الجمـــهورية الـــــجزائرية الديمقـــــراطية الشعـــــــبية
République Algérienne Démocratique et Populaire
وزارة التعليـــــــــم العالـــــــــي و البحث العلمـــــي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université 20 Août 1955 – Skikda		جـامعة 20 أوت 1955- سكيكدة
Faculté des Sciences		كـــــلية العلـــــوم
Departement d'informatique		قسم اعـــــلم اللـــي

Mémoire De fin d'étude en vue de l'obtention du Diplôme de Master en Informatique

Option : Réseaux et systèmes distribués

Sujet :

Conception et réalisation d'une application de correction automatique des copies d'examen QCM

Réalisé par les étudiants :	Dirigé par :
❖ Benhamidcha Romaissa ❖ Hammouda Khawla	❖ Dr Zeghida Djamel

Soutenu le Jeudi 26/06/2025, devant le jury composé de :

Dr Ramdane Ch	Université de Skikda	Président
Dr Bouchaour N	Université de Skikda	Examinateur
Dr Kissoum Y	Université de Skikda	Invité

Année Universitaire 2024-2025

Remerciement

*Tout d'abord, nous adressons nos louanges et notre gratitude à **Dieu** Tout-Puissant, qui nous a accordé la force, la santé et la patience pour mener à bien ce travail, malgré les défis rencontrés.*

*Nous tenons à exprimer nos remerciements les plus sincères à notre encadrant, **Dr. Zeghida Djamel** , pour son accompagnement, sa disponibilité et ses conseils pertinents tout au long de notre projet. Son encadrement bienveillant et rigoureux a été une véritable source de motivation et d'inspiration.*

*Un merci tout particulier également au **Dr.Boucheham** , qui nous a grandement soutenus et guidés pendant les moments où notre encadrant n'était pas disponible. Son aide généreuse et son expertise ont joué un rôle clé dans l'avancement de ce travail.*

*Nous remercions également les **membres du jury** pour l'intérêt qu'ils ont porté à notre mémoire, ainsi que pour leurs remarques constructives et enrichissantes.*

*Nous exprimons toute notre reconnaissance à **nos familles**, qui ont toujours cru en nous, nous ont soutenus moralement et encouragés avec amour tout au long de ce parcours. Leur patience et leurs sacrifices silencieux ont été notre force.*

Enfin, on remercie tous ce qui ont contribué de près ou de loin à

l'élaboration de ce modeste travail

A tous,'MERCI'

Romeissa et khawla

Dédicaces

À mon père,
Toi qui m'as appris que le silence vaut parfois plus que mille mots,
Que l'effort et l'honneur ne font jamais de bruit,
Ce mémoire est une graine semée grâce à ton exemple.

À ma mère,
Pour chaque regard réconfortant, chaque prière murmurée dans l'ombre,
Pour cet amour pur et sans mesure…
Ce chemin est le fruit de ta lumière.

À ma sœur adorée,
Complice de tous les instants, tu es ce sourire qui m'encourage, cette voix qui me rassure. Merci d'avoir été là dans les hauts comme dans les bas.

À mes frères,
Vos encouragements, vos sourires et votre présence ont mis de la lumière sur mon chemin. Merci pour votre amour sincère.

À mon encadreur,
Merci pour votre patience, votre bienveillance et vos conseils précieux. Votre accompagnement a été un repère tout au long de ce travail.

À ma camarade de route,
Ce mémoire est aussi le tien . Ensemble, on a partagé le stress, les doutes, les idées folles… mais aussi les rires, les réussites et les souvenirs inoubliables. Merci d'avoir été là.

Ce travail est un hommage à chaque personne qui m'a tendu la main, un bout de cœur à ceux qui m'aiment. Merci pour tout.

Romeissa

Dédicaces

Avec tous mes sentiments de respect, avec l'expérience de ma reconnaissance, je dédie ma remis de diplôme et ma joie

À mon paradis, à la prunelle de mes yeux, à la source de ma joie et mon bonheur, ma lune et le fil d'espoir qui allumer mon chemin, ma moitié,

Maman,

À celui qui m'a fait une femme, ma source de vie, d'amour et d'affection, à mon support qui était toujours à mes côtés pour me soutenir et m'encourager, à mon prince papa,

À ma sœur, et mon frère pour leur soutien moral Et leur intérêt envers notre travail,

À mon amie d'enfance sirine, complice de mes premiers rêves et souvenirs, pour son affection fidèle et son soutien sincère.

À notre encadrant, pour l'accompagnement qu'il a su nous offrir, dès nos premières réflexions jusqu'à la remise de notre mémoire. Ses remarques, suggestions et sa bonne humeur nous ont été d'un grand soutien tout au long du processus de rédaction.

À mon binôme, merci pour chaque moment de fatigue, de stress et de pression que nous avons traversé ensemble, et pour ton engagement constant dans la réussite de ce projet. Nous avons créé des souvenirs inoubliables qui resteront gravés dans nos esprits.

À tous ce qui ont qui ont participé à ma réussite et a tous qui m'aiment.

Khawla

Table des matières

Remerciement..... 2
Table des matières..... 5
Table des figures..... 10
Table des Tableaux..... 12
Introduction générale..... **13**

Chapitre 01:Reconnaissance des formes et vision par ordinateur

1. Introduction..... 16
2. Généralités sur les traitements d'images..... 15
   2.1. Introduction..... 15
   2.2. Image..... 15
   2.3. Pixel..... 15
   2.4. L'image numérique..... 16
   2.5. Histogramme..... 17
   2.6. Notion de contours dans une image..... 18
   2.7. Textures..... 19
   2.8. Images à niveaux de gris..... 19
   2.9. Le traitement d'image..... 20
3. La reconnaissance des formes (RDF)..... 20

3.1. Processus de reconnaissance..... 21
   3.1.1 Acquisition des données (numérisation)..... 22
   3.1.2 Le prétraitement..... 22
   3.1.3 Extraction des caractéristiques (calcul des représentations)..... 23
   3.1.4 L'apprentissage..... 23
   3.1.5 Classification automatique..... 23
   3.1.6 Post-traitement..... 27

3.2. Domaines d'application..... 27
   3.2.1 Vision par ordinateur..... 27
   3.2.2 La reconnaissance vocale..... 28
   3.2.3 Reconnaissance de texte et d'écriture..... 28
      3.2.3.1. La reconnaissance d'écriture manuscrite..... 29

3.2.3.1.1. Définition..... 29

3.2.3.1.2. Différents aspects de la reconnaissance automatique de l'écriture manuscrite..... 31

3.2.3.1.2.1. Mode d'acquisition (Enligne / Hors-ligne).....31

a. Systèmes de reconnaissance hors ligne.....31
b. Systèmes de reconnaissance en ligne.....31
3.2.3.1.2.2. Approches de reconnaissance de l'écriture manuscrite.....32

a. L'approche Globale ou Holistique.....32

b. L'approche analytique ou locale.....32
4. Conclusion..... 32

Chapitre 02 :Correction automatique des feuilles d'examen QCM

1. Introduction..... 33
2. Méthodes existantes pour la correction des QCM.....33
2.1. Correction semi-automatisée..... 34
2.2.1 Systèmes de Reconnaissance Optique de Marques (OMR)..... 34
2.2. Correction automatique par OCR traditionnels.....35
2.3. Limites et difficultés de ces méthodes..... 35
3. Méthodes modernes de correction automatique.....36
3.1. Applications du deep learning à la reconnaissance et à l'analyse de documents..... 36
3.2. Approche hybride et automatisée du projet OptiQcm..... 37
4. Conclusion.....37

Chapitre 3 : Classifieur basé sur le deep learning : CNN

1. Introduction..... 39
2. Définition et principe des CNN..... 39
3. Architecture fonctionnelle d'un CNN..... 40
3.1. La couche de convolution.....40
3.2. La couche de correction ReLU..... 41
3.3. La couche de pooling.....41
3.4. Couche Fully Connected (FC).....42
4. Les différentes architectures d'un CNN.....43
4.1. LeNet.....43
4.2. AlexNet..... 44
4.3. VGGNet.....45
4.4. ResNet..... 45
5. Le Processus d'apprentissage d'un CNN.....46

5.1. Propagation avant (Forward Propagation).....46

5.2. Calcul de l'erreur..... 46
5.3. Calcul du gradient (Backward Propagation)..... 47
5.4. Mise à jour des poids..... 47
5.5. Répétitions sur plusieurs époques..... 47
5.6. Évaluation sur données de validation..... 47
6. Conclusion..... 48

Chapitre 04 : Analyse et conception

1. Introduction..... 50
2. Présentation d'UML..... 50
2.1. Définition de l'UML..... 50
2.2. Rôle de l'UML dans la modélisation d'un système..... 51
3. Objectif de notre projet..... 51
4. Analyse du système..... 52
4.1. Identification des acteurs..... 52
a) Agent..... 52
4.2. Spécification des cas d'utilisation du système..... 53
4.3. Conception de la base des données..... 54
4.3.1. Description des tables..... 54
4.3.2. Le modèle conceptuel des données (MCD)..... 56
5. Modélisation UML..... 56
5.1. Diagramme de cas d'utilisation..... 56
5.2. Diagramme de classes..... 58
5.3. Diagramme de séquence..... 59
6. Description détaillée du système..... 62
6.1. Module d'acquisition des copies scannées (au format PDF)..... 62
6.2. Module de Prétraitement des images extraites pour améliorer la qualité et l'alignement..... 63
a. Filtrage Gaussien :..... 63
b. Binarisation (Otsu + inversion)..... 63
C. Détection des coins..... 64
D. Correction de l'inclinaison..... 66
E. Détection de la ligne verticale de séparation :..... 67
F. Extraction des coins inférieurs :..... 69
6.3. Détection des zones de réponses et lecture des cases cochées..... 70

6.4. Reconnaissance des identifiants étudiants (nom, prénom, Matricule)..... 73
a. Extraction de la zone identité.....73
b. Détection des contours et filtrage des rectangles.....74
c. Prétraitement et reconnaissance des cases manuscrites.....75
d. Recherche et validation des identifiants dans la base de données..... 76
6.5. Module de correction et calcul des notes..... 76
a. Saisie ou importation du corrigé type..... 76
b. Lancement de la correction automatisée.....77
c. Analyse et évaluation des réponses..... 77
d. Calcul automatique de la note.....77
6.6. Module d'affichage et d'exportation des résultats..... 78
a. Consultation des résultats..... 78
b. Recherche et filtrage..... 78
c. Exportation des résultats..... 78
d. Statistiques et visualisation graphique.....78
7. Conclusion.....79

Chapitre 05 : Implémentation et expérimentation

1. Introduction.....81
2.1 Environnement machine..... 81
2.2. Environnement logiciel (Système d'exploitation)..... 81
3. Les langages de programmation et les technologies utilisés..... 81
3.1. Langage de programmation.....82
Python..... 82
3.2. Développement de l'interface utilisateur.....82
PyQt5..... 82
3.3 Génération de documents PDF..... 83
ReportLab..... 83
3.4 Manipulation et visualisation de données..... 83
Pandas..... 83
Matplotlib..... 84
3.5 Traitement d'images.....85
OpenCV.....85
3.6 Intelligence artificielle et reconnaissance de caractères.....85
TensorFlow..... 85
Keras..... 86

NumPy..... 86
3.7 Gestion des données et base de données locale..... 87
SQLite.....87
DB Browser..... 87
3.8 Environnement de développement.....88
PyCharm.....88
Google colab.....89
4. Implémentation du modèle CNN..... 89
4.1 Résumé de l'entraînement..... 89
4.2 Résultats par époque.....89
4.3 Analyse visuelle des performances..... 90
5. Présentation de quelques interfaces.....91
5.1. Page de couverture.....91
5.2. Page de connexion.....92
5.3 Page d'accueil..... 92
5.4 Page Numérisation et traitement.....93
5.5 Page Traitement..... 93
5.6 Page de correction.....94
5.7 Page de corrigé type..... 94
5.8 Page de correction des copies importées..... 95
5.9 Page statistiques.....95
6. Conclusion.....96
Conclusion générale.....**97**
Références..... 98
Références de figures..... 102

Table des figures

Figure 1 : Représentation d'un pixel dans une image..... 16
Figure 2 : Représentation des pastilles RGB [01]..... 16
Figure 3 : Synthèse des couleurs par variation de l'intensité des sous-pixels RGB..... 17
Figure 4 : Exemple d'histogramme d'une image à niveau de gris..... 18
Figure 5 : Un exemple de contouring d'image..... 18
Figure 6 : Valeurs des niveaux de gris et teintes correspondantes..... 19
Figure 7 : Image en niveaux de gris..... 19
Figure 08 : Schéma d'un système de traitement d'images..... 20
Figure 9 : Analogie entre un système réel et un système de RdF..... 21
Figure 10 : Schéma général d'un système de RdF (Merabti)..... 22
Figure 11: Méthode de décision des k plus proches voisins..... 24
Figure 12:Exemple de squelettisation..... 25
Figure 13:les applications majeures de la vision par ordinateur..... 28
Figure 14: Reconnaissance automatique de la parole..... 28
Figure 15:reconnaissance d'écriture manuscrite..... 29
Figure 16: Dispositifs interactifs pour la reconnaissance en ligne de l'écriture manuscrit..... 30
Figure17 :Exemple d'un ORM..... 35
Figure 18:Exemple d'une convolution..... 41
Figure 19:Fonction d'activation ReLU..... 42
Figure 20:Représentation de max pooling..... 43
Figure 21:Représentation de la couche fully-connected..... 44
Figure 22:Exemple des couches convolutives..... 44
Figure 23: LeNet-5..... 45
Figure 24: AlexNet..... 45
Figure 25:VGGNet..... 46
Figure 26:Bloc résiduel unique..... 47
Figure 27:Logo du langage de modélisation UML (Unified Modeling Language)..... 51

Figure 28 :Le modèle conceptuel des données (MCD).....57

Figure 29 : Diagramme de cas d'utilisation.....58

Figure 30 :Diagramme de classes.....60

Figure 31 : Diagramme de séquence << page connexion>>.....60

Figure 32 : Diagramme de séquence << page Numérisation..... et traitement>> 62

Figure 33 : Diagramme de séquence << page Correction>>.....62

Figure 34 : Exemple de Filtrage Gaussien et Binarisation (Otsu + inversion).....64

Figure 35 : le Centre de gravité des coins de l'image découpée.....66

Figure 36: Les nouvelles coordonnées de centre de gravité de coin droit.....67

Figure 37 :Rotation d'une image autour de son origine.....68

Figure 38 :Exemple de correction de l'inclinaison.....68

Figure 39 :la ligne de séparation détectée.....70

Figure 40:Extraction des coins inférieurs centre de gravité.....71

Figure 41:La zone d'intérêt extraite.....72

Figure 42:Exemple d'extraction de la première colonne et lecture des cases (case cochée = vert, case vide = rouge).....73

Figure 43:Extraction de la zone identité.....74

Figure 44 :Rectangles classés par ligne (Nom/Prénom/Matricule).....75

Figure 45:L'architecture utilisée dans notre model CNN.....76

Figure 46:Exemple de reconnaissance des cases manuscrites.....77

Figure 47:Logo officiel du langage de programmation Python.....83

Figure 48:Logo de PyQt5, bibliothèque Python pour interfaces graphiques.....84

Figure 49:Logo de ReportLab, bibliothèque Python pour la génération de documents PDF 84

Figure 50: Logo de la bibliothèque Pandas pour l'analyse de données en Python.....85

Figure 51: Logo de la bibliothèque Matplotlib pour la visualisation de données en Python...85

Figure 52:Logo officiel de la bibliothèque OpenCV.....86

Figure 53 :Logo de la bibliothèque TensorFlow.....87

Figure 54 :logo de la bibliothèque Keras.....87

Figure 55 :Logo de la bibliothèque NumPy pour la manipulation de données scientifiques en Python.....88

Figure 56: Logo de la base de données relationnelle légère SQLite.....88

Figure 57: logo de DB Browser for SQLite.....89

Figure 58 :logo de Environnement de développement intégré (IDE) PyCharm utilisé pour le développement Python..... 89

Figure 59 : Logo officiel de Google Colab.....90

Figure 60 :Évolution de la précision et de la fonction de perte lors de l'entraînement du modèle CNN..... 92

Figure 61:Page de couverture de notre système.....92

Figure 62:Page de connexion.....93

Figure 63: Page d'accueil de notre application de correction automatique des QCM.....93

Figure 64:Page Numérisation et traitement.....94

Figure 65:Page Traitement des copies importées.....94

Figure 66: Page de correction.....95

Figure 67: Page de corrigé type.....95

Figure 68:Page de correction des copies importées.....96

Figure 69: Page statistiques.....96

Table des Tableaux

Tableau 1 : Spécification des cas d'utilisation du système.....55

Tableau 2 : Table users (utilisateurs du système).....55

Tableau 3 : Table exams (informations sur les examens).....56

Tableau 4 :Table étudiants (étudiants inscrits à un examen).....56

Tableau 5 : Table copies (copies corrigées par le système).....57

Tableau 6: Résultats de l'évolution des performances du modèle CNN au cours des époques d'entraînement.....91

Introduction générale

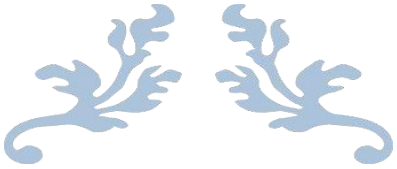
Dans le contexte actuel de la digitalisation de l'enseignement et de la massification des évaluations, la correction manuelle des copies d'examen à choix multiples (QCM) représente un défi important en termes de temps, d'efficacité et de fiabilité. Face à ces contraintes, il devient essentiel de concevoir des solutions automatiques capables de traiter rapidement et avec précision de grandes quantités de copies scannées.

Notre projet s'inscrit dans cette dynamique en proposant un **système complet de correction automatique de QCM** reposant sur les techniques avancées de **vision par ordinateur** et d'**intelligence artificielle**, notamment les **réseaux de neurones convolutifs (CNN)**. Ces derniers, largement utilisés dans la reconnaissance d'images, sont particulièrement adaptés pour détecter et interpréter les réponses cochées ainsi que les informations manuscrites (nom, prénom, matricule) inscrites dans des grilles prédéfinies.

L'objectif principal de ce projet est de développer une application robuste, intuitive et efficace qui permet de :

- Numériser automatiquement des copies d'examen au format PDF ou image,
- Extraire les réponses aux questions via détection des cases cochées,
- Identifier les candidats à partir des zones d'identité manuscrite,
- Corriger les copies selon un barème défini et générer les résultats au format Excel et PDF,
- Fournir des statistiques détaillées sur les performances des étudiants.

En combinant des approches classiques de traitement d'image avec la puissance des CNN pour la reconnaissance des caractères isolés, notre système vise à automatiser l'ensemble du processus de correction tout en garantissant un haut niveau de fiabilit



CHAPITRE 01 : Reconnaissance des formes et vision par ordinateur

1. Introduction:

La reconnaissance des formes et la vision par ordinateur permettent aux machines d'analyser et d'interpréter des images. Les notions essentielles du traitement d'image, comme le pixel, l'histogramme et les contours, facilitent cette analyse. Le processus de reconnaissance passe par plusieurs étapes : acquisition, prétraitement, extraction de caractéristiques et classification. Ces techniques trouvent des applications en vision artificielle, reconnaissance vocale et écriture manuscrite.

2. Généralités sur les traitements d'images

Le **traitement d'images** est un domaine vaste et en constante évolution depuis les années 60, jouant un rôle essentiel dans les sociétés modernes où l'image est omniprésente. Il regroupe un ensemble de techniques permettant de modifier, analyser et extraire des informations d'images numériques. Ses applications sont nombreuses, allant de la détection et la reconstruction d'objets à la biométrie et à la classification. Grâce aux avancées technologiques, notamment en intelligence artificielle, il est devenu un outil clé dans des domaines variés comme la médecine, la sécurité et la robotique.

2.1 Image

Une **image** est une représentation visuelle d'un objet, d'une scène ou d'un concept, réalisée par les arts graphiques, plastiques, la photographie, le cinéma, etc.

Les images peuvent être naturelles (ombres, reflets) ou artificielles (sculptures, peintures)

2.2 Pixel

Le plus petit élément constitutif d'une image numérique (abréviation de

«**Pic**ture **El**ement»),il est caractérisé par sa position (i en abscisse et j en ordonné) et sa valeur (niveau de gris ou couleur), Comme l'indique la *figure 1.* (Nsimichelet91, s.d.)
Le terme "pixels" désigne aussi les points lumineux constituant un écran et détermine la définition et la résolution d'une image affichée. (maxicours, s.d.)

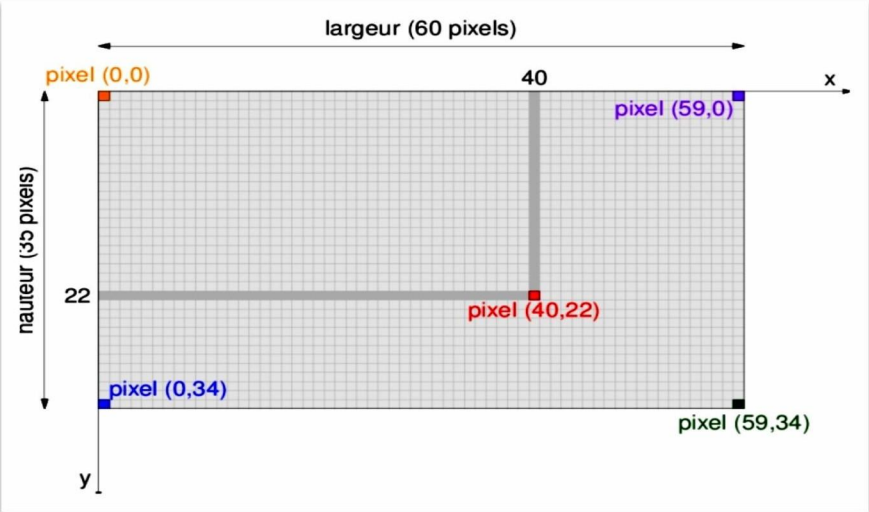


Figure 1 : Représentation d'un pixel dans une image (Nsimichelet91, s.d.)

2.3 L'image numérique

En informatique, une **image numérique** est une représentation d'une image (dessin, icône, photographie, etc.) qui a été acquise, créée, traitée et stockée sous forme binaire, c'est-à-dire sous forme de suites de 0 et de 1. (Techno- Science.net, 2025)
Une image numérique apparaît comme un quadrillage ou un tableau dont chaque case est un pixel d'une couleur spécifique. Chaque pixel est composé de **3 pastilles** (aussi nommées sous-pixels), une rouge, une verte, et une bleue **(RGB)**,dont la luminosité varie pour recréer une large gamme de couleurs. Comme l'indique la *figure 2* et la *figure 3* (Nsimichelet91, s.d.) (maxicours, s.d.)

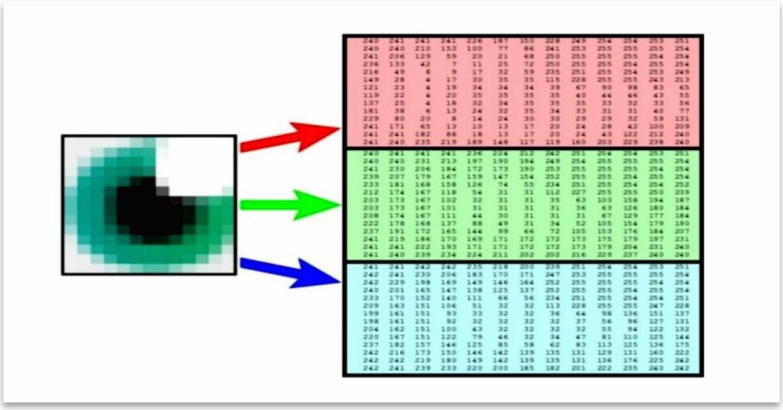


Figure 2 : Représentation des pastilles RGB

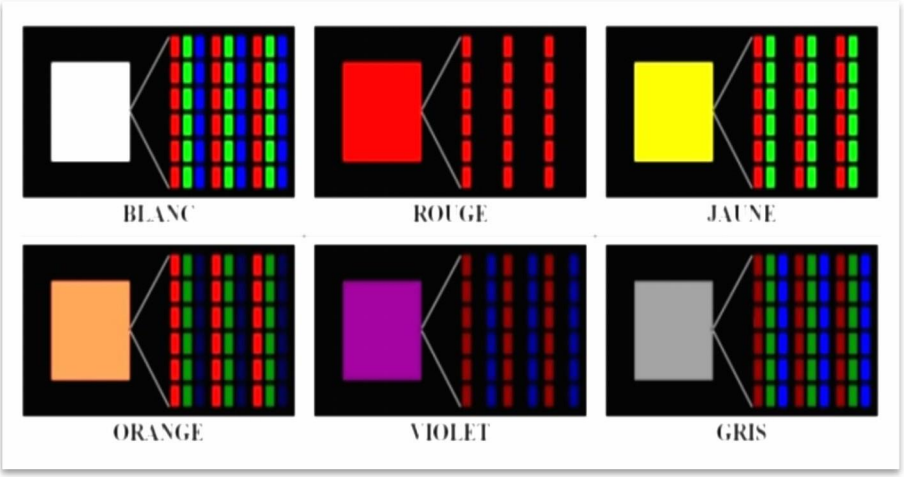


Figure 3: Synthèse des couleurs par variation de l'intensité des sous-pixels RGB

2.4 Histogramme

L'histogramme d'une image constitue une fonction discrète qui associe à chaque niveau d'intensité le nombre de pixels correspondants. Pour une image en niveaux de gris codée sur **L** niveaux, si **nk** *désigne le nombre de pixels ayant l'intensité* ***xk*****,** l'histogramme peut être normalisé en divisant chaque valeur par le nombre total de pixels **n**, ce qui permet d'interpréter chaque valeur comme une probabilité d'occurrence :

$$p_x(x_k)=p(x=x_k)=\frac{nk}{n},\quad 0 \le k < L$$

Ainsi, l'histogramme normalisé représente une estimation de la densité de probabilité des intensités de l'image, facilitant l'analyse statistique de son contenu. (Techno- Science.net, Histogramme (imagerie numérique), 2025)

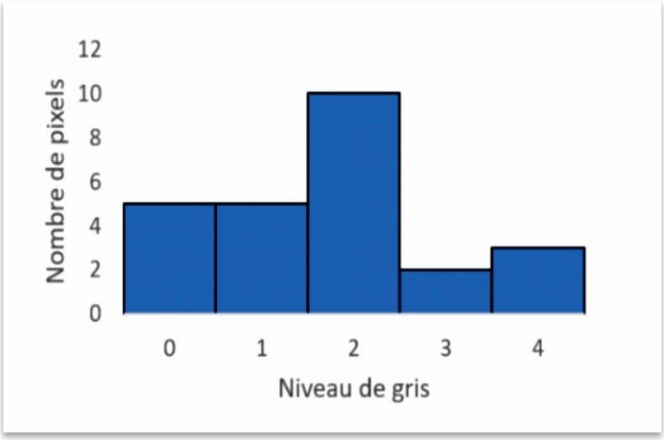


Figure 4: Exemple d'histogramme d'une image à niveau de gris

2.5 Notion de contours dans une image:

En vision par ordinateur, les contours désignent les courbes délimitant les objets dans une image, en reliant des points d'intensité ou de couleur similaire. Leur détection permet de simplifier l'image en extrayant les structures essentielles, facilitant ainsi la segmentation et l'analyse des objets. Les images suivantes montrent un exemple de contouring. Sur l'image, la forme d'une pomme est détectée *(figure 5)*. (M., 2024)

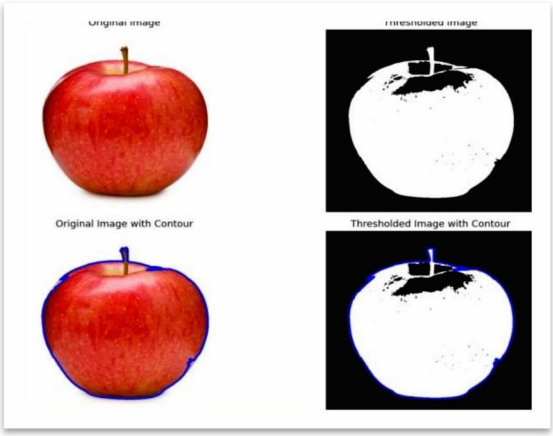


Figure 5 : Un exemple de contouring d'image

2.6 Textures

En informatique et en vision par ordinateur, la **texture** se définit comme l'ensemble des caractéristiques visuelles liées à l'apparence de surface d'un objet, telles que la taille, la forme, la densité et l'organisation spatiale de ses éléments constitutifs. Elle est perçue à travers les variations locales de propriétés simples comme la couleur, l'intensité ou l'orientation dans une image. (Haddon, 2006)

2.7 Images à niveaux de gris

Dans une image en niveaux de gris, la valeur d'un pixel varie du noir au blanc, en incluant un nombre fini de nuances intermédiaires.

Généralement codé sur un octet soit 8 bits, ce qui offre là d'obtenir 256 niveaux de gris (0 pour le noir et 255 pour le blanc). (Figure 6)

Chaque pixel n'est plus représenté par un bit, mais par un octet. Ainsi, le matériel d'affichage doit être capable de restituer les différents niveaux de gris correspondants. (Kaidi, 2017, 26 septembre)

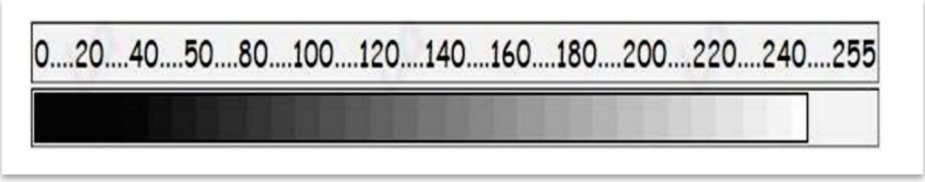


Figure 6 : Valeurs des niveaux de gris et teintes correspondantes

Figure 7: Image en niveaux de gris

2.8 Le processus de traitement d'image

Le traitement d'images est une discipline alliant informatique et mathématiques appliquées, qui étudie les images numériques et leurs transformations. Il regroupe un ensemble de techniques visant à modifier, analyser et améliorer ces images, dans le but d'en extraire des informations utiles ou d'optimiser leur qualité visuelle. De manière générale, le processus de traitement d'image s'articule autour des étapes suivantes : **acquisition**, **prétraitement**, **segmentation**, **extraction de caractéristiques**, **reconnaissance** et étiquetage**.** (Gonzalez, 2018)

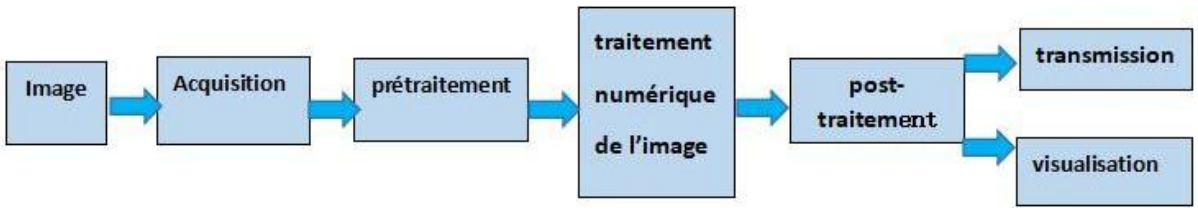


Figure 08 : Schéma d'un système de traitement d'images

3. La reconnaissance des formes (RDF)

La reconnaissance des formes, également connue sous le nom de reconnaissance de motifs ou de pattern recognition, est un domaine scientifique et technique qui s'intéresse à la conception et à la réalisation de systèmes, qu'ils soient matériels ou logiciels, capables de percevoir et, dans une certaine mesure, d'interpréter des signaux issus du monde physique *(J.P. Haton)*.

Elle vise à concevoir des systèmes automatiques ou semi-automatiques qui sont en mesure d'identifier et classer des formes sur la base d'informations qu'ils leur fournissent.L'homme, avec ses capacités naturelles d'analyse et d'interprétation,représente le système de reconnaissance des formes le plus évolué.La RdF se fonde sur la reconstitution, sur des machines, de fonctions typiquement humaines comme :

- **La perception** des données visuelles, sonores ou autres ;
- **La représentation** et l'analyse des informations données ;

- **L'interprétation** des formes et des structures détectées.

Elle constitue un champ d'application essentiel de l'intelligence artificielle (IA), notamment dans la compréhension automatique des données. (Benazzouz, année universitaire 2018-2019.)

3.1 Processus de reconnaissance

Un **système de reconnaissance des formes** vise à déterminer la classe d'appartenance d'une forme inconnue. Pour cela, il suit trois étapes essentielles :(figure 9)

- **L'étape d'acquisition:** l'espace d'observation permet l'acquisition des données à travers des capteurs.
- **L'étape d'extraction des caractéristiques:** le passage de l'espace d'observation vers l'espace de représentation.
- **L'étape de classification :** le passage de l'espace de représentation vers l'espace d'interprétation. (Merabti)

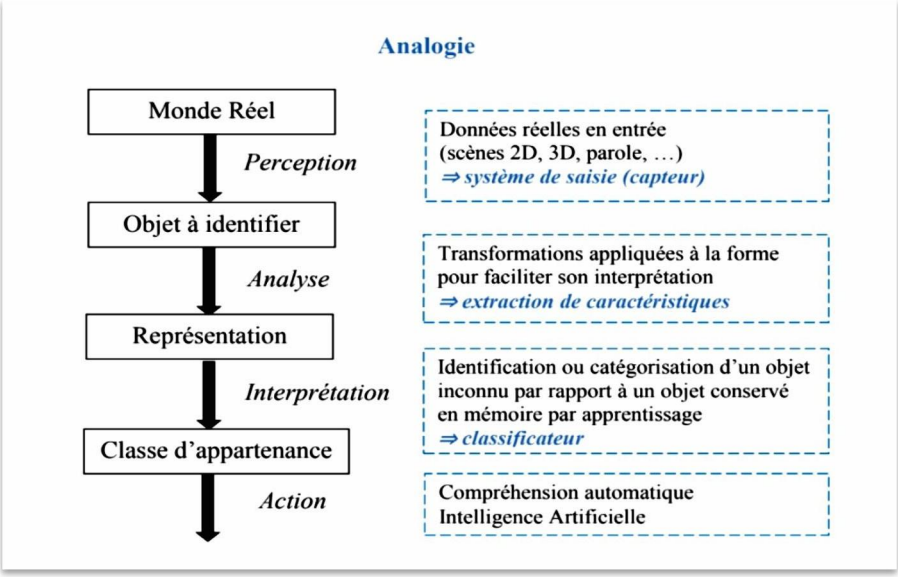


Figure 9 : Analogie entre un système réel et un système de RdF

La majorité des systèmes de RdF ont le schéma de fonctionnement suivant :

(Figure 10). (Merabti)

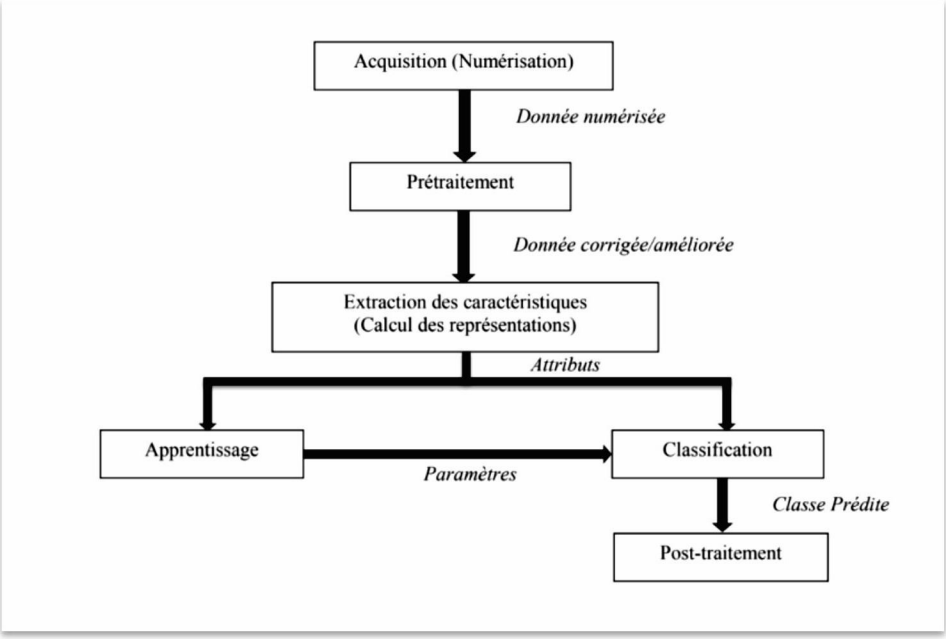


Figure 10 : Schéma général d'un système de RdF (Merabti)

3.1.1 Acquisition des données (numérisation)

L'acquisition des données consiste à convertir les informations du monde réel en une représentation numérique exploitable par la machine. Des capteurs, comme des caméras ou des microphones, transforment les signaux analogiques en signaux numériques, formant un **espace de représentation** riche en informations.

On distingue deux types d'acquisition :

- **Physique :** utilisation de capteurs comme les scanners pour numériser les données.
- **Logique :** exploitation d'un fichier graphique sous forme de matrice pour le traitement numérique. (Merabti)

3.1.2 Le prétraitement

Cette étape vise à extraire les informations essentielles de l'espace de représentation tout en éliminant les éléments inutiles. Elle consiste principalement à :

- **Réduire le bruit** causé par les outils d'acquisition, la qualité du papier, l'encre ou le vieillissement des documents anciens.

- **Préserver les caractéristiques significatives** de l'objet à classer en supprimant les informations redondantes. (Merabti)

3.1.3 Extraction des caractéristiques (calcul des représentations)

Il s'agit de la dernière étape de la préparation des données. Elle consiste à extraire ou sélectionner un ensemble de caractéristiques clés, appelées attributs, à l'aide d'algorithmes spécialisés. Comme le nombre d'attributs est limité, l'espace des paramètres obtenu est bien plus réduit que l'espace initial, tout en conservant les informations essentielles pour l'analyse. (Djabeur Djezzar)

3.1.4 L'apprentissage

L'apprentissage a pour objectif d'identifier les caractéristiques distinctives propres à chaque classe, en s'appuyant sur un modèle général dont les paramètres sont ajustés à partir de données d'entraînement.

L'apprentissage peut se dérouler sous différentes formes : (Merabti)

- **Apprentissage supervisé** : l'algorithme apprend à partir de données déjà étiquetées, c'est-à-dire pour lesquelles la classe ou la réponse correcte est connue.
- **Apprentissage non supervisé** : l'algorithme travaille sur des données non étiquetées, sans information préalable sur les catégories ou les réponses à trouver.

3.1.5 Classification automatique

Cette phase est le noyau de la reconnaissance des formes.Elle consiste à regrouper les objets ou les formes partageant des caractéristiques similaires en classes bien distinctes. L'objectif est que chaque élément soit plus proche des objets de sa propre classe que de ceux des autres classes. Parmi les approches les plus couramment utilisées, on trouve : (Ait Aider)

3.1.5.1 Approches mathématiques

On distingue les méthodes métriques et statistiques.

3.1.5.1.1 Méthodes métriques

Dans ce cas, la forme à reconnaître est représentée par un vecteur de caractéristiques, contenant les valeurs des paramètres calculés pour cet objet. Parmi les méthodes métriques, on retrouve notamment la méthode des **k plus proches voisins (k-NN)**.

Avec cette approche, pour classifier une forme inconnue **x** （ Figure 11 ） , on mesure la distance entre **x** et l'ensemble des échantillons appartenant aux différentes classes. Ensuite, on sélectionne les **k échantillons les plus proches** et **x** est attribuée à la classe la plus représentée parmi eux. (Ait Aider)

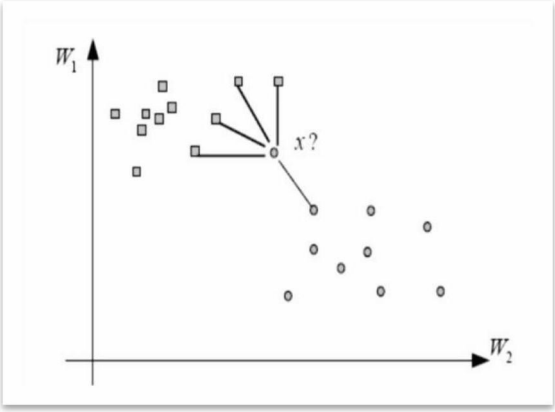


Figure 11: Méthode de décision des k plus proches voisins

Il existe plusieurs choix de distances possibles, la plus utilisée est la distance euclidienne.

3.1.5.1.2 Méthodes statistiques

Ces méthodes reposent sur le concept de **fonction de densité de probabilité** et sur la **théorie de la décision de Bayes**. Lorsqu'une connaissance complète des probabilités a priori des objets est disponible, le **théorème de Bayes** permet de les convertir en **probabilités a posteriori**, facilitant ainsi l'identification des classes les plus probables. (Ait Aider)

3.1.5.2 Approches structurelles

Une **signature structurelle** représente la structure d'une forme en la décomposant en **primitives élémentaires** (lignes, arcs, régions, vecteurs) et en décrivant leurs relations

(jonctions, parallélisme, etc.). Elle peut contenir uniquement ces primitives, uniquement les relations, ou une combinaison des deux.

En complément des **graphes**, certaines approches reposent sur des **grammaires syntaxiques** pour modéliser la construction des formes de manière hiérarchique. Une grammaire se compose d'**éléments de base**, d'**éléments complexes** formés à partir de ces bases et de **règles de construction**. Parmi les grammaires utilisées en reconnaissance des formes, on trouve les **grammaires de chaînes, bidimensionnelles et stochastiques**. (Ait Aider)

3.1.5.2.1 La squelettisation

La **squelettisation** est une technique utilisée en reconnaissance de formes, notamment pour les **chiffres manuscrits**. Elle permet de réduire une forme à un **squelette d'une épaisseur d'un pixel**, tout en conservant sa structure.

L'**algorithme de Marthon** réalise cette transformation de manière itérative en supprimant les points non essentiels. Cependant, pour extraire des **primitives structurelles significatives**, une étape complémentaire est souvent nécessaire: la **construction d'un graphe du squelette**. Il est crucial que les **extrémités des branches soient bien définies** et que les **courbures des formes soient correctement représentées** afin de garantir une bonne analyse des symboles. (Ait Aider)

Figure 12:Exemple de squelettisation

3.1.5.2.2 Extraction de segments primitifs

L'extraction des segments primitifs est une étape clé dans la reconnaissance des formes, permettant d'obtenir une **représentation plus structurée** d'une image en plusieurs niveaux d'information.

- **Décomposition en éléments de base** : L'image est divisée en traits, jonctions et boucles pour une meilleure manipulation.
- **Description des éléments** : Chaque segment est analysé en termes de longueur, nombre de pixels et autres caractéristiques.**Analyse des points caractéristiques**: Les points d'extrémité et de jonction sont identifiés à partir des coordonnées et des relations de voisinage.

Cette approche, fondée sur les **signatures structurelles**, facilite la reconnaissance des formes en contexte et résiste aux variations d'échelle et de rotation. Toutefois, elle reste **sensible au bruit et aux déformations**, nécessitant parfois l'ajout de connaissances a priori. Une fois les segments extraits et analysés, ils sont utilisés pour **classer ou identifier de nouvelles formes** dans la phase de reconnaissance. (Ait Aider)

3.1.5.3 Approches bio-inspirées

Les chercheurs ont optimisé les **algorithmes classiques** en s'inspirant de mécanismes biologiques. Parmi les approches les plus utilisées :

- **Les algorithmes génétiques**, basés sur la sélection naturelle, utilisent des processus de sélection, croisement et mutation pour optimiser les solutions.
- **Les réseaux de neurones artificiels**, inspirés du cerveau humain, sont efficaces en reconnaissance des formes et en apprentissage automatique, notamment avec le **deep learning**.

D'autres méthodes, comme les **colonies de fourmis** et les **essaims de particules**, exploitent des comportements collectifs pour résoudre des problèmes complexes. Ces approches améliorent la **performance et l'adaptabilité** des systèmes intelligents. (Approches incrémentales pour l'apprentissage en reconnaissance de formes) (Approches incrémentales pour l'apprentissage en reconnaissance de formes)

3.1.5.4 Méthodes connexionnistes

Les méthodes connexionnistes représentent un domaine de recherche distinct qui s'est largement imposé en reconnaissance de formes. Elles offrent un cadre pour développer et tester diverses variantes de réseaux, tout en étendant les techniques de

séparation linéaire, comme celle du perceptron, aux séparations non linéaires. Inspirée par la structure neurophysiologique du cerveau, l'approche connexionniste repose sur des neurones formels, qui sont des unités interconnectées au sein d'un réseau structuré. Adjoudj, R. (2006–2007)

3.1.6 Post-traitement

Le post-traitement joue un rôle essentiel dans la **validation et la correction des erreurs** issues de la reconnaissance des formes. Il s'appuie sur différentes sources d'information, telles que les **données lexicales, syntaxiques et sémantiques**, afin d'améliorer la précision des résultats. Bien que facultative, cette phase optimise **significativement** la qualité de la reconnaissance.

Cette phase optimise la qualité de la reconnaissance, notamment dans les systèmes de lecture automatique de documents. (Merabti)

3.2 Domaines d'application

La reconnaissance de formes (RDF) est utilisée dans de nombreux domaines, notamment :

3.2.1 Vision par ordinateur

La vision par ordinateur est un domaine clé de la reconnaissance des formes, avec des applications variées. En **imagerie biomédicale**, elle permet la détection de tumeurs cérébrales via l'analyse d'IRM. En **industrie**, elle est utilisée pour le contrôle de qualité des produits grâce à des systèmes de détection de défauts. D'autres applications incluent **l'analyse d'images satellitaires** et la **biométrie** (reconnaissance faciale, empreintes digitales), essentielles pour la **sécurité** et la **météorologie**. (Ait Aider)

Figure 13:les applications majeures de la vision par ordinateur

3.2.2 La reconnaissance vocale

La reconnaissance automatique de la parole (ASR) est une technologie d'intelligence artificielle qui convertit la voix humaine, captée par un microphone, en texte exploitable par ordinateur. Aussi appelée reconnaissance vocale ou parole-texte, elle facilite une interaction naturelle avec les machines. Zaion AI. (2022, 6 juillet).

figure 14: Reconnaissance automatique de la parole

3.2.3 Reconnaissance de texte et d'écriture

La reconnaissance de caractères (OCR, pour Optical Character Recognition) constitue une application classique de la reconnaissance de formes, visant à convertir un texte manuscrit ou imprimé en format numérique. Le document est d'abord numérisé, puis chaque caractère est analysé afin de lui associer sa

correspondance numérique. L'OCR s'appuie sur des algorithmes avancés pour identifier les lettres et chiffres dans les images, permettant l'automatisation de tâches telles que l'archivage, l'indexation documentaire, le tri postal ou encore le traitement de chèques. (Ait Aider)

3.2.3.1 La reconnaissance d'écriture manuscrite

3.2.3.1.1. Définition :

Elle est un processus informatique qui permet d'identifier et de convertir automatiquement du texte écrit à la main en format numérique. Cette technologie repose sur des algorithmes avancés capables d'analyser la forme, l'orientation et la structure des caractères manuscrits, malgré leur grande variabilité d'un individu à l'autre. Elle doit surmonter plusieurs défis, tels que les différences de style, les variations d'inclinaison, les chevauchements de lettres et les irrégularités des tracés. Son développement s'appuie sur des techniques de traitement d'image et d'IA, permettant d'améliorer la précision et l'adaptabilité aux divers styles d'écriture. (Graves,2008)

Figure 15:reconnaissance d'écriture manuscrite

3.2.3.1.2. Différents aspects de la reconnaissance automatique de l'écriture manuscrite :

3.2.3.1.2.1. Mode d'acquisition (Enligne / Hors-ligne)

À ce jour, aucun système informatique ne parvient à reconnaître l'écriture manuscrite avec la même universalité et précision qu'un être humain alphabétisé. Les méthodes de reconnaissance existantes se divisent en deux grandes catégories : la reconnaissance hors ligne, également appelée « statique », et la reconnaissance en ligne, dite « dynamique ».

A. Systèmes de reconnaissance hors ligne

Dans le cas de la reconnaissance hors ligne, le texte manuscrit est analysé à partir d'un document déjà écrit. L'image du texte est numérisée via un scanner, produisant une représentation discrète sous forme de pixels. L'écriture est alors traitée comme un signal spatial bidimensionnel numérisé. **(Marti & Bunke, 2001)**

B. Systèmes de reconnaissance en ligne

en revanche, la reconnaissance en ligne intervient directement lors de l'écriture. Le texte est saisi à l'aide d'un stylo numérique ou d'une tablette graphique, enregistrant une suite ordonnée de points définis par leurs coordonnées. Dans ce cas, l'écriture est perçue comme un couple de signaux temporels numérisés.**(Reconnaissance de l'écriture manuscrite,2004)**

Figure 16: Dispositifs interactifs pour la reconnaissance en ligne de l'écriture manuscrit

3.2.3.1.2.2 Approches de reconnaissance de l'écriture manuscrite

La reconnaissance de l'écriture manuscrite, peut être abordée à travers deux méthodes principales : **l'approche globale (holistique)** et **l'approche analytique (locale)**.

A. L'approche Globale ou Holistique

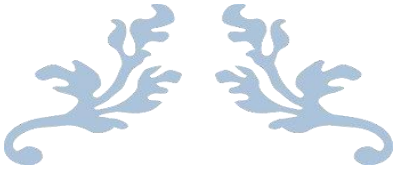
L'approche globale dans la reconnaissance de l'écriture manuscrite traite les mots ou les phrases comme des entités complètes. Au lieu de segmenter chaque caractère, cette méthode analyse l'ensemble du mot, ce qui permet de tirer parti du contexte et d'améliorer la précision de la reconnaissance. Par exemple, en utilisant des modèles de langage, le système peut prédire les mots en fonction des lettres déjà reconnues, ce qui est similaire à la façon dont les humains reconnaissent les mots dans une phrase.**(Merabti,2016)**

B. L'approche analytique ou locale

L'approche analytique se concentre sur l'analyse des caractères individuels ou des segments de l'écriture manuscrite. Cette méthode décompose les mots en lettres ou en traits, permettant une reconnaissance plus précise des détails. Cela est particulièrement utile pour des écritures variées ou mal formées, car elle permet de traiter chaque caractère indépendamment. Cependant, cette approche peut être moins efficace dans des contextes où le style d'écriture est très variable, car elle ne prend pas en compte le contexte global des mots .**(Merabti,2016)**

4. Conclusion

La reconnaissance des formes repose sur un enchaînement d'étapes techniques bien définies et indispensables. Le traitement d'image prépare les données visuelles à être exploitées pour l'identification et la classification automatique. De nombreuses applications utilisent ces méthodes dans des domaines variés comme la santé, la sécurité et l'industrie. L'intégration de l'intelligence artificielle renforce désormais les performances de ces systèmes.



CHAPITRE 02 : Correction automatique des feuilles d'examen QCM

1. Introduction

La correction des feuilles d'examen, notamment celles comportant des questions à choix multiples (QCM), pose de nombreux défis pour les enseignants et les établissements scolaires. En effet, lorsque le nombre de copies à traiter est important, la correction manuelle devient une tâche longue, répétitive et source d'erreurs potentielles, ce qui peut entraîner des retards dans la remise des résultats et affecter l'objectivité de l'évaluation. Dans ce contexte, l'automatisation de la correction, grâce à des technologies telles que la reconnaissance optique de caractères (OCR), apparaît comme une solution innovante et efficace. L'objectif principal de notre projet est de développer un système capable de corriger automatiquement les feuilles d'examen QCM, afin de réduire le temps de traitement, d'augmenter la fiabilité des résultats et d'assurer une évaluation équitable pour tous les candidats. Ce chapitre présente ainsi la problématique de la correction des QCM et expose les objectifs et l'intérêt de l'automatisation à travers une analyse des méthodes existantes basées sur l'OCR.

Ce chapitre commence par un état des lieux des méthodes classiques de correction, puis présente les apports des approches modernes basées sur l'intelligence artificielle, et enfin détaille la solution développée dans le cadre de notre projet.

2. Méthodes existantes pour la correction des QCM

Avec l'essor des technologies numériques, la correction des QCM s'appuie aujourd'hui sur des méthodes de plus en plus automatisées. On distingue principalement la correction semi-automatique, basée sur la reconnaissance optique de marques (OMR), et la correction automatique par des systèmes classiques utilisant la reconnaissance optique de caractères (OCR) et des règles heuristiques. Ces différentes approches, bien qu'efficaces, présentent chacune des limites et des défis, notamment en matière de fiabilité, d'adaptabilité et de coût.

2.1. Correction semi-automatisée

La correction semi-automatisée utilise des grilles de réponse pré-imprimées que les étudiants remplissent. Les scanners optiques, tels que les lecteurs OMR (Optical Mark Reader), lisent ces grilles pour identifier les réponses.

2.2.1 Systèmes de Reconnaissance Optique de Marques (OMR)

L'**OMR,** ou **O**ptical **M**ark **R**ecognition (Reconnaissance Optique de Marques), est une technologie permettant à une machine de détecter la présence de marques (cases cochées, bulles, croix) sur un formulaire papier. Elle est couramment utilisée pour l'extraction de données à partir de feuilles d'examen QCM. Plusieurs logiciels et solutions OMR existent, comme **Gravic Remark Office OMR** (commercial), **AutoMultipleChoice** (open source) ou **Zebra OMR** (open source), qui appliquent cette technique pour automatiser la lecture des réponses sur des formulaires standardisés.

L'OMR fonctionne en utilisant un scanner ou une caméra pour prendre une image du formulaire, puis en analysant cette image pour identifier les marques. Une fois les marques reconnues, les informations sont automatiquement transférées vers un système informatique pour traitement et analyse. PaperSurvey.io. (2023, 7 février).

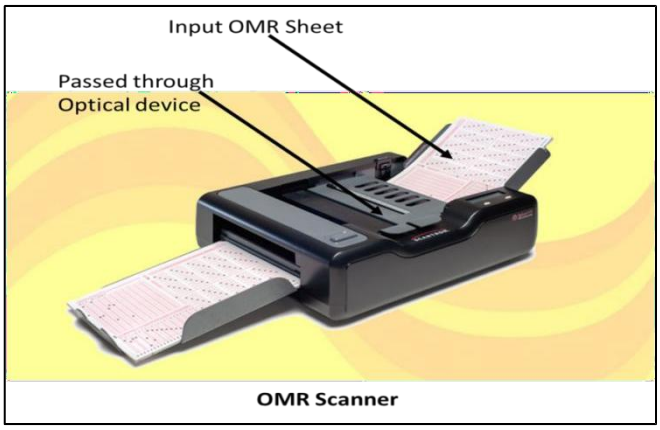


Figure17 :Exemple d'un ORM

2.2. Correction automatique par OCR traditionnels

Les systèmes classiques de correction automatique reposent généralement sur des moteurs d'OCR traditionnels, comme **Tesseract**, associés à des règles heuristiques simples pour analyser les feuilles de QCM. Par exemple, un logiciel peut utiliser Tesseract pour essayer de lire le contenu des cases et déterminer la réponse sélectionnée par l'étudiant. Cette approche fonctionne correctement lorsque le texte est imprimé ou que les cases sont parfaitement cochées. En revanche, dès que les copies présentent des écritures manuscrites, des cases partiellement remplies, l'efficacité du système diminue rapidement.

2.3. Limites et difficultés de ces méthodes:

Les méthodes traditionnelles de correction, qu'elles soient manuelles ou semi-automatisées, présentent plusieurs défis :

- **Temps Consommé** : La correction manuelle des QCM est chronophage, surtout lorsque le nombre de copies est élevé.
- **Risques d'Erreurs:** La correction manuelle reste vulnérable aux erreurs humaines, tandis que même les systèmes OMR peuvent produire des erreurs si les copies ne sont pas correctement lues, ce qui compromet la fiabilité globale des résultats.
- **Impact sur l'Apprentissage:** Les QCM traditionnels sont souvent critiqués pour encourager la mémorisation au détriment de la compréhension profonde. Ils évaluent mal les compétences de raisonnement et de résolution de problèmes.
- **Manque de flexibilité** : Difficulté à s'adapter à différents types de questions, peu adaptés à l'évaluation des compétences complexes ou à la pensée critique.
- **Coût et ressources** : La correction manuelle demande beaucoup de personnel ; l'acquisition de systèmes OMR ou de logiciels spécialisés représente aussi un investissement important.

Ces limites ont conduit au développement de solutions plus avancées, s'appuyant sur l'intelligence artificielle et le deep learning.

3. Méthodes modernes de correction automatique

Avec l'essor de l'intelligence artificielle, et plus particulièrement du **deep learning**, de nouvelles approches transforment la correction automatique des QCM.

3.1. Applications du deep learning à la reconnaissance et à l'analyse de documents

Le **deep learning** a permis l'apparition d'une nouvelle génération d'outils capables de traiter automatiquement des documents variés, allant de la **reconnaissance de texte manuscrit** à la **détection précise de marques** ou de **cases cochées**. Ces technologies sont désormais intégrées dans plusieurs solutions utilisées pour la correction automatique des QCM.

- **EasyOCR** et **Google Cloud Vision API** : ces outils utilisent des modèles de deep learning pour reconnaître aussi bien l'écriture manuscrite que les marques sur les formulaires d'examen. (JaidedAI. (2020)) (Google Cloud. (2023)).
- **DeepOMR** : cette approche basée sur les réseaux de neurones profonds permet de détecter et d'analyser automatiquement les réponses sur des feuilles QCM.
- **YOLO (You Only Look Once)** et autres algorithmes de détection d'objets : ils sont adaptés pour localiser précisément les cases ou les zones de réponses à corriger.
- D'autres méthodes intègrent des architectures comme les **LSTM** (Long Short-Term Memory) ou **Transformers** pour la reconnaissance avancée de texte manuscrit. Par exemple, le modèle **TrOCR** de **Microsoft** combine un encodeur d'image basé sur un Transformer et un décodeur de texte pour une reconnaissance précise de l'écriture manuscrite. Microsoft. (s.d.)

3.2. Approche hybride et automatisée du projet OptiQcm

Dans ce contexte d'innovation, notre projet **« OptiQcm »** propose une approche hybride, spécifiquement adaptée aux contraintes réelles des examens sur papier.

- Pour **la détection des cases cochées**, nous appliquons des techniques spécialisées de **traitement d'image** (analyse morphologique et mesure du taux de remplissage), évitant ainsi les erreurs courantes des systèmes OCR classiques pour ce type de tâche.
- Pour **la reconnaissance de l'écriture manuscrite** dans les champs d'identité, nous utilisons un modèle **CNN** entraîné sur des lettres isolées, ce qui améliore la précision même pour des écritures variées ou peu lisibles.
- Enfin, l'ensemble du processus est entièrement automatisé, de **l'import des copies** jusqu'à **la génération des résultats,** assurant ainsi rapidité, fiabilité et facilité d'utilisation pour les enseignants.

4. Conclusion

En résumé, l'automatisation de la correction des QCM a considérablement évolué, passant des méthodes traditionnelles comme l'OMR et l'OCR aux approches modernes basées sur l'intelligence artificielle et le deep learning. Ces innovations, et en particulier l'introduction des **réseaux de neurones convolutifs (CNN)** pour la reconnaissance de l'écriture manuscrite, permettent désormais d'atteindre une correction plus fiable, rapide et adaptée aux réalités des examens.

CHAPITRE 03 : Classifieurs basé sur le deep learning : *CNN*

1. Introduction:

Les réseaux de neurones convolutifs (CNN) se sont imposés comme une référence incontournable en vision par ordinateur, notamment grâce à leur capacité à extraire automatiquement des caractéristiques visuelles pertinentes. Contrairement aux approches classiques de machine learning, qui nécessitent une extraction manuelle des *features* (comme SIFT) par des experts, les CNN réalisent eux-mêmes cette étape de manière hiérarchique et optimisée au cours de l'entraînement.

Leur percée majeure s'est produite en 2012 avec l'architecture AlexNet, qui a dominé la compétition ILSVRC. Depuis, les CNN sont devenus essentiels dans de nombreuses applications : classification d'images, diagnostic médical, reconnaissance faciale, etc. Merzougui, D. (s.d.)

Dans le cadre de ce mémoire, les CNN sont utilisés pour un cas d'usage concret : la reconnaissance automatique de **l'écriture manuscrite** sur des **copies de QCM scannées**. Cette tâche nécessite l'identification fiable de lettres et de chiffres inscrits dans des cases, une problématique bien adaptée à la puissance d'abstraction des CNN. Ce chapitre présente donc les bases théoriques de ces réseaux, avant que leur mise en œuvre ne soit détaillée dans les chapitres suivants.

2. Définition et principe des CNN:

Les réseaux de neurones convolutifs (CNN) tirent leur nom de l'opération mathématique de **convolution**, qu'ils utilisent dans certaines couches à la place des multiplications matricielles classiques. Cette opération permet d'extraire automatiquement des caractéristiques locales utiles dans les données, notamment les images. Après la convolution, une **fonction d'activation non linéaire** est appliquée pour introduire de la complexité dans les représentations apprises, ce qui permet au réseau de distinguer des différences sémantiques entre les éléments visuels. Gurney, K. (1997)

3. Architecture fonctionnelle d'un CNN:

L'architecture d'un réseau de neurones convolutifs (CNN) repose sur une organisation en **couches successives**, où chaque couche a un rôle spécifique dans l'extraction, la transformation et l'interprétation des informations visuelles. Cette approche modulaire permet au réseau d'apprendre progressivement des représentations de plus en plus abstraites à partir d'une image en entrée. On distingue généralement **quatre types principaux de couches** dans un CNN :

3.1. La couche de convolution:

C'est le cœur du réseau convolutif. où elle est généralement placée en première position. Elle repose sur l'utilisation de **filtres** (ou noyaux), qui sont des petits tableaux de poids permettant d'extraire des **caractéristiques visuelles**. En appliquant ces filtres à l'image d'entrée, la couche produit des **cartes de caractéristiques** (*feature maps*). Par exemple, un filtre 3×3 appliqué à une image 5×5 génère une carte 3×3. Chaque filtre apprend à détecter un motif spécifique (comme un bord ou une texture), et plusieurs filtres permettent d'extraire différentes informations de l'image en parallèle. Merzougui, D. (s.d.)

(Équipe Blent. (2022, 21 juin).)

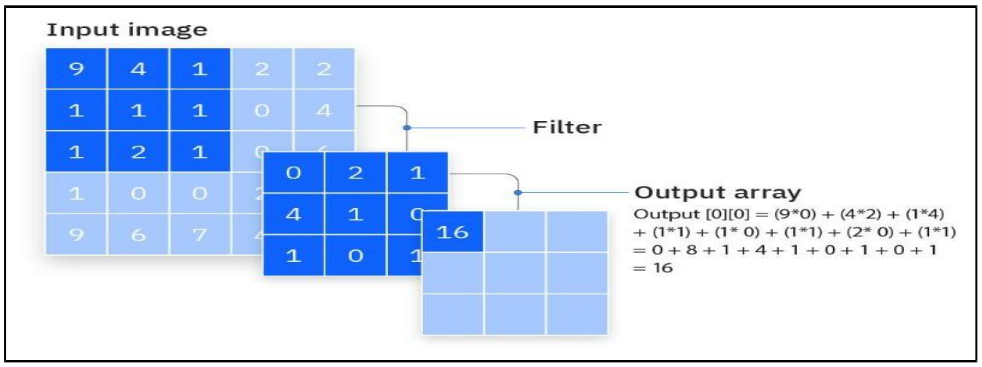


Figure 18:Exemple d'une convolution

3.2. La couche de correction ReLU:

La couche **ReLU** (Rectified Linear Unit) accélère et optimise l'apprentissage en **remplaçant les valeurs négatives par zéro** tout en conservant les valeurs positives. Ce mécanisme, appelé fonction d'activation, permet de ne transmettre à la couche suivante que les caractéristiques activées. MathWorks. (s.d.).

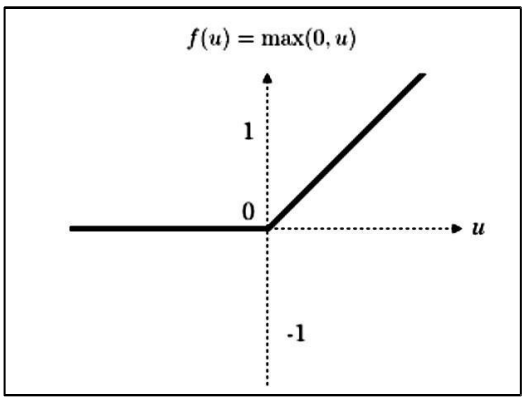


Figure 19:Fonction d'activation ReLU

3.3. La couche de pooling:

La **couche de pooling**, souvent placée entre deux couches de convolution, sert à **réduire la taille des images** tout en **préservant leurs caractéristiques essentielles**. Elle applique à chaque **feature map** une opération de **Max Pooling**, qui consiste à faire glisser une fenêtre (ou **kernel**) de taille fixe (souvent 2×2 ou 3×3) sur l'image avec un certain **pas** (*stride*), et à **conserver la valeur maximale** dans chaque zone.

Cette opération :

- **Diminue le nombre de paramètres** et les calculs du réseau,
- **Améliore l'efficacité** et réduit le **risque de sur-apprentissage**,
- Rend le réseau **moins sensible à la position ou à l'orientation** des objets.

Ainsi, même si les détails sont moins précis après pooling, cela permet une meilleure **invariance spatiale**, utile pour des tâches comme la reconnaissance d'objets (ex. : un chien reste un chien, même si ses oreilles sont légèrement déplacées). Merzougui, D. (s.d.).

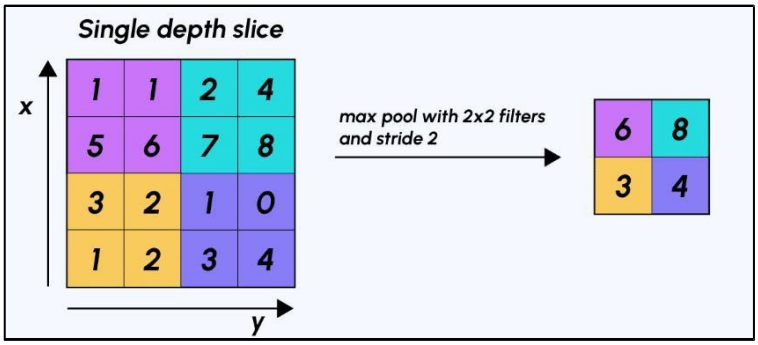


Figure 20:Représentation de max pooling

3.4. Couche Fully Connected (FC) :

La **couche fully-connected** (ou entièrement connectée) se situe généralement **à la fin d'un réseau de neurones**. Elle reçoit en entrée un **vecteur unidimensionnel** et produit un **nouveau vecteur** en sortie, en appliquant une **combinaison linéaire** suivie, si besoin, d'une **fonction d'activation**.

Dans le cadre d'un problème de **classification d'images**, cette dernière couche a une **taille égale au nombre de classes** N. Chaque élément du vecteur de sortie représente la **probabilité** que l'image appartienne à une classe donnée.
Par exemple, pour distinguer un **"A"** d'un **"B"**, la sortie sera un vecteur de taille 2 :
[0.9 ; 0.1] signifie que le modèle attribue **90 % de probabilité** à la classe "A" et **10 %** à la classe "B".

Le calcul se fait en **multipliant** chaque valeur d'entrée par un **poids**, en faisant la **somme** des résultats, puis en appliquant une **fonction d'activation** adaptée :

- **Sigmoïde** ou **fonction logistique** pour deux classes (N = 2),
- **Softmax** pour plus de deux classes (N > 2).

Cette architecture, dans laquelle **chaque neurone d'entrée est connecté à tous les neurones de sortie**, explique le nom **fully-connected**. Merzougui, D. (s.d.).

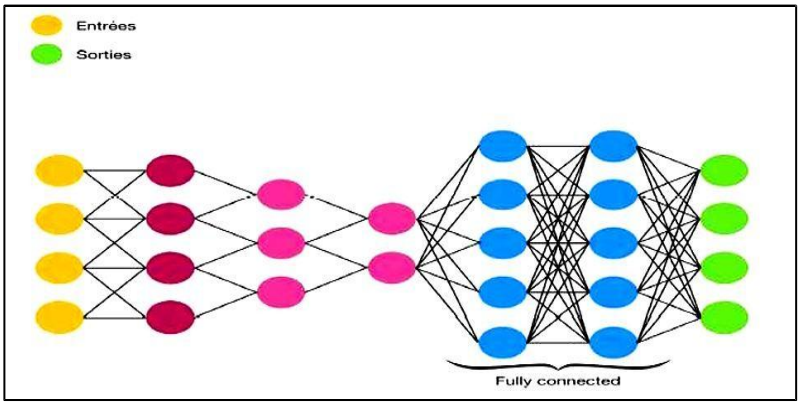


Figure 21:Représentation de la couche fully-connected

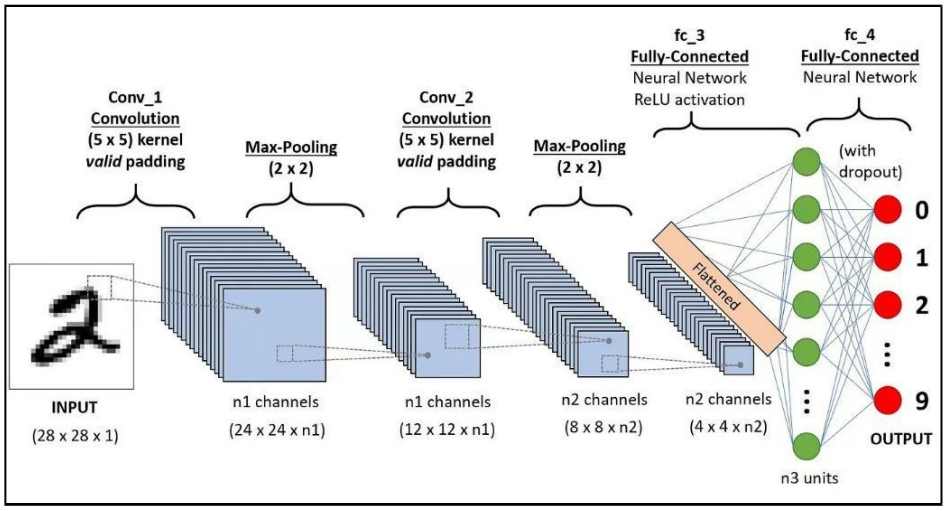


Figure 22:Exemple des couches convolutives

4. Les différentes architectures d'un CNN:

Diverses architectures de CNN ont joué un rôle essentiel dans le développement de l'intelligence artificielle, en particulier dans les domaines de la vision par ordinateur et de la reconnaissance automatique. Chacune a introduit des concepts novateurs qui ont permis d'améliorer la profondeur des réseaux, leur capacité d'apprentissage et leur robustesse face aux données complexes. Voici une présentation des modèles les plus emblématiques.

4.1. LeNet:

Développée dans les années 1990 par Yann LeCun, **LeNet** est l'une des premières architectures de CNN. Composée de sept couches (dont des couches de convolution, de

pooling, de normalisation et des couches fully connected), elle a été conçue à l'origine pour la reconnaissance de chiffres manuscrits, notamment sur la base de données **MNIST**. Son architecture simple mais efficace a posé les bases des réseaux modernes et a été utilisée dans diverses tâches telles que la classification d'images, la reconnaissance de caractères, et même l'analyse de signaux. Équipe Blent. (2022, 21 juin).

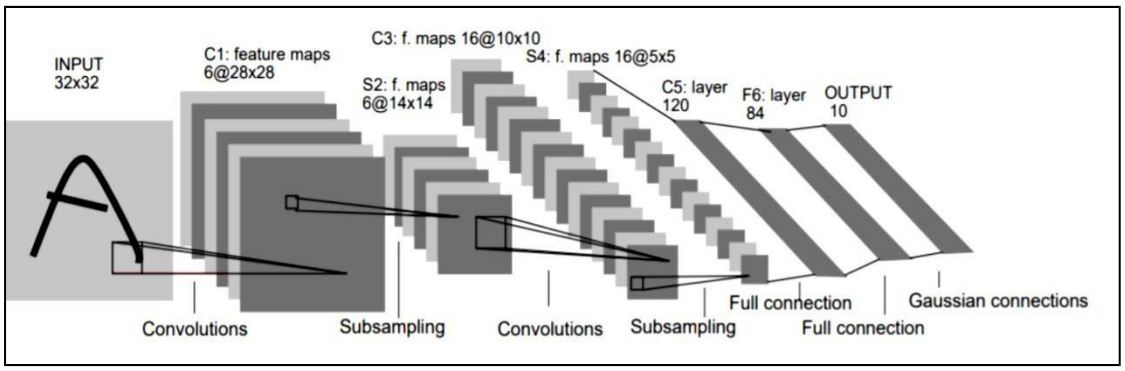


Figure 23: LeNet-5

4.2. AlexNet:

Introduite en 2012, **AlexNet** a marqué un tournant dans l'histoire de l'apprentissage profond en remportant le concours ImageNet (ILSVRC) avec une marge significative. Elle comporte 5 couches de convolution et 3 couches entièrement connectées, combinées à des techniques innovantes pour l'époque telles que la **normalisation locale**, le **Dropout** pour la régularisation, et l'usage de **GPU pour l'entraînement**. AlexNet a démontré l'efficacité des CNN profonds sur de très grands ensembles de données. Vitalflux. (2023, février 25).

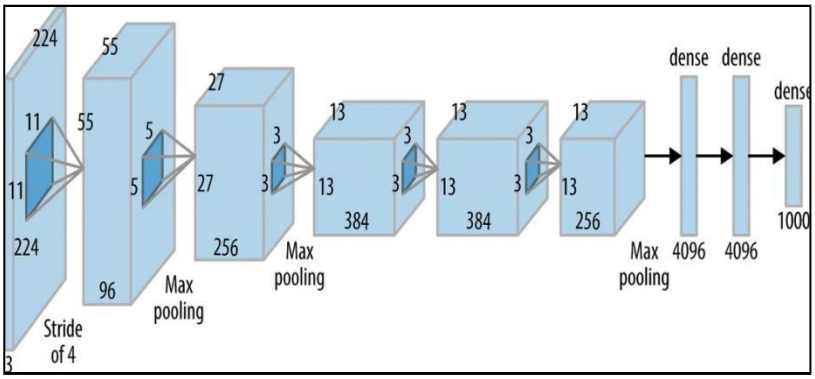


Figure 24: Alex Net

4.3. VGGNet :

Développée par le Visual Geometry Group de l'Université d'Oxford, **VGGNet** a été proposée dans l'article "Very Deep Convolutional Networks for Large-Scale Image Recognition" en 2014. Les variantes les plus connues, **VGG-16** et **VGG-19**, comportent respectivement 16 et 19 couches, essentiellement constituées de convolutions 3×3 empilées. Cette structure régulière et profonde a permis d'obtenir d'excellentes performances tout en facilitant l'analyse des effets de la profondeur dans les réseaux. Certaines versions remplacent les couches de pooling par des convolutions avec stride. Équipe Blent. (2022, 21 juin).

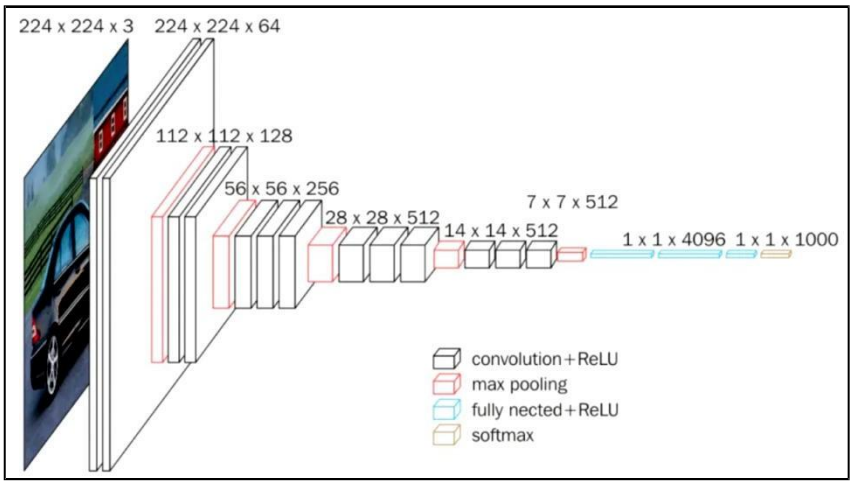


Figure 25: VGGNet

4.4. ResNet:

Proposée en 2015 par Microsoft Research, **ResNet (Residual Network)** introduit le concept de **blocs résiduels** à l'aide de **raccourcis (skip connections)**. Cela permet de construire des réseaux très profonds, jusqu'à 152 couches, sans subir les problèmes habituels d'apprentissage comme la disparition ou l'explosion du gradient. Chaque bloc résiduel ajoute directement l'entrée d'un module à sa sortie, ce qui facilite le flux d'information à travers le réseau. ResNet a établi de nouveaux standards de performance sur de nombreuses tâches de classification d'images. Équipe Blent. (2022, 21 juin).

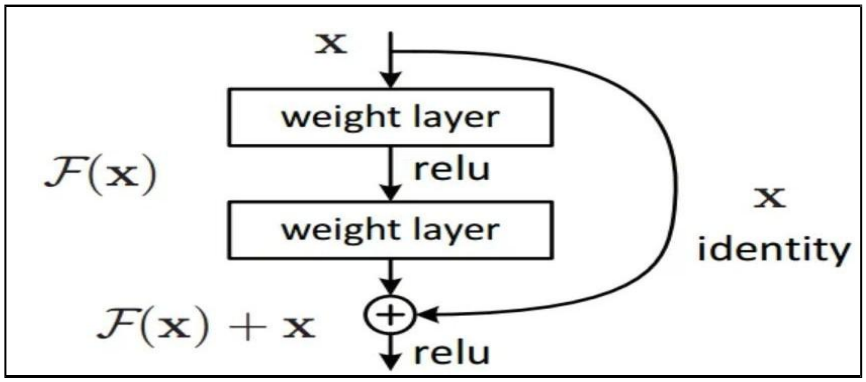


Figure 26:Bloc résiduel unique

5. Le Processus d'apprentissage d'un CNN:

Une fois l'architecture d'un réseau convolutif définie, l'enjeu consiste à ajuster ses **paramètres internes (poids et biais)** afin qu'il puisse effectuer des prédictions fiables. Ce processus d'apprentissage repose sur une succession d'étapes qui permettent au réseau de s'améliorer progressivement en analysant des exemples annotés. On parle alors d'**apprentissage supervisé**.

5.1. Propagation avant (Forward Propagation):

La propagation avant est le processus par lequel un réseau de neurones transforme une entrée en prédiction. Les données circulent de la couche d'entrée à la sortie, en passant par des transformations successives. Chaque couche applique **des poids**, **biais** et **fonctions d'activation** pour extraire des motifs complexes. Ce mécanisme permet au réseau de construire des représentations utiles pour la tâche visée. Tuychiev, B. (2025, 19 mars).

5.2. Calcul de l'erreur:

Le calcul de l'erreur utilise la fonction de perte d'**entropie croisée**, qui mesure la différence entre la distribution de probabilité prédite par le modèle et la distribution réelle des labels. Cette fonction quantifie l'erreur en évaluant combien la prédiction s'écarte de la vérité, et sert à ajuster les poids du réseau pour améliorer ses performances. Une valeur d'entropie

croisée plus faible indique une meilleure précision du modèle. Tuychiev, B. (2024, 16 janvier).

5.3. Calcul du gradient (Backward Propagation):

La rétropropagation calcule les gradients de la fonction de perte en partant de la sortie vers l'entrée du réseau. Ces gradients servent à ajuster les poids et biais de chaque neurone pour corriger l'erreur du modèle. Masdoum, F. (2024, 17 juillet).

5.4. Mise à jour des poids:

Grâce aux gradients calculés lors de la rétropropagation, les poids et les biais du réseau sont ajustés à l'aide d'algorithmes d'optimisation. Le plus classique est la **descente de gradient**, qui met à jour les paramètres pour minimiser la fonction de perte. Aujourd'hui, des méthodes plus avancées comme **Adam** sont souvent préférées, car elles adaptent automatiquement la vitesse d'apprentissage pour accélérer et stabiliser la convergence.

Chaque cycle complet — **propagation avant**, **calcul de l'erreur**, **rétropropagation**, puis **mise à jour des poids** — constitue une **itération** d'apprentissage. Masdoum, F. (2024, 17 juillet).

5.5. Répétitions sur plusieurs époques:

L'ensemble du processus est répété sur de nombreux **mini-lots de données (mini-batches)** pendant plusieurs **époques** (*epochs*), càd plusieurs passages complets sur le jeu de données d'entraînement.

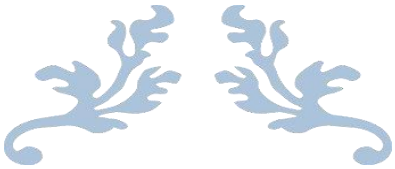
5.6. Évaluation sur données de validation:

Une partie des données est réservée à la **validation** : le réseau est évalué sur des exemples qu'il n'a pas vus pendant l'entraînement. Cela permet de :

- Surveiller la performance réelle du modèle,
- Détecter un **surapprentissage** (overfitting) si la précision en validation diminue.

6. Conclusion:

Les réseaux de neurones convolutifs représentent une avancée majeure dans le traitement d'images, grâce à leur capacité à apprendre automatiquement des caractéristiques complexes. Leur processus d'apprentissage structuré permet une amélioration progressive des performances. En combinant des couches spécialisées et des algorithmes d'optimisation efficaces, les CNN offrent une solution puissante aux tâches de reconnaissance visuelle. Ils constituent ainsi un pilier fondamental du deep learning moderne.



CHAPITRE 04 : Analyse et conception de notre système

1. Introduction

Ce chapitre présente l'analyse et la conception de notre système de correction automatique de QCM. Nous commençons par définir le langage UML et son rôle dans la modélisation des systèmes informatiques. Ensuite, nous rappelons les objectifs de notre projet et identifions les différents acteurs et cas d'utilisation. La conception de la base de données est également détaillée. Nous proposons ensuite plusieurs diagrammes UML (cas d'utilisation, classes et séquence) pour modéliser les interactions et la structure du système. Enfin, nous décrivons en détail les modules développés, depuis l'acquisition des copies scannées jusqu'à l'exportation des résultats. Cette modélisation a permis de structurer efficacement le développement de notre application.

2. Présentation d'UML

2.1. Définition de l'UML

L'UML (Unified Modeling Language), ou langage de modélisation unifié en français, est un langage graphique utilisé pour modéliser des systèmes informatiques. Il constitue aujourd'hui la norme en matière de modélisation orientée objet. Cette approche consiste à représenter des éléments du monde réel (tels qu'un immeuble, un ingrédient, une personne, un logo, ou un organe du corps humain) ou virtuel (comme le temps, le prix ou une compétence) sous forme d'entités appelées « objets ». (Booch et al., 1999).

Figure 27:Logo du langage de modélisation UML (Unified Modeling Language)

2.2. Rôle de l'UML dans la modélisation d'un système

L'UML (Unified Modeling Language) joue un rôle essentiel dans la modélisation des systèmes informatiques.

Il fournit un ensemble standardisé de diagrammes permettant de représenter de manière graphique la structure statique et le comportement dynamique d'un système. Grâce à l'UML, les concepteurs peuvent **visualiser**, **spécifier**, **construire** et **documenter** efficacement toutes les étapes du développement logiciel.

UML permet notamment de :

- Comprendre les exigences du système en identifiant les acteurs et leurs interactions (via les diagrammes de cas d'utilisation).
- Décomposer le système en classes, objets et relations (avec les diagrammes de classes).
- Décrire les scénarios d'exécution et l'échange d'informations entre les composants (diagrammes de séquence, d'activités).
- Faciliter la communication entre tous les acteurs du projet (développeurs, analystes, clients) grâce à une représentation visuelle claire et partagée.

Ainsi, UML constitue **un langage universel** pour modéliser des systèmes complexes, tout en améliorant la cohérence, la compréhension et la qualité des projets logiciels.

3. Objectif de notre projet

L'objectif principal de notre projet est de concevoir et de réaliser une application capable de corriger automatiquement des copies d'examen sous forme de questionnaires à choix multiples (QCM).

Cette application vise à **réduire considérablement le temps** et **l'effort humain** nécessaires pour corriger un grand volume de copies, tout en **améliorant la précision** et **l'objectivité** de l'évaluation.

Plus précisément, notre système doit permettre :

- La **numérisation** des copies d'examen via un scanner.
- Le **traitement automatique** des images afin de détecter et d'analyser les réponses des étudiants.
- La **lecture** et la **reconnaissance** des cases cochées pour chaque question.
- L'**extraction des informations d'identification** de l'étudiant (nom, prénom, matricule) à partir des formulaires.
- Le **calcul automatique de la note** en comparant les réponses extraites avec un corrigé type préenregistré.
- La **génération de rapports** clairs et exploitables, permettant une exportation facile des résultats.

À travers ce projet, nous cherchons à intégrer des techniques de **traitement d'images**, de **reconnaissance de formes** et d'**intelligence artificielle légère** pour aboutir à une solution pratique, fiable et accessible aux établissements éducatifs.

4. Analyse du système

Cette section présente l'analyse fonctionnelle du système de correction automatisée de QCM. Elle permet d'identifier les acteurs impliqués et de détailler les cas d'utilisation associés à chaque acteur.

4.1. Identification des acteurs

a) Agent :

L'agent est l'utilisateur humain du système. Il se connecte à l'application, sélectionne les examens à corriger, scanne ou importe les copies QCM, configure le corrigé type, lance la correction automatique et consulte les résultats et statistiques. Il peut également exporter ces données sous différents formats.

4.2. Spécification des cas d'utilisation du système

Tâche	Acteur principal	Description
CU1:Authentification	Agent	Se connecter au système en saisissant ses identifiants (nom utilisateur,mot de passe).
CU2 : Sélection d'un examen	Agent	Choisir l'examen à corriger dans la liste des examens disponibles.
CU3 : Numérisation ou importation des copies QCM	Agent	Scanner les copies ou importer des copies déjà scannées dans le système.
CU4: Lancement du traitement des copies.	Agent	Démarrer le traitement automatique des copies importées pour extraction des informations.
CU5: Lecture du nom, prénom, matricule	Système (via CNN)	Extraire les informations d'identité à partir des zones manuscrites
CU6: Lecture des réponses	Système	Extraire les réponses des copies.
CU7: Chargement du corrigé type	Agent	Saisir manuellement ou importer le corrigé officiel correspondant à l'examen.

CU8 : Lancement du correction automatique.	Agent	Démarrer l'analyse automatique des copies.
CU9: Génération des notes	Système (automatique)	Analyser les réponses extraites, comparer avec le corrigé type et calculer les notes.
CU10 : Consultation et Exportation des résultats	Agent	Afficher les résultats individuels et globaux, et enregistrer les résultats au format Excel.
CU11:Génération des statistiques	Système (automatique)	Produire et afficher des statistiques détaillées sur les performances des étudiants (moyenne, taux de réussite, etc.).
CU12: Consultation et exportation des statistiques	Agent	Consulter les statistiques générées automatiquement par le système et les exporter si nécessaire.

Tableau 1 : Spécification des cas d'utilisation du système

4.3. Conception de la base des données

Une Base de Données, communément appelée BD ou BDD, est un système structuré et organisé permettant de stocker de vastes quantités d'informations afin de faciliter leur exploitation, notamment l'ajout, la mise à jour et la recherche de données. Concrètement, une base de données se traduit par un ensemble de fichiers physiquement enregistrés sur un disque.

4.3.1. Description des tables

- **Table users:**

Champ	Type	Description
Id(CP)	INTEGER	Identifiant d'utilisateur
username	TEXT	Le nom d'utilisateur
password	TEXT	Mot de passe d'utilisateur

Tableau 2 : Table users (utilisateurs du système)

- **Table exams:**

Champ	Type	Description
Id(CP)	INTEGER	Identifiant d'examen
name	TEXT	Le nom d'examen

university	TEXT	université
exam_date	TEXT	Date d'examen
Folder_path	TEXT	Le chemin du dossier
logo_path	TEXT	Le chemin du logo
exam_type	TEXT	Le type d'examen
total_questions	INTEGER	Le nombre des questions
grading_mode	TEXT	Le mode de notation
Status	TEXT	Le statut d'examen
date_created	TIMESTAMP	La date de création d'examen
corrige	TEXT	Le corrigé type d'examen

Tableau 3 : Table exams (informations sur les examens)

- **Table etudiants:**

Champ	Type	Description
Id(CP)	INTEGER	Identifiant d'étudiant
matricule	TEXT	Matricule d'étudiant
nom	TEXT	Le nom d'étudiant
prenom	TEXT	Le prénom d'étudiant
exam_name	TEXT	Le nom d'examen

Tableau 4 :Table étudiants (étudiants inscrits à un examen)

- **Table copies:**

Champ	Type	Description
id_copies(CP)	INTEGER	Identifiant d'un copie
id_examen	INTEGER	Identifiant d'examen
Matricule	INTEGER	Matricule d'étudiant
reponses	TEXT	Réponses extraites du document (json)
note	INTEGER	La note calculée

chemin_copie	TEXT	Le chemin du copie

Tableau 5 : Table copies (copies corrigées par le système)

4.3.2. Le modèle conceptuel des données (MCD)

Le modèle conceptuel des données (MCD) a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information. Il s'agit donc d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'entités.

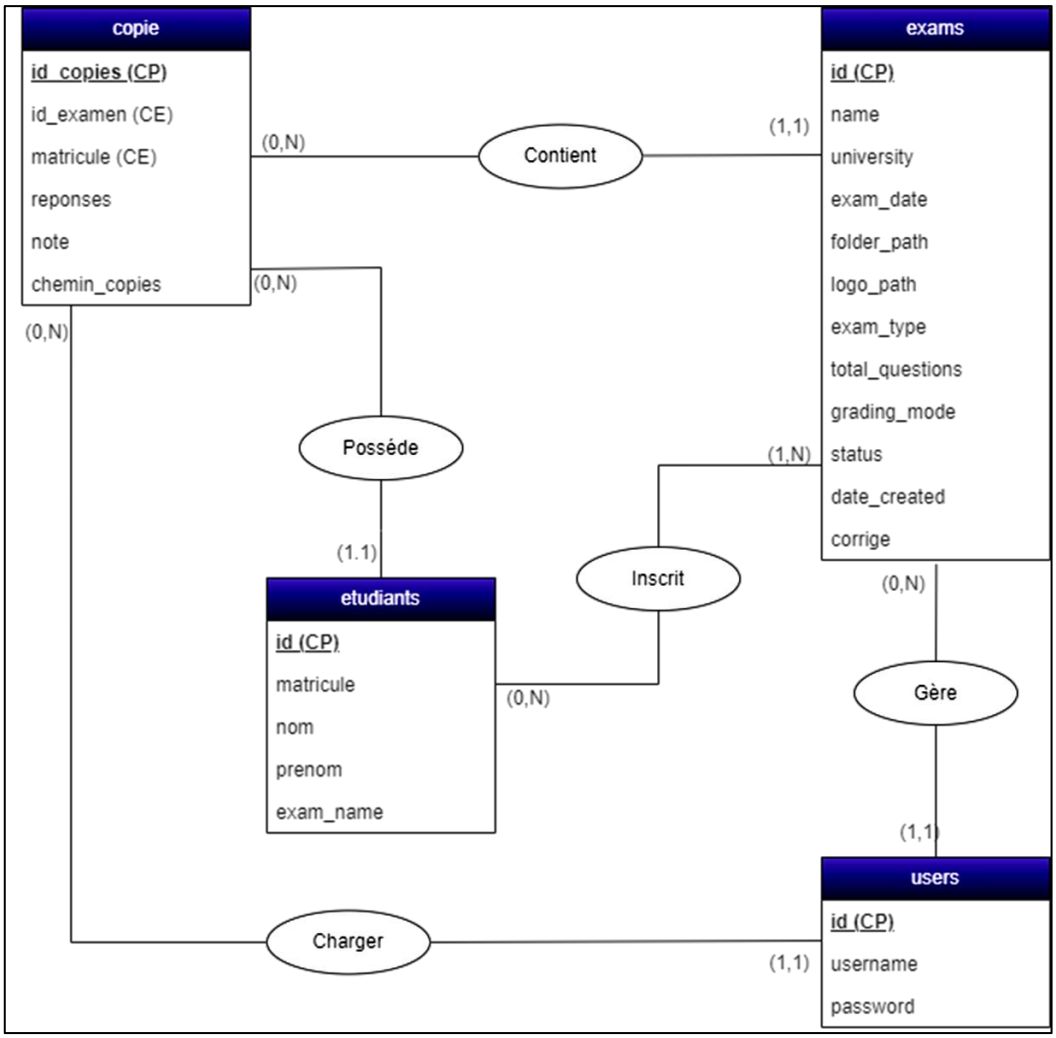


figure 28 :Le modèle conceptuel des données (MCD)

5. Modélisation UML

5.1. Diagramme de cas d'utilisation

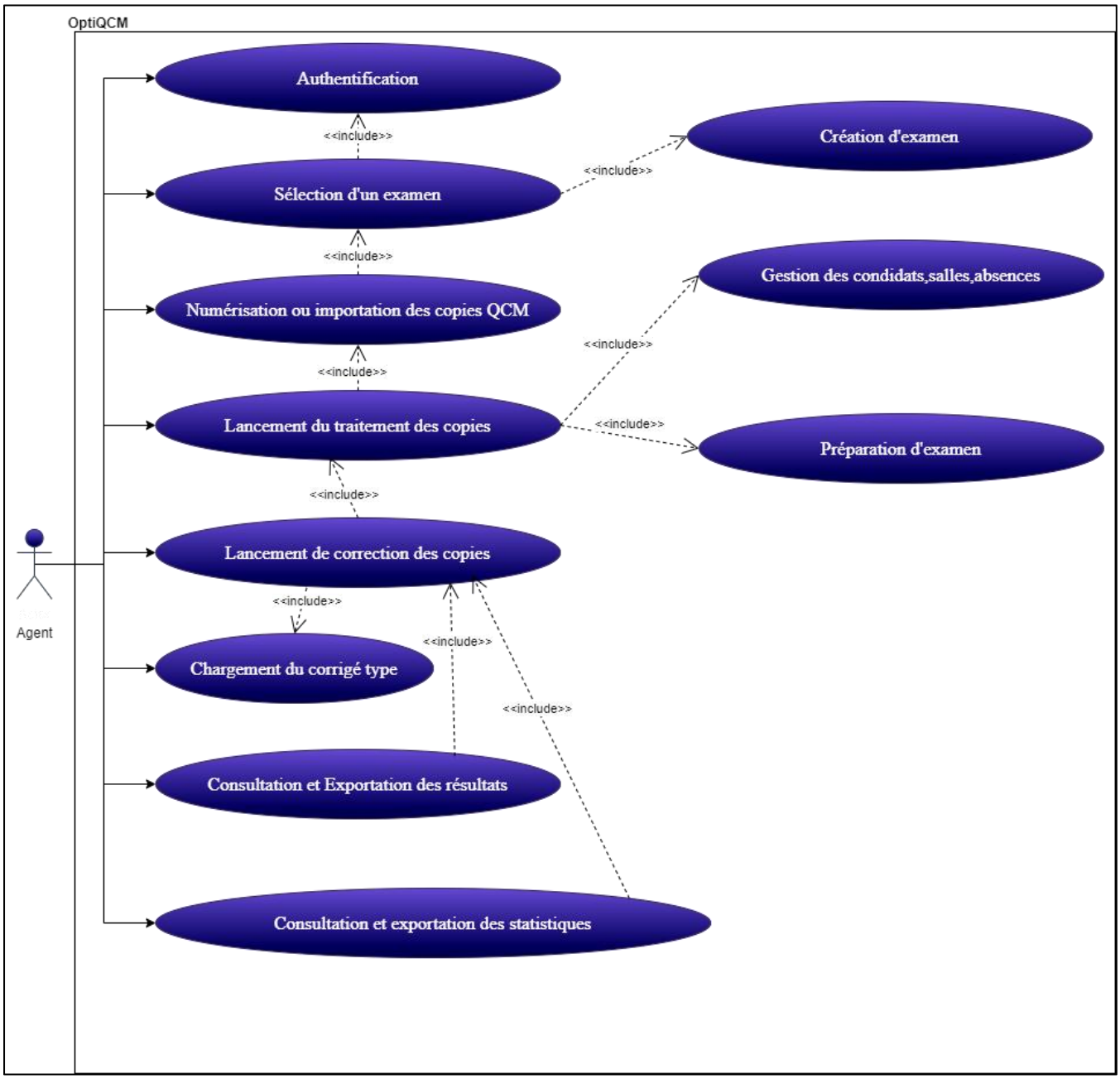


figure 29 :Diagramme de cas d'utilisation

5.2. Diagramme de classes

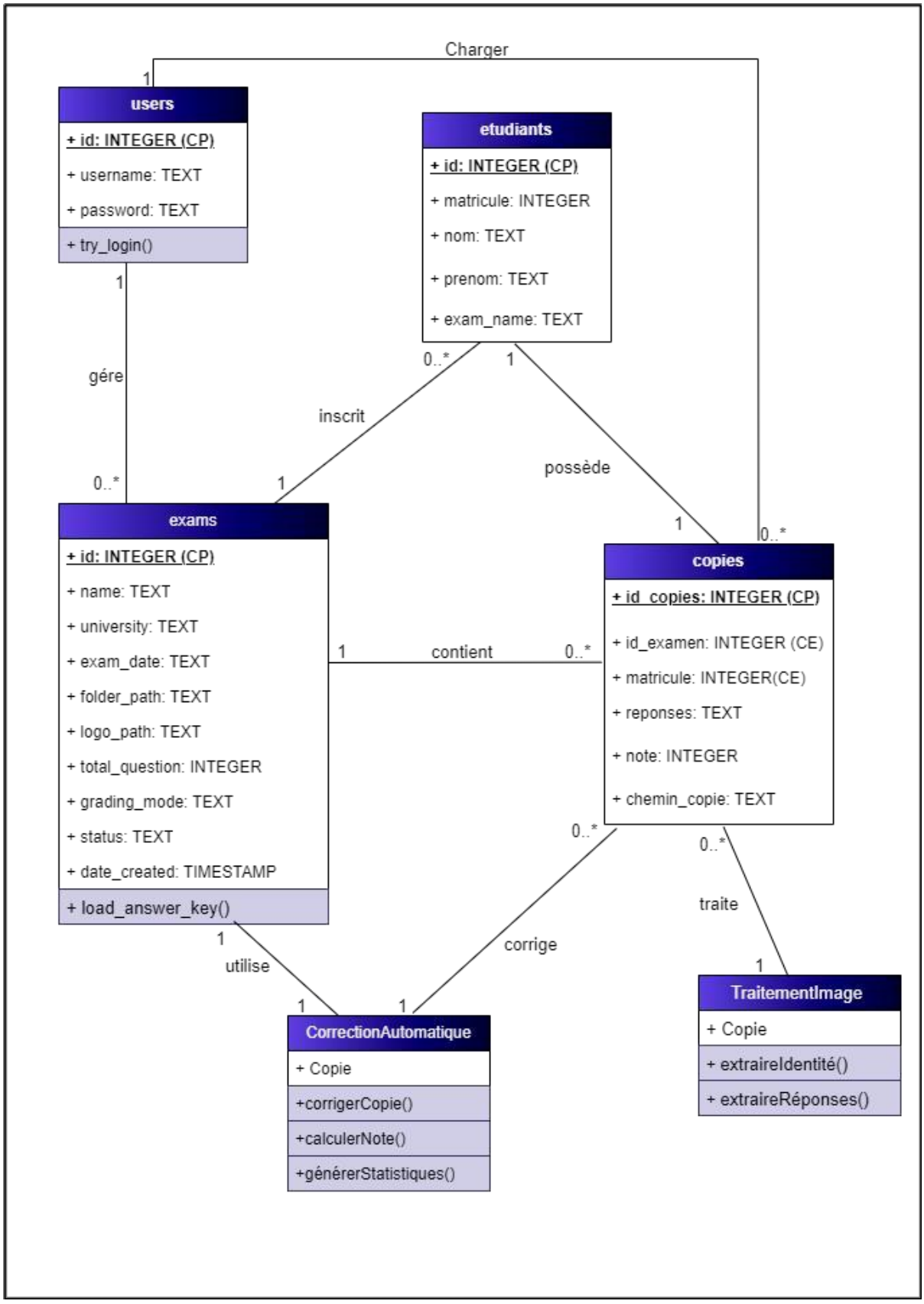


figure 30 :Diagramme de classes

5.3. Diagramme de séquence

- **Diagramme de séquence page <<connexion>>**

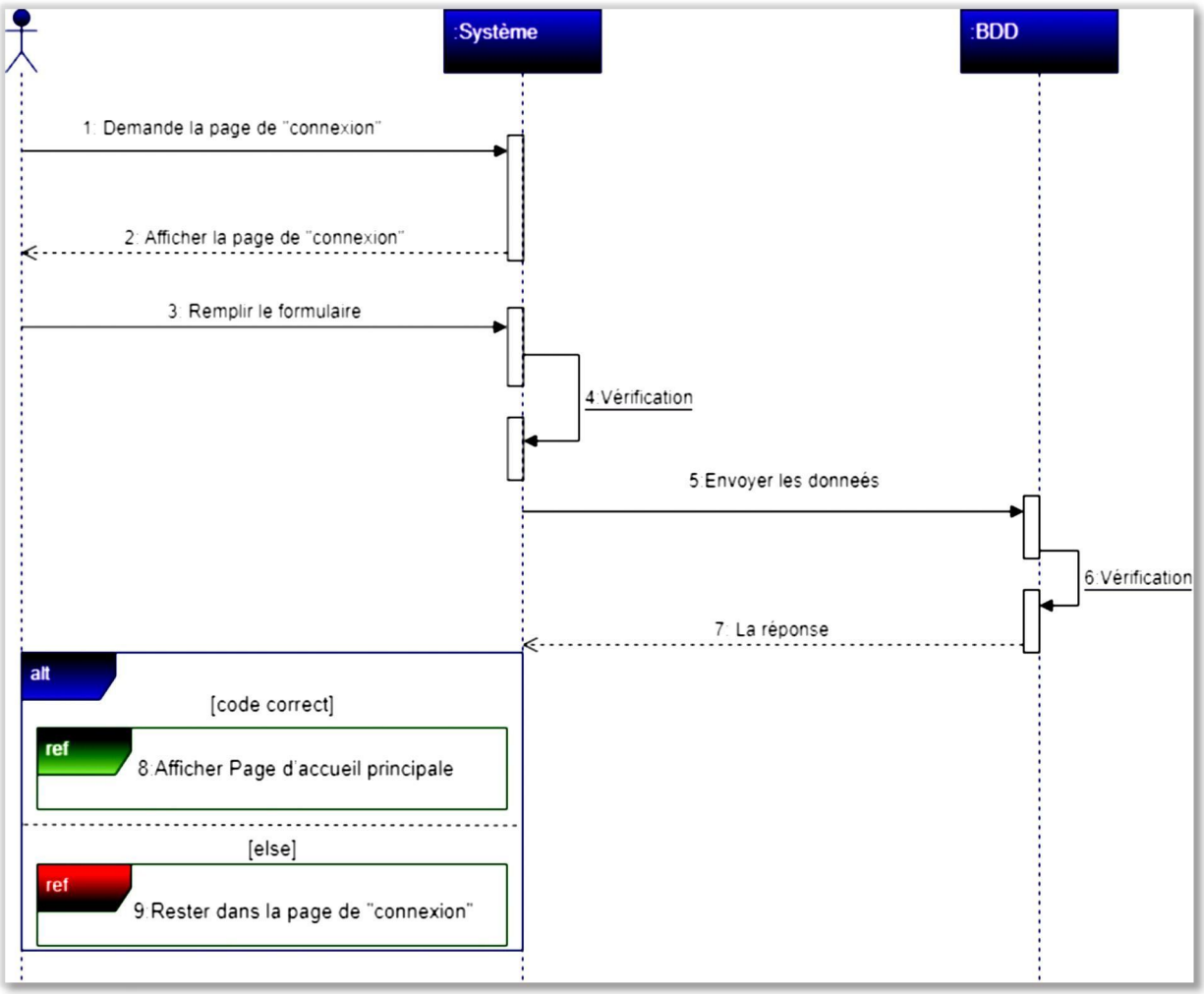


Figure 31 : Diagramme de séquence << page connexion>>

- **Diagramme de séquence page << Numérisation et traitement>>**

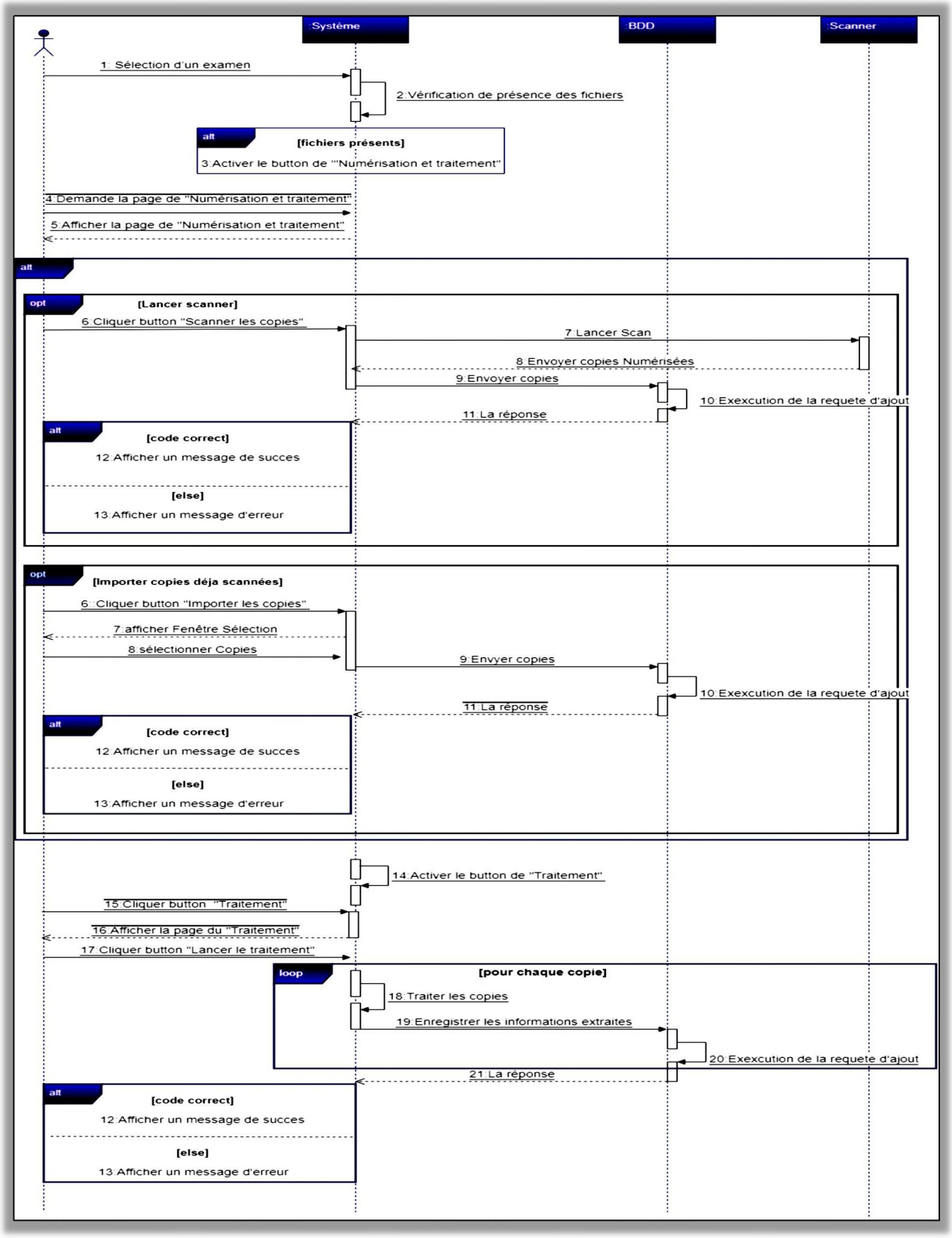


Figure 32 : Diagramme de séquence << page Numérisation et traitement>>

- **Diagramme de séquence page <<Correction>>**

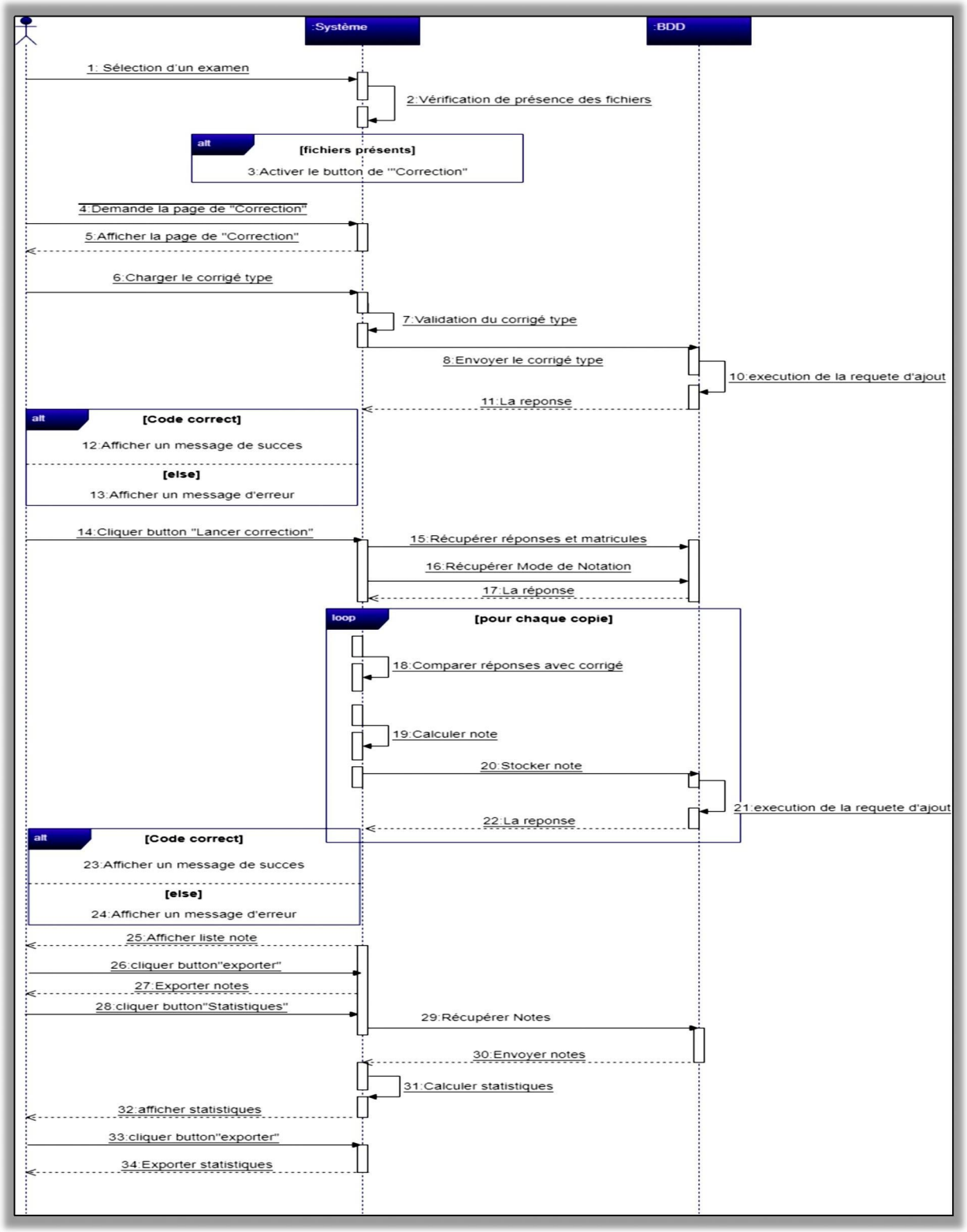


Figure 33 : Diagramme de séquence << page Correction>>

6. Description détaillée du système.

Le système développé a pour objectif de numériser, analyser et corriger automatiquement des copies d'examen à choix multiples (QCM), scannées au format PDF. Il est conçu de manière modulaire, chaque composant prenant en charge une étape spécifique du traitement : de l'acquisition des copies à la génération des résultats corrigés. Cette approche garantit une grande flexibilité et une évolutivité facilitée du système.

Le fonctionnement global repose sur une chaîne de traitement composé des modules suivants :

- Acquisition des copies scannées (au format PDF),
- Prétraitement des images extraites pour améliorer la qualité et l'alignement,
- Détection des zones de réponses et lecture des cases cochées,
- Reconnaissance des identifiants étudiants (nom, prénom, Matricule),
- Correction automatique des réponses et calcul des notes,
- Affichage des résultats, exportation des données et génération de statistiques.

Chaque module interagit avec les autres pour assurer une correction rapide, fiable et automatisée, réduisant au maximum l'intervention manuelle.

6.1. Module d'acquisition des copies scannées (au format PDF)

Le système commence par l'importation des copies d'examen, fournies sous forme de fichiers PDF scannés. Chaque fichier est traité individuellement. À l'aide de la bibliothèque **pdf2image**, la première page de chaque PDF est convertie en image haute résolution (300 dpi), ce qui permet une analyse fine des éléments visuels comme les cases à cocher ou les zones manuscrites.

6.2. Module de Prétraitement des images extraites pour améliorer la qualité et l'alignement

Une fois l'image obtenue, plusieurs étapes de prétraitement sont appliquées pour garantir la précision de la détection :

a. Filtrage Gaussien : un **filtre gaussien de taille 7×7** est appliqué pour **réduire le bruit** tout en **préservant les contours essentiels**, notamment les bordures des cases. Cette opération améliore la qualité de l'image et prépare efficacement l'étape suivante.

b. Binarisation (Otsu + inversion) : réalisée en utilisant la méthode d'**Otsu**, qui détermine automatiquement un seuil optimal pour séparer le fond du contenu pertinent. Les pixels dont la valeur est supérieure au seuil sont convertis en blanc (valeur 255), tandis que les pixels inférieurs au seuil sont convertis en noir (valeur 0). Cette image binaire obtenue constitue une base solide pour les étapes de détection des zones et de reconnaissance visuelle qui suivent.

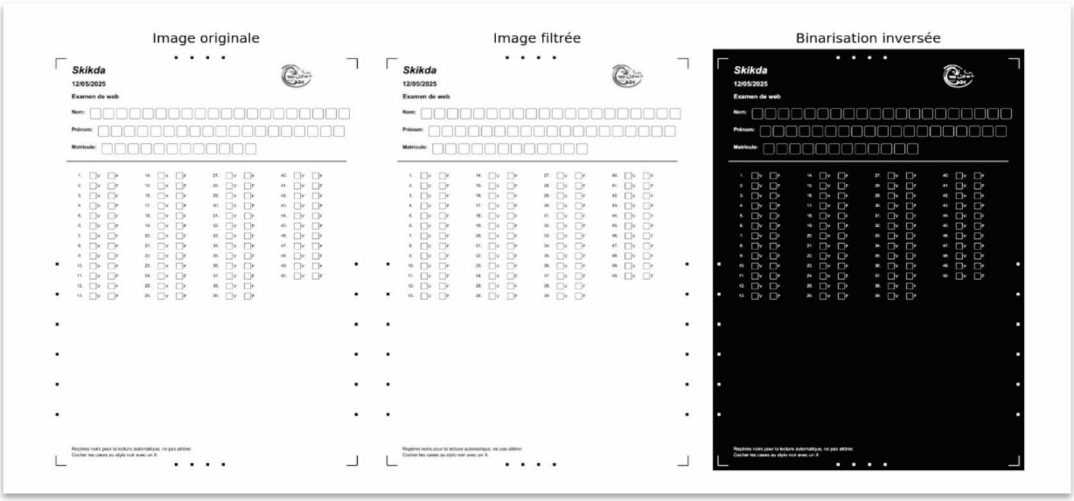


Figure 34 : Exemple de Filtrage Gaussien et Binarisation (Otsu + inversion)

C. Détection des coins : Pour localiser les coins supérieurs d'une copie d'examen, deux zones spécifiques sont extraites de l'image binaire en niveaux de gris :

→ **Coin gauche** : (X=0, Y=0, largeur=380 px, hauteur=360 px)

→ **Coin droit** : (X=2100, Y=0, largeur=380 px, hauteur=360 px)

Chaque zone est ensuite traitée pour identifier les **composantes connexes** (objets distincts), puis le **nombre de pixels blancs** est calculé pour chaque composante. L'objet contenant le plus grand nombre de pixels blancs est retenu comme **coin**.

Pour chaque coin identifié, on calcule les coordonnées de son **centre de gravité** (Xg ,Yg) selon les équations suivantes :

- Aire (ou représente le nombre de pixels de l'objet détecté) :

$$m_{00} = \sum_{x=0}^{M-1} \sum_{Y=0}^{N-1} f(X,Y) \dots\dots\dots\dots\dots\dots (01)$$

- Coordonnée Xg (abscisse du centre) :

$$Xg = \frac{\sum_{x=0}^{M-1} \sum_{Y=0}^{N-1} f(X,Y) .X}{m_{00}} \dots\dots\dots\dots\dots\dots (02)$$

- Coordonnée Yg (ordonnée du centre) :

$$Yg = \frac{\sum_{x=0}^{M-1} \sum_{Y=0}^{N-1} f(X,Y) .Y}{m_{00}} \dots\dots\dots\dots\dots\dots (03)$$

Où : f (x, y) est la valeur du pixel à la position (x, y) dans l'image binaire (0 ou 255).

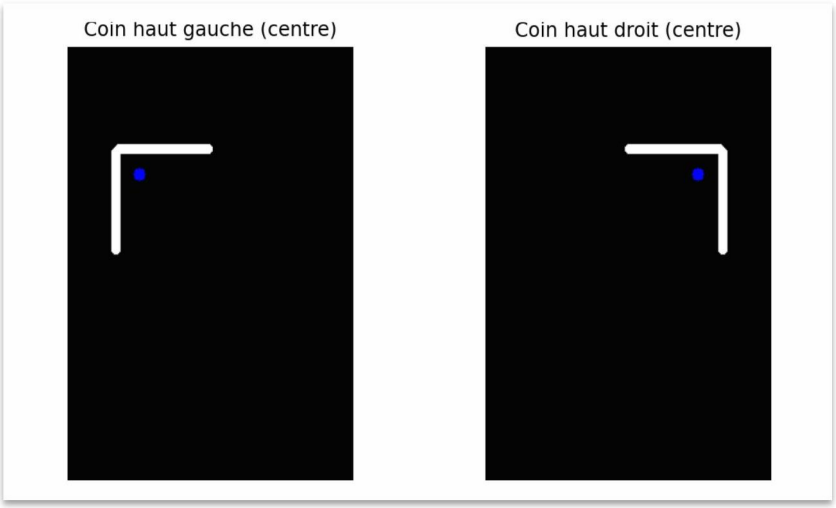


Figure 35 : le Centre de gravité des coins de l'image découpée.

Les coordonnées du centre de gravité du coin gauche sont exactes dans l'image initiale, tandis que celles du coin droit nécessitent une correction à l'aide des équations suivantes :

$$X_{g1} = X_{g1\ ancien}$$

$$y_{g1} = Y_{g1\ ancien}$$

$$x_{g2} = m - (L - X_{g2\ ancien})$$

$$y_{g2} = Y_{g2\ ancien}$$

x_{g1}, y_{g1}, x_{g2}, y_{g2} : les nouvelles coordonnés.

$X_{g1\ ancien}$, $Y_{g1\ ancien}$, $X_{g2\ ancien}$, $Y_{g2\ ancien}$ *: les anciens coordonnés.*

m : largeur du l'image initial (2480 pixels).

L : largeur de partie coupée (380 pixels).

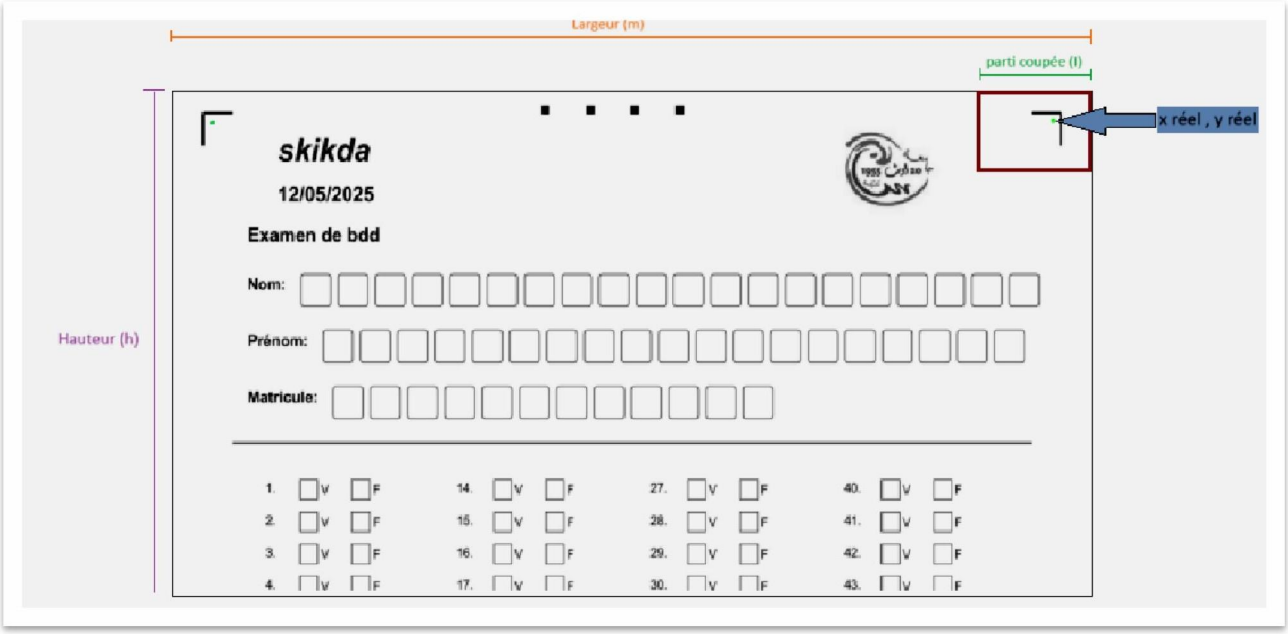


Figure 36: Les nouvelles coordonnées de centre de gravité de coin droit.

D. Correction de l'inclinaison

Pour corriger l'inclinaison, il faut tout d'abord calculer l'angle de rotation à appliquer. Cet angle est déterminé à partir des coordonnées des centres de gravité des deux coins détectés précédemment :

- Coin gauche : (xg1,yg1)
- Coin droit : (xg2,yg2)

L'angle de rotation se calcule à l'aide de la formule suivante :

$$\text{angle}=\text{atan}\left(\frac{yg2-yg1}{xg2-xg1}\right)$$

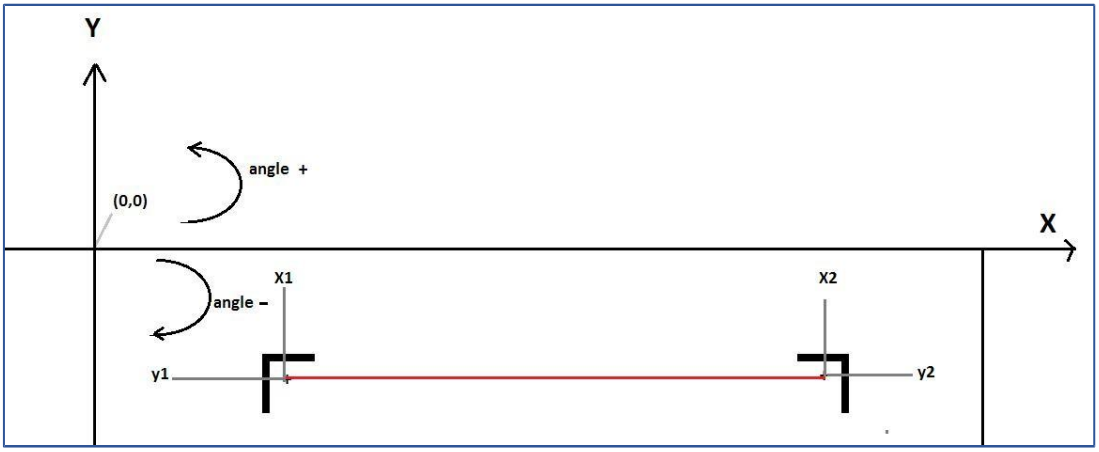


Figure 37 :Rotation d'une image autour de son origine.

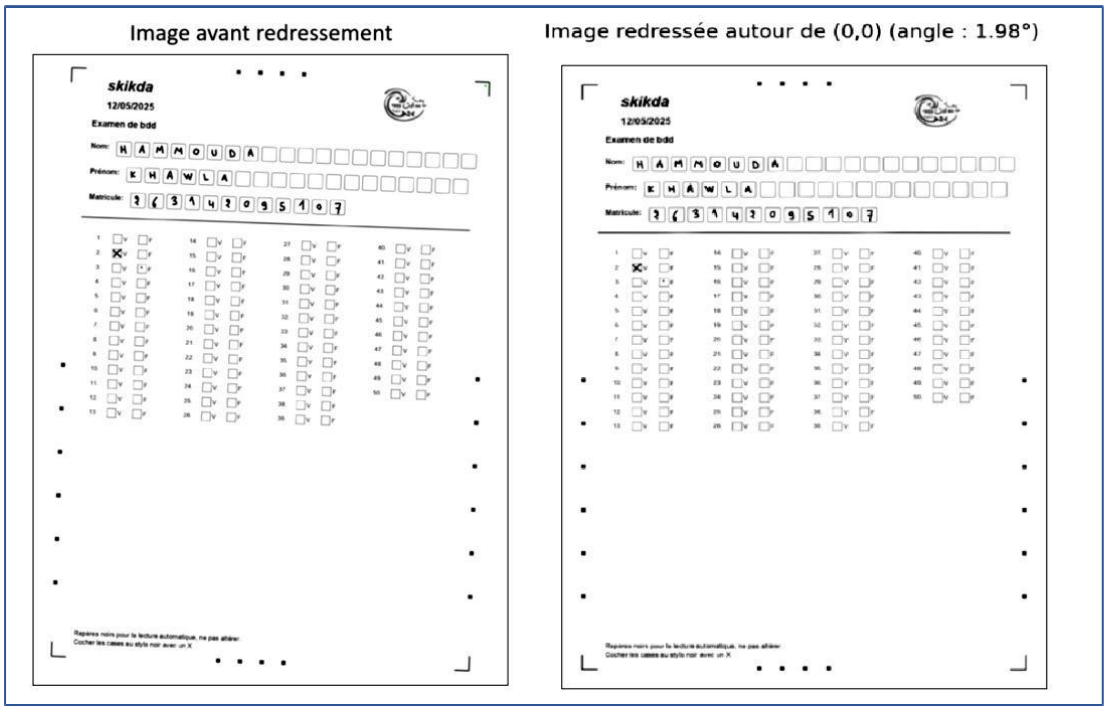


Figure 38 :Exemple de correction de l'inclinaison

E. Détection de la ligne verticale de séparation : Pour détecter précisément la position de la ligne de séparation dans la feuille QCM, on procède selon les étapes suivantes :

- **Découpage de la zone de recherche**

On extrait expérimentalement une sous-image de la feuille, définie par :

- X=0
- Y=850
- Largeur=2480
- Hauteur=260

Cette zone contient normalement la ligne de séparation recherchée.

- **Binarisation et extraction de la ligne**

  - On effectue une binarisation (par exemple, méthode d'Otsu), puis une inversion pour que la ligne noire apparaisse en blanc.
  - On applique un étiquetage des composantes connexes pour ne conserver que la plus grande composante (supposée être la ligne).

- **Recherche des positions extrêmes**

On parcourt tous les pixels de la composante extraite, et on calcule :

- Mini: la plus petite coordonnée verticale (ligne)
- Maxi : la plus grande coordonnée verticale (ligne)
- Minj : la plus petite coordonnée horizontale (colonne)
- Maxj: la plus grande coordonnée horizontale (colonne)

Pour chaque pixel (i,j) de la composante, si image(i,j)=alors :

- Si i > Maxi , Maxi=i
- Si i < Mini , Mini=i
- Si j > Maxj , Maxj=j
- Si j < Minj , Minj=j

- **Recalage dans l'image initiale**

Les coordonnées verticales sont recalculées par :

- $Mini_{image}$ =Mini+Y
- $Maxi_{image}$=Maxi+Y

Les coordonnées horizontales ne changent pas (car X=0) :

- $Minj_{image}$=Minj, $Maxj_{image}$=Maxj

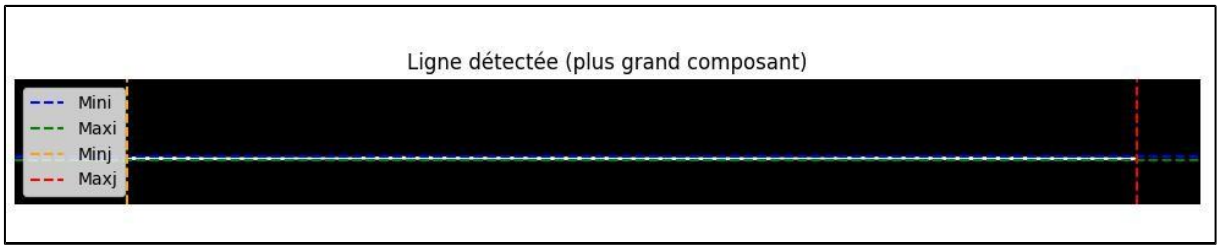


Figure 39 :la ligne de séparation détectée.

F. Extraction des coins inférieurs :

- **Découpage des coins bas**

  Deux sous-images fixes sont extraites dans la partie inférieure de l'image :

- **Partie gauche** : coordonnées (X=0, Y=3137), largeur = 380 pixels, hauteur = 360 pixels
- **Partie droite** : coordonnées (X=2100, Y=3137), largeur = 380 pixels, hauteur = 360 pixels

- **Binarisation et étiquetage**

  Chaque sous-image est binarisée à l'aide de la méthode d'Otsu (noir/blanc) puis inversée si nécessaire pour obtenir les formes en blanc sur fond noir.

  Un étiquetage des composantes connexes est ensuite appliqué afin d'identifier la plus grande zone blanche, supposée correspondre au repère de coin

- **Calcul du centre de gravité**

On utilise les équations (01), (02), (03) pour calculer le centre de gravité :

Coin bas gauche (xg3, yg3) et coin bas droite (xg4, yg4)

- **Calcul des nouvelles coordonnées dans l'image globale**

Les coordonnées globales des centres de gravité sont recalculées selon :

- $xg3 = Xg3_{ancien}$
- $yg3=nb-(h-(Yg3_{ancien}-left\_y))$
- $xg4=mb-(l-(Xg4_{ancien}-right\_x))$
- $yg4=nb-(h-(Yg4_{ancien}-right\_y))$

Où :

mb : largeur de l'image complète (ex. 2480 pixels)

nb : hauteur de l'image complète (ex. 3507 pixels)

l : largeur de la zone découpée (380 pixels)

h : hauteur de la zone découpée (360 pixels)

left_y, right_x, right_y : positions des sous-images extraites

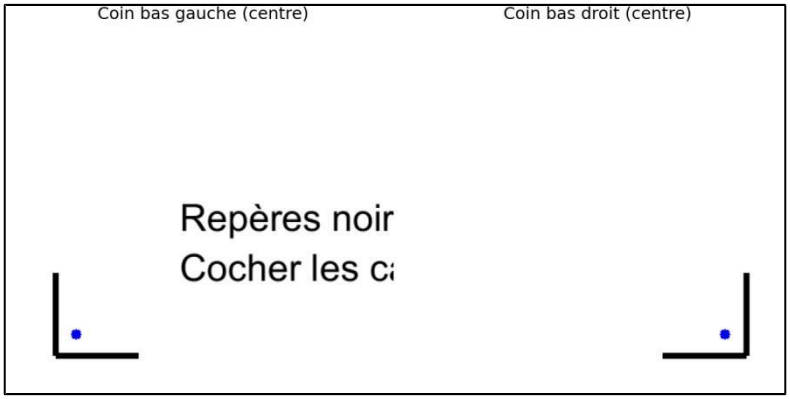


Figure 40:Extraction des coins inférieurs centre de gravité

6.3. Détection des zones de réponses et lecture des cases cochées

Les coordonnées des coins bas gauche et bas droit (**xg3, yg3, xg4**) ainsi que la position verticale supérieure (**Maxi**) sont utilisées pour déterminer la zone contenant les réponses du QCM.

La zone d'intérêt est extraite à partir de l'image principale selon :

- $x=x_{g3}$

- y=Maxi
- largeur=$x_{g4}-x_{g3}$
- hauteur=$y_{g3}-Maxi$

Cette opération permet d'isoler précisément la partie de la feuille où se trouvent toutes les cases à analyser.

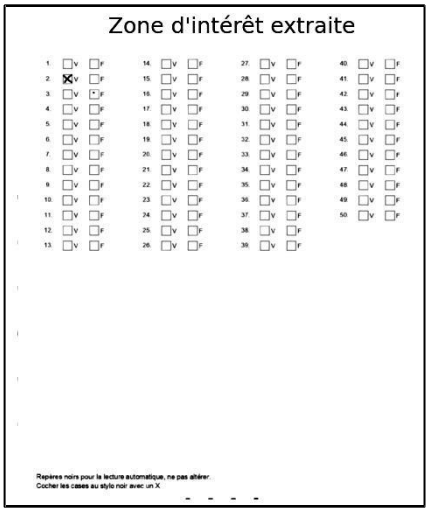


Figure 41:La zone d'intérêt extraite

- **Découpage de la zone de réponses en colonnes:**

La largeur totale de la zone de réponses est notée W, et le nombre de colonnes est 4.

- **Largeur d'une colonne :**

Pour chaque colonne i (avec $i \in [0,4-1]i$), la largeur est donnée par :

$$Largeur_{col}=\left[\frac{W}{4}\right]$$

- **Début et fin de chaque colonne :**

$$x_{start}=i*Largeur_{col}$$

$x_{end}=(i+1)*Largeur_{col}$(sauf pour la dernière colonne, ou $x_{end}=W$)

- **Nombre de lignes par colonne :**

$$L_{col}=\left[\frac{Nq}{4}\right]$$

où N_q est le nombre total de questions.

- **Détection automatique des cases par histogramme de projection**
  - Pour chaque colonne, l'image est binarisée (noir/blanc) et nettoyée par filtrage.
  - Des **histogrammes de projection horizontaux et verticaux** sont calculés pour localiser les lignes (lignes de questions) et colonnes (cases de réponses) dans la grille, grâce à la méthode de détection des transitions (zéros/début/fin).
  - Les coordonnées de chaque case sont alors identifiées automatiquement, évitant un découpage manuel.

- **Extraction individuelle et analyse de chaque case**

Chaque case détectée est extraite sous forme de petite image.

Pour chaque case :

- Application d'une détection de marque "**X**" à l'aide de la transformée de Hough sur l'image binarisée de la case.
- Si deux diagonales sont détectées (≈45° et ≈-45°), la case est considérée comme cochée.

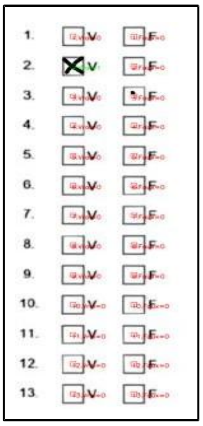


Figure 42:Exemple d'extraction de la première colonne et lecture des cases (case cochée = vert, case vide = rouge).

6.4. Reconnaissance des identifiants étudiants (nom, prénom, Matricule)

A. Extraction de la zone identité

On commence par extraire une bande horizontale située juste au-dessus de la ligne de séparation (détectée précédemment à l'aide de **Mini**).

Les coordonnées de la zone à extraire sont calculées ainsi :

Position horizontale :

- $x=x_{g3}$ (coordonnée du coin bas gauche)
- $w=x_{g4}-x_{g3}$(largeur entre les deux coins bas détectés)

Position verticale :

- y=Mini−hauteur (juste au-dessus de la ligne de séparation)
- h=hauteur(valeur fixée, par exemple 500 pixels)

La zone identité extraite correspond donc à l'image restreinte à ces coordonnées :

- zone_identite=image[y:y+h, x:x+w]

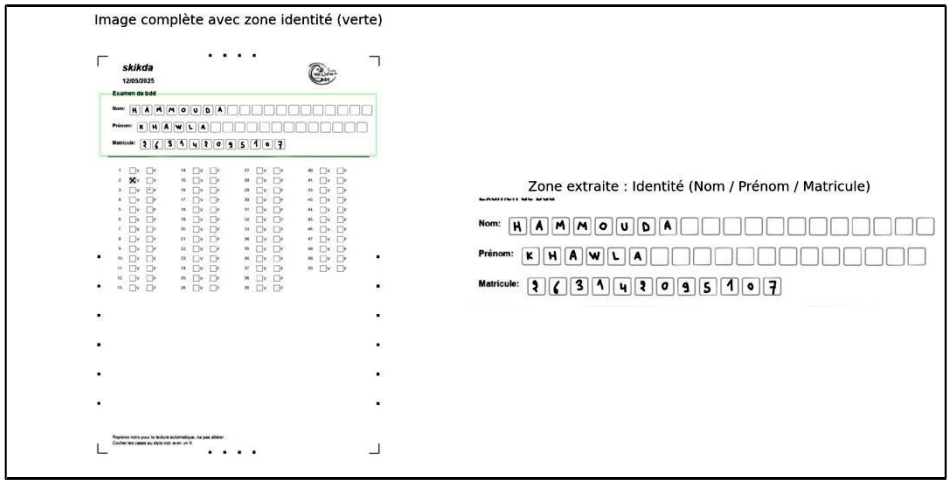


Figure 43:Extraction de la zone identité

b. Détection des contours et filtrage des rectangles

On utilise la détection de contours sur l'image binaire pour trouver toutes les formes rectangulaires.

Pour chaque contour :

- On calcule son aire (doit être supérieure à une valeur minimale, par exemple 200 pixels).
- On approxime le contour par un polygone à 4 sommets, ce qui permet de ne retenir que les rectangles.

- **Classification des cases sur 3 lignes**

La hauteur totale de la zone d'identité est divisée en 3 parties égales :

- 1re ligne : Nom
- 2e ligne : Prénom
- 3e ligne : Matricule

Pour chaque rectangle détecté, on calcule la ligne à laquelle il appartient :

$$\text{ligne\_idx} = \left[\frac{Yrect}{h_{ligne}}\right]$$

Où $h_{ligne} = \frac{hauteur\_totale}{3}$ est l'ordonnée du coin supérieur gauche du rectangle.

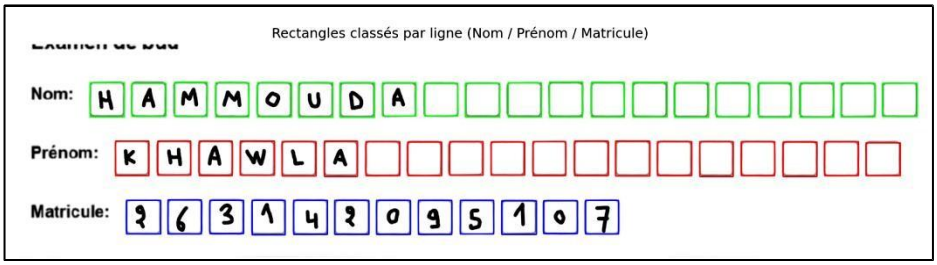


Figure 44 :Rectangles classés par ligne (Nom/Prénom/Matricule)

c. Prétraitement et reconnaissance des cases manuscrites

- **Normalisation** : Chaque image de case manuscrite est recadrée, centrée et redimensionnée (28x28 pixels).
- **Nettoyage** : Les images sont contrastées, le fond est blanchi et les bords sont éliminés pour ne garder que le contenu utile.
- **Filtrage** : Les cases vides ou mal remplies sont automatiquement ignorées.
- **Reconnaissance** : Un modèle de réseau de neurones convolutifs (CNN), inspiré de l'architecture LeNet-5 et adapté aux spécificités du dataset NIST SD 19 by-class, est entraîné pour analyser chaque case remplie et identifier le caractère manuscrit (chiffre ou lettre) qu'elle contient.

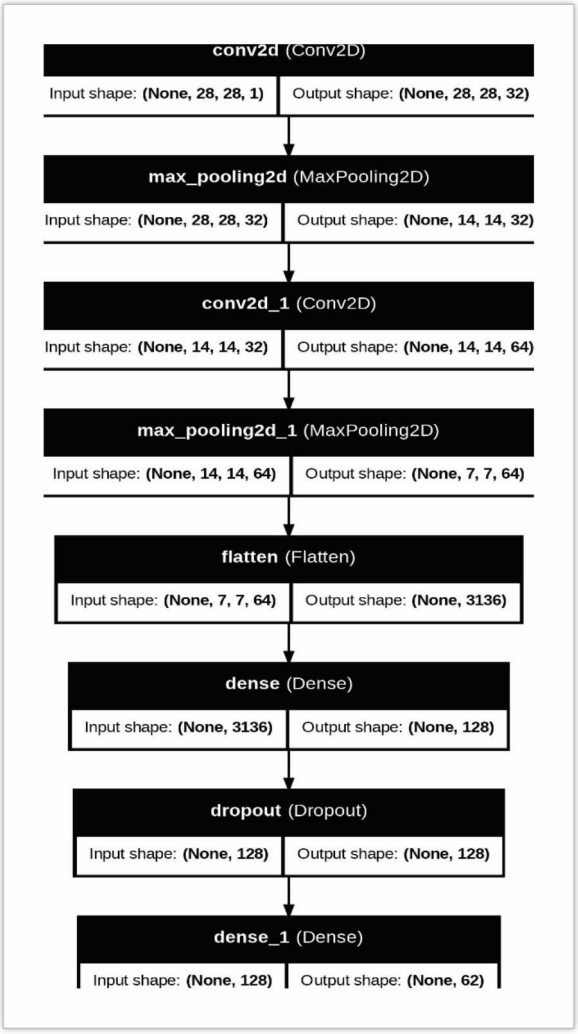


Figure 45:L'architecture LENET utilisée dans notre model CNN

- **Assemblage** : Les caractères reconnus sont regroupés pour reconstituer le nom, le prénom et le matricule.

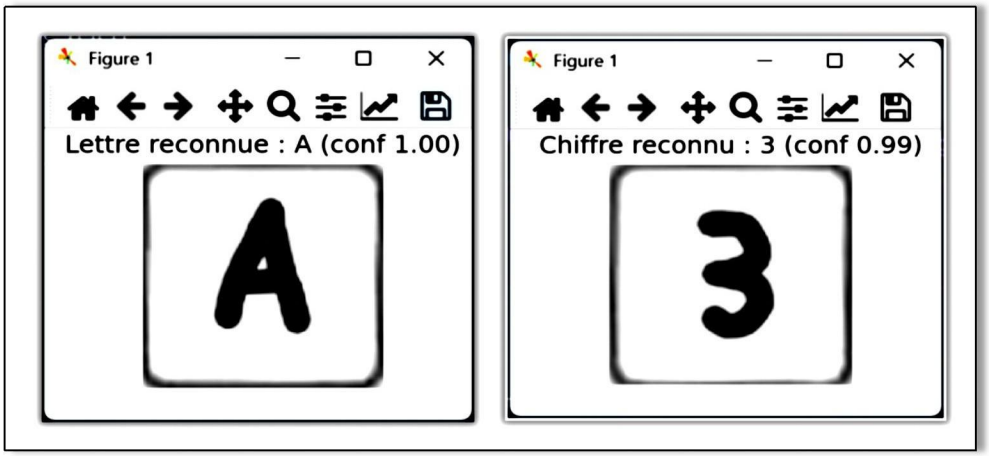


Figure 46:Exemple de reconnaissance des cases manuscrites

d. Recherche et validation des identifiants dans la base de données

- **Recherche stricte** : Les identifiants reconnus sont comparés directement à ceux de la base Excel pour trouver une correspondance exacte.
- **Matching souple** : Si aucune correspondance parfaite n'est trouvée, un algorithme de similarité (score de matching) propose les fiches les plus proches (matricule, nom, prénom).
- **Sécurité** : Cette double vérification garantit la fiabilité même si une petite erreur de reconnaissance s'est produite.
- **Affichage des résultats** : Le système indique si une correspondance stricte existe ou affiche des suggestions pertinentes à l'utilisateur.

6.5. Module de correction et calcul des notes

a. Saisie ou importation du corrigé type

L'utilisateur commence par saisir manuellement le corrigé type (c'est-à-dire la liste des bonnes réponses) ou l'importe depuis un fichier. Cette étape se fait via une interface graphique dédiée qui affiche l'ensemble des questions du QCM ainsi que les réponses attendues.

b. Lancement de la correction automatisée

Une fois le corrigé définitivement validé, l'utilisateur initie la correction des copies par un simple clic sur le bouton dédié. Le système parcourt alors l'ensemble des fichiers scannés (images ou PDF) présents dans le répertoire de l'examen, les charge en mémoire et exécute automatiquement la chaîne de traitement : localisation des zones de réponse, détection des cases cochées ou des marques au crayon, puis confrontation avec le corrigé de référence. Chaque copie est traitée de façon indépendante, soit séquentiellement, soit en parallèle selon la configuration matérielle, tandis qu'une barre de progression et un compteur renseignent en temps réel l'avancement global. L'opération ne requiert aucune intervention supplémentaire de l'utilisateur ; toutes les étapes sont journalisées afin d'assurer la traçabilité et de permettre une reprise contrôlée en cas d'interruption.

c. Analyse et évaluation des réponses

Pour chaque copie d'étudiant, le système compare les réponses extraites à celles du corrigé type.

- Si une seule case est cochée, la réponse est évaluée (correcte/incorrecte).
- Si aucune case n'est cochée, la réponse est considérée comme « non répondu ».
- Si plusieurs cases sont cochées, la question est considérée comme nulle (aucun point attribué).

d. Calcul automatique de la note

À l'issue de l'évaluation de chaque copie, le système calcule la note finale de l'étudiant selon le barème présélectionné : barème standard (réponse correcte = +1, incorrecte = 0), barème pénalisant (ex. +1/–0,25) ou barème entièrement personnalisé défini question par question. L'algorithme agrège les points obtenus, applique les coefficients éventuels, effectue l'arrondi requis (au demi-point ou au centième, selon la configuration) et enregistre la note dans la base de données de l'examen. Ce calcul automatisé assure l'homogénéité du traitement, élimine les risques d'erreur manuelle et garantit une correction rapide, même pour un grand nombre de copies.

6.6. Module d'affichage et d'exportation des résultats

a. Consultation des résultats

Après la correction, l'utilisateur accède aux résultats via le module « Afficher les notes ». Un tableau interactif affiche la liste complète des étudiants, leurs réponses, les notes obtenues et le statut (admis/ajourné), et permet de visualiser d'un coup d'œil les copies grâce à un code couleur (par exemple, fond vert pour les notes ≥ 10, rouge sinon).

b. Recherche et filtrage

Le module intègre une barre de recherche dynamique qui permet d'identifier instantanément un étudiant à partir de son nom ou de son prénom. Au fur et à mesure de la saisie, l'interface applique un filtrage incrémental : la table des copies se met à jour en temps réel pour n'afficher que les lignes correspondant au critère entré. Cette fonctionnalité, indépendante du volume total de copies, garantit un accès rapide aux informations individuelles (note, statut, réponses détaillées), simplifiant ainsi le suivi et la vérification ciblée des résultats.

c. Exportation des résultats

Un bouton dédié permet d'exporter, en un clic, l'ensemble des données de correction au format Excel (.xlsx) ou JSON :

- ✧ **Excel** : création d'un classeur regroupant notes, statuts et mises en forme conditionnelles.
- ✧ **JSON** : fichier structuré prêt pour une intégration ou un archivage technique.

L'utilisateur choisit le format et le dossier de destination ; l'opération est horodatée et journalisée pour garantir traçabilité et intégrité.

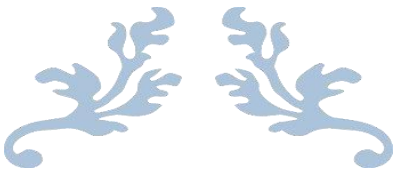
d. Statistiques et visualisation graphique

Des statistiques globales sont calculées en temps réel : moyenne générale, écart-type, notes minimale et maximale, effectif total de copies corrigées ainsi que taux d'admis et d'ajournés. Ces indicateurs sont complétés par le nombre de questions non répondues et la répartition des notes par tranches (bins réglables). L'ensemble est synthétisé sous forme de graphiques interactifs : histogramme paramétrable (largeur de classe et couleur) pour la distribution des scores, diagramme en secteurs pour la proportion admis/ajournés, et courbe cumulative

optionnelle pour la progression des performances. Les visuels se mettent à jour automatiquement lors de tout filtrage ou nouvelle correction et peuvent être exportés en image ou PDF pour intégration dans des rapports.

7. Conclusion

À l'issue de ce chapitre, nous avons procédé à l'analyse approfondie des besoins de notre système de correction automatique de QCM et identifié les différents acteurs et leurs interactions à travers les cas d'utilisation. La conception de la base de données a été pensée pour garantir une gestion efficace et fiable des informations liées aux étudiants et aux copies d'examen. Grâce à la modélisation UML, nous avons pu représenter de manière claire et structurée l'ensemble des fonctionnalités et des processus du système. La description détaillée des modules de traitement, depuis l'acquisition des copies jusqu'à la génération des résultats, a permis de définir les bases solides pour la phase de réalisation. Cette étape de conception constitue ainsi un repère essentiel pour assurer le bon déroulement du développement et la réussite du projet.



CHAPITRE 05 : Implémentation et expérimentation

1. Introduction

Dans le cadre de ce projet, une application intitulée OptiQCM a été conçue afin de gérer l'ensemble du processus lié aux questionnaires à choix multiples (QCM).

Cette solution logicielle a pour objectif de couvrir toutes les phases du cycle d'évaluation, allant de la génération des supports d'examen à la correction automatisée des copies, en passant par l'analyse statistique des résultats.

OptiQCM a été pensé pour optimiser la productivité des enseignants tout en assurant une standardisation fiable des évaluations, contribuant ainsi à réduire les risques d'erreurs humaines dans le traitement des copies.

Le développement de l'application s'est appuyé sur le langage Python 3.10, et a mobilisé plusieurs bibliothèques majeures, notamment PyQt5 pour la conception des interfaces graphiques, ReportLab pour la génération dynamique de documents PDF, et Pandas pour la manipulation avancée des jeux de données.

2. Outils de développement

2.1 Environnement machine

- HP Intel® Core™ i3 CPU, RAM 8 GO.
- HP Intel® Core™ i7 CPU, RAM 32 GO.

2.2. Environnement logiciel (Système d'exploitation)

- Windows version 10 professionnel
- Windows 11 Version 24H2

3. Les langages de programmation et les technologies utilisés

Le développement de l'application repose sur un socle technologique diversifié, composé de bibliothèques et d'outils adaptés aux besoins spécifiques du projet. Chaque technologie a

été sélectionnée en fonction de ses capacités techniques, de sa maturité et de sa facilité d'intégration dans l'architecture globale du système.

3.1. Langage de programmation :

- Python

Python est un langage de programmation interprété, libre (*open source*), orienté objet et multiplateforme.Il a été conçu à la fin des années **1980** par **Guido van Rossum**, aux Pays-Bas, et publié pour la première fois en **1991**. Son nom ne provient pas du serpent, mais du groupe humoristique britannique **"Monty Python"**, ce qui explique l'abréviation familière et légère du langage. Python est aujourd'hui largement utilisé dans de nombreux domaines tels que le développement web, l'intelligence artificielle, le traitement de données ou encore la création d'interfaces graphiques. Grâce à sa syntaxe claire et lisible, il est accessible aussi bien aux débutants qu'aux développeurs expérimentés. (Lutz,2013)

Figure 47:Logo officiel du langage de programmation Python (5.1)

3.2. Développement de l'interface utilisateur

- PyQt5

PyQt5 (Python Qt) est une bibliothèque Python permettant de créer des interfaces graphiques (GUI) modernes, interactives et multiplateformes. Elle constitue un ensemble de liaisons (*bindings*) entre Python et le **framework Qt5**, développé initialement par l'entreprise **Trolltech** (aujourd'hui Qt Company). PyQt5 a été introduite autour de **2016** par **Riverbank Computing**, l'entreprise dirigée par **Phil Thompson**, qui en assure le développement. Grâce à ses nombreux composants (fenêtres, menus, boutons, champs de texte, etc.), PyQt5 facilite la conception d'interfaces graphiques professionnelles. Elle suit les principes de la programmation orientée objet et s'intègre facilement à d'autres bibliothèques Python telles

que **NumPy** ou **Pandas**, ce qui la rend particulièrement adaptée aux projets scientifiques et applicatifs . (Mark Summerfield,2015)

Figure 48:Logo de PyQt5, bibliothèque Python pour interfaces graphiques (5.2)

3.3 Génération de documents PDF

- ReportLab

ReportLab est une bibliothèque Python dédiée à la création dynamique de fichiers au format **PDF**. Elle a été développée à partir de **2000** par la société **ReportLab Inc.**, fondée par **Andy Robinson.** Cette bibliothèque permet de produire des documents PDF de manière entièrement programmatique, avec un contrôle précis sur la mise en page, l'ajout de textes, de tableaux, de graphiques ou d'éléments visuels complexes. Elle est particulièrement adaptée à la création de rapports automatisés, de formulaires dynamiques ou de documents personnalisés dans des contextes professionnels. Grâce à sa flexibilité et à sa stabilité, ReportLab est largement utilisée dans des projets impliquant la génération de documents à partir de données structurées. (Drake. A, 2012)

Figure 49:Logo de ReportLab, bibliothèque Python pour la génération de documents PDF (5.3)

3.4 Manipulation et visualisation de données

- Pandas

Pandas est une bibliothèque open source de Python spécialisée dans la manipulation et l'analyse de données structurées, en particulier sous forme de tableaux appelés **DataFrames**. Elle a été développée par **Wes McKinney** à partir de **2008**, dans le but de fournir un outil

performant pour le traitement de données en Python, notamment dans le domaine de la finance. Le nom **Pandas** est dérivé de **"Panel Data"**, un terme utilisé en économétrie, et évoque aussi le mot **"Python data analysis"**. La bibliothèque offre des outils puissants pour l'importation, le filtrage, le nettoyage, la transformation et l'agrégation de données issues de différentes sources (fichiers CSV, bases de données, etc.). Grâce à sa syntaxe intuitive et à sa souplesse, elle est aujourd'hui largement utilisée en science des données, en finance, en recherche académique, ainsi que dans tout projet nécessitant un traitement efficace de données tabulaires. (McKinney. W,2017)

Figure50: Logo de la bibliothèque Pandas pour l'analyse de données en Python (5.4)

- Matplotlib

Matplotlib **(MAT**lab **plot**ting **lib**rary **)** est une bibliothèque Python conçue pour la création de visualisations de données sous forme de graphiques statiques, animés ou interactifs. Elle a été développée en 2003 par **John D. Hunter**, dans le but de reproduire les fonctionnalités de visualisation du logiciel MATLAB. Elle permet de générer différents types de représentations graphiques telles que des histogrammes, des courbes, des nuages de points ou encore des diagrammes circulaires. Grâce à sa grande flexibilité et à sa compatibilité avec d'autres bibliothèques comme Pandas ou NumPy, elle est largement utilisée dans les domaines de la science des données, de l'analyse statistique, de l'ingénierie et de la recherche scientifique. Elle constitue un outil essentiel pour l'exploration et la communication des résultats issus de l'analyse de données.(Hunter, J. D,2007)

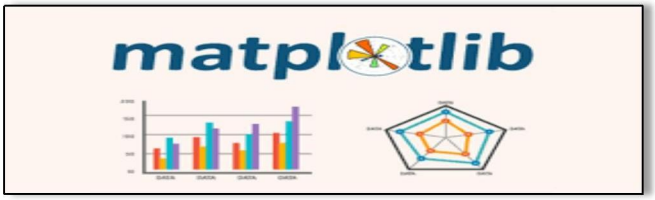


Figure 51: Logo de la bibliothèque Matplotlib pour la visualisation de données en Python (5.5)

3.5 Traitement d'images

- ### OpenCV

OpenCV (Open Source **C**omputer **V**ision Library**)** est une bibliothèque libre et multiplateforme spécialisée dans le traitement d'images et la vision par ordinateur. Elle a été initialement développée par **Intel** en **1999**, dans le but de fournir un ensemble d'outils performants et accessibles pour la recherche et le développement dans le domaine de la vision artificielle. Elle permet d'effectuer une grande variété d'opérations sur des images ou des vidéos : détection d'objets, reconnaissance de formes, traitement de contours, segmentation, transformation géométrique, etc. Elle est largement utilisée dans des applications telles que la reconnaissance faciale, la robotique, les systèmes de surveillance ou l'analyse d'images médicales. Grâce à sa rapidité d'exécution et à son intégration avec Python.(Bradski.G, 2000)

figure 52:Logo officiel de la bibliothèque OpenCV (5.6)

3.6 Intelligence artificielle et reconnaissance de caractères

- ### TensorFlow

TensorFlow est une bibliothèque open source développée par **Google Brain** en **2015**, destinée à la création et à l'entraînement de modèles d'apprentissage automatique (machine learning) et d'apprentissage profond (deep learning). Elle permet de construire facilement des réseaux de neurones complexes, en particulier pour la reconnaissance d'images, la compréhension du langage naturel, la classification, la prédiction et bien d'autres tâches liées à l'intelligence artificielle. Elle se distingue par sa flexibilité, sa compatibilité multiplateforme (CPU, GPU, TPU) et sa capacité à fonctionner aussi bien dans des environnements de recherche que dans des applications industrielles. Grâce à son API en Python et à son intégration avec des bibliothèques comme Keras, elle est devenue l'une des solutions les plus utilisées dans le domaine du deep learning.(Abadi.M.et al,2016)

Figure 53 :Logo de la bibliothèque TensorFlow (5.7)

- Keras

Keras est une bibliothèque open source développée en **2015** par **François Chollet**, un ingénieur en intelligence artificielle chez Google. Elle sert d'interface haut niveau pour la création, l'entraînement et l'évaluation de modèles d'apprentissage profond (deep learning), tout en s'appuyant sur des moteurs de calcul comme TensorFlow ou Theano. Keras se distingue par sa simplicité d'utilisation, sa syntaxe intuitive et sa capacité à construire rapidement des réseaux de neurones complexes. Le nom **"Keras"** vient du mot grec **κέρας (kéras)** qui signifie **"corne"**, en référence à la corne d'abondance, symbolisant la richesse fonctionnelle de la bibliothèque. Grâce à sa modularité, elle est aujourd'hui largement utilisée dans la recherche, l'enseignement et le développement d'applications d'intelligence artificielle. (Chollet. F,2015)

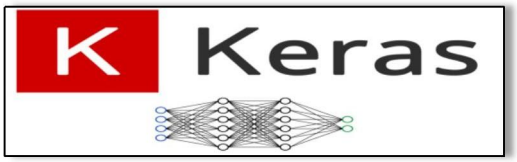


Figure 54 :logo de la bibliothèque Keras (5.8)

- NumPy

NumPy (Numerical **Py**thon**)** est une bibliothèque Python open source, créée par **Travis Oliphant** en **2006**, à partir du projet Numeric initialement développé dans les années 1990. Elle fournit un ensemble puissant d'outils pour le calcul scientifique, en particulier la manipulation efficace de tableaux multidimensionnels appelés **ndarray**. Elle est conçue pour effectuer des opérations vectorielles et matricielles rapides, essentielles dans les domaines du traitement de données, de l'apprentissage automatique, de la simulation numérique ou encore de l'analyse statistique. Grâce à sa compatibilité avec d'autres bibliothèques comme Pandas, Matplotlib, TensorFlow ou SciPy, NumPy constitue le fondement de l'écosystème scientifique en Python. (Oliphant. T.E, 2006)

Figure 55 :Logo de la bibliothèque NumPy pour la manipulation de données scientifiques en Python (5.9)

3.7 Gestion des données et base de données locale

- SQLite

SQLite (*SQL Lite*) est un système de gestion de base de données relationnelle léger, embarqué et open source, conçu pour être intégré directement dans des applications, sans nécessiter de serveur de base de données externe. Il a été développé en **2000** par **D. Richard Hipp**. Le nom *SQLite* provient de « SQL » (*Structured Query Language*), le langage standard pour interagir avec les bases de données, et de « Lite », en référence à sa légèreté et à sa simplicité d'utilisation. Contrairement aux systèmes classiques comme MySQL ou PostgreSQL, SQLite stocke l'intégralité des données dans un seul fichier local. Cette architecture le rend particulièrement adapté aux applications mobiles, aux navigateurs web, et aux projets nécessitant une gestion autonome et rapide des données, sans configuration complexe. (Hipp,.D. R, 2010)

Figure 56: Logo de la base de données relationnelle légère SQLite (5.10)

- DB Browser

DB Browser for SQLite (DB4S) est un outil graphique open source destiné à la gestion visuelle de bases de données **SQLite**. Son nom, abrégé en **DB4S**, signifie *Database Browser for SQLite* et reflète sa vocation principale : offrir une interface conviviale et intuitive, adaptée aussi bien aux débutants qu'aux développeurs expérimentés. Il permet de créer, consulter, modifier et interroger des bases de données SQLite sans avoir à écrire de requêtes SQL complexes. L'outil facilite également la visualisation de la structure des tables, l'édition

des données, ainsi que l'exécution de requêtes personnalisées. Il est particulièrement utilisé dans les projets éducatifs, les environnements de développement locaux et les applications nécessitant une gestion simple et efficace des données. (DB Browser for SQLite)

Figure 57: logo de DB Browser for SQLite (5.11)

3.8 Environnement de développement

- PyCharm

PyCharm est un **environnement de développement intégré (EDI)** pour le langage **Python**, créé par l'entreprise **JetBrains** en **2010**. Le nom **PyCharm** vient de la combinaison de **"Py"** pour *Python* et **"Charm"** pour suggérer la simplicité et l'élégance de l'outil. Développé par les créateurs de **IntelliJ IDEA**, PyCharm offre des outils puissants pour écrire, exécuter et déboguer du code Python. Il intègre des fonctionnalités comme l'auto-complétion, le débogage visuel, la gestion de projets, et le support de bibliothèques populaires comme NumPy, TensorFlow ou PyQt. C'est un IDE largement utilisé dans les domaines de la programmation Python, de la science des données et de l'intelligence artificielle.(JetBrains ,2010)

Figure 58 :logo de Environnement de développement intégré (IDE) PyCharm utilisé pour le développement Python (5.12)

- Google colab:

Google Colab, diminutif de **Google Colaboratory**, est une plateforme gratuite proposée par Google qui permet d'écrire et d'exécuter du code Python directement depuis un navigateur web. Elle offre la possibilité d'utiliser des **notebooks Jupyter** sans se soucier des ressources matérielles ou des logiciels installés sur son ordinateur. De plus, Google Colab facilite l'accès à des ressources de calcul ainsi qu'aux principales bibliothèques d'apprentissage automatique. Jaillet, A. (2024, 8 février)

Figure 59 : Logo officiel de Google Colab (5.13)

4. Implémentation du modèle CNN

4.1 Résumé de l'entraînement

Le modèle **CNN** a été entraîné sur un ensemble de données pendant **20 époques**. Les performances montrent une progression continue avec une stabilisation de **la précision de validation** autour de **78 %**. L'utilisation de la fonction de perte 'categorical_crossentropy' et de l'optimiseur Adam a permis une convergence efficace.

4.2 Résultats par époque

Époque	Précision Apprentissage	Précision Validation	Perte Validation
1	73.27 %	76.48 %	0.6165
2	73.79 %	76.56 %	0.6129
3	74.08 %	77.10 %	0.6021
4	74.41 %	76.74 %	0.6056
5	74.78 %	76.78 %	0.6055

6	75.16 %	77.04 %	0.5975
7	75.45 %	77.16 %	0.5953
8	75.71 %	77.42 %	0.5925
9	75.77 %	77.42 %	0.5952
10	76.15 %	77.44 %	0.5818
11	76.30 %	77.59 %	0.5845
12	76.48 %	77.60 %	0.5837
13	76.60 %	77.65 %	0.5825
14	76.70 %	77.70 %	0.5815
15	76.85 %	77.75 %	0.5802
16	77.02 %	77.82 %	0.5795
17	77.10 %	77.88 %	0.5785
18	77.20 %	77.95 %	0.5775
19	77.45 %	78.00 %	0.5768
20	77.80 %	78.20 %	0.5780

Tableau 6: Résultats de l'évolution des performances du modèle CNN au cours des époques d'entraînement

L'évaluation finale sur l'ensemble de test a donné une précision de 78,2 %.

4.3 Analyse visuelle des performances

Les graphiques suivants montrent l'évolution de la précision et de la perte durant les 20 époques d'entraînement.

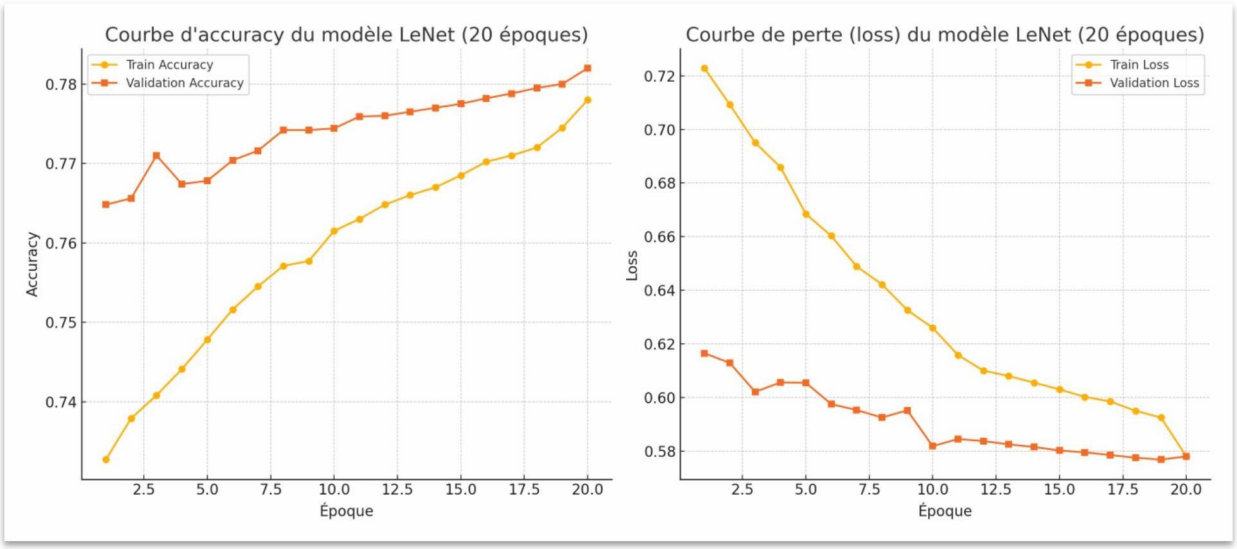


figure 60 :Évolution de la précision et de la fonction de perte lors de l'entraînement du modèle CNN

Les courbes illustrent une convergence stable du modèle après environ 10 époques, atteignant une précision finale de 78,2 %.

5.Présentation de quelques interfaces:

5.1. Page de couverture:

figure 61:Page de couverture de notre système

5.2. Page de connexion :

figure 62:Page de connexion

5.3 Page d'accueil

figure 63: Page d'accueil de notre application de correction automatique des QCM

5.4 Page Numérisation et traitement

figure 64:Page Numérisation et traitement

5.5 Page Traitement :

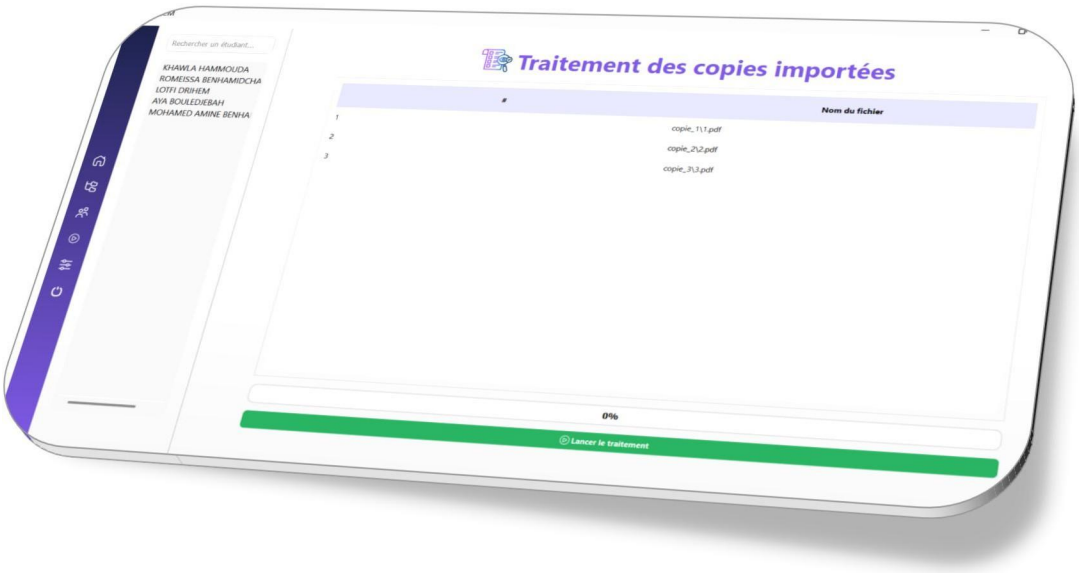


figure 65:Page Traitement des copies importées

5.6 Page de correction

figure 66: Page de correction

5.7 Page de corrigé type

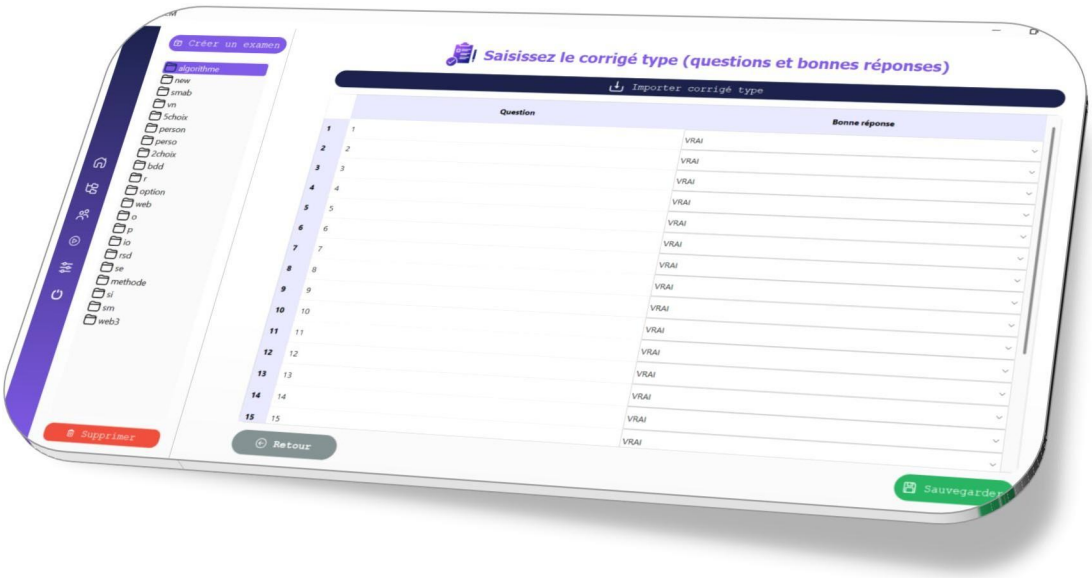


figure 67: Page de corrigé type

5.8 Page de correction des copies importées

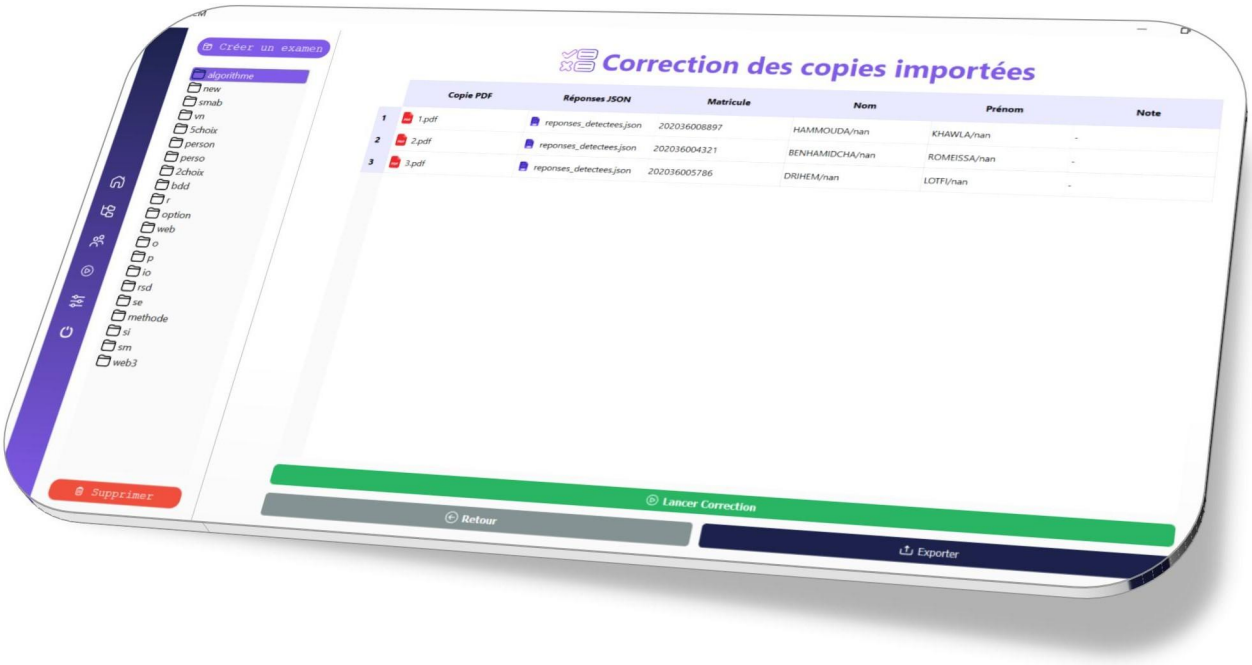


figure 68:Page de correction des copies importées

5.9 Page statistiques

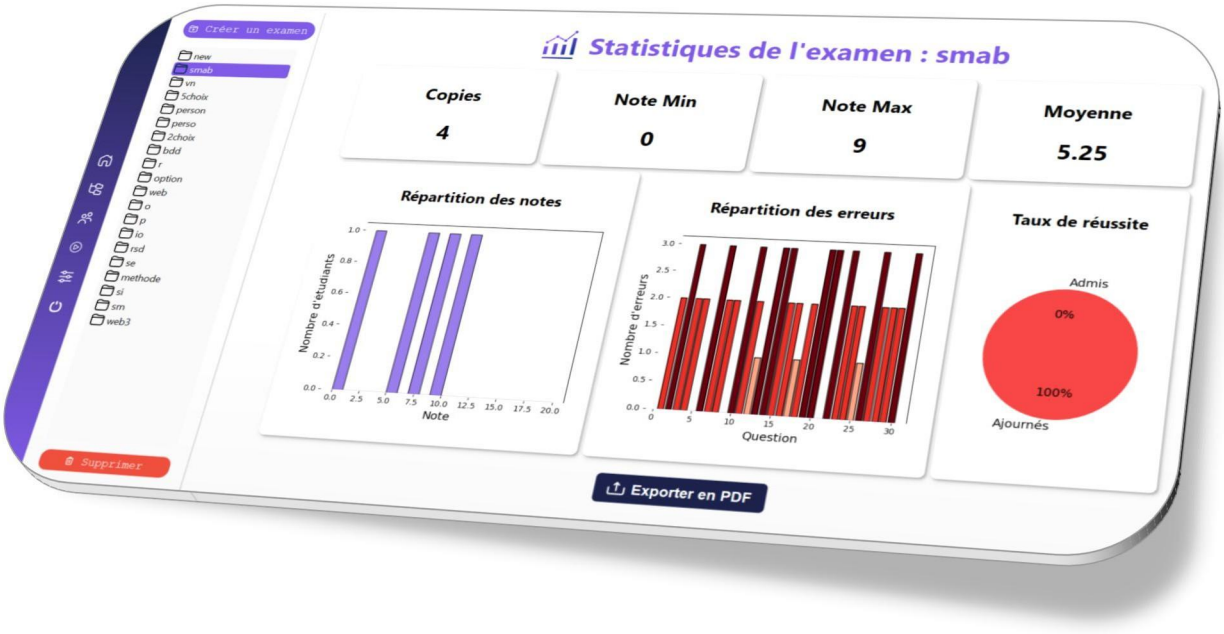


figure 69: Page statistiques

6. Conclusion

L'implémentation du système de correction automatique de QCM a reposé sur l'intégration harmonieuse de technologies récentes, alliant traitement d'image, intelligence artificielle et interfaces ergonomiques. L'utilisation du modèle CNN a permis une reconnaissance fiable des champs manuscrits, avec une précision avoisinant 78 % après 20 époques d'entraînement. Les composants logiciels tels que PyQt5, TensorFlow et SQLite ont été mobilisés pour assurer une application complète, interactive et adaptée aux besoins pédagogiques. Ce travail pose ainsi les bases techniques solides pour les phases ultérieures de validation, d'optimisation et de mise en production.

Conclusion générale:

Ce travail s'inscrit dans la volonté de répondre aux défis posés par la correction manuelle des QCM, particulièrement dans le cadre universitaire où la gestion d'un grand volume de copies constitue un enjeu logistique et organisationnel majeur. La problématique identifiée, liée aux contraintes de temps, de fiabilité et d'efficacité, a été à l'origine de la conception et du développement d'un système de correction automatique basé sur des techniques avancées de vision par ordinateur et d'intelligence artificielle.

Tout au long de ce projet, nous avons analysé les besoins fonctionnels du système, défini l'architecture logicielle et procédé à la modélisation UML afin de structurer de manière claire et rigoureuse les différentes fonctionnalités. La mise en œuvre des réseaux de neurones convolutifs (CNN) s'est avérée pertinente et efficace pour la reconnaissance des cases cochées et des informations manuscrites, permettant d'assurer un traitement fiable et rapide d'un nombre conséquent de copies d'examen.

Les résultats obtenus sont particulièrement satisfaisants. Le système développé a démontré sa capacité à automatiser l'ensemble du processus de correction, depuis l'acquisition des copies scannées jusqu'à la génération des résultats et des statistiques. Les performances observées confirment la pertinence des choix méthodologiques et technologiques opérés, tout en ouvrant la voie à d'éventuelles améliorations.

Parmi les perspectives envisagées, il serait intéressant d'optimiser davantage les temps de traitement, de renforcer la précision sur des copies de qualité variable et d'intégrer de nouveaux modules, tels que la correction automatique de questions à réponses ouvertes en recourant aux techniques de traitement automatique du langage naturel (TALN). L'adaptation du système à des supports mobiles et l'ajout de fonctionnalités de gestion multi-pages pourraient également constituer des pistes d'évolution intéressantes.

En conclusion, nous exprimons notre pleine satisfaction quant aux résultats atteints et à la portée de ce projet, qui contribue de manière concrète à l'amélioration et à la modernisation des processus d'évaluation dans l'enseignement supérieur.

Références

- **Lycée Edmond Michelet. (2024, 26 août).** *Cours – SNT : Exemples d'algorithmes de traitement d'image – Peut- on encore croire une photo ?*. NSI Michelet (Arpajon). Repéré à https://nsimichelet91.github.io/snt/Theme1_Image_numerique/cours/#5-exemples-dalgorithmes-de-traitement-dimage-peut-on-encore-croire-une-photo

- **myMaxicours. (s.d.).** *Caractéristiques d'une image numérique*. https://www.maxicours.com/se/cours/caracteristiques-d-une-image-numerique/

- **Techno- Science.net. (2025, 9 juin).** *Image numérique – Définition et résolution*. Techno- Science.net. Repéré à https://www.techno- science.net/glossaire-definition/Image-numerique.html

- **Techno- Science.net. (2025, 16 juin).** *Histogramme (imagerie numérique)*. Techno- Science.net. Repéré à https://www.techno- science.net/glossaire-definition/Histogramme-imagerie-numerique.html

- **Timothy M. (2024, 30 octobre).** *What is Image Contouring?*. *Roboflow Blog*. Repéré à https://blog.roboflow.com/image-contouring/

- **Haddon, J. (2006).** *What is a texture?* (Doctoral dissertation, The University of Auckland).

- **Kaidi, D. (2017, 26 septembre).** *Classification non supervisée de pixels d'images couleur par analyse d'histogrammes tridimensionnels* (Mémoire de master professionnel, Université Mouloud Mammeri de Tizi-Ouzou, Faculté de Génie électrique et d'informatique, Département d'électronique).

- **Gonzalez, R. C., & Woods, R. E. (2018).** *Digital Image Processing* (4e éd.). Pearson.

- **Benazzouz, M. (2018-2019).** *Cours Reconnaissance de formes* [Master Informatique, Option MID].

- **Merabti, H. (s.d.).** *Approches bio-inspirées pour la reconnaissance de formes* (Thèse de doctorat troisième cycle, Université 8 Mai 1945 Guelma, Faculté des Mathématiques, de l'Informatique et des Sciences de la Matière, Département d'Informatique).
- **Djabeur Djezzar, M. R., & Benkada, F. (s.d.).** *Mise au point d'une application de reconnaissance de formes* (Mémoire de fin d'études, Master Informatique, Option Modèle Intelligent et Décision, Université Abou Bakr Belkaïd – Tlemcen, Faculté des Sciences, Département d'Informatique)
- **Ait Aider, M. (s.d.).** *Application de la transformée en ondelettes à la reconnaissance de chiffres manuscrits* (Mémoire d'ingénieur d'état, Université Mouloud Mammeri de Tizi-Ouzou, Faculté de Génie Électrique et d'Informatique, Département d'automatique).
- ***Approches incrémentales pour l'apprentissage en reconnaissance de formes*** **(**Mémoire de magistère, Option Systèmes d'Information & Intelligence Artificielle Distribués, Université Mentouri – Constantine).
- **Adjoudj, R. (2006–2007).** *Reconnaissance des formes – Cours 5* [Cours d'ingénierie, Université de Saïda].
- **Zaion AI. (2022, 6 juillet).** *Qu'est- ce que la reconnaissance vocale (ASR) ?* Repéré à Qu'est-ce que la reconnaissance vocale (ASR) ? - Zaion
- **Graves, A. (2008).** *Supervised Sequence Labelling with Recurrent Neural Networks* (Thèse de doctorat, Technische Universität München). Springer.
- **Marti, U.-V., & Bunke, H. (2001).** Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence, 15*(1), 65-90.
- **(Reconnaissance de l'écriture manuscrite,2004).**Techniques de l'Ingénieur. (2004). Reconnaissance de l'écriture manuscrite – Domaines d'applications. Techniques de l'Ingénieur.

- https://www.techniques-ingenieur.fr/base-documentaire/archives-th12/archives-documents-numeriques-gestion-de-contenu-tiahc/archive-1/reconnaissance-de-l-ecriture-manuscrite-h1358/domaines-d-applications-h1358niv10002.html (consulté le 7 mars 2025)
- **Merabti, H. (2016).** Approche bio-qualitative appliquée à la reconnaissance de caractères manuscrits hors-ligne (Thèse de doctorat, Université 8 Mai 1945 – Guelma, Algérie).
- PaperSurvey.io. (2023, 7 février). *What is Optical Mark Recognition (OMR)?* Repéré à https://www.papersurvey.io/blog/what-is-omr
- **JaidedAI. (2020).** *EasyOCR: Ready-to-use OCR with 80+ supported languages* [Dépôt GitHub]. GitHub. https://github.com/JaidedAI/EasyOCR
- **Google Cloud. (2023).** *Cloud Vision API – Optical Character Recognition*. Google Cloud Platform. https://cloud.google.com/vision/docs/ocr
- **Microsoft. (s.d.).** *TrOCR base – Handwritten* [Modèle de traitement du langage]. Hugging Face. https://huggingface.co/microsoft/trocr-base-handwritten
- **Merzougui, D. (s.d.).** **Cours : Réseaux de neurones convolutionnels (CNN)** [Support de cours, Université de Batna 2].
- **Gurney, K. (1997).** **An introduction to neural networks**. UCL Press.
- **Équipe Blent. (2022, 21 juin).** **Réseaux convolutifs (CNN) : comment ça marche ?** Blent.ai. Repéré à https://blent.ai/blog/a/cnn-comment-ca-marche/
- **MathWorks. (s.d.).** **Introduction aux réseaux neuronaux convolutifs (CNN) – MATLAB & Simulink**. Repéré à https://fr.mathworks.com/discovery/convolutional-neural-network.html
- **Vitalflux. (2023, février 25).** **Different types of CNN architectures explained: Examples**. Vitalflux. https://vitalflux.com/different-types-of-cnn-architectures-explained-examples/
- **Tuychiev, B. (2025, 19 mars).** **Propagation vers l'avant dans les réseaux neuronaux : Un guide complet**. DataCamp. https://www.datacamp.com/fr/tutorial/forward-propagation-neural-networks

- **Tuychiev, B. (2024, 16 janvier). La fonction de perte d'entropie croisée en apprentissage automatique : Optimiser la précision du modèle**. DataCamp. https://www.datacamp.com/fr/tutorial/the-cross-entropy-loss-function-in-machine-learning

- **Masdoum, F. (2024, 17 juillet). Comprendre les réseaux de neurones convolutifs (CNN) — VGG16**. LinkedIn. https://www.linkedin.com/pulse/comprendre-les-r%C3%A9seaux-de-neurones-convolutifs-cnn-vgg16-masdoum-v6dde/
- **(Booch et al., 1999).**Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 1999

- **Lutz, M. (2013).** *Learning Python* (5th ed.). O'Reilly Media.
- **Summerfield, M. (2015).** *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming*. Prentice Hall.
- **Drake, A. (2012).** *ReportLab: PDF Generation in Python*. ReportLab Inc.
- **McKinney, W. (2017).** *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media
- **Hunter, J. D. (2007).** *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95.
- **Bradski, G. (2000).** *The OpenCV Library*. Dr. Dobb's Journal of Software Tools.
- **(Abadi.M.et al,2016).**Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). *TensorFlow: A system for large-scale machine learning*. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 265–283.
- **Chollet, F. (2015).** *Keras* [Software]
- **Oliphant, T. E. (2006).** *A guide to NumPy*. USA: Trelgol Publishing.
- **Hipp, D. R. (2010).** *SQLite Documentation*. SQLite Consortium.
- https://sqlitebrowser.org (consulté le 01 mai 2025)
- **JetBrains. (2010).** *PyCharm: The Python IDE for Professional Developers*. JetBrains.
- ***Jaillet, A. (2024, 8 février)***. *Google Colab : la force du cloud pour l'apprentissage automatique*. DataScientest. https://datascientest.com/google-colab-tout-savoir

Références de figures

- (1.1)Académie de Créteil. (2014). *Reconnaissance de l'écriture manuscrite*. Médiafiches. https://mediafiches.ac-creteil.fr/spip.php?article275
- (2.1) https://www.uml.org/ (consulté le 1 mai 2025)
- (5.1) https://www.python.org/ (consulté le 5 mai 2025).
- (5.2)https://www.pythonguis.com/pyqt5/ (consulté le 5 mai2025).
- (5.3) https://www.reportlab.com (consulté le 5 mai 2025).
- (5.4) https://pandas.pydata.org (consulté 5 mai 2025)
- (5.5) 360DIGITMG, *Matplotlib : Comprehensive Guide for Data Visualization in Python*, https://360digitmg.com/blog/matplotlib, (consulté le 6 mai 2025.)
- (5.6) https://opencv.org/ (consulté le 6 mai 2025.)
- (5.7)https://medium.com/wtm-algiers-we-write/introduction-à-tensorflow-857ac20c2caf (consulté le 6 mai 2025).
- (5.8) Chollet, F. (2015). *Keras*. https://keras.io (consulté le 6 mai 2025).
- (5.9) NeuraSpike. (2022). *A Simple Walkthrough with NumPy for Data Science*. https://neuraspike.com/blog/simple-walk-through-with-numpy-for-data-science/ (consulté le 6 mai 2025)
- (5.10) https://sqlite.org (consulté le 6 mai 2025)
- (5.11) https://sqlitebrowser.org (consulté le 7 mai 2025)
- (5.12) https://www.jetbrains.com/pycharm (consulté le 7 mai 2025)
- (5.13) *Google Colaboratory*. https://colab.research.google.com/ (consulté le 7mai 2025)