



République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene



Faculté d'Électronique et d'Informatique Département Informatique

Mémoire de Licence

Filière : Informatique

Spécialité : Ingénierie des Systèmes d'informations et logiciels

---

## Thème :

Système de codes QR pour la gestion des doctorants et du personnel de l'université

---

**Proposé et encadré par :**

**Prof. LARABI Slimane.**

**Réalisé par :**

**BOUROUINA Rania**

**CHIBANE Ilies**

**Soutenu devant le jury  
composé de :**

- **M<sup>r</sup> Belache M. (Président)**
- **M<sup>me</sup> Derbal K. (Membre)**

Projet No : ISIL 054/21

# Remerciement

Nous remercions Dieu le tout puissant de nous avoir aidé et donné la force et la volonté pour accomplir ce travail.

Nous tenons tout d'abord à exprimer notre profonde reconnaissance à Mr. LARABI Slimane pour nous avoir proposé ce sujet, sa contribution dans l'élaboration de ce travail par son encadrement et sa disponibilité ainsi que la confiance qu'il nous a accordée.

Nous remercions les membres de notre jury pour l'honneur qu'ils nous ont accordé en acceptant d'examiner notre travail.

Et finalement nous sommes reconnaissantes envers nos familles respectives en particulier nos parents qui nous ont été d'un grand soutien moral.

# Table des matières

Introduction générale .....	8
Chapitre I. État de l'art.....	2
Introduction.....	2
1 Les codes QR .....	2
1.1 Définition et historique.....	2
1.2 La structure du code QR.....	2
1.3 Les caractéristiques du code QR .....	3
1.4 Les caractéristiques du code QR .....	4
1.5 Les variantes du code QR.....	4
1.6 Le décodage du code QR.....	5
1.7 Avantages du code QR.....	8
1.8 Inconvénients du code QR.....	8
2 Détection des codes QR .....	8
2.1 Méthode de Viola et Jones .....	8
2.2 Méthodes basées sur une approche neuronale .....	9
3 Le réseau YOLO (You Only Look Once).....	11
4 YOLOv5 .....	12
4.1 Architecture de YOLOv5 .....	12
4.2 La fonction d'activation .....	14
4.3 La fonction d'optimisation .....	14
4.4 La fonction coût .....	14
Conclusion .....	15
Chapitre II. L'implémentation de l'algorithme de détection du code QR.....	16
Introduction.....	16
1 Implémentation de l'algorithme YOLOv5 .....	16
1.1 L'environnement de l'entraînement .....	16
1.2 La préparation des données .....	17
1.3 La configuration du modèle .....	18
1.4 L'entraînement et l'évaluation du modèle.....	19
1.5 Le test du modèle .....	20
2 Expériences .....	21
2.1 Expérience 1 : YOLOv5 Basique sans augmentation.....	21
2.2 Expérience 2 : YOLOv5 Basique avec augmentation .....	23

2.3 Expérience 3 : YOLOv5 Avec du Transfer Learning.....	24
2.4 Expérience 4 : YOLOv5 Extra Large.....	25
3 Discussion .....	26
3.1 La détection du code QR.....	26
3.2 L'importance des données.....	27
3.3 Le Transfer Learning.....	27
Conclusion .....	27
Chapitre III. Conception .....	28
Introduction.....	28
1 UML.....	28
2 Le diagramme de cas d'utilisation .....	28
2.1 Définition.....	28
2.2 Le diagramme de cas d'utilisation du système .....	28
Conclusion .....	29
Chapitre IIII. Réalisation et Tests .....	30
Introduction.....	30
1 Les outils utilisés.....	30
2 Présentation des outils logiciels .....	31
3 Système du codage et décodage .....	31
4 Présentation de l'outil.....	33
4.1 Les fonctionnalités de l'application .....	33
5 Tests de réalisation.....	34
5.1 Interface d'ouverture.....	34
5.2 Génération d'un certificat de scolarité .....	34
5.3 Affichage des informations du doctorant .....	36
5.4 Implémentation du modèle dans l'interface .....	37
Conclusion .....	37
Conclusion générale .....	38
Bibliographie.....	39
Webographie .....	41
Résumé.....	43

# Liste des figures

FIGURE I.1: Code QR [Web2].....	2
FIGURE I.2: Structure d'un code QR [De Almeida].....	3
FIGURE I.3: Micro code QR [Web1] .....	5
FIGURE I.4: Code iQR [Web1] .....	5
FIGURE I.5: Code FrameQR [Web1] .....	5
FIGURE I.6: processus de décodage de code QR [De Almeida] .....	6
FIGURE I.7: QR code à décoder [Web3].....	6
FIGURE I.8: Image représentant les bits du format du code QR (en vert) [Web3].....	6
FIGURE I.9 : Les différents types de masque binaire que peut avoir un code QR [web4] .....	7
FIGURE I.10 : Découpage du code QR en blocs de 8 bits [Web3] .....	7
FIGURE I.11 : Chemin à suivre lors du décodage d'un code QR [Web31] .....	7
FIGURE I.12: Fonctionnement du modèle R-CNN [van de Sande†] .....	9
FIGURE I.13: Architecture du Fast R-CNN [Girshick] .....	9
FIGURE I.14: illustration du fonctionnement du Region Proposal Network (RPN) [Girshick and al].	10
FIGURE I.15: segmentation du Mask R-CNN [Girshick and al] .....	10
FIGURE I.16: MultiBox Detector [Anguelov and al].	11
FIGURE I.17: Fonctionnement de Yolo [Redmon and al] .....	11
FIGURE I.18 : Comparaison des performances des versions de YOLOv5[Web31].....	12
FIGURE I.19 : Aperçu de l'architecture de YOLOv5[Web24] .....	13
FIGURE II.1 : Les outils utilisés .....	16
FIGURE II.2 : Un exemple de codes QR étiqueté.....	18
FIGURE II.3 : Un exemple de codes QR étiqueté.....	18
FIGURE II.4 : L'état d'utilisation du GPU alloué .....	19
FIGURE II.5 : La formule de la fonction Intersection over Union [Do Thuan] .....	19
FIGURE II.6 : Les métriques de l'évaluation pour les 500 epochs de l'expérience 1 .....	22
FIGURE II.7 : Les métriques de l'évaluation pour les 500 epochs de l'expérience 1 avec Adam.....	23
FIGURE II.8 : (a) La détection en utilisant la fonction d'optimisation SGD (b) La détection en utilisant la fonction d'optimisation Adam.....	23
FIGURE II.9 : Les métriques de l'évaluation pour les 500 epochs de l'expérience 2.....	24
FIGURE II.10 : La détection en utilisant le dataset 1 avec YOLOv5s.....	24
FIGURE II.11 : Les métriques de l'évaluation pour les 500 epochs de l'expérience 3.....	25
FIGURE II.12 : La détection en utilisant le Transfer Learning avec YOLOv5s .....	25
FIGURE II.13: Les métriques de l'évaluation pour les 500 epochs de l'expérience 4.....	26
FIGURE II.14: La détection en utilisant le dataset 2 avec YOLOv5x .....	26
Figure III.1 : Diagramme de cas d'utilisation .....	29
FIGURE III.1: Logos des outils de développement .....	31
FIGURE III.2 : Comparaison des performances des différentes librairies. [Web14].....	32
FIGURE III.3 : Comparaison des performances des différentes librairies [Web19] .....	33

FIGURE III.4 : Comparaisons du temps d'exécution des différentes librairies [Web20] .....	33
FIGURE III.5: Interface d'ouverture.....	34
FIGURE III.6: Génération d'un certificat de scolarité .....	35
FIGURE III.7: Le Scan du code QR .....	35
FIGURE III.8: Le certificat de scolarité généré.....	36
FIGURE III.9: L'affichage des informations du doctorant .....	37

# Liste des tables

Table I.1: Les caractéristiques du code QR..... 4

Table II.1: Résumé des métriques des expériences ..... 26

Table II.2: Résumé des temps d’entraînement et de détection des expériences..... 26

# Liste des équations

Eq (1.1) [Web3] ..... 7

Eq (1.2) [Web5]..... 8

# Introduction générale

L'évolution de l'informatique et des différentes technologies permettent de faciliter les quotidiens de nombreuses personnes et notamment dans le cadre professionnel, et spécialement au niveau de l'administration.

De nombreux logiciels ont été développés afin de faciliter et d'automatiser ce domaine et l'émergence de l'intelligence artificielle pourrait être un véritable plus.

Le nouveau staff administratif de la faculté s'est fixé un objectif de moderniser et numériser les différents services. Il nous a été confié une tâche qui consiste à développer une application de bureau permettant la gestion des doctorants et du personnel de l'université en utilisant la technologie des codes QR, à laquelle nous ajouterons des méthodes de vision artificielle et d'apprentissage profond afin d'améliorer les performances et de faciliter le travail administratif au niveau des scolarités de notre université.

Le travail présenté dans ce mémoire concerne le service de post-graduation et l'application développée sera également exploitée et appliquée au service du personnel en changeant les données à traiter. Concrètement, le travail réalisé et présenté dans ce mémoire est scindé en trois chapitres :

**Le chapitre 1** sera dédié à la présentation des différentes technologies utilisées telle que la vision par ordinateur, l'apprentissage profond pour la détection d'objet et les codes QR.

**Le chapitre 2** sera dédié au réseau CNN utilisé, les données utilisées ainsi que les différentes expériences réalisées seront présentées.

**Le chapitre 3** présente le processus conceptuel de la gestion des doctorants et du personnel de l'université.

**Dans le dernier chapitre**, on présentera la réalisation de notre application de bureau, ainsi que l'implémentation de notre réseau de neurone.



# Chapitre I. État de l'art

## Introduction

Le code QR a fait l'objet de plusieurs études en intelligence artificielle plus précisément celui de la vision par ordinateur ainsi que de l'apprentissage profond. Par conséquent, de nombreux travaux et recherches ont été faits et il est important de les retracer afin de mieux appréhender notre travail.

## 1 Les codes QR

### 1.1 Définition et historique

Le terme « QR Code » signifie « Quick Response Code », cela sous-entend que le décodage est rapide. Cette technologie est apparue à la suite des limitations des caractéristiques technologiques du code à barres linéaire unidimensionnel, également appelé code à barres conventionnel ou classique. Le code QR est une matrice bidimensionnelle composée de plusieurs modules colorés, qui codent des données. Différentes versions sont disponibles permettant des capacités de données adaptées [Aktaş], voir un exemple de code QR dans la figure I.1.



FIGURE I.1: Code QR [Web2]

En 1994, la société japonaise Denso-Wave a inventé la technologie du code QR pour assurer un suivi rapide des pièces automobiles de la marque Toyota. Cette société a décidé de publier sa création en licence libre et elle s'est vite propagée au Japon, puis dans le monde entier [Petrova and al].

### 1.2 La structure du code QR

Le code QR est basé sur une matrice de structure cellulaire formatée dans un carré. Il est composé de petits carrés noirs et blancs qui s'appellent modules et chacun représente une valeur binaire (0 : blanc, 1 : noir). Il comporte également plusieurs zones pour une lecture facile [Winter] Voir figure I.2.

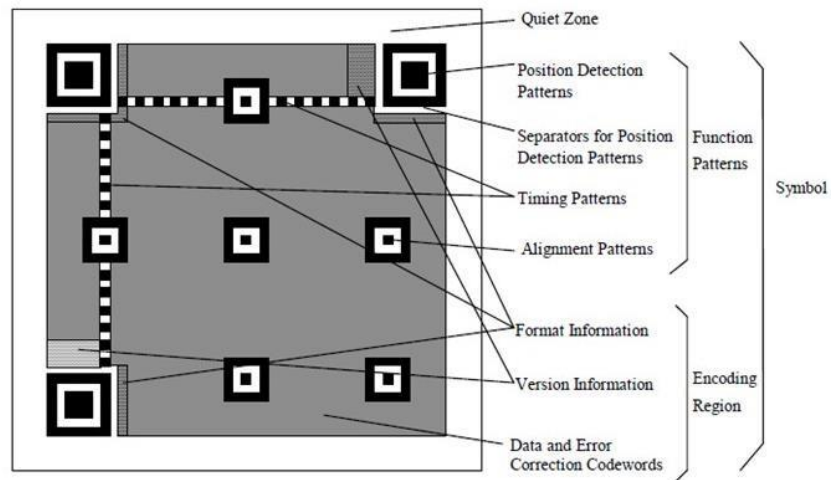


FIGURE I.2: Structure d'un code QR [De Almeida]

**La zone silencieuse (Marge).** La marge est un espace de marge qui mesure 4 modules de largeur. Cette marge est fortement recommandée car elle facilite la lecture en délimitant de code QR [Soon].

**Les patterns (motifs) de détection de position.** Les trois motifs de détection de position nous permettent de détecter le code QR ainsi qu'avoir des informations sur la taille et l'angle du symbole. Ils sont omnidirectionnels et sont toujours placés en haut à gauche, en haut à droite et en bas à gauche [Sharma].

**Les séparateurs.** Les séparateurs permettent de séparer les motifs de détection de position du reste des modules [De Almeida].

**Les motifs de synchronisation (timing Patterns).** Les motifs de synchronisation densifient le code QR par l'alternance des carrés noirs et blancs, et sont fixés dans les deux sens, horizontal et vertical. Ils sont utilisés pour corriger la coordonnée centrale du module d'information lorsque le symbole est déformé ou lorsqu'il y a une erreur. On peut également déterminer la version du code QR à l'aide de ces motifs [Sharma].

**Les motifs d'alignement (alignement patterns).** Les motifs d'alignement sont composés de 3 carrés concentriques superposés et de taille 5x5 modules pour le premier noir, 3x3 modules pour le blanc et un seul module central noir. Ils sont très efficaces pour ajuster les distorsions non linéaires du code QR. Le nombre de ces motifs d'alignement dépend de la version du code généré [Soon].

**La zone des données.** La zone des données est la région où toutes les données encodées sont stockées ainsi que le code de correction d'erreur, les informations de version et de format. L'information sera codée en nombres binaires et sont remplacés par des cases noires et blanches et sont ensuite corrigés. Des codes Reed-Solomon seront incorporés dans la zone d'information pour les données stockées et la fonction de correction des erreurs [Sharma].

### 1.3 Les caractéristiques du code QR

Les caractéristiques du code QR sont décrites dans le tableau I.1

La taille du code	Min. 21x21 modules - Max. 177x177 modules	
Volume de données selon le type	Caractères numériques	7 089 caractères au maximum
	Caractères Alphanumériques	4 296 caractères au maximum
	Binaire (8 bits)	2 953 caractères au maximum
	Caractères de Kanji/Kana	1 817 caractères au maximum
Efficacité de conversion	Mode de caractères numériques	3.3 modules/caractère
	Mode de caractères alphanumériques	5,5 modules/caractères
	Mode binaire (8 bits)	8 modules/caractère
	Mode de caractères de Kanji/Kana (13bits)	13 modules/caractère
La fonction de correction d'erreur	Niveau L	Environ 7 % de la surface du symbole restaurée au maximum
	Niveau M	Environ 15 % de la zone du symbole restaurée au maximum
	Niveau Q	Environ 25 % de la surface du symbole restaurée au maximum
	Niveau H	Environ 30 % de la surface du symbole restaurée au maximum
Annexe structurée	Peut être divisé en 16 symboles au maximum	

Table I.1: Les caractéristiques du code QR

## 1.4 Les caractéristiques du code QR

- La taille du symbole est choisie en fonction du volume de données à stocker et de la méthode de lecture.
- Plusieurs types de données peuvent coexister dans le même code QR.
- Chaque mode de conversion a fait l'objet de considérations visant à améliorer son efficacité de conversion.
- La fonction de correction d'erreur permet la restauration des données. Il existe quatre niveaux de restauration différents, de sorte qu'on puisse sélectionner le niveau qui correspond à chaque environnement d'utilisation.
- Un symbole de code QR peut contenir plusieurs symboles plus petits, chacun contenant des informations différentes et uniques [Nyon].

## 1.5 Les variantes du code QR

Il existe quarante versions du code QR avec des niveaux de correction d'erreurs basés sur le code de Reed-Solomon Algorithm. En allant d'une version à la suivante, on ajoute quatre modules pour chaque côté. Par la suite plusieurs variantes de ce code ont vu le jour [Web1].

**MicroQR.** Pour les symboles de taille encore plus petite, il est possible d'utiliser les codes microQR (voir figure I.3). Ils contiennent moins de données (35 caractères numériques au maximum) et sont conçus pour des espaces très réduits, par exemple sur les circuits imprimés et d'autres composants électroniques [Winter].

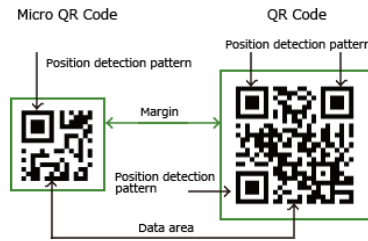


FIGURE I.3: Micro code QR [Web1]

**Iqr.** Un code iQR peut être utilisé dans un large éventail de domaines car il peut être transformé en rectangle, voir exemple dans la figure I.4. Il peut être imprimé sous la forme d'un code retourné, d'un code d'inversion noir et blanc ou d'un code à motifs depoints [Chang].



FIGURE I.4: Code iQR [Web1]

**SQRC (secure QR code).** Le SQRC est un code qui peut contenir des « données publiques » et des « données privées » en même temps. Il a une fonction de restriction qui bloque l'accès aux données sensibles sauf pour un lecteur dédié avec une clé de cryptage [Web1]. Ce symbole ressemble à un code QR normal.

**FrameQR.** Le FrameQR a une zone à l'intérieur du code où on peut mettre une image, des illustrations ou des logos, voir exemple dans la figure I.5. La forme et la couleur du cadre peuvent être modifiées librement [Chang].



FIGURE I.5: Code FrameQR [Web1]

## 1.6 Le décodage du code QR

Au début de l'apparition des codes unidimensionnels et bidimensionnels, des appareils à laser ont été mis au point pour permettre de détecter ces codes. Aujourd'hui, les logiciels utilisent des techniques de traitement d'images afin d'extraire le symbole à partir de l'image. Une fois le code QR détecté, le scanner commence le décodage. Le schéma I.6 représente les étapes de cette technique.

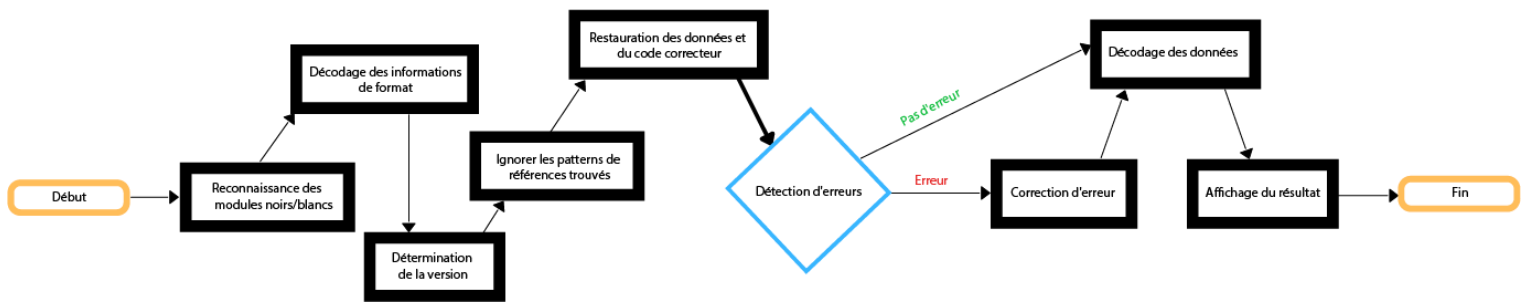


FIGURE I.6: processus de décodage de code QR [De Almeida]

Il est toutefois possible d'expliquer plus en détail le processus de décodage d'un code en s'attardant sur chaque étape.

Afin de mieux illustrer comment fonctionne le décodage prenons le code QR de la figure I.7.



FIGURE I.7: QR code à décoder [Web3]

La première étape consiste à extraire le format du code QR en lisant les zones en vert illustrées dans la figure I.8.



FIGURE I.8: Image représentant les bits du format du code QR (en vert) [Web3]

Une fois extrait on obtient une valeur sur 15bit à laquelle on applique un ou exclusif (XOR) avec le masque 101010000010010 [Web32].

Avec le format du code QR obtenu l'étape suivante consiste à démasquer les données pour cela on doit d'abord connaître le niveau de correction du code QR (de L jusqu'à H comme préciser précédemment) représenté par les 2 premiers bits de la séquence ainsi que le masque binaire utilisé sur le code QR afin d'encoder les données, qui se trouve sur les 3 bits suivant ceux du niveau de correction. Le masque binaire consiste à encoder le code QR en inversant la couleur des modules noir en blanc et vice versa en suivant un certain pattern. Au nombre de huit chacun se base sur une certaine fonction mathématique. Voir les types de masque dans la figure I.9.

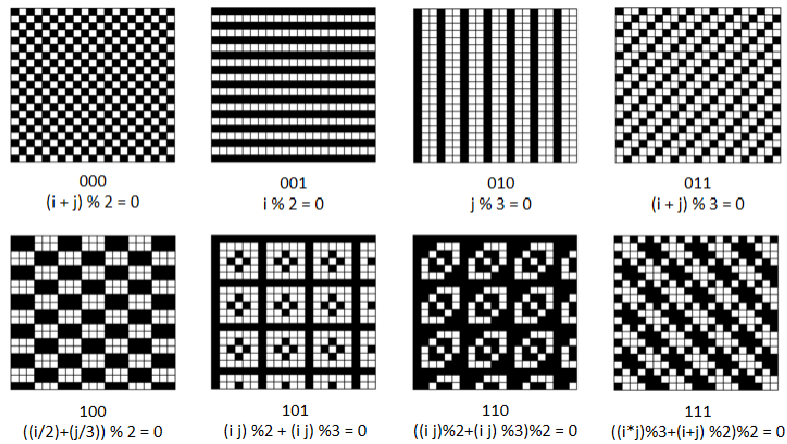


FIGURE I.9 : Les différents types de masque binaire que peut avoir un code QR [web4]

Ayant le masque binaire adéquat il suffit par la suite de l'appliquer l'intégralité du code QR à l'exception des détections pattern ce qui nous permet d'obtenir notre code démasqué prêt à être lu.

Maintenant que la lecture du code QR est possible il est nécessaire de découper le code QR en blocs de bits, commençant en bas à droite du code, les 4 premiers bits précisent le type d'encodage (numérique, alphanumérique ...), les 8 bits suivants indiquent la taille du message contenu dans le code QR, par la suite nous continuons de découper le code QR en blocs de 8 bits en allant de haut en bas et de droite à gauche. À noter qu'un bloc de 4 bits vide (entièrement blanc) sépare le message et la correction d'erreur. Voir l'exemple du des étapes du découpage dans la figure I.10 et la figure I.11.

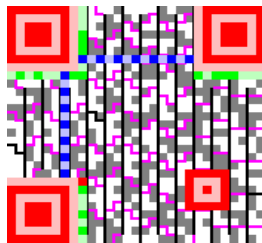


FIGURE I.10 : Découpage du code QR en blocs de 8 bits [Web3]

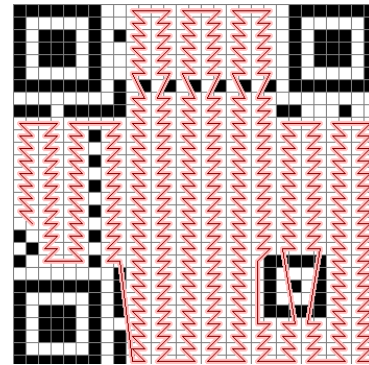
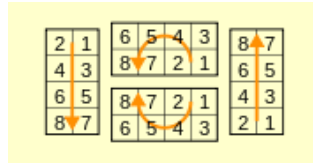


FIGURE I.11 : Chemin à suivre lors du décodage d'un code QR [Web3]

Chaque bloque de code contient une lettre du message codifié sous format ascii, chacun des 8 bits représentant une puissance de 2 qui une fois assemblé en utilisant les formules Eq (1.1) et Eq (1.2) donne le résultat final.

$$n = \overline{b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0}_2 = \sum_{i=0}^7 b_i 2^i$$

Eq (1.1) [Web3]



Eq (1.2) [Web5]

## 1.7 Avantages du code QR

- Les codes QR peuvent contenir jusqu'à plusieurs centaines de fois plus d'informations qu'un code à barres classique.
- Une lecture rapide et fiable.
- Ils peuvent stocker la même quantité d'informations dans un dixième de l'espace des codes habituels.
- Ils ont une capacité intégrée de correction d'erreur (système Reed-Solomon) qui aide à la récupération des données même si elles sont endommagées jusqu'à 30%.
- Ils sont omnidirectionnels et peuvent être scannés en les tenant à n'importe quel angle sur 360 degrés.
- Ils sont capables d'encoder les caractères utilisés dans les systèmes d'écriture logographiques et phonémiques (par exemple, Kana et Kanji)

## 1.8 Inconvénients du code QR

- Les algorithmes qui analysent les codes QR sont complexes vu la diversité des données qu'ils peuvent encoder.
- Scanner un code QR inconnu peut poser de sérieux problèmes de sécurité

## 2 Détection des codes QR

### 2.1 Méthode de Viola et Jones

La méthode de Viola et Jones est une méthode de détection d'objet dans une image numérique, proposée par les chercheurs Paul Viola et Michael Jones en 2001. Cette méthode consiste à entraîner un modèle d'apprentissage supervisé en utilisant plusieurs milliers d'images de l'objet souhaitant être détecté. Cette méthode est considérée comme étant une pionnière dans le cadre de la détection d'objet et de la vision artificielle, elle est d'ailleurs implémentée dans de nombreuses bibliothèques de vision artificielle dont la plus connue, OpenCV. Cette méthode peut être divisée en 3 éléments distincts :

*Les caractéristiques* : Connue sous le nom des caractéristiques pseudo-Haar [Viola and al] qui consistent à obtenir une représentation synthétique et informative des pixels en calculant la différence des sommes des différents pixels d'une sous-région d'une image. Une autre méthode de calcul plus optimale aura été inventée par la suite du nom d' « image intégrale ».

*Sélection par boosting* : Cet élément consiste en la sélection des meilleures caractéristiques en construisant un classifieur dit « fort » à partir des classifieurs dits « faibles ». Les plus performants d'un ensemble de classifieurs « faibles » sont entraînés et sélectionnés par la suite d'un processus itératif déterminé.

*Cascade de classifieurs* : La méthode de Viola et Jones étant basée sur une recherche

exhaustive, son coût de calcul est extrêmement élevé. Afin de palier à ce problème une cascade de classifieurs est utilisée dans l'algorithme. Une méthode itérative qui a pour effet de réduire considérablement le temps de calcul.

## 2.2 Méthodes basées sur une approche neuronale

### 2.2.1 R-CNN

R-CNN, ou Region Based Convolutional Neural Networks, est un modèle de détection d'objets et d'apprentissage automatique utilisé dans le domaine de la vision par ordinateur qui utilise des CNN et qui analyse un grand ensemble de régions dans une image ou une vidéo afin de localiser et de segmenter des objets. Ce modèle a été conçu par Ross Girshick afin de contourner le problème de la sélection d'un grand nombre de régions à analyser. Ces régions sont générées à l'aide de l'algorithme de recherche sélective.

Le R-CNN est composé des 3 modules [Girshick], voir figure I.12 :

- Un générateur de proposition de régions indépendante par catégorie
- Un grand réseau neuronal convolutif qui extrait une caractéristique des régions
- Un ensemble de SVM (support vector machine) linéaires spécifiques à une classe

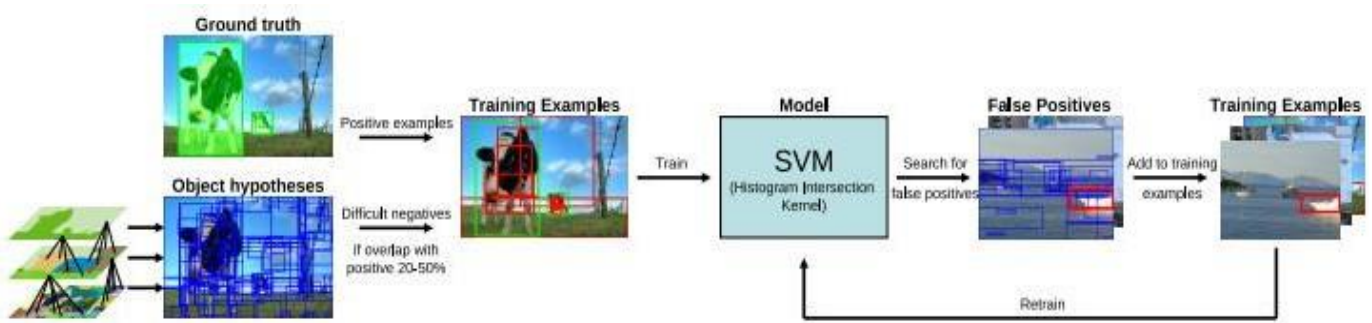


FIGURE I.12: Fonctionnement du modèle R-CNN [van de Sande†]

### 2.2.2 Fast R-CNN

Le Fast R-CNN est une amélioration du R-CNN publiée par le même auteur. Cette version est très similaire au R-CNN mais en plus rapide et avec des inconvénients en moins, l'ajout de la technique RoI Pooling (Region of Interest) [Girshick] est l'un des plus grands changements ajoutés. Voir l'architecture du Fast R-CNN dans la figure I.13.

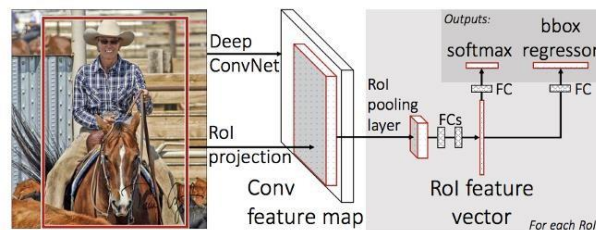


FIGURE I.13: Architecture du Fast R-CNN [Girshick]



### 2.2.3 Faster R-CNN

Le Faster R-CNN est une amélioration du Fast R-CNN toujours proposée par Ross Girshick. Elle est composée de deux modules. Le premier module est un réseau entièrement convolutif qui propose des régions, et le deuxième module est le détecteur Fast R-CNN [Girshick and al]. L'amélioration majeure réside dans la suppression par la recherche sélective pour découvrir les propositions de région. Par un réseau entièrement convolutif du nom de Region Proposal

Network (RPN), augmentant considérablement ses performances. Le reste de l'algorithme est similaire au Fast R-CNN. Voir figure I.14.

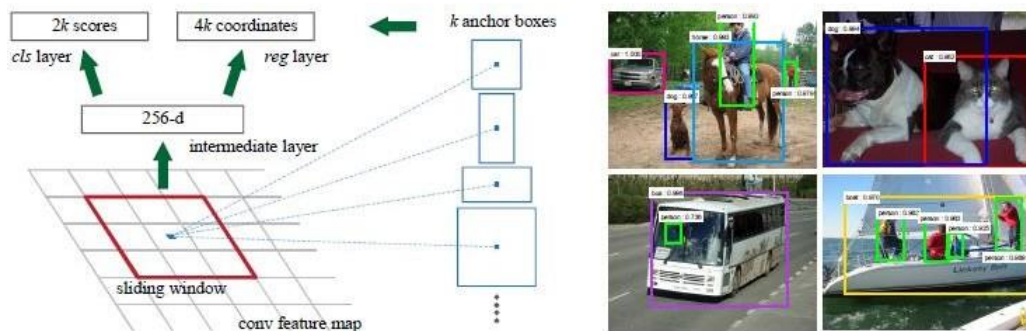


FIGURE I.14: illustration du fonctionnement du Region Proposal Network (RPN) [Girshick and al].

### 2.2.4 Mask R-CNN

Le Mask R-CNN (figure I.15) est construit à partir Faster R-CNN. Qui, en utilisant une segmentation au niveau des pixels, permet d'identifier chaque objet séparément. Cet algorithme adopte les mêmes étapes que le Faster R-CNN, avec une première étape identique (RPN). Dans la deuxième étape, parallèlement à la prédiction de la classe et de la boîte offset, Mask R-CNN génère également un masque binaire pour chaque RoI [Girshick and al]. Un dernier changement important vient dans l'ajout des FPNs (Feature Pyramid Networks) [Web6].

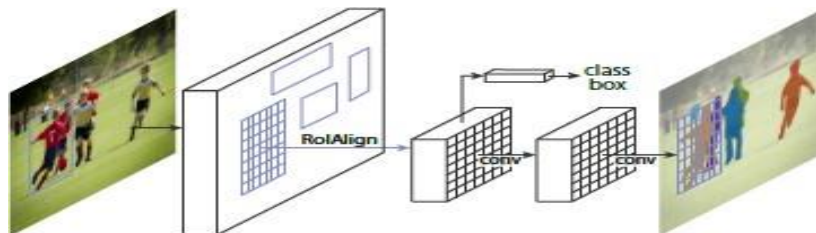


FIGURE I.15: segmentation du Mask R-CNN [Girshick and al]

### 2.2.5 SSD (Single Shot Detector)

Contrairement aux modèles cités précédemment le modèle SSD se démarque en étant conçu spécialement pour la détection d'objets en temps réel. Les algorithmes précédents ayant une précision de détection assez satisfaisante ils manquaient néanmoins d'une vitesse nécessaire à la détection en temps réel. Le SSD est capable d'accélérer le processus en éliminant le besoin du réseau de proposition de région utilisant à la place un MultiBox Detector (figure I.16) et une fonction de perte [Anguelov and al].

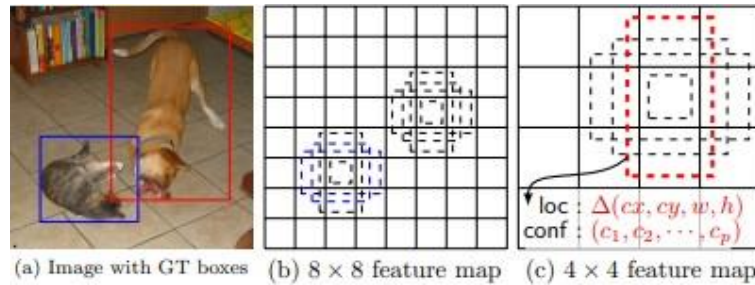


FIGURE I.16: MultiBox Detector [Anguelov and al].

### 2.2.6 YOLO (You Only Look Once)

YOLO ou You Only Look Once est un algorithme de détection d'objets dont un seul réseau convolutif prédit les « bounding box » et les probabilités qu'elles contiennent l'objet recherché, voir figure I.17. Il existe plusieurs versions de YOLO, les plus utilisées actuellement sont yolov3 [Redmon] et yolov4 [Bochkovskiy and al], mais la version la plus récente est yolov5.

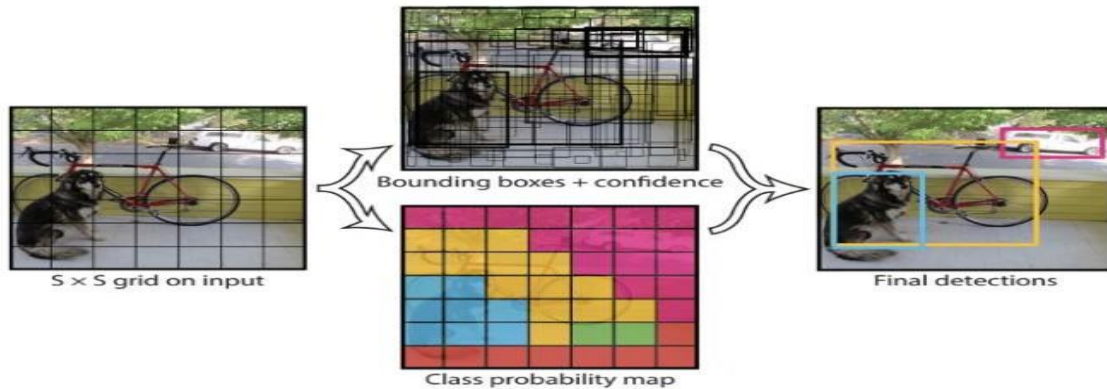


FIGURE I.17: Fonctionnement de Yolo [Redmon and al]

## 3 Le réseau YOLO (You Only Look Once)

You only look once est un CNN de détection d'objet qui a comme particularité de détecter les « bounding box » d'un objet en une seule itération (d'où son nom), son architecture est assez simple mais puissante le rendant beaucoup plus précis et rapide que les CNN présentés précédemment tels que Faster R-CNN, SSD et Mask R-CNN pour la détection en temps réel. Il divise l'image en plusieurs petites cellules qui s'occupent de manière indépendante de détecter l'objet qu'elles contiennent (elles le détectent uniquement si le centre de l'objet se trouve en elle). Ensuite, le prédit la « bounding box » associée contenant les informations suivantes : x, y, w, h, et la « confidence ». Les coordonnées (x,y) représentent les valeurs du centre de la

« bounding box » tandis que les valeurs w et h qui eux représentent la hauteur et la largeur de

la « bounding box » et pour un score de « confidence ».

Le terme « bounding box » décrit le rectangle qui entoure un objet à détecter tandis que la « confidence » qui représente un score de confiance nous indiquant à quel point le détecteur est certain que la « bounding box » contient effectivement un objet. Elle est calculée de la manière suivante :

$$C = \Pr(\text{Object}) * \text{IoU}_{\text{pred}}^{\text{truth}}$$

$\Pr(\text{Object})$  est la probabilité de la classe, tandis que  $\text{IoU}_{\text{pred}}^{\text{truth}}$  est l'intersection sur l'union entre la bounding box prédite et la vérité du terrain.

Il est à savoir que le terme « Anchors » est utilisé pour désigner les tailles des objets de l'image qui ont été redimensionnés à la taille du réseau [Web24].

Depuis sa création YOLO a connu de nombreux changements et d'amélioration le rendant le CNN de détection d'objet le plus performant et à ce jour, 5 versions de YOLO existent.

## 4 YOLOv5

YOLOv5 a été publié sur Github est annoncé comme étant la dernière version en date de YOLO. Reprenant la même architecture que YOLOv4, cette version a comme particularité d'être développée en Python contrairement aux autres versions développées en C. Etant à présent utilisable sur « Pytorch » l'une des bibliothèques python les plus populaires pour le « deep learning », cela permet une évolution plus rapide du modèle, la communauté Pytorch étant plus active que celle de DarkNet. Cela a eu aussi pour effet de faciliter son utilisation et son implémentation lors d'un projet, notamment dans le domaine de l'IoT. Etant en quelque sorte un modèle YOLOv4 dans un autre langage, leurs performances sont extrêmement similaires mais après plusieurs tests, il a été constaté que YOLOv5 est plus performant dans certains types de projet notamment grâce à la facilité de son implémentation [Do Thuan].

Il existe plusieurs versions de YOLOv5, chacune avec un modèle ayant des avantages et des inconvénients en termes de performance, rapidité d'entraînement et moyenne de précision, voir figure I.18.

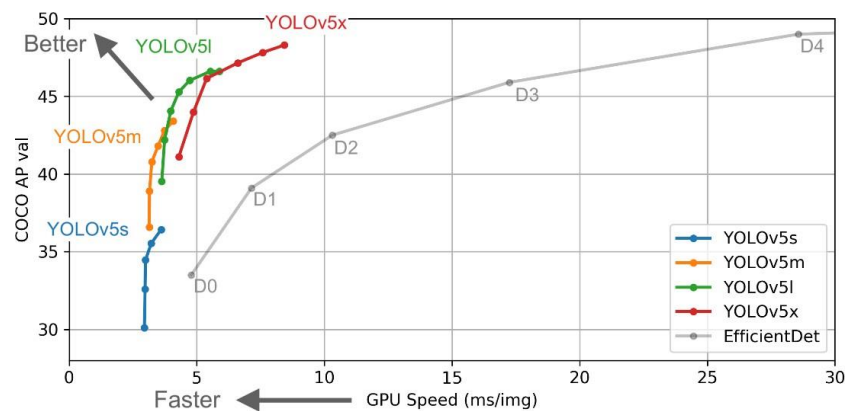


FIGURE I.18 : Comparaison des performances des versions de YOLOv5 [Web31]

### 4.1 Architecture de YOLOv5

Puisque YOLOv5 est un détecteur d'objets en une seule itération, son architecture est constituée de 3 parties comme tout autre détecteur du même type, voir le schéma de l'architecture dans la figure I.19.

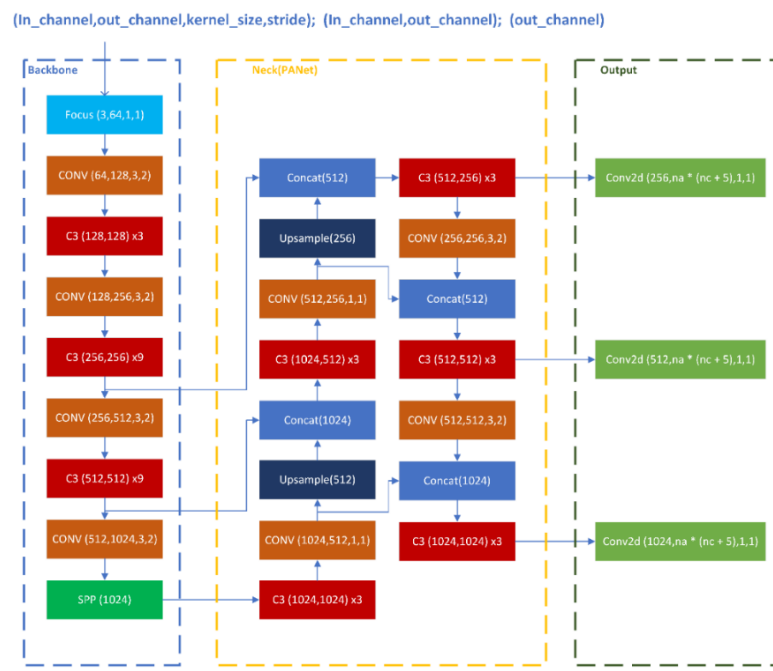


FIGURE I.19 : Aperçu de l'architecture de YOLOv5[Web24]

## Model Backbone : La couche Focus, réseau CSP

Cette partie du modèle est principalement utilisée pour faciliter l'extraction des « features » importantes d'une image d'entrée donnée.

La couche « Focus » est utile pour la transformation de la largeur en profondeur. Dans YOLOv5, on a voulu réduire le coût du calcul « Conv2d » en en faisant du remodelage tensoriel pour réduire l'espace (résolution) et augmenter la profondeur (nombre de canaux).

CSPNet (Cross Stage Partial Networks) est un réseau qui est utilisé pour améliorer la capacité d'apprentissage d'un CNN. Il représente la colonne vertébrale du modèle YOLOv5 étant donné que c'est à partir de ce réseau là que l'extraction des caractéristiques riches en informations d'une image d'entrée commence. CSPNet a montré une amélioration remarquable du temps de traitement avec des réseaux plus profonds. De plus, il est facile à mettre en œuvre et peut s'adapter aux architectures basées sur DenseNet, ResNeXt et ResNet [Wang and al].

## Model Neck : Le bloc SPP, PANet

Cette partie sert à générer des pyramides de caractéristiques « Feature Pyramids ». Ces dernières aident les modèles à se généraliser et à identifier un objet avec des tailles différentes. Ainsi, elles permettent l'obtention des bonnes performances sur des données non encore vues.

Puisque plusieurs modèles basés sur les CNN contiennent des couches entièrement connectées qui n'acceptent que des images d'entrée de dimensions spécifiques, SPP est créé pour générer une sortie de taille fixe, quelle que soit la taille de l'entrée. De plus, SPP permet également d'extraire des caractéristiques importantes en produisant des versions multi-échelles de lui-même. Après la version 2 de YOLO, le modèle basé sur le FCN (réseau à convolution complète) qui

permet l'entrée d'images de différentes dimensions. En revanche, il doit faire des prédictions sur les coordonnées pour localiser les « bounding box » sur un frame ou une image. Il n'est donc pas souhaitable de transformer les « feature maps » -qui sont à la base bidimensionnelles- en un vecteur à une seule dimension. C'est pourquoi le bloc SPP a été modifié dans YOLO pour préserver la dimension spatiale de sortie [Do Thuan].

D'autre part, nous savons que, plus l'image passe par les couches profondes du modèle, plus la complexité des caractéristiques apprises augmente. Par conséquent, la résolution spatiale des « feature maps » diminue en raison du sous-échantillonnage (down sampling) et cela entraîne une perte d'informations. Pour remédier à ça, la technique « Feature Pyramid » a été rajoutée au modèle YOLO. Ainsi, il est devenu possible de prédire les petits objets dans un détecteur à grande échelle. Il existe plusieurs modèles qui utilisent différents types de techniques de génération comme FPN, BiFPN, PANet, etc. Dans YOLO v5, PANet est utilisé [Web22].

### **Model Head : YOLOv3 head avec GIoU-loss**

Cette partie représente la dernière étape de la détection. Dans, YOLOv5, cette partie est restée inchangée et est donc la même que celle des deux versions précédentes (YOLOv4 et YOLOv3). Pour faire la détection, le modèle applique coordonnées (centre, hauteur, largeur) des « anchors » prédites sur les caractéristiques extraites préalablement afin de générer des vecteurs avec des probabilités de classe, des « bounding box » et des scores d'objectalité [Web21].

## **4.2 La fonction d'activation**

Le choix de la fonction d'activation est extrêmement important dans un réseau de neurones. Dans le modèle YOLOv5, deux fonctions ont été utilisées : Leaky ReLU et Sigmoid.

La fonction d'activation Leaky ReLU est utilisée dans les couches intermédiaires/cachées tandis que la fonction d'activation Sigmoid -qui est souvent utilisée pour la classification- est utilisée dans la couche de détection finale.

La fonction Leaky ReLU est une amélioration de la fonction ReLU (Rectifier Linear Unit). Cette dernière attribue un gradient égal à 0 pour toutes les entrées qui sont inférieures à zéro, chose qui entraîne la désactivation des neurones de la région concernée. Leaky ReLU a résolu ce problème en définissant une composante linéaire extrêmement petite de chaque entrée  $x$  négative [Favorskaya].

## **4.3 La fonction d'optimisation**

Dans YOLOv5, il y a deux options de fonctions d'optimisation : SGD et Adam.

SGD (Stochastic Gradient Descent) est la fonction utilisée par défaut. Cette approche est à la fois simple et efficace pour adapter des classificateurs et des régresseurs linéaires à des fonctions de coût convexes. SGD n'a reçu que récemment une attention considérable dans le cadre de l'apprentissage à grande échelle. Par ailleurs, Adam est une extension de l'algorithme SGD qui maintient un taux d'apprentissage par paramètre qui améliore les performances sur les problèmes de vision par ordinateur [Web23].

## **4.4 La fonction coût**

Le calcul du coût du modèle YOLO est basé sur la probabilité de classe, les scores de régression des « bounding box » et les scores d'objectalité. Par ailleurs, GIoU-loss (Generalized Intersection over Union) est la fonction utilisée pour mesurer la perte dans les « bounding box » en plus de la fonction « binary cross entropy » avec « Logits » qui est utilisée pour calculer le coût de probabilité de classe et du « Target score » [Web21].

Binary Cross Entropy est une fonction de coût utile pour les tâches de classification binaire. C'est-à-dire répondre à une question avec le choix oui ou bien non. Il est également possible de répondre à plusieurs questions indépendantes en même temps, comme dans la classification multi-labels. D'autre part, Logits informe simplement la fonction de coût qu'elle opère sur la sortie non échelonnée des couches, c'est-à-dire que les valeurs de sortie générées par le modèle ne sont pas normalisées.

## **Conclusion**

Le domaine de la vision par ordinateur et de l'apprentissage profond aura connu de nombreuses recherches ayant été utilisé avec les codes QR. Et reste en développement constant. Nous pouvons donc nous attendre à une grande évolution et innovation dans ce cadre de recherche et un énorme potentiel à offrir dans le monde scientifique et plus particulièrement celui de l'intelligence artificielle.

# Chapitre II. L'implémentation de l'algorithme de détection du code QR

## Introduction

Les quantités de données augmentent au fur et à mesure que la société humaine se développe. L'être humain fait face à un nombre immense de données où la gestion prend beaucoup de temps. Notre projet de fin d'études consistait à faciliter l'accès aux données des doctorants et du personnel en utilisant de code QR. Pour cela nous avons mis en œuvre un scanner de code QR. La première tâche à faire pour scanner ce dernier est la détection.

Cependant, l'existence de plusieurs solutions pour la détection nous a emmené à les étudier et opter pour une nouvelle méthode en utilisant une approche neuronale et l'implémenter dans notre travail.

## 1 Implémentation de l'algorithme YOLOv5

Pour réaliser la détection en temps réel des Codes QR, un modèle de YOLO doit être utilisé étant donné qu'il soit le plus performant parmi tous ceux qui existent. Nous avons choisi YOLOv5 c'est la version qui a eu les meilleurs résultats parmi les cinq. De plus, c'est une solution déjà implémentée qui nous permettra d'épargner le temps de son implémentation et l'investir dans le développement d'un modèle efficace pour la détection des codes QR.

### 1.1 L'environnement de l'entraînement



FIGURE II.1 : Les outils utilisés

**Roboflow :** Cette plateforme facilite le déploiement des modèles de vision par ordinateur personnalisés. En l'utilisant pour YOLOv5, on peut facilement y mettre un dataset contenant les images et leurs coordonnées dans un fichier texte portant le même nom. Ces dernières seront ensuite converties au format d'annotation. On peut également détecter les mauvaises annotations, les ajuster, évaluer la santé de notre dataset et même l'augmenter. Cependant, c'est gratuit que pour les 1000 premières images [Web25].

**Google Colaboratory :** Cet outil développé par Google est un notebook qui marche dans le cloud. Il permet à l'utilisateur d'écrire et exécuter des programmes en Python très facilement. De plus, il ne requiert aucune configuration, il facilite le partage entre les codeurs et finalement, il donne un accès gratuit aux GPU pendant 12 heures [Web26].

**Pytorch :** C'est une bibliothèque d'apprentissage automatique qui est opensource et dédiée au langage Python. Elle est principalement utilisée pour les applications du Deep Learning utilisant

les GPU et les CPU pour l'entraînement du réseau de neurones. Elle a été développée par l'équipe de l'intelligence artificielle de Facebook [Web27].

**TensorBoard** : C'est un outil qui fournit toutes les visualisations nécessaires au cours de l'apprentissage. Il permet de suivre les métriques de l'expérience tels que le coût (loss) et la précision (accuracy) ainsi que de visualiser le graphe de la performance du modèle pendant l'entraînement et la phase de test, et bien plus encore [Web28].

## 1.2 La préparation des données

L'un des plus gros problèmes des projet de data science en général est l'absence des datasets disponibles publiquement sur Internet. Ce problème est une limitation qui n'est pas facile à surmonter car les modèles de réseaux de neurones ont tendance à être moins performant lorsqu'il a peu de données sur lesquelles s'entraîner.

### Les annotations YOLO

Les annotations YOLO sont représentées sous forme d'étiquettes contenant des informations sur le Code QR dans l'image. Le format de ces étiquettes est le suivant : {classe x y w h}, tel que classe est l'identificateur de l'objet à détecter, x et y étant les coordonnées du centre de l'objet et finalement, w et h la largeur et la hauteur de la « bounding box », respectivement. Une normalisation par rapport à la taille de l'image est appliquée sur les coordonnées ce qui fait que toutes leurs valeurs sont comprises dans l'intervalle [0...1]. Il est à savoir que dans notre projet, nous avons une seule classe qui est QR-Code.

### Dataset 1

Pour une première itération, nous avons pris les images du dataset [Web29]. Nous l'avons ensuite fusionné avec le premier dataset trouvé dans [Web30]. Au total, ils contiennent 735 images. Nous avons ensuite effectué les étapes suivantes afin de préparer ce dataset :

- 1- Créer un script qui change les noms des classes du dataset tiré de [Web30], crée des fichiers texte avec des coordonnées erronées et enfin, renommer toutes les images du nouveau dataset ainsi que leurs fichiers texte associés.
- 2- Mettre le dataset dans Roboflow afin de détecter les mauvaises annotations.
- 3- Ajuster manuellement toutes les étiquettes du dataset.
- 4- Faire le preprocessing et l'augmentation des images pour avoir 1758 images.
- 5- Diviser le dataset en deux : le train set contenant 1539 images (88%), le validation set avec 146 images (8%) et finalement le test set avec 73 images (4%).

Roboflow a simplifié le processus de la préparation du dataset car nous l'avons utilisé pour redimensionner les images à 416x416 et les augmenter. Pour cette dernière, nous avons rajouté 4 types de rotation : la rotation dans le sens des aiguilles d'une montre, la rotation dans le sens contraire des aiguilles d'une montre, la rotation à l'envers et une rotation entre  $-15^\circ$  et  $+15^\circ$ . Nous avons également rajouté une exposition entre -10% et +10%, et un bruit qui va jusqu'à 3% de pixels. Nous avons utilisé la technique de l'augmentation des images du dataset pour améliorer les résultats du modèle pour les images déformées et éviter le « overfitting ». Voir un échantillon du dataset dans la figure II.2.



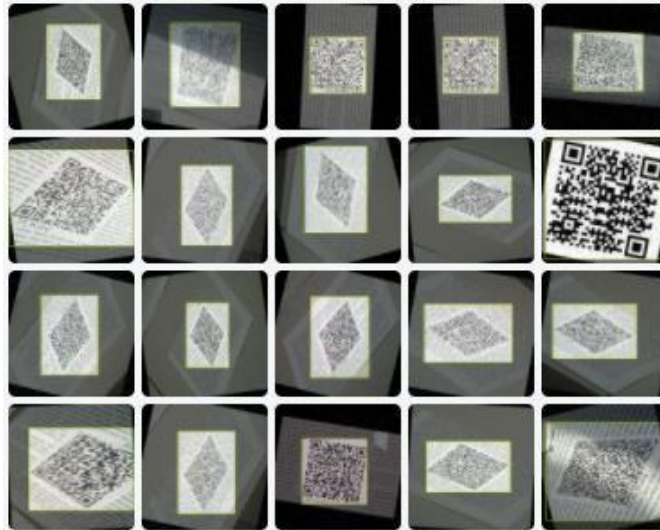


FIGURE II.2 : Un exemple de codes QR étiqueté

## Dataset 2

Pour construire le deuxième dataset, nous avons fusionné le dataset 1 avec les deux datasets restants de [Web30]. Nous avons eu donc un dataset qui contient 49 806 images de taille 416x416. Nous l'avons aussi divisé en train set contenant 43581 images (88%), un validation set avec 4 150 images (8%) et un test set avec 2 075 images (4%). Il est à savoir que nous n'avons effectué aucune augmentation en plus sur ce dataset car YOLOv5 prend en charge cette tâche lors de l'entraînement, voir l'augmentation effectuée par YOLOv5 dans la figure II.3.

```
'hsv_h': (1, 0.0, 0.1), # image HSV-Hue augmentation (fraction)
'hsv_s': (1, 0.0, 0.9), # image HSV-Saturation augmentation (fraction)
'hsv_v': (1, 0.0, 0.9), # image HSV-Value augmentation (fraction)
'degrees': (1, 0.0, 45.0), # image rotation (+/- deg)
'translate': (1, 0.0, 0.9), # image translation (+/- fraction)
'scale': (1, 0.0, 0.9), # image scale (+/- gain)
'shear': (1, 0.0, 10.0), # image shear (+/- deg)
```

FIGURE II.3 : Un exemple de codes QR étiqueté

## 1.3 La configuration du modèle

Puisque les algorithmes de deep learning nécessitent une puissance et une vitesse de traitement très élevées (généralement basées sur les GPU), chose qui n'est pas à la portée des ordinateurs ordinaires qui ne sont pas équipés de GPU, nous avons décidé d'utiliser Google Colaboratory qui fournit des GPU Tesla K80 pour entraîner notre modèle, voir les détails dans la figure II.4.

NVIDIA-SMI 465.27				Driver Version: 460.32.03		CUDA Version: 11.2	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile Uncorr. ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	MIG M.
0	Tesla K80	Off	00000000:00:04.0	Off	0%	Default	N/A
N/A	32C	P8	29W / 149W	3MiB / 11441MiB			

Processes:							
GPU	GI	CI	PID	Type	Process name	GPU Memory	Usage
ID	ID	ID					
No running processes found							

FIGURE II.4 : L'état d'utilisation du GPU alloué

Tout comme les versions précédentes de YOLO, le modèle YOLOv5 a été construit théoriquement ensuite a été déployé et publié via un repo Github. Ultralytics a facilité l'utilisation du modèle en le construisant sur le framework Pytorch. Contrairement aux architectures préliminaires, celle-ci peut être configurée pour obtenir les meilleurs résultats en fonction des problèmes. On peut par exemple y ajouter des couches, supprimer des blocs, y intégrer des méthodes de traitement supplémentaires, modifier les méthodes d'optimisation ou les fonctions d'activation, etc.

## 1.4 L'entraînement et l'évaluation du modèle

Avant de commencer, il est important de savoir que YOLOv5 a des paramètres prédéfinis pour les personnes qui veulent utiliser la version de base. Parmi les plus importants on trouve :

- *Lr0* : le Learning rate initial qui est 1E-2 pour SGD et 1E-3 pour Adam.
- *Lrf* : un hyper-paramètre fait pour calculer le OneCycleLR learning rate en utilisant la formule ( $\text{lrf} * \text{lr}$ ), le learning rate final sera  $\text{hyp}[\text{'lrf'}] * \text{hyp}[\text{'lr0'}]$  en utilisant le schedulerLR en cosinus.
- *Momentum* : un hyper-paramètre de la fonction SGD qui permet d'accélérer les vecteurs de gradients dans les bonnes directions lors de l'apprentissage. Son équivalent pour la fonction Adam dans YOLOv5 est  $\beta_1$ . Ce dernier est un taux de décroissance initial utilisé lors de l'estimation des premiers et deuxièmes moments du gradient, qui est multiplié de manière exponentielle à la fin de chaque étape d'entraînement (Batch). La diminution de  $\beta_1$  rendra l'apprentissage plus lent.
- *Anchor* : taille de carré par grille de sortie (0 à ignorer).
- *Weight decay* : C'est une technique de régularisation qui consiste à ajouter une petite pénalité, généralement la norme L2 des poids à la fonction de coût.
- *Iou\_t* : le seuil de l'IoU lors de l'entraînement (figure II.5)


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


FIGURE II.5 : La formule de la fonction Intersection over Union [Do Thuan]

L'IoU est utilisé pour déterminer si la prédiction est un :

1. *Vrai positif* : L'IoU est supérieur à la valeur seuil et la classe prédite correspond à la vérité terrain.
2. *Faux positif* : L'IoU est inférieur à la valeur seuil.
3. *Faux négatif* : L'IoU est supérieur à la valeur seuil mais la classe prédite est incorrecte, ou aucune prédiction n'a été effectuée en présence d'une vérité de terrain.

Pour entraîner le modèle YOLOv5, il est possible de configurer plusieurs paramètres. Parmi les plus pertinents, on peut citer :

- *Weights* : Le chemin vers le fichier des poids initiaux à entraîner.
- *Cfg* : Le chemin vers le fichier .yaml contenant l'architecture du modèle à utiliser.
- *Data* : Le chemin vers le fichier data.yaml qui contient les chemins du dataset de l'entraînement et de la validation.
- *Hyp* : Le chemin du fichier des hyperparamètres à utiliser.
- *Epochs* : Le nombre de fois que le modèle va travailler sur l'ensemble des données.
- *Batch-size* : Le nombre d'exemples d'images utilisées dans une itération.
- *Img-size* : La taille de l'image en entrée, YOLO a toujours les meilleurs résultats avec des tailles multiples de 32.
- *Evolve* : Un booléen qui permet aux hyperparamètres d'évoluer au fur et à mesure de l'entraînement.
- *Cache-images* : Un booléen qui utilise le cache d'image pour permettre un entraînement plus rapide.
- *Adam* : Un booléen qui permet à l'utilisateur d'utiliser la fonction d'optimisation Adam au lieu de SGD.
- *Nom* : Le nom du fichier où l'on stock les poids entraînés.

Dans la détection, nous utilisons souvent AP et mAP pour évaluer la qualité du modèle, mais mAP.5, mAP.95, recall and precision sont les métriques utilisées pour évaluer le modèle entraîné avec YOLOv5. Dans notre cas, nous nous intéressons au map.5 et mAP.5.95. Ces deux métriques représentent la précision moyenne du modèle pour un IoU égale à 0.5 et compris entre 0.5 et 0.95 respectivement.

## 1.5 Le test du modèle

Une fois le modèle est entraîné et évalué, il sera prêt pour l'utilisation en temps réel. Pour ce faire, nous avons exporté les meilleurs poids générés lors de l'entraînement, ensuite nous avons copié le repository de YOLOv5 de Github pour tester la détection avec la webcam en local. Voici quelques arguments à préciser à la fonction pour faire la détection :

- *Weights* : Le chemin vers le fichier des poids après l'entraînement.
- *Source* : Le chemin vers le dossier du dataset à tester.
- *Conf-thres* : Le seuil de la confiance à partir de laquelle le détecteur si un objet est un Code QR.
- *Iou-thres* : Le seuil de la fonction coût.
- *Max-det* : Le nombre de détections à faire par image.
- *View-img* : Un booléen pour afficher l'image à tester.
- *Hide-labels* : Un booléen pour cacher les étiquettes.
- *Hide-conf* : Un booléen pour cacher la valeur de la confiance.

## 2 Expériences

### 2.1 Expérience 1 : YOLOv5 Basique sans augmentation

Pour l'expérience 1, nous avons pris le dataset 2 et nous avons entraîné le modèle en compilant le fichier `train.py` avec ses arguments configurables.

La taille d'images la plus commune dans les modèles YOLO est 416, nous l'avons donc divisé ainsi étant donné que c'était déjà la taille des images que nous avons recensées.

En ce qui concerne le batch-size, nous savons qu'envoyer des images simultanément dans un réseau de neurones fait augmenter le nombre de poids qu'apprend ce dernier en une seule itération, et c'est pour cela qu'il faut diviser les images en lots qui passeront dans le réseau pour donner des résultats qui seront stockés dans la RAM. On en déduit que plus le nombre de lots est élevé, plus la consommation de mémoire sera importante. Dans notre cas, nous avons choisi une taille de lot égale à 16 pour un dataset qui contient un train set de images, ce qui donne  $43\,581/16 = 2\,723$  lots d'entraînement.

Pour le nombre d'itérations (epochs), nous avons choisi 500 epochs en se basant sur nos tests, il est à noter que les modèles YOLOv5 sont d'habitude entraînés pour 3000 epochs et plus.

Pour l'argument `data`, nous avons simplement mis le chemin vers le fichier « `data.yaml` » qui contient le chemin du train set et le validation set ainsi que le nombre des classes.

Pour l'argument `cfg` qui sert à choisir une version de YOLOv5 parmi celles déjà citées, nous avons choisi YOLOv5s étant donné que c'est la version la plus rapide à entraîner pour effectuer nos tests.

Par ailleurs, nous n'avons pas opté pour le Transfer Learning pour cette expérience et donc nous n'avons spécifié aucun chemin vers un fichier de poids à ajuster. De plus, nous avons utilisé les hyperparamètres de YOLOv5 de base que nous avons vus dans la section 1.7.5.

Le temps moyen pour effectuer l'entraînement sur un lot de 16 est 23 secondes pour l'entraînement et 1 seconde pour la validation. Le temps d'exécution total est 3h 26min et 40s et le mAP de la dernière itération est 99% (mAP.5) et 93% (mAP.5.95). Il faut noter que les meilleurs poids obtenus lors de l'entraînement ne sont pas forcément ceux appris lors de la dernière itération. C'est pour cela que nous avons utilisé TensorBoard afin de visualiser clairement l'ensemble du processus de l'entraînement. Voir figure II.6.

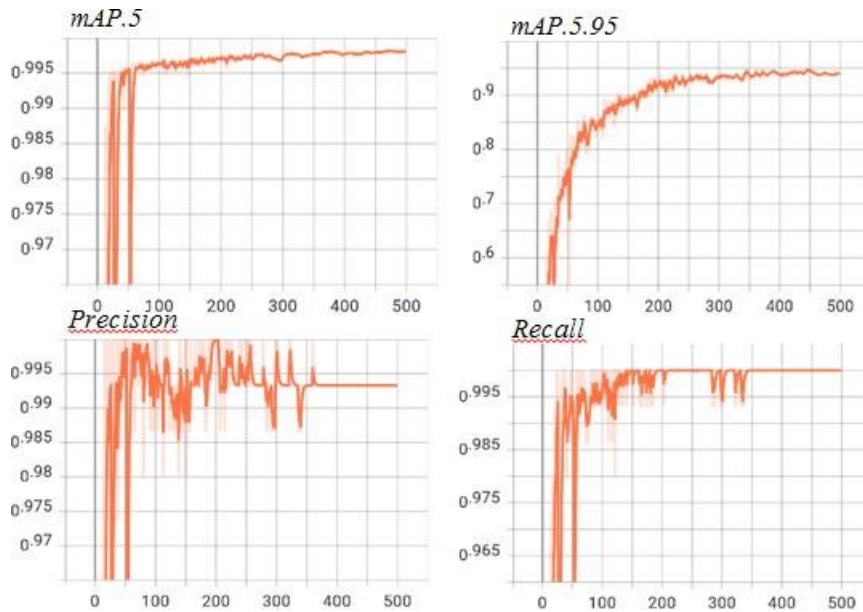


FIGURE II.6 : Les métriques de l'évaluation pour les 500 epochs de l'expérience 1

Comme on peut le voir, la précision du modèle est de 99.8%. Cela prouve qu'en utilisant l'architecture de base de YOLOv5, le modèle est non seulement rapide, mais la précision est également élevée sans aucune autre méthode d'optimisation.

Par ailleurs, le modèle enregistre deux fichiers de résultats (poids) : le last.pt qui contient les poids de la dernière epoch et le best.pt qui contient les poids de la dernière epoch qui a la meilleure précision.

La taille de ces fichiers est assez petite, rendant leur intégration dans notre interface plus facile tout en maintenant la précision du modèle.

Afin d'effectuer plus de tests, nous avons remplacé la fonction d'optimisation SGD (par défaut) avec la fonction Adam. Les résultats obtenus sont dans la figure II.7.

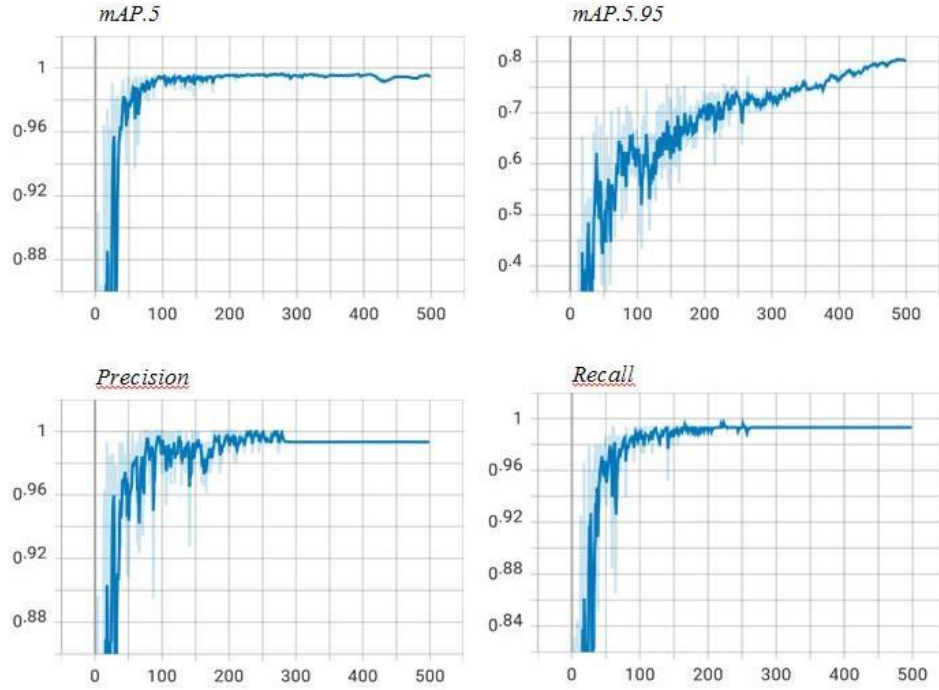


FIGURE II.7 : Les métriques de l'évaluation pour les 500 epochs de l'expérience 1 avec Adam

Puisqu' il est plus intéressant de voir visuellement les résultats que de mesurer le mAP. Dans la figure II.8 on trouve certaines inférences réalisées par les deux modèles.

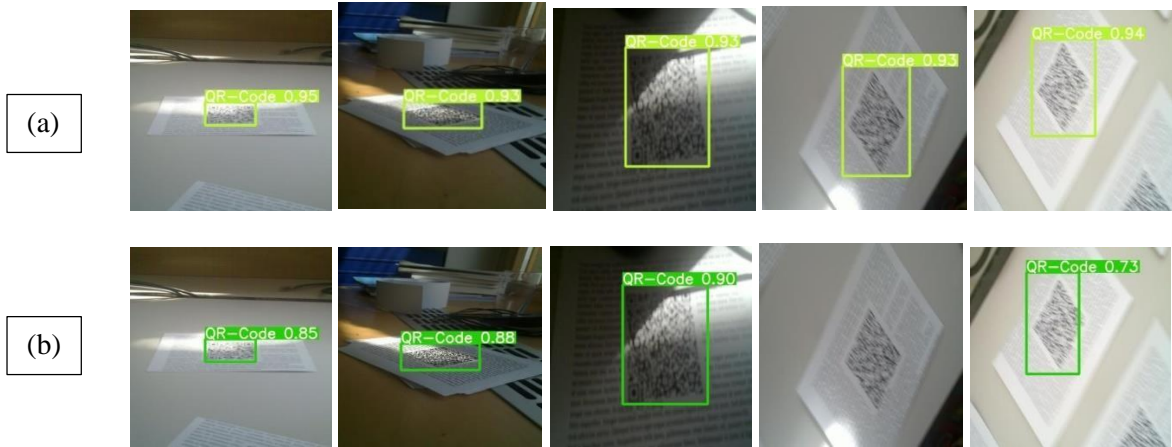


FIGURE II.8 : (a) L(a) a détection en utilisant la fonction d'optimisation SGD (b) La détection en utilisant la fonction d'optimisation Adam

## 2.2 Expérience 2 : YOLOv5 Basique avec augmentation

Pour l'expérience 2, nous avons gardé les mêmes paramètres et hyperparamètres du modèle de l'expérience 1 avec la fonction SGD et nous l'avons entraîné avec le Dataset 1. Les métriques obtenues sont représentées par les graphes de la figure II.9.

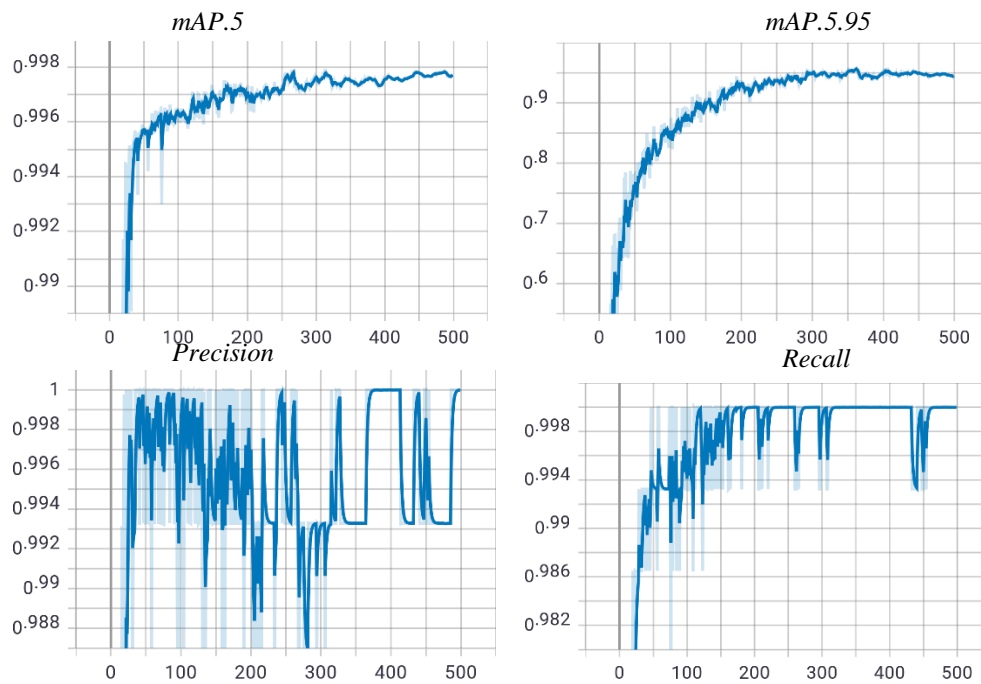


FIGURE II.9 : Les métriques de l'évaluation pour les 500 epochs de l'expérience 2

Après 1h et 25 minutes d'entraînement, le modèle a également atteint un mAP de 99.8%. Les inférences réalisées par ce modèle sont dans la figure II.10.

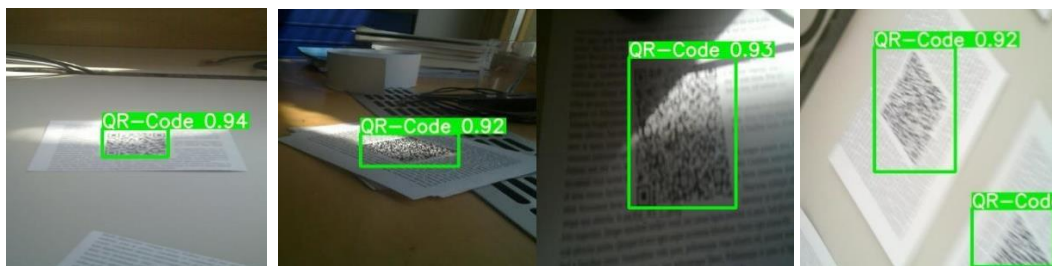
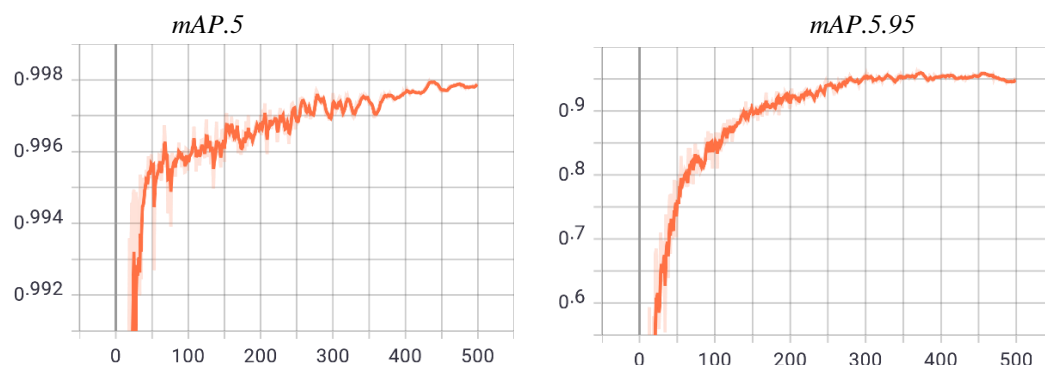


FIGURE II.10 : La détection en utilisant le dataset 1 avec YOLOv5s

## 2.3 Expérience 3 : YOLOv5 Avec du Transfer Learning

Pour l'expérience 3, nous sommes allés chercher s'il y a un « feature extractor » entraîné avec un dataset commun qui peut être utilisé pour l'entraînement de notre détecteur de code QR. C'est pour cela que nous avons opté pour les poids de Darknet 53 entraînés avec ImageNet qui contient plus de 14 millions d'images. Les résultats obtenus après avoir entraîné le modèle en utilisant la technique du Transfer Learning sont représentés dans la figure II.11 et la figure II.12.





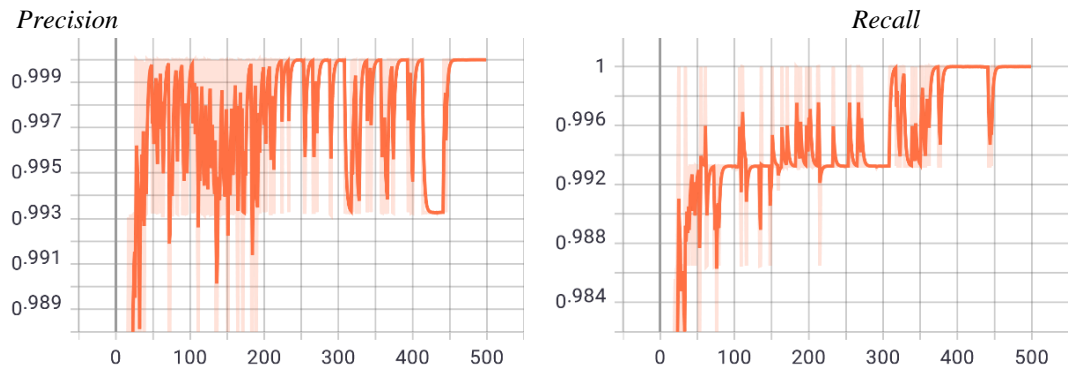


FIGURE II.11 : Les métriques de l'évaluation pour les 500 epochs de l'expérience 3

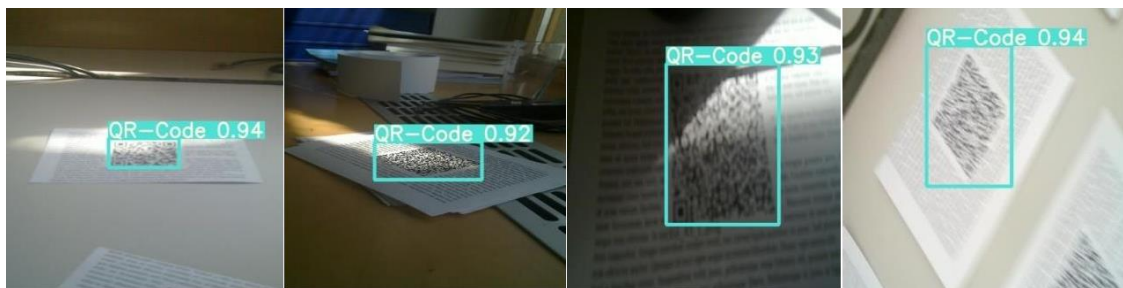
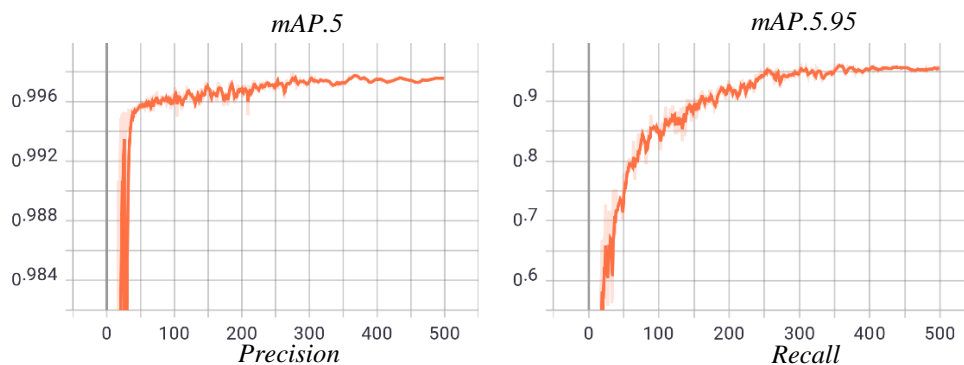


FIGURE II.12 : La détection en utilisant le Transfer Learning avec YOLOv5s

## 2.4 Expérience 4 : YOLOv5 Extra Large

Finalement pour notre dernière expérience, nous avons également gardé les mêmes paramètres et hyperparamètres des expériences précédentes. Nous avons choisi d'entraîner le modèle avec le dataset 2 en utilisant le même fichier des poids à ajuster (Transfer Learning) avec la fonction d'optimisation SGD. Toutefois, nous avons changé la version de YOLOv5 en optant pour celle qui a l'architecture la plus complexe : YOLOv5x. Après 2h et 53 secondes d'entraînement, nous avons obtenu les résultats de la figure II.13 et la figure II.14.





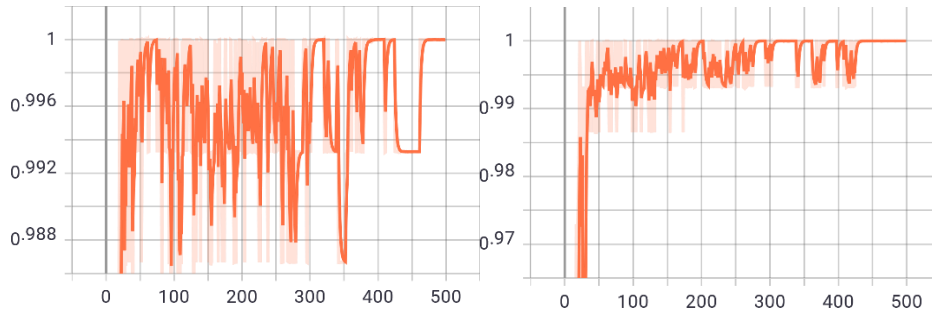


FIGURE II.13: Les métriques de l'évaluation pour les 500 epochs de l'expérience 4

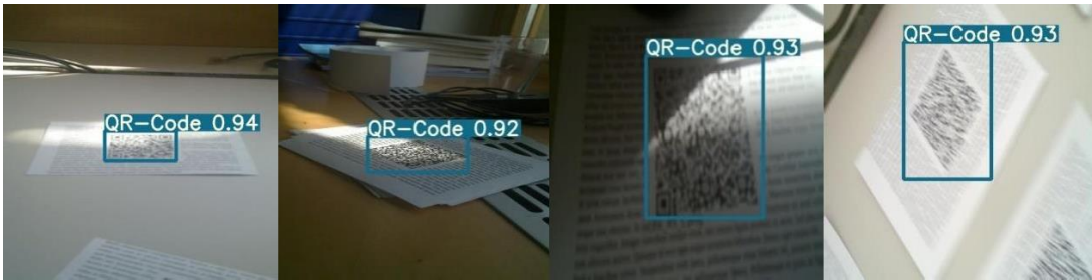


FIGURE II.14: La détection en utilisant le dataset 2 avec YOLOv5x

### 3 Discussion

La table II.1 et la table II.2 résument tous les résultats pertinents que nous avons obtenus 4 expériences effectuées.

Métrique	Expérience 1- SGD	Expérience 1- Adam	Expérience 2	Expérience 3	Expérience 4
mAP.5	0.998 (epoch 215)	0.997 (epoch 266)	0.998 (epoch 167)	0.998 (epoch 255)	0.998 (epoch 253)
mAP.5.95	0.948	0.805	0.958	0.961	0.962
Recall	1	0.993	1	1	1
Precision	1	0.993	1	1	1

Table II.1: Résumé des métriques des expériences

	Expérience 1- SGD	Expérience 1- Adam	Expérience 2	Expérience 3	Expérience 4
Temps d'entraînement	3h 26min 40s	2h 1min 57s	1h 48min 12s	2h 4min 5s	2h 53s
Temps de détection sur image (max)	0.014s	0.012s	0.010s	0.012s	0.016s

Table II.2: Résumé des temps d'entraînement et de détection des expériences

#### 3.1 La détection du code QR

La tâche de détection nécessaire pour décoder le code QR a été réalisée avec des résultats variés. Dans toutes les expériences, nous avons utilisé presque les mêmes paramètres en ne changeant qu'un seul à la fois pour voir comment cela peut influencer sur la précision du modèle. Par exemple dans l'expérience 1, nous avons fait une comparaison entre la détection d'un modèle

entraîné avec la fonction d'optimisation SGD et un modèle entraîné avec Adam. Les résultats ont démontré que le modèle entraîné avec SGD est beaucoup plus performant que celui entraîné avec Adam. Cependant dans l'expérience 2, nous avons changé le dataset en optant pour un autre plus petit auquel nous avons effectué une augmentation. Étonnamment, celui-ci a été plus performant que le premier en termes de mAP.5.95 et de temps de détection en plus du fait qu'il était plus rapide à entraîner et plus rapide à atteindre un mAP.5 de 0.998. Par ailleurs, dans l'expérience 3, nous avons utilisé la technique du « Transfer Learning » en fournissant au modèle des poids à ajuster au lieu de les apprendre à partir de zéro. En faisant ça, le mAP.5.95 a augmenté. Finalement dans l'expérience 4, nous avons opté pour la version la plus complexe de YOLOv5 dans l'espoir d'augmenter la précision du modèle.

Pour conclure cette partie, nous dirons que toutes nos expériences prouvent que le modèle YOLOv5 est effectivement capable d'effectuer les détections correctement.

### **3.2 L'importance des données**

L'un des plus gros problèmes auxquels nous avons été confrontés est l'absence des datasets YOLO volumineux pour faire l'entraînement. Dans l'expérience 1 et 2, nous avons vu qu'un dataset avec des données augmentées peut être aussi performant qu'un dataset avec des données réelles seulement.

Disposer d'un dataset prêt à être entraîné par YOLO était assez difficile. Les deux datasets générés ont consommé pas mal de temps pour faire le processus de l'étiquetage en prenant en considération la durée limitée pour la réalisation du projet. Si on avait pu avoir plus de temps, faire l'étiquetage et l'entraînement avec un dataset suffisamment grand aurait eu de meilleurs résultats.

### **3.3 Le Transfer Learning**

Le principe du « Transfer learning » se repose sur le fait de pouvoir utiliser les connaissances apprises précédemment pour résoudre rapidement les nouveaux problèmes et obtenir de meilleurs résultats.

Utiliser le Transfer Learning a augmenté les performances du modèle en termes de mAP.5.95 de 0.003.

## **Conclusion**

Dans ce chapitre, nous avons mis en œuvre un modèle pour la détection des codes QR. Nous avons commencé par la présentation de la méthode choisie, ensuite la description de l'environnement de l'entraînement du modèle et les différentes expériences et comparaisons pour augmenter la performance de celui-ci. Dans le prochain chapitre, nous allons implémenter le meilleur détecteur et commencer le décodage et la réalisation de l'interface.

# Chapitre III. Conception

## Introduction

L'étude conceptuelle nous permet de visualiser l'aspect statique et dynamique de notre système en fournissant une description globale de sa structure et son fonctionnement. Il est donc devenu possible de définir et recenser toutes les fonctionnalités d'un système grâce à des schémas facilement compréhensibles par l'utilisateur. Dans ce chapitre, nous allons représenter les fonctionnalités de notre outil en utilisant le langage UML.

## 1 UML

*Unified Modeling Language* (UML) est un langage de modélisation graphique universel qui permet au concepteur de représenter un système logiciel à travers des diagrammes. Ces derniers sont une spécification pour le réalisateur qui s'en sert pour la construction d'un système tout en étant une documentation pour l'utilisateur futur. UML modélise les systèmes créés en utilisant une approche orienté objet et permet la visualisation de sa vue statique et dynamique.

## 2 Le diagramme de cas d'utilisation

### 2.1 Définition

Le diagramme de cas d'utilisation est d'un des diagrammes UML qui représentent l'aspect statique d'un système. C'est un diagramme qui nous permet de visualiser les interactions entre les différents utilisateurs et le système ainsi qu'identifier les fonctionnalités proposées par ce dernier dans le but de définir les besoins de l'utilisateur dans la phase d'analyse de la conception.

### 2.2 Le diagramme de cas d'utilisation du système

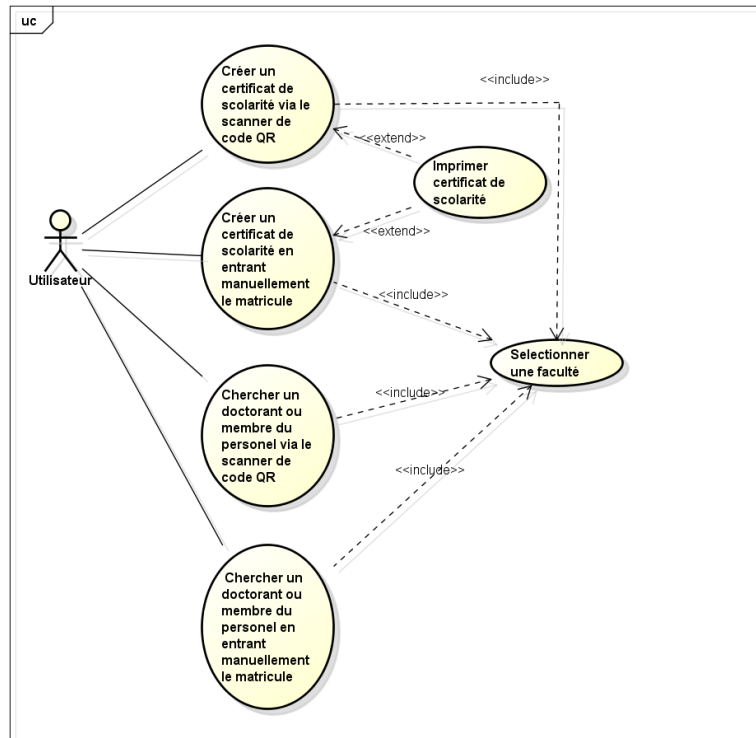


Figure III.1 : Diagramme de cas d'utilisation

## Conclusion

Après avoir étudié l'implémentation de l'architecture Yolov5 et la préparation de données pour la détection, nous avons pu élaborer dans ce chapitre le diagramme de cas d'utilisation d'UML pour représenter globalement l'aspect statique de notre système. Cela nous a permis de dégager les fonctionnalités principales de notre application. Nous verrons dans le dernier chapitre l'implémentation de cette dernière.

# Chapitre III. Réalisation et Tests

## Introduction

Dans le premier et deuxième chapitre, nous avons expliqué les notions de base qui sont nécessaires pour comprendre le processus de la détection, codage et décodage du QR Code. Tout cela a pour but de faciliter la tâche de saisie des matricules des doctorants de notre université. Dans ce chapitre, nous allons expliquer le fonctionnement de l'interface que nous avons mis en œuvre pour faciliter la gestion des certificats de scolarité. La même application sera déployée pour le service du personnel. Nous allons également illustrer les différents scénarios d'utilisation de l'application.

## 1 Les outils utilisés

Pour cette partie, nous avons utilisé deux machines : une pour le développement et une machine pour les tests dont les caractéristiques sont décrites ci-dessous.

### Machine 1 :

- *Modèle : HP Omen.*
- *Processeur : Intel ® Core ™ i7-8750H CPU @ 2.20GHz.*
- *Carte graphique : GeForce GTX 1050TI.*
- *Mémoire installée (RAM) : 8,00 Go.*
- *Édition Windows : Windows 10 Famille.*
- *Type du système : Système d'exploitation 64 bits.*

### Machine 2 :

- *Modèle : HP Probook.*
- *Processeur : Intel ® Core ™ i7-8550U CPU @ 1.80GHz.*
- *Carte graphique : GeForce 930MX.*
- *Mémoire installée (RAM) : 8,00 Go.*
- *Édition Windows : Windows 10 Professionnel.*
- *Type du système : Système d'exploitation 64 bits.*

**Partie Software :** Nous avons utilisé un ensemble d'outils dont les logos sont illustrés par la figure III.1

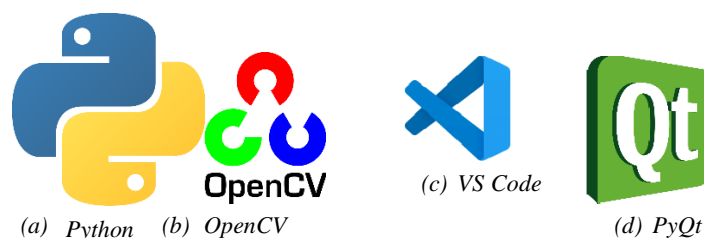




FIGURE III.1: Logos des outils de développement

## 2 Présentation des outils logiciels

Nous présenterons d'abord le langage de développement ainsi que le Framework et les outils utilisés pour concevoir cette interface.

**Python** : Nous l'avons choisi car il est multiplateforme et multiparadigme. Il favorise la programmation orientée objet, fonctionnelle et impérative structurée

**PyQt5** : C'est une librairie permet d'utiliser le Framework Qt GUI à partir de Python. Sachant que Qt lui-même est écrit en C++, en l'utilisant à partir de Python, il est possible de construire des applications beaucoup plus rapidement tout en sacrifiant un petit peu de la vitesse du C++. PyQt5 fait référence à la plus récente version 5 de Qt [Web7] voir figure III.1 (d).

**Visual Studio Code** : C'est un éditeur de code source multiplateforme avec des extensions pour plusieurs langages de programmation, notamment le langage Python [Web8], voir figure II.1 (c).

**Pycharm** : C'est un éditeur de code source spécialement conçu pour les programmeurs en Python. Il facilite la maintenance et l'écriture du code grâce aux contrôles, l'assistance aux tests, aux refactorisations intelligentes et à de nombreuses inspections [Web9].

## 3 Système du codage et décodage

La partie du codage de notre projet consiste à créer un code QR du matricule afin de faciliter l'accès aux informations du doctorant par la suite. Afin de développer cette fonctionnalité, nous avons eu besoin de tester plusieurs fonctions de différentes librairies tel que QrCode [Web10], PyQrCode [Web11], Qrcodegen [web12], et la librairie pour laquelle nous avons finalement opté est Segno.

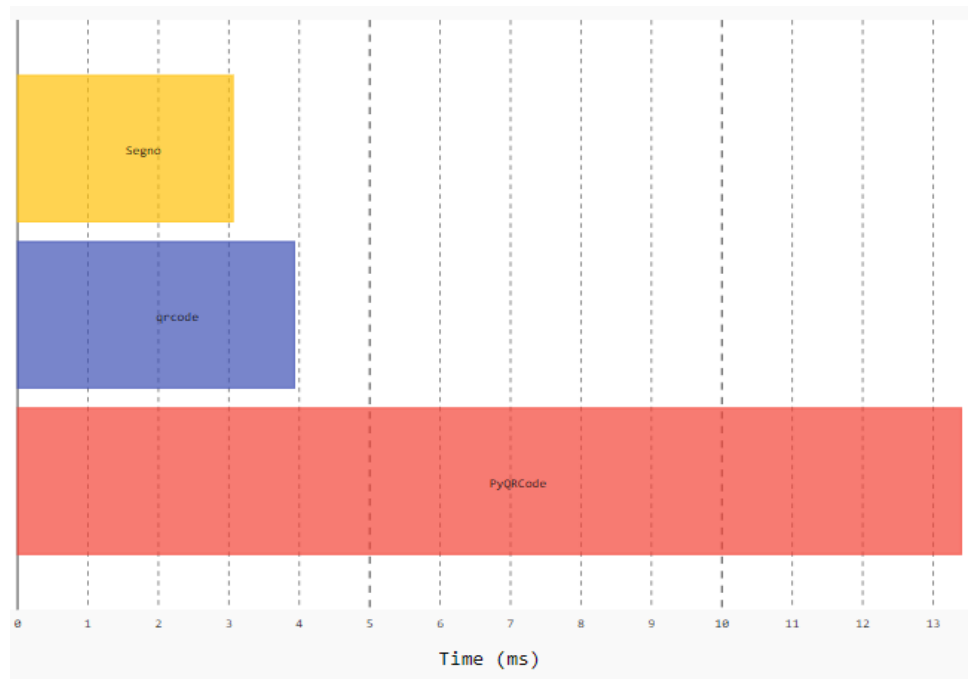


FIGURE III.2 : Comparaison des performances des différentes librairies. [Web14]

**Segno** : Ce module est un générateur de codes QR complètement écrit en python. Il est possible de créer des codes QR, des micros codes QR ainsi que des codes QR colorés. Il possède aussi de nombreuses options de personnalisation du code QR [Web13].

Segno ayant de loin les meilleures performances, nous avons opté pour cette librairie lors de la réalisation de notre projet.

La partie du décodage de notre projet consiste à décoder le code QR conçu au préalable afin d'avoir la nouvelle version du certificat de scolarité en question, ou bien afficher toutes les informations du doctorant. Afin de développer cette fonctionnalité, nous avons également eu besoin de tester plusieurs fonctions de différentes librairies tel que OpenCV [Web16] Python\_Quirc [Web33], PyZXing [Web18], Pyboof [Web17] et pour finir Pyzbar.

**PyZbar** : C'est un module de décodage de codes QR et de codes-barres disponibles en python. Il est utile pour la détection d'un ou plusieurs codes à partir d'une image et le décodage du code QR [Web15].

Afin de sélectionner la librairie que nous allons utiliser dans notre projet nous avons non seulement comparé leurs performances, mais aussi leurs vitesses d'exécution, ce qui nous a donné les résultats de la figure III.3 et la figure III.4

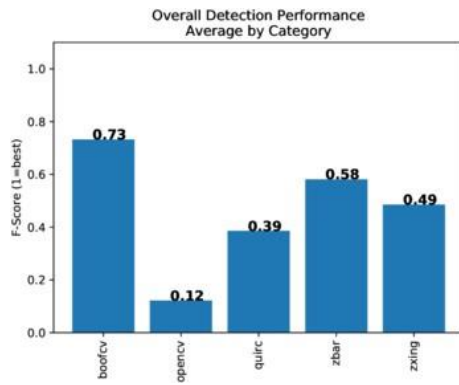


FIGURE III.3 : Comparaison des performances des différentes librairies [Web19]

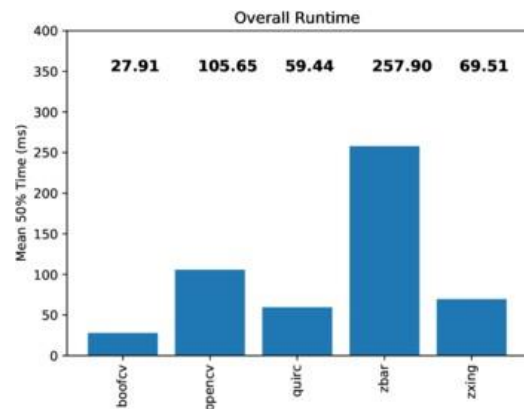


FIGURE III.4 : Comparaisons du temps d'exécution des différentes librairies [Web20]

Après comparaison, il est évident que BoofCV et par extension PyBoof est la plus performante, cependant la nécessité de fournir un environnement java nous a plutôt fait opter pour Pyzbar. Ayant la deuxième performance la plus élevée et n'ayant aucune dépendance (les DLL de zbar étant incluses dans la version Windows de pyzbar) en a fait le meilleur malgré le fait qu'elle soit la librairie avec le temps d'exécution le plus élevé cela reste malgré un temps assez court et ne pose aucun inconvénient lors de l'utilisation pour le décodage en temps réels.

## 4 Présentation de l'outil

Notre outil est une application Desktop qui permet aux employés de la faculté d'électronique et d'informatique de gérer les certificats de scolarités de ses doctorants plus facilement et de l'utiliser dans le service personnel. Tout employé ayant accès à cet outil pourra générer un nouveau certificat de scolarité à l'aide d'un scanner, le visualiser et l'imprimer avec la nouvelle date et le nom de la faculté spécifié. Il est également possible d'afficher toutes les informations du doctorant.

### 4.1 Les fonctionnalités de l'application

#### 4.1.1 Génération et impression des certificats de scolarité

Cette fonctionnalité consiste à générer un certificat de scolarité pour un doctorant cela peut être réalisé de deux manières différentes, soit en rentrant manuellement le matricule du doctorant ainsi que l'année universitaire, ou bien en scannant le code QR du doctorant, dans le cas échéant, l'année universitaire en cours sera prise par défaut. Pour cela nous avons créé une méthode scanner qui analyse en temps réel la vidéo prise par la webcam et en utilisant la fonction *decode()* de *pyzbar* détecte et décode le code QR obtenant le matricule du doctorant et déduit l'année universitaire en cours en se basant sur la date du jour.

Par la suite la méthode *GenCertificat* se lance automatiquement générant le certificat de scolarité en utilisant la librairie *reportlab*.

Avant la génération du certificat une vérification est faite afin de confirmer si le certificat n'a pas déjà été créé au préalable, si cela est le cas le programme ne génère pas un nouveau certificat et se contente d'afficher le certificat déjà existant.

Dans le cas où l'utilisateur n'utilise pas la fonctionnalité scanner, il se doit de rentrer le matricule ainsi que l'année universitaire manuellement dans les champs à cette effigie et appuyer sur le bouton *générer*. Dans ce cas précis le programme vérifie si un code QR contenant le matricule du doctorant n'a pas déjà été créé. Dans le cas contraire un code QR est généré en



utilisant la librairie *segno* et il est par la suite sauvegardé dans le dossier des codes QR.

Une fois cela l'utilisateur peut imprimer le certificat en appuyant sur le bouton *imprimer*.

#### 4.1.2 Recherche des doctorants

Cette fonctionnalité consiste à afficher les informations du doctorant soit entrant son matricule soit manuellement soit en scannant son code QR, cela aura pour effet de chercher le doctorant dans la liste et afficher l'intégralité de ses informations. La méthode *scanner* utilisée est la même que celle décrite dans la partie précédente à l'exception qu'elle exécute par la suite la méthode *AffichInfo* qui récupère l'intégralité des informations du doctorant et les affiche sur l'interface. Le code QR du doctorant est aussi affiché dans le cas où celui-ci n'en possède pas, une image indiquant l'inexistence du code QR est affichée.

Comme dans la fonctionnalité précédente, il est aussi possible de rentrer le matricule manuellement dans le champ spécifié, dans ce cas de figure la méthode « *AffichInfo* » s'exécute de la même manière après que l'utilisateur est appuyé sur le bouton *Chercher*.

## 5 Tests de réalisation

Après avoir fait la description de l'environnement dans lequel nous avons travaillé, Nous allons procéder à la présentation de l'interface de notre outil. Nous allons dans chaque section montrer l'interface et les tests établis sur l'une des fonctionnalités.

### 5.1 Interface d'ouverture

La première interface de notre outil est celle représentée par la figure III.5. Celle-ci contient le logo de l'université ainsi que le nom avec une « *comboBox* » pour choisir la faculté qui va apparaître dans le certificat du doctorant.



FIGURE III.5: Interface d'ouverture

### 5.2 Génération d'un certificat de scolarité

Une fois que l'utilisateur accède à l'outil et choisisse le nom de la faculté à mettre dans le certificat de scolarité (voir figure III.6). Il pourra ensuite générer un certificat en spécifiant la date en premier lieu, ensuite en saisissant le matricule manuellement ou bien à l'aide de la fonction scanner que nous avons conçu. Il pourra ensuite choisir de, soit générer le certificat et le visualiser, soit l'imprimer directement, ou les deux. Un Code QR du matricule sera généré et mis automatiquement dans le certificat pour faciliter l'accès.



FIGURE III.6: Génération d'un certificat de scolarité

### 5.2.1 Scanner du Code QR

Grâce au scanner, l'utilisateur peut éviter de faire la saisie manuelle du QR Code. Le travail du scanner est divisé en deux parties : *La détection et le décodage*.

La partie de la détection consiste à localiser l'emplacement du code QR. Cette dernière fait grâce aux « *Finder Patterns* » qui représentent la caractéristique principale d'un code QR. Il sera donc délimité et prêt pour le décodage.

Après la détection du Code QR vient le décodage. Cette dernière étape consiste à traduire les modules du Code QR en suivant la méthode décrite auparavant pour avoir en sortie le matricule en question. Une fois cela est fait, ce dernier sera saisi automatiquement dans son champ et l'utilisateur est libre de choisir la fonctionnalité qu'il veut par la suite, voir figure III.7



FIGURE III.7: Le Scan du code QR

### 5.2.2 Génération du Code QR

Dans le cas où le code QR n'a pas été généré auparavant, le matricule du doctorant est codé en suivant la méthode décrite auparavant pour avoir en sortie le code QR qui par la suite sera affiché en bas à gauche du certificat de scolarité.

### 5.2.3 Génération du PDF

Une fois les informations du doctorant nécessaires sont récupérées, le certificat est généré et sauvegarder en format PDF, et sera affiché à l'utilisateur qui sera libre de le consulter ou de l'imprimer par la suite, voir figure III.8.

جامعة جوارى بومدين للعلوم والتكنولوجيا  
Université des Sciences et de Technologie Houari Boumediene  
Faculté d'électronique et d'informatique

**CERTIFICAT DE SCOLARITE  
POST-GRADUATION**

Le Doyen de la Faculté d'électronique et d'informatique de l'Université des Sciences  
et de la Technologie Houari Boumediene Certifie que :

**Mr:** ABIDAT Redouane  
**Né le :** 1991-11-27 à : Alger  
**Nationalité :** Algérienne  
**Matricule :** C16026ERER  
**Domaine :** Sciences et Technologies  
**Filière :** Genie electrique  
**Spécialité :** Instrumentation  
est inscrite au titre de l'année universitaire 2020/2021  
En: 5<sup>ème</sup> Année Doctorat.

Bab-Ezzouar, le : 28/05/2021  
Le Doyen

Faculté d'électronique et d'informatique  
USTHB, BP. 32, El Ala, Bab Ezzouar 16111, Alger  
Tel: +213203934056, Fax: +213203934056, email: pghe@usthb.dz

FIGURE III.8: Le certificat de scolarité généré

### 5.2.4 Impression du PDF

Après avoir correctement généré le certificat de scolarité l'utilisateur a la capacité de l'imprimer en appuyant sur le bouton créer à cette occasion, lançant le processus d'impression.

## 5.3 Affichage des informations du doctorant

Après la détection du Code QR vient le décodage. Cette dernière étape consiste à traduire les modules du Code QR en suivant la méthode décrite auparavant pour avoir en sortie le matricule en question. Une fois cela est fait, ce dernier sera saisi automatiquement dans son champ.

Après la saisie du matricule une recherche est effectuée au niveau du fichier data contenant l'intégralité des doctorants, une fois celui-ci trouvé, on procède à la récupération de l'intégralité de ses informations qui seront par la suite affichées au niveau de l'interface graphique, voir figure III.9.

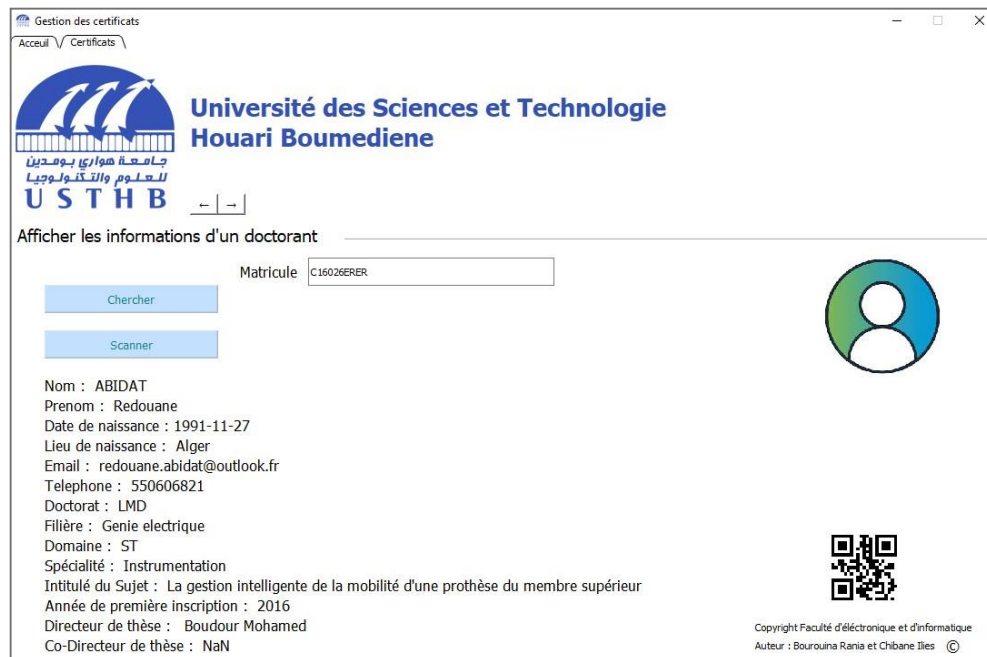


FIGURE III.9: L'affichage des informations du doctorant

## 5.4 Implémentation du modèle dans l'interface

Au niveau de l'implémentation du CNN nous avons utilisé le script `detect.py` [Web34] de la repository de Github ainsi que les dossiers contenant le modèle et le dossier utiles contenant des scripts nécessaires à son fonctionnement.

De nombreuses modifications ont été apportées au script afin d'adapter et de simplifier le code afin de garantir une exécution optimale. La première étape a été de supprimer le

« main » afin de conserver uniquement la fonction « detect », certaines parties du code notamment celles traitant des formats d'input non utilisés par notre application ont aussi été supprimées. Par la suite, quelques modifications ont été faites vers la fin du script. Une fois les coordonnées du code QR récupérées, le script est censé dessiner une bounding box autour du code QR et préciser la confiance du modèle. A la place, nous avons mis un code qui dans le cas où la confiance est supérieure ou égale à 80% le code QR se retrouve découper de l'image en utilisant les coordonnées fournies par le CNN et est envoyé au programme principal en utilisant un return afin de procéder au décodage.

Dans le cas où l'utilisateur quitte le scanner en ayant rien détecter la valeur None sera retournée.

## Conclusion

Dans ce chapitre, nous avons présenté l'environnement de notre travail avec les différents outils, langages et bibliothèques que nous avons utilisées pour réaliser cette interface. Nous avons également présenté le mécanisme et les fonctionnalités en expliquant les étapes de l'utilisation de chacune avec des captures d'écran.

# Conclusion générale

Dans le cadre de ce travail, nous nous sommes intéressés au développement d'un système de codes QR permettant la gestion des doctorants et du personnel de l'université en utilisant une approche neuronale.

À travers une étude de l'état de l'art, nous avons analysé le fonctionnement des codes QR ainsi que des différentes méthodes de détection d'objets utilisant l'apprentissage profond. Puis, en nous basant sur cette étude, nous avons sélectionné l'architecture neuronale la plus adaptée à nos besoins et nous lui avons apporté des modifications afin de créer notre propre détecteur de code QR ayant des performances très satisfaisantes.

En effet, notre détecteur est capable de localiser le code QR très rapidement et dans des conditions non optimales (flou, inclinaison, rotations...etc.).

Les résultats obtenus restent prometteurs et l'apport d'une telle application reste d'un intérêt incontestable apportant ainsi une contribution au développement de la détection d'objet et permettant de mettre en avant cette technologie en l'implémentant dans un cadre aussi important que la gestion universitaire.

## Bibliographie

[Nyon] Marc Meury NYON : Les QR Codes en bibliothèque, un exemple de médiation numérique au service des usagers. Travail Final De Certificat Septembre 2013. Ecole Nationale Supérieure des Sciences de l'Information et des Bibliothèques (ENSSIB).

[Aktaş] Celalettin AKTAŞ: The Evolution and Emergence of QR Codes. Cambridge Scholars Publishing (2017).

[Petrova and al] Dangers Krassie Petrova, Adriana Romanello, B. Dawn Medlin and Sandra A. Vannoy : QR Codes Advantages and Advantages. Department of Computer Information Systems, School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand Department of Management, Universidad Rey Juan Carlos, Madrid, Spain Department of Information Systems, Appalachian State University, Boone, U.S.A. Department of Computer Information Systems, Appalachian State University, Boone, U.S.A. (2016).

[Winter] Mick Winter: Scan Me. Everybody's guide to the magical world of QR Codes. Barcodes, Mobile Devices and Hyperlinking the Real to the Virtual. Westsong Publishing. Napa, California (2010).

[De Almeida] Mickael DE ALMEIDA : XPosé - QR Code. École supérieure d'ingénieurs de Paris-Est, 2012.

[Sharma] Divya Sharma : A Review of QR code Structure for Encryption and Decryption Process. Computer Science, SKIT, Jaipur, Rajasthan, India. Volume 2, Issue 2, February – 2017. International Journal of Innovative Science and Research Technology.

[Soon] Tan Jin Soon : QR Code. Synthesis journal. EPCglobal Singapore Council Chairman, Automatic Data Capture Technical Committee (2008).

[Chang] Jae Hwa chang : An introduction to using QR codes in scholarly journals. Application of advanced information technology to scholarly journal publishing during the 12th EASE General Assembly and Conference (2014) .InfoLumi, Seongnam, Korea

[Viola and al] Paul Viola and Michael Jones : Boosting: Foundations and Algorithms. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, volume 1, pages I{I. IEEE, 2001.

[Girshick] Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5) Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik UC Berkeley 22 Octobre 2014.

[van de Sande† and al] Selective Search for Object Recognition J.R.R. Uijlings\*1,2, K.E.A. van de Sande†2, T. Gevers2 , and A.W.M. Smeulders2 1University of Trento, Italy 2University of Amsterdam, the Netherlands Technical Report 2012, submitted to IJCV (2013).

[Girshick] Fast R-CNN Ross Girshick Microsoft Research 27 Septembre 2015.

[Girshick and al] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun 6 Janvier 2016.

[Girshick and al] Mask R-CNN Kaiming He Georgia Gkioxari Piotr Doll'ar Ross Girshick Facebook AI Research (FAIR) 24 Janvier 2018.

[Anguelov and al] SSD : Single Shot MultiBox Detector Wei Liu1 , Dragomir Anguelov2 , Dumitru Erhan3 , Christian Szegedy3 , Scott Reed4 , Cheng-Yang Fu1 , Alexander C. Berg1 1UNC Chapel Hill 2Zoox Inc. 3Google Inc. 4University of Michigan, Ann-Arbor 29

Décembre 2016.

[Redmon] YOLOv3 : An Incremental Improvement Joseph Redmon Ali Farhadi 8 Avril 2018.

[Bochkovskiy and al] YOLOv4: Optimal Speed and Accuracy of Object Detection Alexey Bochkovskiy Chien-Yao Wang Hong-Yuan Mark Liao 23 Avril 2020.

[Redmon and al] You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon<sup>1</sup>, Santosh Divvala<sup>2</sup>, Ross Girshick, Ali Farhadi<sup>3</sup> University of Washington, Allen Institute for AIy, Facebook AI Research.

[Wang and al] CSPNet: A New Backbone that can Enhance Learning Capability of CNN. Conference: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). June 2020.

[Do Thuan] Evolution of YOLO Algorithm and YOLOv5: The state-of-the-Art Object Detection Algorithm. Do Thuan. Oulu University of Applied Sciences.

[Favorskaya] THE STUDY OF ACTIVATION FUNCTIONS IN DEEP LEARNING FOR PEDESTRIAN DETECTION AND TRACKING. Reshetnev Siberian State University of Science and Technology, Institute of Informatics and Telecommunications, 31, Krasnoyarsky Rabochy ave., Krasnoyarsk, 660037 Russian Federation.

## Webographie

- [Web1] DENSO WAVE. Qrcode.com, denso wave. <https://www.qrcode.com/en/>, 2020.
- [Web2] Le code QR de [www.futura-sciences.com](https://www.futura-sciences.com). <https://www.futura-sciences.com/tech/definitions/informatique-code-qr-11969/>, 2018
- [Web3] <https://connect.ed-diamond.com/GNU-Linux-Magazine/GLMF-194/Decoder-un-code-QR>, 2016
- [Web4] <https://developpaper.com/qr-code-details-2/>, 2020
- [Web5] <https://merricx.github.io/qrazybox/help/getting-started/about-qr-code.html>
- [Web6] Deep object detectors. <https://www.slideshare.net/IldooKim/deep-object-detectors-1-20166>, 2017
- [Web7] PyQt5 5.15.4. <https://pypi.org/project/PyQt5/>, 2018
- [Web8] Visual Studio Code. <https://code.visualstudio.com/>, 2015
- [Web9] JetBrains, Pycharm. <https://www.jetbrains.com/fr-fr/pycharm/>, 2000
- [Web10] QR Code. <https://github.com/endroid/qr-code>, 2017
- [Web11] pyqrcode. <https://github.com/mnooner256/pyqrcode>, 2016
- [Web12] Qrcodegen <https://www.nayuki.io/page/qr-code-generator-library>, 2021
- [Web13] Segno <https://segno.readthedocs.io/en/latest/>, 2016
- [Web14] Create a QR code and serialize it as PNG  
<https://segno.readthedocs.io/en/stable/comparison-qrcode-libs.html#create-a-qr-code-and-serialize-it-as-png>, 2021
- [Web15] pyzbar 0.1.8. <https://pypi.org/project/pyzbar/>, 2019
- [Web16] OpenCV. <https://opencv.org/>, 2021
- [Web17] PyBoof <https://github.com/lessthanoptimal/PyBoof>, 2015
- [Web18] PyZXing <https://github.com/ChenjieXu/pyzxing>, 2020
- [Web19] Detection Results  
[https://boofcv.org/index.php?title=File:Qrcode\\_detection\\_summary.png](https://boofcv.org/index.php?title=File:Qrcode_detection_summary.png), 2019
- [Web20] Runtime Results [https://boofcv.org/index.php?title=File:Qrcode\\_runtime\\_summary.png](https://boofcv.org/index.php?title=File:Qrcode_runtime_summary.png), 2019
- [Web21] YOLO V5 — Explained and Demystified. <https://pub.towardsai.net/yolo-v5-explained-and-demystified-4e4719891d69>, 2020
- [Web22] Jonathan Hui. YOLOv4. <https://jonathan-hui.medium.com/yolov4-c9901eaa8e61>, 2020
- [Web23] Stochastic Gradient Descent. <https://scikit-learn.org/stable/modules/sgd.html>, 2007
- [Web24] YOLOv5 Github repository. <https://github.com/ultralytics/yolov5>, 2020
- [Web25] Roboflow. <https://roboflow.com/>, 2021



- [Web26] Google Colaboratory.  
[https://colab.research.google.com/notebooks/intro.ipynb?hl=fr#scrollTo=5fCEDCU\\_qrC0](https://colab.research.google.com/notebooks/intro.ipynb?hl=fr#scrollTo=5fCEDCU_qrC0) , 2018
- [Web27] Pytorch. <https://pytorch.org/> , 2016
- [Web28] TensorBoard : le kit de visualisation de TensorFlow.  
<https://www.tensorflow.org/tensorboard?hl=fr> , 2016
- [Web29] Real-time precise detection of regular grids and matrix codes.  
[http://www.fit.vutbr.cz/research/groups/graph/pclines/pub\\_page.php?%20id=2012-JRTIP-MatrixCode](http://www.fit.vutbr.cz/research/groups/graph/pclines/pub_page.php?%20id=2012-JRTIP-MatrixCode) , 2014
- [Web30] YOLO-QR-labeled. <https://www.kaggle.com/hamidl/yoloqrlabeled> , 2020
- [Web31] <https://www.programmingsought.com/article/96534106915/> , 2019
- [Web32] <https://www.qrcode.com/en/howto/generate.html> , 2019
- [Web33] <https://python-quirc.readthedocs.io/en/latest/>
- [Web34] <https://github.com/ultralytics/yolov5>

## Résumé

La quantité des données ne cesse d'augmenter au fil des années, notamment dans administrations des établissements où il faut garder trace de toutes ses ressources humaines.

Le but de notre projet est de faciliter la gestion des données des doctorants et du personnel en mettant en œuvre un outil qui permet de faire cette tâche facilement.

En plus de pouvoir accéder aux données en utilisant des codes QR, ces derniers sont détectés avec une nouvelle méthode que nous avons déployée en utilisant une approche neuronale.