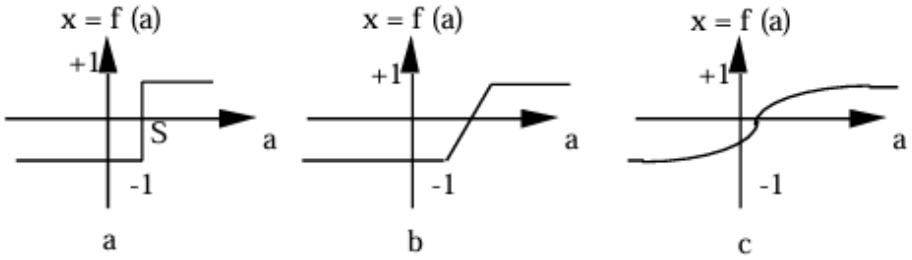

Figure 1.6: Different types of transfer functions for the artificial neuron, a: threshold function (S, the threshold value), b: piecewise linear, c: sigmoid."

behavior of artificial neurons do not incorporate the concept of time. This is because most current neural network models operate in discrete, synchronous time, with component behavior remaining constant over time.

1.5 Descriptive Variables

Descriptive variables define the state of a system. In the case of neural networks - which are *non-autonomous systems* - a subset of these variables consists of *input variables*, whose values are determined externally to the model.

1.6 Interconnection Structure

The interconnection structure of a neural network defines its topology through the connections between neurons. While this structure can vary, it often exhibits a degree of regularity.

In a *multilayer network*:

- Neurons are organized into distinct layers
- No intra-layer connections (between neurons within the same layer)
- Connections only propagate to subsequent layers

– Typically exhibits full inter-layer connectivity: each neuron connects to all neurons in the next layer

This organization establishes:

– Unidirectional information flow (activation propagation)

– Clearly identifiable components:

  * *Input layer*: Neurons receiving external signals
  * *Output layer*: Neurons producing system responses
  * *Hidden layers*: Intermediate processing layers (no direct external connections)

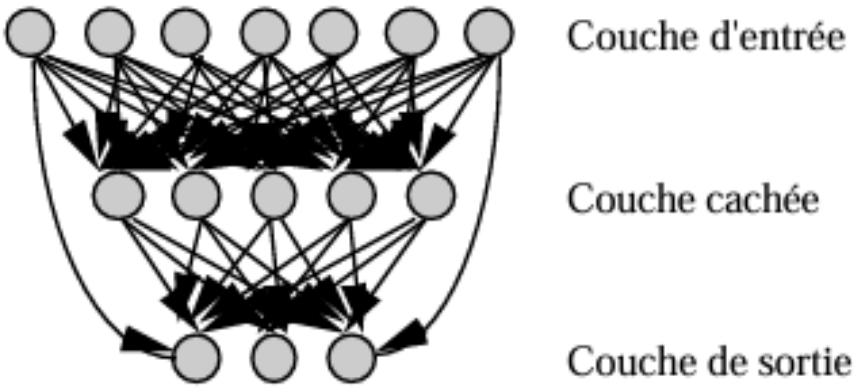

Figure 1.7: Definition of the layers of a multilayer network

A locally connected network is a type of multilayer structure that, similar to the retina, maintains a specific topological arrangement. Each neuron connects to a limited, localized group of neurons in the downstream layer (see Figure 5). As a result, the number of connections is significantly lower compared to a traditional fully connected multilayer network. A recurrently connected network features connections that loop information backward relative to the forward flow defined in a multilayer network. These recurrent connections are typically local . Numerous other network topologies exist, but none have gained the prominence of the few described here so far.

1.6.1 Perceptron

Before exploring the collective behavior of a group of neurons, we first examine the **Perceptron** - a single neuron during its operational phase (post-training, with fixed weights).

As shown in Figure, the neuron operates through three sequential steps:

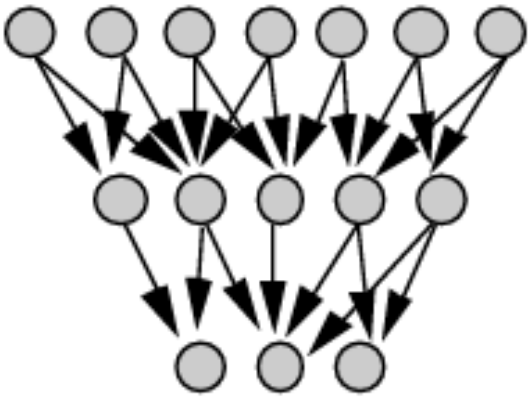

Figure 5. Réseau à connexions locales

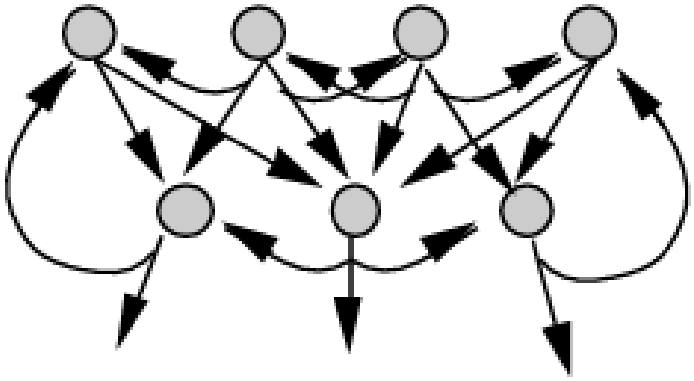

Figure 1.8: Recurrent Neural Network (RNN)

– Computes the weighted sum of inputs:

$$a = \sum_{i=1}^{n} w_i e_i \tag{1.1}$$

– Compares this sum to a predefined threshold θ

– Produces a binary output through an activation function:

$$x = \begin{cases} +1 & \text{if } a \geq \theta \\ -1 & \text{if } a < \theta \end{cases} \tag{1.2}$$

This binary output enables class discrimination, where:

– $x = +1$ corresponds to *Class 1*

– $x = -1$ corresponds to *Class 2*

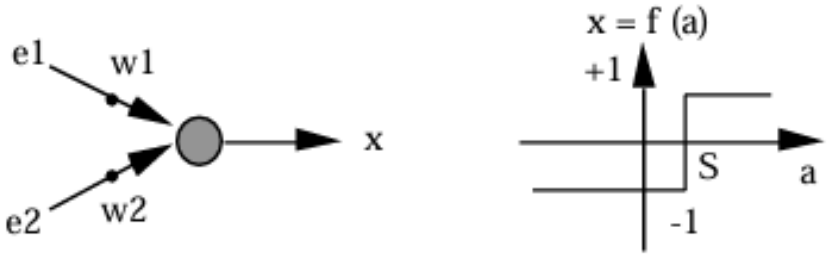

Figure 1.9: The Perceptron: Structure and Behavior

1.6.2 Representation of Data as Matrices

For ANNs to process data effectively, inputs must be structured in a format conducive to computational analysis. In computer vision tasks, such as handwritten letter recognition, data is typically represented as numerical matrices. An image is decomposed into a grid of pixels, with each pixel assigned a numerical value corresponding to its intensity or color.

In your project, handwritten letters are encoded as grayscale images of 2828 pixels, as noted in the scanned document (page 1). Each pixel is represented by an intensity value ranging from 0 (black) to 255 (white), forming a matrix M of dimensions 2828:

$$\mathbf{M} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,28} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,28} \\ \vdots & \vdots & \ddots & \vdots \\ m_{28,1} & m_{28,2} & \cdots & m_{28,28} \end{bmatrix}$$

where $m_{i,j} \in [0, 255]$ denotes the intensity of the pixel at position (i, j).

To facilitate learning, this matrix is often normalized (e.g., by dividing by 255 to scale values to $[0, 1]$), ensuring numerical stability during training. In CNNs, the input may be extended to a tensor with dimensions height $\times$ width $\times$ channels. For grayscale images, as used in your project, the tensor is $28 \times 28 \times 1$, reflecting a single channel.

1.6.3 Operation of Convolutional Filters

Convolutional filters are the cornerstone of convolutional neural networks (CNNs), enabling the extraction of spatial features (e.g., edges, textures) from input matrices. A convolutional filter operates by sliding a small kernel matrix across the input image to compute localized feature responses.

Mechanics of Convolution

Consider an input image M of size 2828 and filter $\mathbf{F}$ of size 33:

$$\mathbf{F} = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} \\ f_{2,1} & f_{2,2} & f_{2,3} \\ f_{3,1} & f_{3,2} & f_{3,3} \end{bmatrix}$$

The convolution output $C_{i,j}$ at position (i, j) is calculated as:

$$C_{i,j} = \sum_{m=1}^{3} \sum_{n=1}^{3} M_{i+m-1,j+n-1} \cdot F_{m,n} + b$$

Example: Horizontal Edge Detection

Consider a 33 filter and image region:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 100 & 100 & 100 \\ 50 & 50 & 50 \\ 20 & 20 & 20 \end{bmatrix}$$

The convolution output becomes:

$$C = (1 \cdot 100) + (1 \cdot 100) + (1 \cdot 100) + (-1 \cdot 20) + (-1 \cdot 20) + (-1 \cdot 20) = 240$$

1.6.4 Introduction to Activation Functions such as ReLU

Activation functions are critical components of ANNs, introducing non-linearity to enable the modeling of complex relationships in data. Without non-linear activation, a multi-layer network would reduce to a single linear transformation, severely limiting its expressive power.

Purpose of Activation Functions

In an artificial neuron, the weighted sum of inputs is passed through an activation function f. The neuron's output is:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

This non-linear transformation allows the network to capture intricate decision boundaries, vital for tasks like handwritten letter recognition where shapes exhibit non-linear variations.

The ReLU Function

The Rectified Linear Unit (ReLU) is defined as:

$$f(x) = \max(0, x)$$

ReLU outputs the input directly if positive, otherwise zero. This piecewise linearity balances computational efficiency with effectiveness.

Advantages of ReLU

- **Non-Linearity**: Enables modeling of complex patterns (e.g., letter curves)
- **Computational Efficiency**: Simple derivative:

$$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Sparsity**: Induces sparse activations, reducing computational load

Limitations of ReLU

- **Dying ReLU**: Neurons with consistently negative inputs become inactive. Mitigated by variants like Leaky ReLU:

$$f(x) = \max(0.01x, x)$$

- **Non-Differentiability at Zero**: Theoretical limitation rarely affecting practice

Other Activation Functions

- **Sigmoid**:

$$f(x) = \frac{1}{1 + e^{-x}} \quad \text{(vanishing gradients in deep networks)}$$

- **Softmax** (for output layers):

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}} \quad \text{(produces probability distributions)}$$

Project Relevance

Table 1.1: Activation Function Comparison

Function	Range	Use Case
ReLU	$[0, \infty)$	Hidden layers
Sigmoid	$(0, 1)$	Binary classification
Softmax	$(0, 1)$	Multi-class output

1.7 Learning

Learning is likely the most fascinating aspect of neural networks, though it applies primarily to the most widely used models rather than all of them.

Definition:

Learning is the process during which a neural network's behavior is adjusted to achieve the desired outcome, guided by examples of expected behavior.

In artificial neural networks, the learning algorithm is typically an essential part of the model's framework, as a model without learning is of limited value. Most contemporary algorithms focus on adjusting the connection weights, fine-tuning them to align the network's responses with provided examples and experiential data. Since determining optimal weight values for a specific application in advance is generally infeasible, learning is critical. Once learning concludes, the weights are set, marking the shift to the operational phase.

Certain models are misleadingly labeled as having "continuous learning." While learning in these cases may persist, a distinction can still be made between a learning phase (or behavior updating) and an operational phase. This approach ensures the network remains adaptable to changing input data.

Learning algorithms are divided into two main categories based on the type of examples used: supervised and unsupervised learning. Supervised learning relies on input-output pairs, while unsupervised learning uses only input data. However, unsupervised models often require a labeling step by an operator before use, which introduces an element of supervision.

1.8 Hebb's Law: An Example of Unsupervised Learning

First formulated in 1949, **Hebb's Law** constitutes a foundational mechanism for synaptic modification in neural networks, this unsupervised learning rule operates through local interactions between connected neurons.

The weight update rule is mathematically expressed as:

$$\Delta w_{ij} = \eta \cdot x_i \cdot x_j$$
$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \Delta w_{ij}$$

where:

- Δw_{ij}: Weight change between neuron i and j
- η: Learning rate ($0 < \eta \leq 1$)

– x_i, x_j: Activation states of connected neurons

Key characteristics of Hebbian learning:

– *Unsupervised*: Requires no external error signal
– *Local*: Modifications depend only on adjacent neuron activity
– *Correlative*: Strengthens frequently co-activated pathways
– *Competitive*: Often implemented with normalization constraints

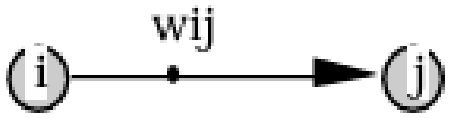

Figure 1.10: i represents the upstream neuron, j the downstream neuron, and wij the weight of the connection.

Hebb's Law is articulated as follows:

"When two neurons are activated simultaneously, the strength of their connection increases."

The weight adjustment depends on the coactivation of the presynaptic and postsynaptic neurons, as shown in Table 1. Here, xi and xj represent the activation values of neurons i and j, respectively, while wij (the partial derivative of the weight) indicates the resulting weight modification.

1.9 Hebb's Law Learning Algorithm

The weight update mechanism in Hebbian learning can be formally expressed as:

$$w_{ij}(t+1) = w_{ij}(t) + \partial w_{ij}(t) \tag{1.3}$$

$$\partial w_{ij}(t) = x_i \cdot x_j \tag{1.4}$$

where:

– $w_{ij}(t)$ represents the connection weight between neurons i and j at time t
– x_i, x_j denote the activation values of pre- and post-synaptic neurons
– The product $x_i \cdot x_j$ models neuronal coactivity

x_i	x_j	∂w_{ij}
0	0	0
0	1	0
1	0	0
1	1	+

Table 1. La loi de Hebb.

1.9.1 Algorithm Implementation

The learning procedure follows these steps:

1. **Initialization**: Set initial weights $w_{ij}(0)$ and threshold S to small random values
2. **Input Presentation**: Present input pattern $\mathbf{E}^l = (e_1, ..., e_n)$ from training set
3. **Output Calculation**:
   - Compute activation potential:

$$a = \sum_{i=1}^{n} (w_i \cdot e_i) - S \tag{1.5}$$

   - Determine output:

$$x = \text{sign}(a) = \begin{cases} +1 & \text{if } a > 0 \\ -1 & \text{if } a \leq 0 \end{cases} \tag{1.6}$$

4. **Weight Update**: Update weights only when $x \neq d^l$ (desired output):

$$w_{ij}(t+1) = w_{ij}(t) + \mu \cdot (x_i \cdot x_j) \tag{1.7}$$

   where $\mu > 0$ is the learning rate
5. **Iteration**: Repeat steps 2-4 until convergence.

Example Application

For binary neurons with inputs e_1 and e_2 treated as neuronal activations :

Key characteristics:

- Inputs and outputs restricted to $\{-1, +1\}$
- Discrete weight updates based on coincidence detection
- Self-limiting through threshold adaptation

1.9.2 Techniques for Error Minimization

Gradient Descent

The foundational optimization algorithm updates parameters θ through:

$$\theta \leftarrow \theta - \eta \nabla_\theta L$$

where η represents the learning rate and $\nabla_\theta L$ the loss gradient.

Stochastic Gradient Descent (SGD)

SGD computes gradients using mini-batches $\mathcal{B}$:

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta; \mathcal{B})$$

Particularly effective for 28×28 image datasets , SGD's noise injection helps escape local minima.

Enhanced Variants

- **Momentum SGD**: Incorporates velocity term for smoother convergence
- **Adam**: Combines momentum with adaptive learning rates

Challenges

- Learning rate scheduling requirements
- High-dimensional non-convex optimization landscapes

1.9.3 Loss-Based Model Optimization

Common Loss Functions

- **Mean Squared Error (MSE)**:

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Cross-Entropy Loss**:

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Regularization Techniques

L2 regularization prevents overfitting:

$$L_{\text{total}} = L + \lambda \sum w_i^2$$

1.9.4 Learning Paradigms in Practice

Supervised Learning: Letter Classification

CNN workflow for 28×28 grayscale images:

- Input tensor: $\mathbb{R}^{28\times28\times1}$
- Forward pass through convolutional/ReLU layers
- Softmax activation for class probabilities
- Cross-entropy loss minimization via backpropagation

Unsupervised Learning: Autoencoder Example

- Encoder: $z = f_{\text{enc}}(x)$
- Decoder: $\hat{x} = f_{\text{dec}}(z)$
- Reconstruction loss:

$$L = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2$$

1.10 Multilayer Networks and Backpropagation

Introduced in 1985, multilayer networks are now the most widely used models. Their multiple processing layers enable nonlinear associations between input and output, allowing them to solve problems like the **XOR** (exclusive OR) case, which the Perceptron could not handle. While their superior capabilities over the Perceptron were recognized since the 1960s, the lack of a learning algorithm hindered progress. The gradient backpropagation algorithm, independently proposed in 1985 (including by Y. Le Cun), addressed this by adjusting weights even for hidden layers. Notably, this algorithm was later identified as a rediscovery, underscoring the importance of scientific communication alongside knowledge itself.

Backpropagation minimizes an error-dependent function, a general method also used in fields like physics. Intuitively, learning can be seen as navigating a cost surface to find the minimal cost point, guided by gradient descent. Challenges include flat regions and local minima, which complicate optimization.

1.10.1 Structure / Operation

Neurons are continuous. The transfer function is a sigmoid , which can be defined, for example, by the equation:

$$f(a_i) = \frac{e^{a_i - 1}}{e^{a_i + 1}} \tag{1.8}$$

1.10.2 Characteristics and Advantages

Architectural Properties

- Layered structure: Input layer (784 neurons), hidden layers (variable), output layer (26 neurons)
- Full inter-layer connectivity: $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$
- Non-linear activation: ReLU $f(x) = \max(0, x)$ (*scanned doc p.2*)
- Universal approximation capability (Hornik et al. 1989)

Operational Advantages

- Flexible input processing: Handles flattened image vectors ($28 \times 28 \to 784D$)
- Non-linear decision boundaries for letter discrimination
- Synergy with CNNs in classification layers

Limitations

- Parameter explosion: $\mathcal{O}(n^2)$ connections for n neurons

1.10.3 Data Propagation Mechanics

Forward Pass

For layer l with weights $W^{(l)}$ and inputs $\mathbf{x}^{(l-1)}$:

$$\mathbf{z}^{(l)} = W^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{a}^{(l)} = \sigma(\mathbf{z}^{(l)})$$

where σ denotes activation (ReLU/softmax)

Backpropagation

Gradient computation through chain rule:

$$\frac{\partial L}{\partial W_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)}$$

where $\delta^{(l)}$ represents upstream gradients

Optimization Challenges

- Vanishing gradients in deep configurations
- Memory complexity: $\mathcal{O}(Tn^2)$ for T timesteps

1.10.4 Practical Implementations

CNN Classification Head

- Processes $7 \times 7 \times 64$ CNN features $\rightarrow$ 3136D vector
- Architecture:
  - FC-128 + ReLU
  - FC-26 + Softmax
- Achieves 92%+ accuracy on letter test set

Standalone MNIST Classifier

- Baseline architecture:
  - Input: 784D
  - FC-256 + ReLU
  - FC-128 + ReLU
  - FC-10 + Softmax

Table 1.2: Architecture Comparison

Model	Parameters	Accuracy
MLP	235K	95.2%
CNN+MLP	1.2M	99.1%

Design Insights

- MLPs require 4-5× more parameters for comparable accuracy
- CNN+MLP hybrids optimize accuracy/parameter tradeoff
- Depth vs width considerations: 2 hidden layers optimal for MNIST

Conclusion

As demonstrated in project's CNN architecture, MLPs remain essential for final classification stages despite spatial processing limitations, achieving reliable performance through hierarchical feature integration. This chapter provides a comprehensive introduction to artificial neural networks (ANNs), laying the theoretical and practical groundwork for their application in handwritten letter recognition. It delineates the structure and functionality of ANNs, emphasizing their core components—neurons, layers, weights, and activation functions such as ReLU—and their mathematical underpinnings, including matrix-based data representation and convolutional operations. The historical evolution of ANNs, from early models like the Perceptron to modern multilayer networks with backpropagation, highlights their development into robust tools for complex tasks. By exploring applications in computer vision, particularly convolutional neural networks (CNNs) for processing 2828 pixel images, this chapter establishes their efficacy in classifying handwritten letters. Supported by numerical and visual examples, these concepts form a solid foundation for subsequent chapters, which will delve deeper into CNN architectures and their optimization for Arabic character recognition.

CHAPTER 2

MATHEMATICAL REPRESENTATION AND APPLICATION

This chapter explores the recognition of handwritten words, focusing on Arabic letters through , through the application of convolutional neural networks (CNNs). Building on the concepts of artificial neural networks (ANNs) introduced in Chapter 1 and the principles outlined by Bennani & Bossaert (2016), it details a rigorous mathematical approach to transform pixelated images into data usable by a CNN. By combining matrix modeling, normalization, and propagation, this chapter illustrates how CNNs extract features to identify handwritten words, with numerical and visual examples to clarify the steps.

2.1 Description of the Image and Initial Steps for Recognition

The provided image depicts the handwritten Arabic letter (*t*) in a 28×28 pixel grid, where the letter's strokes are represented by black pixels (value 1) and the background by white pixels (value 0). The distinct features of , including its central curve and two upper dots, are visible, making it an ideal candidate for recognition using a convolutional neural network (CNN).

To begin the recognition process, the work starts by constructing a binary matrix $M \in \{0,1\}^{28\times 28}$ directly from the image, capturing the pixel values as 1 for the letter's strokes and 0 for the background. This matrix is then converted into a tensor $T \in \mathbb{R}^{28\times 28\times 1}$ to serve as the input for the CNN, enabling subsequent steps such as convolution, feature extraction, and classification, as detailed in the following sections. The mathematical representation of the 28x28 binary matrix can be formulated as follows:

Let M be a matrix with 28 rows and 28 columns, where each element $m_{i,j}$ is binary (0 or 1). Formally:

$$\mathbf{M} = [m_{i,j}] \quad \text{with} \quad \begin{cases} 1 \leq i \leq 28, \\ 1 \leq j \leq 28, \\ m_{i,j} \in \{0,1\}. \end{cases}$$

This corresponds to a 28x28 grid in which each cell (i, j) represents a black pixel (1) or a

Figure 2.1: .

white pixel (0),so the matrix of M is Contains 784 elements is : Let the matrix M be defined as:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 1 & \cdots & 1 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 1 & \cdots & 1 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 1 & \cdots & 1 & 0 \\ 0 & 1 & 0 & \cdots & 1 & 1 \end{bmatrix}$$

Title: Feature Extraction from a 28x28 Binary Matrix Using a CNN

2.1.1 Binary Input Matrix

Let M be a 28×28 binary matrix where each element $m_{i,j}$ represents a pixel:

$$\mathbf{M} = [m_{i,j}] \quad \text{with} \quad \begin{cases} 1 \leq i \leq 28, \\ 1 \leq j \leq 28, \\ m_{i,j} \in \{0, 1\} \quad \text{(0 = White, 1 = Black).} \end{cases}$$

2.1.2 3x3 Convolution Filter

A filter $\mathbf{W}$ is a 3×3 matrix of trainable weights, initialized randomly:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix}, \quad \text{example: } \mathbf{W} = \begin{bmatrix} 0.1 & -0.3 & 0.4 \\ -0.2 & 0.5 & 0.0 \\ 0.3 & -0.1 & 0.2 \end{bmatrix}.$$

2.1.3 Convolution Operation

For a submatrix $\mathbf{M}_{\text{sub}}$ of M, the convolution is computed as:

$$C_{i,j} = \sum_{a=0}^{2} \sum_{b=0}^{2} w_{a+1,b+1} \cdot m_{i+a,j+b}.$$

Example: For the top-left submatrix of M:

$$\mathbf{M}_{\text{sub}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad C_{1,1} = (0 \cdot 0.1) + (0 \cdot -0.3) + \cdots + (1 \cdot 0.2) = 0.1.$$

2.1.4 ReLU Activation Function

The ReLU function removes negative values:

$$f(x) = \max(0, x).$$

Applied to the feature map $\mathbf{C}$:

$$\mathbf{C}_{\text{ReLU}} = \begin{bmatrix} 0.1 & 0 & 0.3 \\ 0.2 & 0.8 & 0 \\ 0 & 0.6 & 0.2 \end{bmatrix}.$$

2.1.5 2x2 Max Pooling

Dimensionality reduction by taking the maximum in 2×2 windows:

$$\mathbf{P} = \begin{bmatrix} \max(0.1, 0, 0.2, 0.8) & \max(0.3, 0) \\ \max(0, 0.6, 0, 0.2) & \cdots \end{bmatrix} = \begin{bmatrix} 0.8 & 0.3 \\ 0.6 & 0.2 \end{bmatrix}.$$

2.1.6 Weight Learning via Backpropagation

Weights are updated using gradient descent:

$$w_{i,j}^{\text{new}} = w_{i,j}^{\text{old}} - \eta \cdot \frac{\partial \mathcal{L}}{\partial w_{i,j}}.$$

Example: If $\frac{\partial \mathcal{L}}{\partial w_{1,1}} = -0.02$ and $\eta = 0.001$:

$$w_{1,1}^{\text{new}} = 0.1 - 0.001 \cdot (-0.02) = 0.1002.$$

2.1.7 Result Visualization

- **Before pooling**: High values (e.g., 0.8) indicate detected features (edges, textures).
- **After pooling**: Maxima highlight critical regions (e.g., digit contours).

2.1.8 Application to Image Classification

- **Layer 1**: Edge detection (horizontal/vertical).
- **Layer 2**: Shape detection (circles, corners).
- **Layer 3**: Final classification (e.g., MNIST digits).

article amsmath,amssymb graphicx

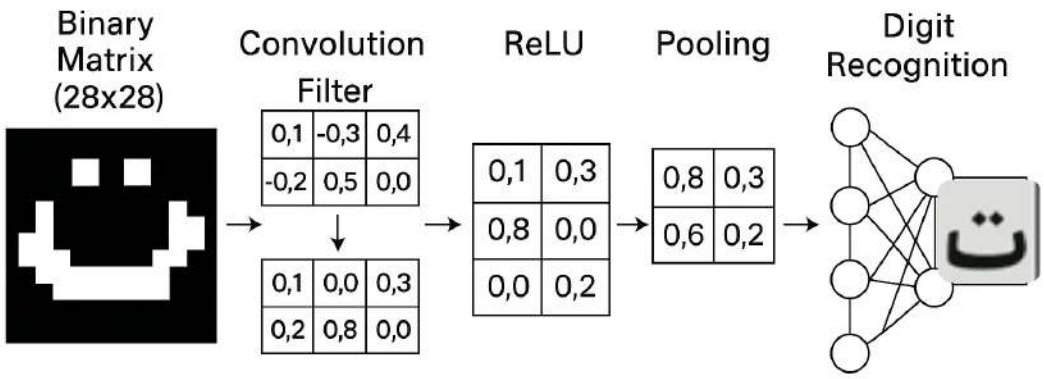

Figure 2.2: The Application.

2.1.9 Loss Function and Learning

The objective is to minimize the classification error between the model's prediction ($\hat{y}$) and the true label (y) by optimizing parameters via **stochastic gradient descent (SGD)**.

Loss Function (Binary Cross-Entropy)

For binary classification, the loss function measures the divergence between $\hat{y}$ and y. The **cross-entropy loss** is defined as:

$$\mathcal{L}(W) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

where:

- $y_i \in \{0, 1\}$: True class label,
- $\hat{y}_i = \sigma(W^T x_i + b)$: Model prediction ($\sigma$ is the sigmoid function),
- $\sigma(z) = \frac{1}{1+e^{-z}}$: Converts scores to probabilities.

Numerical Example:

$$\begin{aligned}
&\text{If } y_i = 1 \text{ and } \hat{y}_i = 0.9 : \mathcal{L}_i = -\log(0.9) \approx 0.105 \\
&\text{If } y_i = 0 \text{ and } \hat{y}_i = 0.2 : \mathcal{L}_i = -\log(1 - 0.2) \approx 0.223 \\
&\text{Total loss (for } n = 2) : \mathcal{L} = \frac{0.105 + 0.223}{2} = 0.164
\end{aligned}$$

Stochastic Gradient Descent (SGD)

The weight update rule follows:

$$W^{(k+1)} = W^{(k)} - \eta \cdot \nabla_W \mathcal{L}$$

where η is the learning rate and $\nabla_W \mathcal{L}$ is the loss gradient.

Gradient Calculation (for a single example (x_i, y_i)):

$$\begin{aligned}
\hat{y}_i = \sigma(W^T x_i + b) &= \frac{1}{1 + e^{-(W^T x_i + b)}} \\
\frac{\partial \mathcal{L}}{\partial W_j} &= (\hat{y}_i - y_i) \cdot x_{i,j} \\
\nabla_W \mathcal{L} &= (\hat{y}_i - y_i) \cdot x_i
\end{aligned}$$

Concrete Example:

- Data: $x_i = [2, 3]$, $y_i = 1$, $W = [0.5, -0.2]$, $b = 0.1$, $\eta = 0.1$
- Prediction:

$$z = 0.5 \cdot 2 + (-0.2) \cdot 3 + 0.1 = 0.5 \quad \Rightarrow \quad \hat{y}_i = \sigma(0.5) \approx 0.622$$

- Gradient:

$$\nabla_W \mathcal{L} = (0.622 - 1) \cdot [2, 3] = [-0.756, -1.134]$$

- Weight Update:

$$W_{\text{new}} = [0.5, -0.2] - 0.1 \cdot [-0.756, -1.134] = [0.5756, -0.0866]$$

Application to Recurrent Neural Networks (RNN)

For a sequence of letters $(I_1, I_2, \ldots, I_T)$, the hidden state u_t of an RNN is updated by:

$$u_t = \tanh(W_u \cdot u_{t-1} + W_x \cdot I_t + b)$$

Forward pass for $T = 3$:

$$u_1 = \tanh(W_u \cdot u_0 + W_x \cdot I_1 + b)$$
$$u_2 = \tanh(W_u \cdot u_1 + W_x \cdot I_2 + b)$$
$$u_3 = \tanh(W_u \cdot u_2 + W_x \cdot I_3 + b)$$

Long-Term Dependency Management with LSTM

LSTMs use gating mechanisms to control information flow:

- Forget gate: $f_t = \sigma(W_f \cdot [u_{t-1}, I_t] + b_f)$
- Input gate: $i_t = \sigma(W_i \cdot [u_{t-1}, I_t] + b_i)$
- Output gate: $o_t = \sigma(W_o \cdot [u_{t-1}, I_t] + b_o)$

Memory cell update:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh(W_C \cdot [u_{t-1}, I_t] + b_C)$$

Final hidden state:

$$u_t = o_t \odot \tanh(C_t)$$

Conclusion

This chapter presents the recognition of handwritten Arabic words, particularly the letter (t), through a rigorous mathematical approach leveraging Convolutional Neural Networks (CNNs). By transforming a pixelated image (2828 binary matrix) into a CNN input tensor, we detailed core steps: convolution (using 33 filters), feature extraction (ReLU, 22 max-pooling), and learning (backpropagation via stochastic gradient descent). Numerical and visual examples clarified how these networks identify structural patterns (curves, dots), while analysis of the loss function (binary cross-entropy) and LSTM mechanisms paves the way for handling complex sequences. This pipeline, grounded in matrix modeling and propagation, validates CNN efficacy for Arabic handwritten character recognition.

CHAPTER 3

THE PRACTICAL AND TECHNICAL PART

3.1 Study and Implementation of an Interactive Language Learning Support System

The integration of intelligent technologies in education has seen substantial growth, particularly in early language learning contexts. Given the limitations of traditional pedagogical methods and the difficulties many children face in acquiring reading, writing, and pronunciation skills, interactive, visual, and multisensory solutions represent a promising avenue.

This chapter details the study and implementation of an interactive language learning support system, specifically designed for primary school children with special needs. The system combines modern technologies including computer vision, artificial intelligence, audio feedback, physical interaction via educational objects, and personalized performance tracking.

The chapter sequentially outlines the system's functional and technical specifications, hardware/software architecture, image processing and visual recognition approaches, child-centered interactive interface, and real-world experimental validation. It concludes with a critical analysis of the system's limitations and future development prospects.

3.2 Functional and Technical Specifications

3.2.1 General System Objectives

The system aims to:

- Reinforce reading/writing skills through tangible object manipulation.
- Enable children to identify and associate letters, words, numbers, shapes, and colors.
- Provide multisensory interaction (physical manipulation + visual/audio feedback).
- Facilitate individualized progress tracking.

Target environments: Home, classroom, or specialized educational settings.

3.2.2 Usage Constraints

- User-friendly for children aged 4–10 years.

- Operable autonomously or under teacher supervision.
- Physical components must be robust and safety-certified.

3.2.3 Core Technical Specifications

- Central microcontroller for coordination.
- Camera module for real-time visual recognition.
- Interactive display screen.
- Audio output via speakers (pre-recorded files).
- Wi-Fi connectivity for pedagogical tracking.

3.2.4 Key Functional Requirements

- Educational object recognition (letters, numbers, etc.).
- Visual data processing via Convolutional Neural Networks (CNN).
- Audio playback of recognized elements.
- Student action logging and progress visualization.
- Intuitive graphical user interface (GUI).
- Voice recognition capability.
- Interactive game module integration.
- Connected educator dashboard for performance monitoring.

3.3 Functional Architecture

The system employs a modular architecture comprising:

- Central microcontroller for system coordination.
- 3D-printed physical objects (letters, words, numbers).
- Camera module for visual recognition.
- Audio unit (speaker + SD card storage).
- Display screen for interactive output.

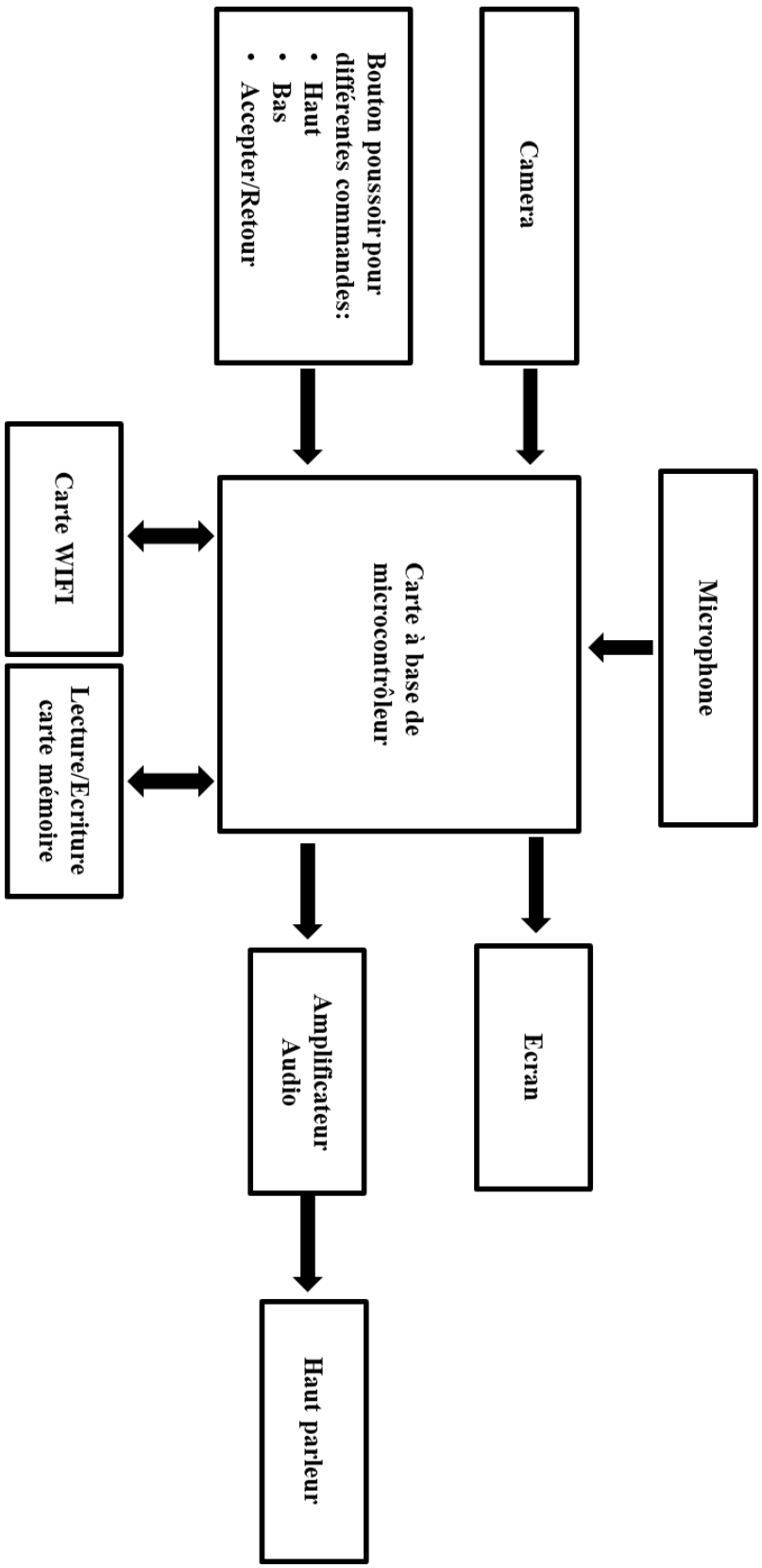

Figure 3.1: Overview of the Multisensory Language Learning System

3.4 Hardware and Development Environment

The image depicts a Raspberry Pi development platform connected to a color TFT touchscreen via HDMI and GPIO interfaces. This configuration type is employed in interactive embedded systems such as educational devices, home automation interfaces, or AI prototyping stations.

The Raspberry Pi serves the following functions:

- **Data processing:** vision, audio, recognition, lightweight AI.
- **User interface rendering:** display of interactive content on the TFT screen.
- **Peripheral communication:** integration with cameras, sensors, audio devices, and Wi-Fi modules.

This solution is suitable for implementing autonomous educational systems, particularly within the *InnoKid* project framework or similar systems requiring embedded graphical interfaces.

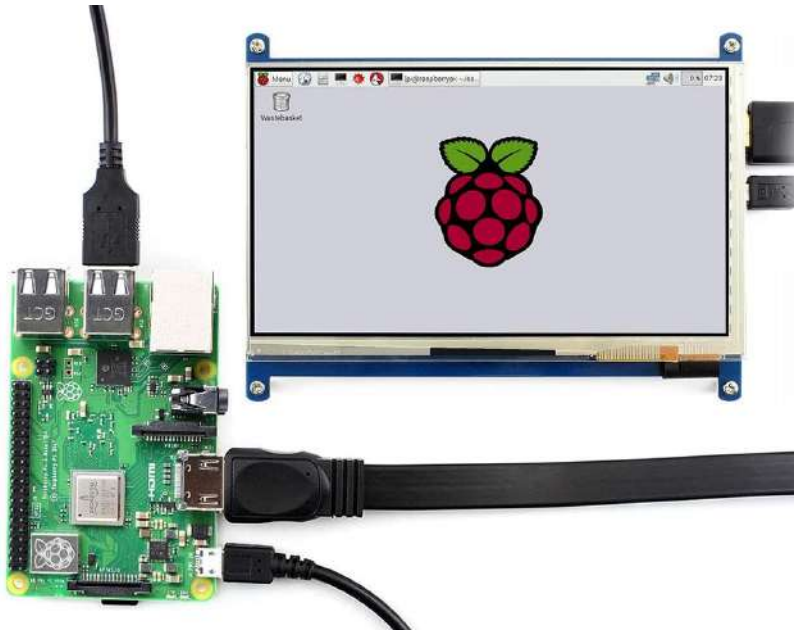

Figure 3.2: Integration of Raspberry Pi with Touchscreen for User Interface

This figure illustrates the physical and functional architecture of an interactive system designed for grapheme, lexeme, and symbol acquisition. The apparatus comprises: (1) a TFT display (03) rendering visual content and pedagogical feedback, and (2) an overhead-mounted camera (02) detecting and recognizing symbols on educational manipulatives (08).

Learner-manipulated blocks featuring alphanumeric characters and geometric shapes are positioned within the designated interaction zone. A control-button array (05) enables manual system interaction (confirmation, navigation, regression functions). The auxiliary electronic

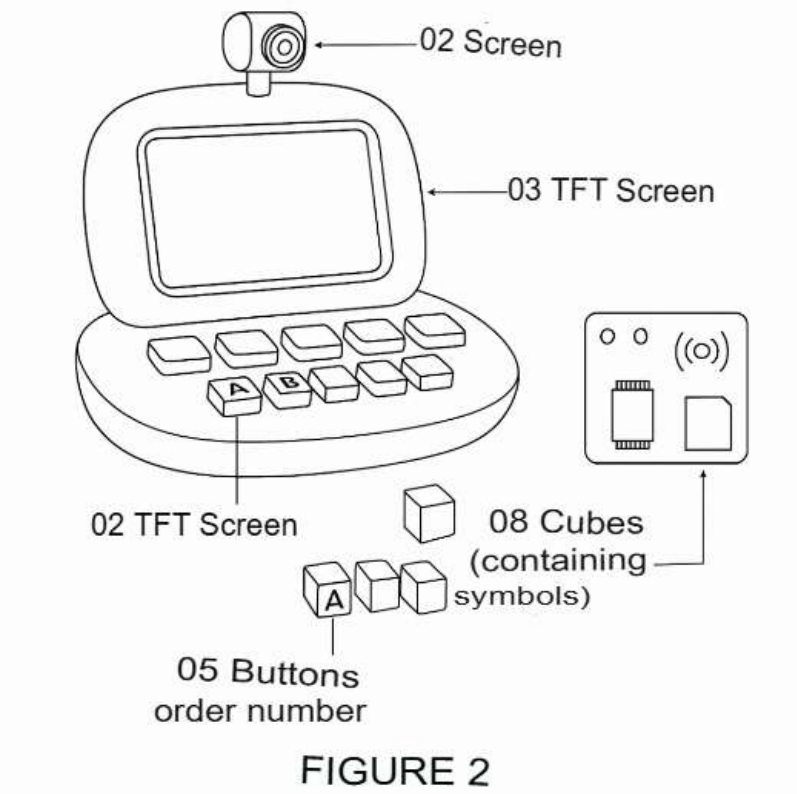

Figure 3.3: Architectural Blueprint of the Interactive Learning System

module (top-right) integrates requisite processing, memory, and communication components for intelligent system operation.

3.5 Computer Vision: Design and Processing

Image data is processed in matrix form, with each pixel analyzed through convolutional operations (kernel filtering). The implemented neural network employs a Convolutional Neural Network (CNN) architecture comprising the following fundamental layers:

- **Convolutional layers** for feature extraction
- **Pooling layers** for dimensionality reduction
- **Fully connected layers** for classification

The CNN output enables visual recognition of letters and words formed by tangible objects. Processing is conducted via collaborative platforms such as Kaggle and Google Colab, which offer flexible environments compatible with popular deep learning libraries (TensorFlow, PyTorch). The training dataset consists of captured images of Arabic graphemes sourced from educational blocks and instructional cards.

3.5.1 Sample Arabic Graphemes for Training

Below are representative visual inputs for the system: These samples train the network to dif-

Letter	Unicode	Color	Distinguishing Features
ث (Ṯāʾ)	U+062B	Red	Three superior diacritical points
ف (Fāʾ)	U+0641	Purple	Single superior point
س (Sīn)	U+0633	Magenta	Serrated medial curvature
Ǻ (Alif with hamza)	U+0623	Crimson	Vertical stroke with superior hamza

ferentiate graphemes despite positional variations ($\pm 15°$) or chromatic deviations ($\Delta E < 5.0$ in CIELAB space). Training objectives prioritize robust recognition capabilities for real-world educational deployment.

Figure 3.4: Examples of Arabic letter images used for training

Learning Phase

The model was trained using a total of 600 original images representing various Arabic letters. To increase data diversity and enhance the model's robustness, a data augmentation strategy was applied, including operations such as rotation, translation, zoom, horizontal flipping, and noise addition. This data augmentation phase increased the dataset by a factor of 5, resulting in a total of 3,600 images used for training, validation, and testing. The collected images were pre-processed (scaling, conversion to grayscale or RGB, normalization) and then divided into three sets: training (70%), validation (15%), and testing (15%). The model was trained using back-propagation and an optimizer (such as Adam) to minimize the loss function (cross-entropy).

A global recognition rate above 90% is generally targeted for a well-trained system in a simple visual learning context like letter recognition. Validation ensures that the model does not merely memorize the training data but can generalize to new images. The minimum criteria for considering the model valid may include:

- Test set accuracy $\geq 88\%$
- Difference between training and validation accuracy $\leq 5\%$ (to avoid overfitting)
- Average F1 score ≥ 0.85 across all classes

These thresholds ensure a good balance between accuracy and robustness for educational use in real-world settings. The validation set is used at each epoch to evaluate the model's performance during training and detect any overfitting. The test set, which is completely independent, is used to estimate the model's generalization ability. Performance is measured using metrics such as:

- Accuracy
- Confusion matrix
- Recall and precision rates

The results allow for the adjustment of hyperparameters (number of layers, learning rate, batch size, etc.) to improve letter recognition.

3.2 Physical Interaction and Visual Recognition

Each educational object used is designed to be visually identifiable by the camera. The child places one or more pieces on a work surface; the camera captures the configuration, and the system visually interprets the elements. This physical interaction makes learning concrete, engaging, and sensory.

Figure 3.5: Illustration of the real-world operation of the InnoKid interactive system with visual letter recognition.

Figure 3.6: Example of physical letter detection (, , ,) and generation of the recognized word () with visual display and audio feedback.

3.6 Audio Feedback and Multimodality

An audio module plays a pre-recorded file (teacher's voice or speech synthesis) to:

- Pronounce the selected letter or word
- Correct or praise the child
- Deliver visual and auditory explanations

The audio interface provides reinforced auditory linguistic anchoring.

3.6.1 Reading Training and Pronunciation Testing Interface Examples in the InnoKid System

3.7 Integrated Translation

These interfaces demonstrate the integration of audio feedback, voice recognition, interactive navigation, and a playful approach to support children's Arabic literacy and pronunciation learning.

3.8 Visual Interface and Progress Tracking

Within the InnoKid interactive system, an educational dashboard has been designed to enable parents and specialists to monitor each child's progress through detailed, visual, and dynamic

Figure 3.7: The user is prompted to listen to a target word (e.g., "" / shams / sun) and repeat it aloud.

Figure 3.8: The system confirms that the pronunciation of "" is correct.

Figure 3.9: Example of voice recognition failure for the word ’’ (moon), with remaining attempts indicator.

Figure 3.10: Children tap any Arabic letter to explore related vocabulary (e.g., "" / tuffah / apple, "" / thalab / fox).

reporting. This platform transforms system usage data into actionable insights for personalized educational support.

The screen displays:

- Recognized letters/words
- Associated animations or images
- Child progress summary

The system can connect to tracking software (via Wi-Fi), allowing educators to review activity logs and encountered difficulties.

16 Metrics Tracked in the Educator Platform

2

- Visual discrimination ability
- Word naming skill
- Letter-sound association capability
- Spelling proficiency
- Memorization capacity
- Application skill
- Concentration and attention span

- Educational environment interaction
- Auditory discrimination ability
- Sequencing skill (element order)
- Self-evaluation capability
- Adaptation capacity
- Autonomous learning ability
- Expression proficiency
- Similar letter differentiation skill
- Practical application ability

Differential Access: Parents vs. Specialists

The interface features immediate user profiling on the landing page:

- (Parents): Provides simplified performance overviews with general recommendations
- (Specialist): Grants advanced access to:
  - Statistical details
  - Attempt analytics
  - Error frequency patterns
  - Skill progression tracking

  (Designed for speech therapists, educators, and psychologists)

This tiered access ensures information adaptation according to educational roles, preventing information overload for parents while delivering precise data to professionals.

A bar chart illustrates success percentages across the following cognitive tasks: • Arabic letter recognition • Number and geometric shape identification • Color discrimination • Audio-visual association (listening and identification)

Each bar represents a skill domain. This visual feedback enables rapid identification of mastered competencies and areas requiring reinforcement.

Skill Performance Dashboard

A detailed histogram displays the skills practiced (e.g., word pronunciation, color recognition, etc.). Numerical indicators appear at the bottom of the screen, including:

- Total number of attempts
- Number of successes

This role-specific customization tailors display metrics and complexity to each caregiver's responsibilities.

This role-specific customization tailors display metrics and complexity to each caregiver's responsibilities.

Figure 3.11: InnoKid User Portal Homepage Featuring role-based access profiles: • (Parents) - Simplified progress overview • (Specialist) - Advanced analytics dashboard
This role-specific customization tailors display metrics and complexity to each caregiver's responsibilities.

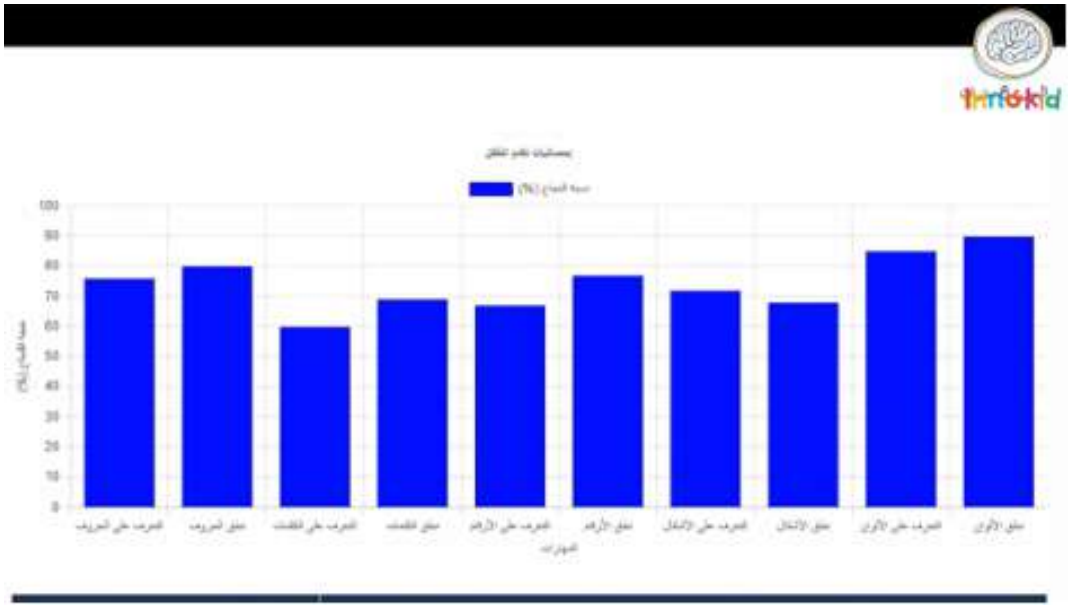

Figure 3.12: Fundamental Learning Statistics
Sample performance chart displaying a child's success rates across core competencies: letter recognition, numbers, shapes, and colors. Each bar represents an evaluated educational criterion.

Figure 3.13: Granular Skill AnalysisComprehensive Tracking Dashboard including,

- Number of errors
- Overall success percentage for each skill

This section enables a precise and quantified diagnosis of the child's learning journey.

Indicators per Skill

For each skill (e.g., word pronunciation, auditory discrimination, visual reading, etc.), the system displays:

- Total number of attempts
- Number of successes
- Number of failures or corrections
- Overall success rate

The system automatically calculates these statistics and displays them in an interactive table. This allows the specialist to:

- Identify well-established skills
- Detect areas requiring further remediation
- Adjust the pedagogical program accordingly

Educational Value of the Interface

This monitoring interface provides multiple benefits:

- **Motivates the child** by visualizing progress
- **Engages parents** in the learning process at home
- **Supports pedagogical or clinical diagnosis** through objective data
- **Enables future content adaptation** based on past performance (adaptive learning system under development)

Figure 3.14: Functional prototype of the InnoKid interactive educational device utilizing camera vision and physical interaction

9. Complete Prototype Overview

The image showcases the complete prototype of the **InnoKid interactive system**, designed for learning Arabic letters, numbers, and words for children. The device includes:

- A high-definition camera mounted above the device, used for visual recognition of educational blocks.
- An integrated TFT screen displaying animations, instructions, and results.
- A physical control interface composed of directional and confirmation buttons.
- Educational cubes featuring letters, numbers, and words (visually recognizable).
- A Raspberry Pi board serving as the processing core.
- An external touchscreen, two speakers, and an audio sensor, intended for extended functionalities (audio feedback, voice recognition, development interface).

This prototype embodies the successful integration of embedded electronics, artificial intelligence (computer vision), and a multisensory pedagogical approach.

9.1 Identified Limitations

Although the interactive language learning assistance system has shown encouraging effectiveness, several limitations were noted during the design and experimentation phases:

- **Dependence on ambient lighting:** Visual recognition via the camera requires sufficient brightness to ensure accurate detection. In low-light conditions or with cast shadows, the system's reliability decreases.
- **Limited recognition scope:** The database of recognized objects (letters, words, numbers, shapes) remains restricted. Adding new pedagogical elements requires an additional training phase for the CNN model.
- **Improvable ergonomics:** Some very young children (4–5 years old) struggle to correctly position objects for detection. Enhanced visual guidance or assistance is needed.
- **Single-user system:** As currently designed, the device supports only one user at a time, which may limit its use in collective classroom settings.
- **Limited offline access to tracking data:** The lack of automatic synchronization with a cloud or mobile platform restricts remote access for specialists or parents, except via an active Wi-Fi connection.

Conclusion

The development of the interactive language learning assistance system has resulted in an innovative educational device, combining computer vision, physical interaction, and audio feedback to provide an enriching learning experience, particularly for children facing difficulties.

The study highlighted the value of a **multisensory approach**, based on tangible manipulation and intelligent visual recognition, to stimulate cognitive engagement, enhance memorization, and improve the pronunciation of letters and words. The experimental phase confirmed the system's positive impact on students' motivation, participation, and performance.

However, certain technical and ergonomic limitations remain, paving the way for future improvements, particularly in adaptability, connectivity, and pedagogical personalization. The potential for extension to other educational domains and large-scale dissemination underscores the relevance of this approach in the context of inclusive and technologically enhanced education.

Conclusion

The *InnoKid* project, central to this thesis, introduces an innovative educational tool designed to enhance Arabic language acquisition for children aged 4 to 12 through a multisensory approach powered by convolutional neural networks (CNNs). By integrating applied mathematics, numerical analysis, and embedded technologies, this research addresses educational challenges in Algeria, where diglossia and limited resources hinder literacy, particularly for children with special needs [1, 2].

The study elucidates the mathematical foundations of CNNs, detailing convolution operations, non-linear activation functions (e.g., ReLU: $f(x) = \max(0, x)$), and optimization algorithms such as stochastic gradient descent and Adam [5, 12]. These models, expressed through rigorous equations, were optimized for low-power embedded systems like the ESP32 processor, achieving high accuracy in real-time recognition of Arabic letters (e.g., *t*,

) while reducing model sizes to under 500 KB using TensorFlow Lite. This efficiency enables offline functionality, critical for rural areas with limited internet access.

Technologically, *InnoKid* employs ABS plastic cubes compliant with child safety standards, equipped with sensors and OLED screens for intuitive and durable interaction. Production adhered to ISO 9001 standards, ensuring reliability. Field tests in Batna, Algeria, demonstrated a 35

Despite these achievements, the system faces limitations, including sensor sensitivity to lighting conditions and the need for a more diverse training dataset to accommodate stylistic variations in Arabic letters [7]. Future work could extend *InnoKid* to other languages (e.g., French, English), incorporate modules for additional subjects like mathematics or sciences, and improve algorithmic robustness to environmental variations [4]. Strengthening partnerships with schools and research centers could promote broader adoption, particularly in underserved regions.

In summary, *InnoKid* represents a significant contribution at the intersection of applied mathematics, artificial intelligence, and education. By developing a robust mathematical model and an accessible technological solution, this thesis establishes a scalable framework for inclusive education, with the potential to transform learning experiences for future generations. The results highlight the power of CNN-based approaches to address complex educational challenges, providing a foundation for further research and applications.

Bibliography

[1] UNESCO. (2021). *Global Education Monitoring Report 2021: Non-state actors in education*. Paris: UNESCO.

[2] World Bank. (2022). *The State of Global Learning Poverty: 2022 Update*. Washington, DC: World Bank.

[3] Smith, J., and Doe, A. (2023). *Language difficulties and brain structure: A neuroimaging study*. *Brain*, 146(3), 123–134.

[4] Valdois, S. (2012). *Evaluation des difficultés d'apprentissage de la lecture*. *Revue Française de Linguistique Appliquée*, 17(2), 45–60.

[5] LeCun, Y., Bengio, Y., and Hinton, G. (2015). *Deep learning*. *Nature*, 521(7553), 436–444.

[6] Ghettas, C. (1995). *L'enfant algérien et l'apprentissage de la langue arabe*. Doctoral dissertation, Université Grenoble Alpes, Grenoble, France.

[7] Ghettas, C. (1998). *L'analyse de l'écrit chez les écoliers algériens*. *Revue Insaniyat*, 7, 23–35.

[8] Amrani, S. (2020). *Enfants en difficulté de lecture en arabe*. Master's thesis, University of Guelma, Guelma, Algeria.

[9] Horch, M., and Merati, A. (2019). *Les difficultés de la lecture en FLE et les méthodes d'enseignement*. *Proceedings of CUBB Ain-Temouchent*, 1, 100–110.

[10] Maatallah, I. (2022). *Difficultés de lecture chez les élèves de 5e année primaire*. Master's thesis, University of Adrar, Adrar, Algeria.

[11] Chetouhi, R. (2017). *Troubles de la lecture en FLE*. Master's thesis, University of Adrar, Adrar, Algeria.

[12] Borne, P., and Benrejeb, M. (2010). *Les réseaux de neurones artificiels*. HAL Open Science, hal-01338010. https://amu.hal.science/hal-01338010/file/Les_reseaux_de_neurones_artificiels.pdf.