RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÉRE DE L'ENSEIGNEMENT SUPÉRIEUR ET LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ MOHAMMED SEDDIK BENYAHIA
JIJEL
FACULTÉ DE SCIENCES EXACTES ET INFORMATIQUE

Mémoire de fin d'études

Présenté pour l'obtention du diplôme de

MASTER

EN INFORMATIQUE

Option : INTELLIGENCE ARTIFICIELLE

Thème

Systéme basé sur l'apprentissage profond pour la gestion des tours dans les administrations Algériennes

Réalisé par
Kihal Merouane
Hafsaoui Abderraouf

Encadré par
Dr.Zennir Mohammed Nadjib

Promotion 2020

Remerciements

Louange à **Allah** le Tout Puissant Qui nous a accordé la foi, le courage et la patience pour mener ce travail.

Le travail présenté dans ce manuscrit n'aurait pu être achevé sans le support constant d'un grand nombre de personnes. Nous tenons, en ces quelques lignes à leur adresser nos plus sincères remerciements.

Nous tenons à remercier infiniment **Dr. ZENNIR Mohammed Nadjib** pour sa patience et ses conseils précieux qui nous ont été très utiles. Permettez-nous Monsieur de vous exprimer nos reconnaissances et nos remerciements les plus sincères.

Nous remercions également, **les membres de jury** qui nous ont honorés en acceptant de juger et d'enrichir ce modeste travail.

Nous remercions aussi, vivement le PAPC de Jijel et les responsables des annexes, pour nous avoir autoriser à manipuler les plaques de tours, se trouvant au niveaux des différentes annexes de la commune de Jijel.

Nous adressons nos vifs remerciements à nos enseignants du département d'informatique de l'université de Jijel.

Nous ne pouvons oublier d'adresser un grand remerciement à **nos familles**, et particulièrement **nos parents**, pour leur soutien et leurs encouragements. Tous les mots ne sauraient exprimer notre gratitude et notre reconnaissance. Puissiez-vous, trouver ici, le fruit de vos efforts, vos dévouements et l'expression de notre profond amour.

Dédicace

J'ai le grand plaisir de dédier ce modeste travail

A mes **chers parents**, qui m'ont toujours accompagné moralement. Je les remercie pour l'éducation, soin, encouragements et conseils qu'ils m'ont prodigués.

A la mémoire de **mes grands parents**, j'aurais souhaité votre présence pour partager ma joie.

A **ma grand-mère**, qui n'a jamais cessé de prier pour moi.

A toute **ma famille**.

A mon cher ami **MAROUANE**.

A tous **mes amis**,

Abderraouf

Dédicace

J'ai le grand plaisir de dédier cet humble travail

A mes **chers parents**, pour tous leurs sacrifices, leurs amours, leurs soutien et leurs prières tout au long de mes études

A mes **chers frères**, pour leurs appui et leurs encouragement.

A toute **ma famille**.

A toutes les personnes qui m'ont aider durant mon cursus universitaire.

A mon cher ami **ABDERRAOUF**.

A tous **mes amis**,

Marouane

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

IA : Intelligence Artificielle

MLP : Multi Layer Perceptron

ReLu : Rectifier Linear Unit

DL : Deep Learning

ML : Machine Learning

CNN : Convolutional Neural Network

RNN : Recurrent Neural Network

LSTM :Long Short Term Memory

SGD : Stochastic Gradient Descent

ADAM : Adaptative Moment Estimation

Introduction Générale

L'intelligence artificielle est un sujet d'actualité qui se démocratise de plus en plus et qui envahi notre quotidien. Elle est souvent abrégé par « IA » (AI en anglais), et elle est définie par Marvin Minsky, l'un de ces créateur comme : "La construction d'un programme informatique qui se donne à des tâches qui sont, pour l'instant accomplies de façon plus satisfaisante par des êtres humains, car elle demande des processus mentaux de haut niveau"

L'un des sous domaines vedette de l'IA est l'apprentissage profond (Deep Learning en anglais DL), qui est une branche de l'apprentissage automatique, entièrement basé sur les réseaux de neurones artificiels qui permettent d'imiter le cerveau humain.

Dans notre travail, nous avons utilisé des techniques de l'apprentissage profond dans le cadre de la reconnaissance des chiffres, spécialement, la reconnaissance des chiffres des compteurs électroniques des tours, présents dans les mairies.

Cette plaque de compteur a pour but d'organiser les tours au niveau des mairies et des administrations Algériennes. Elle affiche le numéro du tour actuel sous le format sept segments. Le problème délicat qui se pose ici, est la surcharge des personnes au niveau des files d'attentes notamment, lors de cette pandémie de COVID 19.

Notre mémoire est constitué de cinq chapitres : Le premier présente la théorie des réseaux de neurones, il explique les notions de neurone comme élément essentiel des réseaux et présente l'algorithme de propagation et rétropropagation du gradient.

Le deuxième est consacré aux notions fondamentales de l'apprentissage profond et

présente les types de réseaux de neurones profonds. Nous nous attarderons sur les réseaux de neurones convolutionnels CNN.

Dans le troisième chapitre, nous présenterons de manière générale l'état de l'art de la reconnaissance des chiffres . Cependant, faute de travaux antérieurs concernant notre problématique, nous avons opté de nous focaliser sur des études qui traitent des problématiques assez proches de la nôtre.

Dans le quatrième chapitre, nous allons détailler notre contribution pour la gestion des tours dans les administrations Algériennes.

Dans le dernier chapitre, nous allons présenter les différentes expérimentations paramétriques effectuées et les résultats obtenus, tout en montrant les interfaces de l'application.

Nous terminons notre mémoire par une conclusion générale sur notre projet de fin d'étude, en montrant les limites et les perspectives possibles.

Chapitre 1

Réseaux de neurones artificiels

1.1 Introduction

Dans ce premier chapitre, nous commençons par introduire les réseaux de neurones artificiels, qui sont la base de l'apprentissage profond, en donnant différentes notions de base et la méthode de leur mise en oeuvre dans le procédé d'apprentissage.

1.2 Neurone biologique

Dans l'antiquité, il n'était pas évident que le cerveau constituait l'organe essentiel de la pensée. Ainsi, pour Aristote « le coeur est au centre des processus sensitifs, le cerveau ayant alors un simple rôle de réfrigération de ce dernier »[1]

Au début du XX^{eme} siècle, Santiago Ramon y Cajal, neuro-scientifique espagnol, met en évidence que les cellules du cerveau sont des entités fonctionnelles autonomes connectées entre elles.

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites.[2]

C'est par les dendrites que l'information est acheminée de l'extérieur vers le noyau du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone pour être transmise aux autres neurones. La jonction entre deux neurones est appelée la synapse.

1.2.1 Composants d'un neurone biologique

Le tableau (1.1) illustre les composants d'un neurone biologique ainsi que leur rôle.

Composant	**Rôle**
Corps cellulaire	intégration des messages nerveux
Dendrites	réception des messages nerveux
Axone	conduction des messages nerveux
Arborisation terminale	transmission des messages nerveux à un autre neurone

Table 1.1: Composants d'un neurone biologique

La figure (1.1) représente un schéma d'un neurone biologique[1].

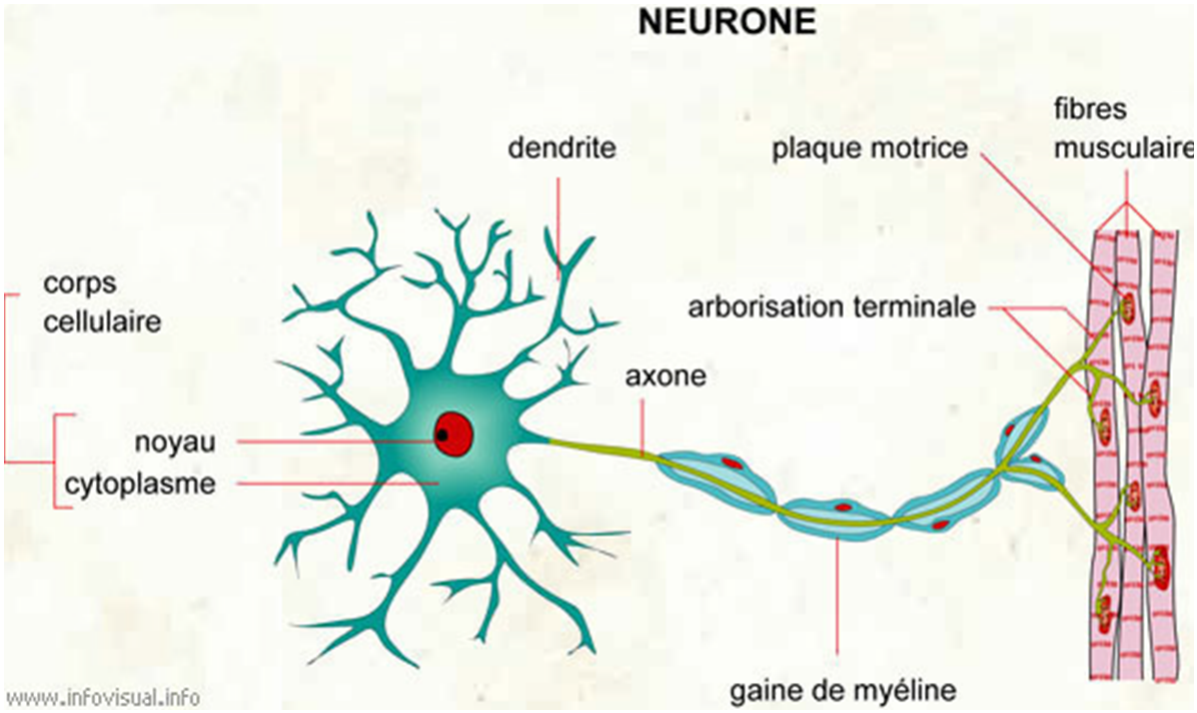


FIGURE 1.1: Schéma du neurone biologique

1.3 NEURONE FORMEL

C'est en 1943 que Mc Culloch (un neurophysicien) et Pitts (logicien) ont proposé les premières notions de neurone formel binaire. Ce concept fut ensuite, mis en réseau avec une couche d'entrée et une sortie par Rosenblatt en 1959, pour simuler le fonctionnement rétinien et tacher de reconnaître des formes. C'est l'origine du perceptron [3].

Cette approche, dite connexionniste, a atteint ses limites technologiques, compte tenu de la puissance de calcul de l'époque.

Un neurone formel est une représentation mathématique et informatique d'un neurone biologique. Le neurone formel possède généralement, plusieurs entrées et une seule sortie, qui correspondent respectivement, aux dendrites et au cône d'émergence du neurone biologique.[4] De manière schématique, un neurone reçoit des signaux entrant via les dendrites et envoie le signal sortant via l'axone. Cela ressemble à une fonction mathématique

1. https://www.infovisual.info/fr/corps-humain/neurone

à plusieurs variables f(x1,x2,x3,xn) avec des coefficients synaptiques (poids), il effectue une sommation pondérée via les poids et déclenche un signal de sortie y tel que :

$$y = w1.x1 + w2.x2 + w3.x3 + ... + wn.xn + b$$

la sortie finale est f(y) tel que f s'appelle la fonction d'activation. L'équation (1.1) illustre l'approximation mathématique du neurone et la figure (1.2) représente un schéma d'un neurone formel.

$$s = \varphi(\sum_{i=0}^{i=p} w_i e_i) \tag{1.1}$$

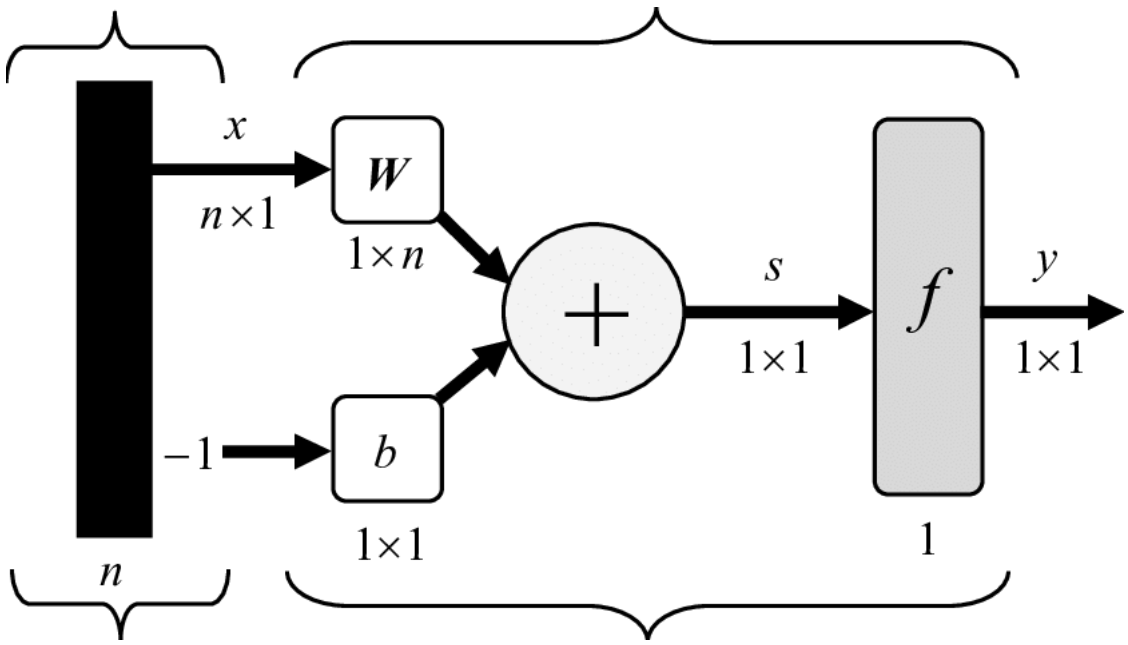


FIGURE 1.2: Schéma du neurone formel [5]

1.3.1 Composants du neurone formel

Dans cette section nous allons présenter les principaux composants d'un neurone formel

- **Entrées :** les entrées peuvent provenir directement du système ou d'autres neurones.
- **Biais :** permet d'ajouter de la flexibilité au réseau, en permettant de varier le seuil de déclenchement du neurone.
- **Poids :** Facteurs multiplicateurs qui affectent l'influence de chaque entrée sur la sortie du neurone.

- **Noyau :** Intègre toutes les entrées et le biais et calcule la sortie du neurone selon une fonction d'activation.
- **Sortie :** Peut être soit, directement une des sorties du système ou distribuée vers d'autres neurones

1.3.2 A quoi sert un perceptron

Le perceptron simple sert comme classifieur linéaire binaire, si le problème est linéairement séparable. Il permet de classifier l'ensemble de données d'entrée en deux différentes classes [6].

Exemple :

$$f(x) = \begin{cases} +1 & si \quad < w, x > +b > 0 \\ -1 & sinon \end{cases} \tag{1.2}$$

ou :

w : le poids synaptique du neurone.

x : l'entrée du neurone.

b : Le biaie du neurone.

1.3.3 Interprétation géométrique

<w,x>+b=0 est l'équation d'un hyperplan qui sépare X en deux demi-espaces correspondants aux deux classes.

Exemple du AND

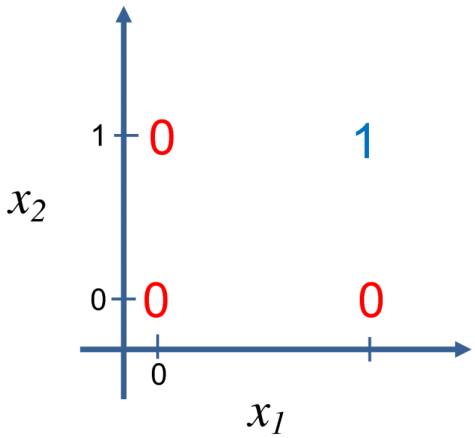


FIGURE 1.3: Schéma de l'opérateur AND [7]

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 1.2: Table de vérité AND

D'après la figure (1.3), il est clair qu'on peut séparer l'ensemble en deux classes par un seul hyperplan, donc on peut déduire que l'opérateur "And" est linéairement séparable, alors il peut être discrétisé par un perceptron linéaire à seuil.

Exemple du XOR

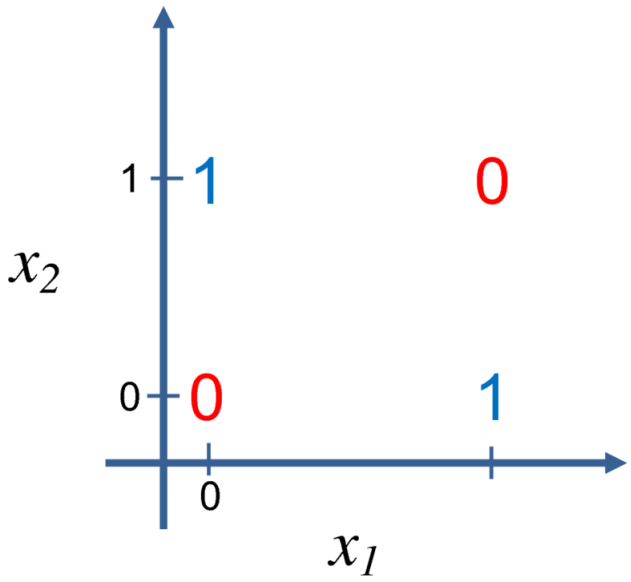


FIGURE 1.4: Schéma de l'opérateur Xor [7]

A	B	A Xor B
0	0	0
0	1	1
1	0	1
1	1	0

TABLE 1.3: Table de vérité Xor

On remarque que le Xor est non linéairement séparable car il est impossible de séparer les deux classes par un tenseur de dimension 1 (une droite). C'est cette limite du perceptron simple qui donne lieu au perceptron multicouches (MLP) qui fera l'objet de la section suivante.

1.4 PERCEPTRON MULTICOUCHE

En anglais « multi layer perceptron MLP» est un ensemble de neurones artificiels, organisés en plusieurs couches successives, au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement. Il s'agit donc d'un réseau de type feedforward [8], chaque couche est constituée d'un nombre variable de neurones, les neurones de la couche de sortie correspondent toujours aux sorties du système. La couche d'entrée reçoit les signaux (ou variables) d'entrée et la couche de sortie fournit les résultats.

Enfin, les neurones des autres couches (couches cachées) n'ont aucun lien avec l'extérieur et sont appelés neurones cachés [9]. La figure (1.5) représente le schéma d'un perceptron multi couches.

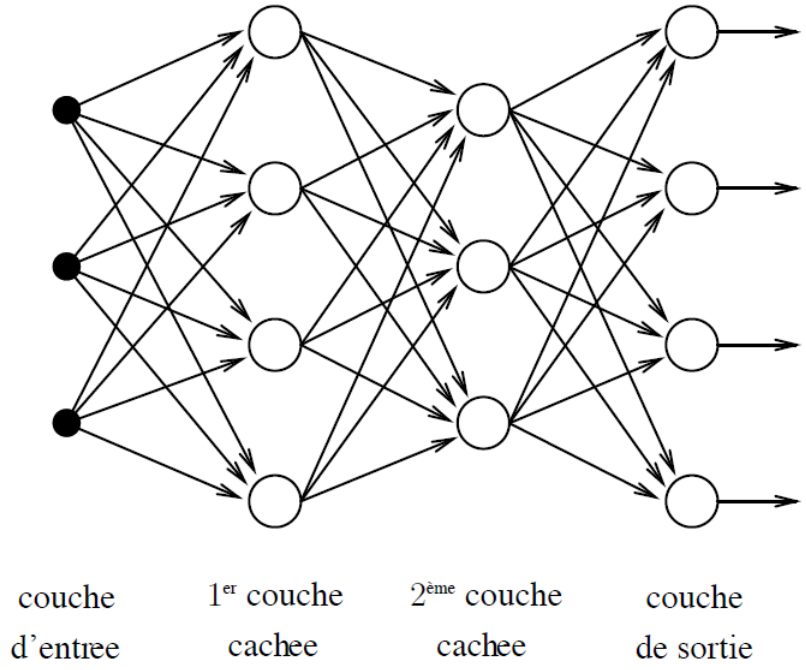


FIGURE 1.5: Schéma MLP [9]

1.5 FONCTION D'ACTIVATION

La fonction d'activation est une composante élémentaire des réseaux de neurones, elle prend en entrée la valeur du produit scalaire et le biais, et elle renvoie la valeur de la sortie. Les différents types de neurones se distinguent par la nature de leur fonction d'activation, les principaux types de ces fonctions sont :

- **Identité** :

$$f(x) = x \tag{1.3}$$

- **Seuil** :

$$f(x) = \begin{cases} 1 & si \quad x > 0 \\ 0 & si \quad x \leq 0 \end{cases} \tag{1.4}$$

- **Sigmoïde** :

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.5}$$

- **Tangente Hyperbolique** :

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{1.6}$$

– **ReLu** :

$$f(x) = \begin{cases} 0 & si \quad x < 0 \\ x & si \quad x \geq 0 \end{cases} \tag{1.7}$$

1.6 La propagation et la rétropropagation du gradient

C'est un algorithme qui permet de stabiliser un réseau à plusieurs couches. Cet algorithme a été trouvé, de façon indépendante, par plusieurs équipes des chercheurs (Rumelhart, Parker et LeCun). La topologie d'un tel réseau est donc formée de plusieurs couches de neurones, sans communication à l'intérieur d'une même couche. Le but est encore de minimiser l'erreur quadratique entre les sorties obtenues et celles souhaitées, qui correspond de nouveau à une descente du gradient.

Le principe est de redistribuer sur toutes les couches, y compris les couches cachées, une partie de l'erreur de manière récursive, en partant de la couche de sortie et en remontant vers la couche d'entrée [10]. La figure (1.6) montre le gradient de l'erreur totale du réseau.

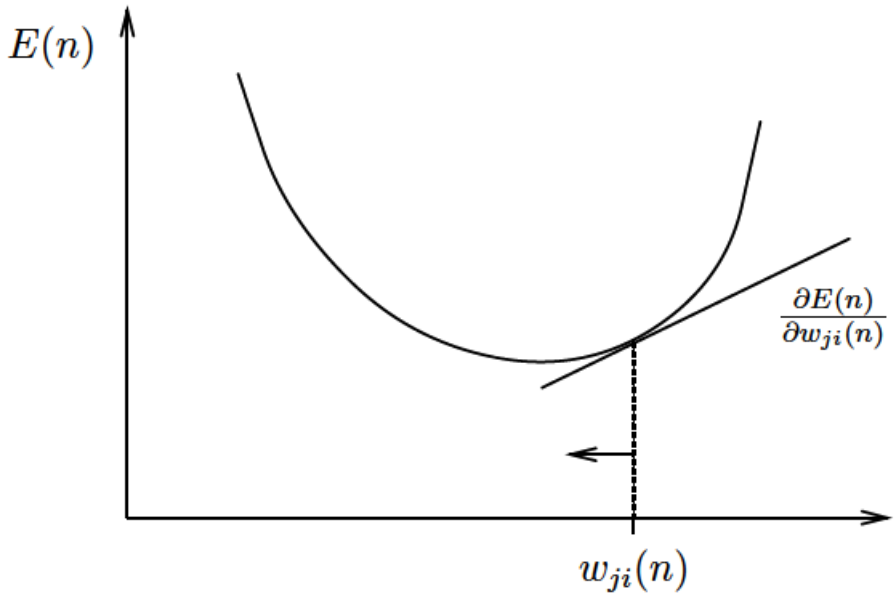


Figure 1.6: Gradient de l'erreur totale du réseau [10]

1.6.1 Formalisation de l'apprentissage

Pour un réseau multi-couches à n sorties, composé de plusieurs couches (couches cachées et couche de sortie), la somme de la j-éme unité est donnée par [10] :

$$S_j = \sum_{j=0}^{i=n} W_{ji} X_i + \theta_j \tag{1.8}$$

$$I_j = f\left(S_j\right) = f\left(\sum_{j=0}^{i=n} W_{ji} X_i + \theta_j\right) \tag{1.9}$$

ou :
W_i : poids de la connexion i^{me} entrée.
e_j : bais du neurone j
F : fonction d'activation de ce neurone j

L'objectif de la rétro-propagation est l'adaptation des paramètres W_{ij}, de telle façon à minimiser la fonction du coût donnée par :

$$E_p = \frac{1}{2} \sum_{k=1}^{T} E = \frac{1}{2} \sum_{k=1}^{T} (\delta)^2 \tag{1.10}$$

$$\delta_k = Y_k - R_k \tag{1.11}$$

Avec :
E_p : somme de l'erreur quadratique moyenne de moindres carrés
R_K : sortie actuelle du réseau
y_K : sortie désirée
T : longueur de l'ensemble d'apprentissage

1.6.2 Ajustement des poids

Après avoir calculer la sortie R_k et l'erreur E correspondantes à l'ensemble des entrées, à partir de l'équation associée, les poids du réseau sont alors ajustés par la méthode du gradient [10] :

$$W_{ji}^L(n+1) = W_{ji}^L(n) + \Delta W_{ji}^L(n) \tag{1.12}$$

$$\Delta W_{ji}^{L}(n) = -\mu \frac{\delta E}{\delta W_{ji}^{L}(n)} \tag{1.13}$$

Avec :
μ : pas d'apprentissage représentant la vitesse de convergence, dont la valeur est généralement choisie expérimentalement ($0 < \mu < 1$)
Si μ est trop petit, la convergence est lente mais la direction de la descente est optimale.
Si μ est trop grand, la convergence est rapide mais la précision est médiocre.

L'application des poids du réseau est faite, tout d'abord, pour la couche de sortie puis pour les couches cachées
on a :

- Pour la couche de sortie :

$$W_{kj}^{R}(n+1) = W_{kj}^{R}(n) + \mu \delta_{k}^{R} I_{j} \tag{1.14}$$

- Pour la couche cachée :
  On note que chaque adaptation des poids dans la couche cachée dépend de l'erreur totale de la couche de sortie, ce qui conduit à la notion de rétro-propagation.
  L'équation d'adaptation des poids dans ce cas est :

$$W_{ji}^{l}(n+1) = W_{ji}^{i}(n) + \mu \delta_{j}^{l} X_{i} \tag{1.15}$$

Toutes ces étapes sont résumées dans le pseudo-algorithme

Algorithm 1 propagation et rétropropagation

1-Initialiser les poids W_{ij} et les biais des neurones à des petites valeurs aléatoires.
2-Présenter le vecteur d'entrée et de sortie désirés.
3-Calculer La somme des entrées des neurones de la couche cachée
3-Calculer Les sorties de neurones de la couche cachée
4-Calculer La somme des entrées des neurones de la couche de sortie
5-Calculer Les sorties des réseaux
6-Calculer l'erreur pour les neurones de la couche de sortie
7-Réinjecter l'erreur de sortie
8-Ajuster Les poids de la couche de sortie
9-Ajuster Les poids de la couche cachée
10-Calculer l'erreur E
if $E - E_p < \varepsilon$ **then**
  FIN
else
  étape 6
end if

1.7 Conclusion

Dans ce chapitre, les principales notions des réseaux de neurones et leur apprentissage ont été présentés. En effet, le neurone n'est rien d'autres qu'une approximation formelle d'un neurone biologique. Le perceptron multi-couches est très efficace pour quelques problèmes. En revanche, il est limité pour d'autres.

Nous avons abordé, également, les principaux types de fonctions d'activation, les plus utilisées. Puis, nous avons expliqué les démarches de l'algorithme de propagation et rétropropagation du gradient et comment il corrige les poids synaptiques, lors de la phase de l'apprentissage du réseau de neurones.

Le chapitre suivant sera consacré à une description de l'apprentissage profond (Deep Learning en Anglais) et les différents types de réseaux de neurones profonds.

Chapitre 2

Apprentissage Profond

2.1 Introduction

Dans ce chapitre, nous présenterons l'apprentissage profond et les réseaux de neurones profonds, c'est-à-dire des réseaux de neurones avec plusieurs couches cachées. Ils sont capables d'apprendre des abstractions de caractéristiques des exemples d'entrée, de comprendre les caractéristiques de base des exemples et de faire des prédictions basées sur ces caractéristiques.

2.2 Apprentissage profond

Selon les fondateurs Yann LeCun, Yoshua Bengio Geoffrey Hinton : "L'apprentissage profond permet aux modèles informatiques, composés de plusieurs couches de traitement, d'apprendre des représentations de données avec plusieurs niveaux d'abstraction."[11]

Une autre définition des auteurs :"L'apprentissage profond est une classe de techniques d'apprentissage automatique, où l'information est traitée en couches hiérarchiques pour comprendre les représentations et les caractéristiques des données dans des niveaux de complexité croissants."[12]

En d'autres termes, Deep Learning est un sous-ensemble des méthodologies et techniques de Machine Learning (ML) qui utilisent le réseau neuronal artificiel. C'est l'adaptation des réseaux neuronaux qui imite la structure du cerveau humain [12].
La figure (2.1) illustre le contexte du DL par rapport à l'IA.

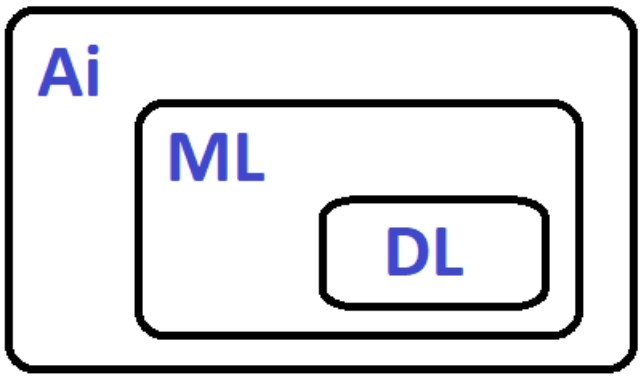


FIGURE 2.1: Contexte de l'apprentissage profond

2.2.1 Comparaison entre l'apprentissage profond et l'apprentissage automatique

Dans cette section nous allons présenter la différence majeur entre le DL et ML,qui est l'intervention humaine. Les systèmes d'apprentissage automatique (Machine Learning) ont besoin de l'intervention humaine pour identifier et coder manuellement les fonctionnalités et les caractéristiques, en fonction du type de données. Par contre, un système de l'apprentissage profond (Deep Learning) tente d'apprendre ces fonctionnalités, sans une intervention humaine supplémentaire. Les quantités de données impliquées dans ce processus sont énormes. Au fur et à mesure que l'entraînement s'enchaine, la probabilité des réponses correctes augmente [13]. La figure (2.2) illustre cette différence.

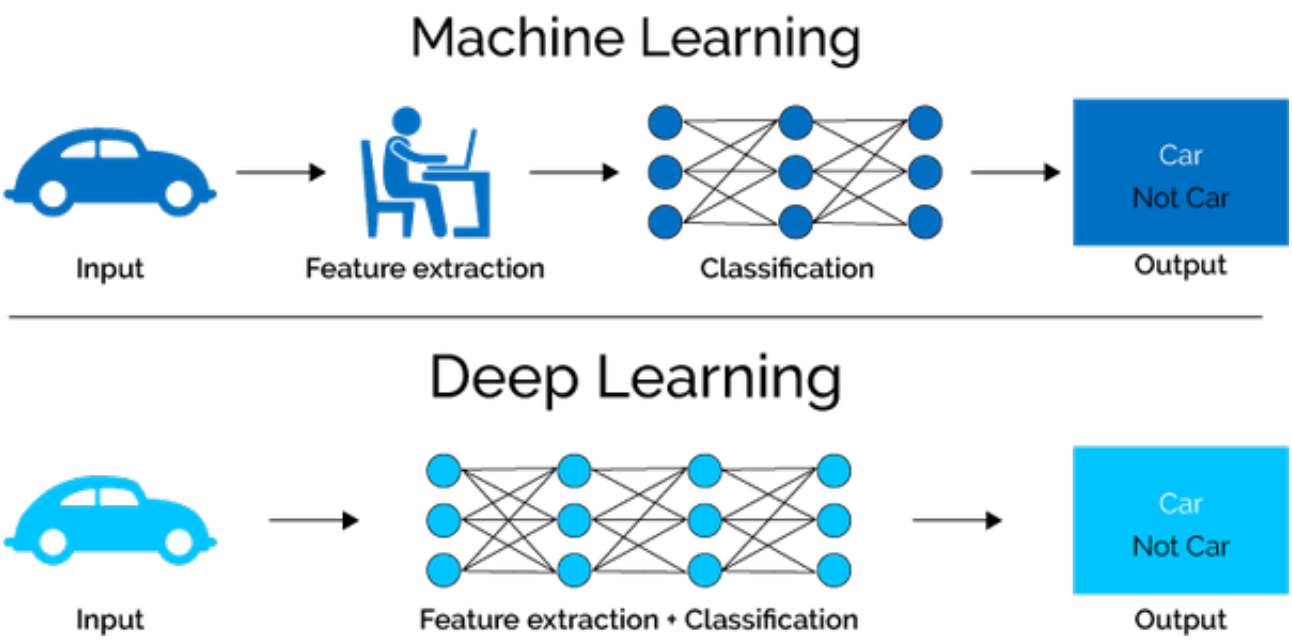


FIGURE 2.2: Comparaison entre l'apprentissage profond et l'apprentissage automatique [14]

2.2.2 Historique de l'apprentissage profond

Dans [15], se trouve un aperçu du contexte historique de l'apprentissage profond, il présente les étapes majeures qui ont mené à ce que nous avons maintenant. Ces étapes sont résumées dans le tableau (2.1) :

Année	Contributeur	Contribution
300 AC	Aristotle	Introduction de l'associationnisme,qui essayent de comprendre le cerveau
1873	Alexander Bain	Introduction des premiers modèles de réseaux de neurones
1943	McCulloch and Pitts	Proposition d'un modèle considéré comme L'ancêtre des réseaux de neurones
1949	Donald Hebb	il introduit la règle d'apprentissage de Hebb
1958	Frank Rosenblatt	Introduction du premier perceptron
1974	Paul Werbos	Introduction de la retro-propagation
1980	Teuvo Kohonen	Introduction des cartes auto organisatrices
1980	Kunihiko Fukushima	Introduction du Neocognitron,
1982	John Hopfield	Introduction des réseaux de Hopfield
1985	Hilton and Sejnowski	Introduction des machines de Boltzmann
1986	Paul Smolensky	Introduction de Harmonium
1986	Michael I. Jordan	Définition et introduction des réseaux de neurones récurrents
1990	Yann LeCun	Introduction de LeNet
1997	Schuster and Paliwal	Introduction des réseaux de neurones récurrents bidirectionnels
1997	Hochreiter and Schmidhuber	Introduction de LSTM,
2006	Geoffrey Hinton	Introduction des Deep belief Network
2009	Salakhutdinov and Hinton	Introduction des Deep Boltzmann Machines
2012	Alex Krizhevsky	Introduction d'AlexNet qui remporta le challenge ImageNet

TABLE 2.1: Historique de l'apprentissage profond [15]

2.3 UTILITÉ DE L'APPRENTISSAGE PROFOND

Dans cette section, nous allons présenter les principaux points qui montrent l'importance du Deep Learning et l'utilité par rapport aux autres techniques de l'apprentissage automatique.[16]

- Les fonctionnalités conçues manuellement, sont souvent sur-spécifiées, incomplètes et prennent beaucoup de temps à concevoir et à valider.
- Les fonctionnalités apprises sont faciles à adapter, rapides à apprendre.
- L'apprentissage profond fournit un cadre très souple, pour représenter des informations.
- Peut apprendre à la fois sans surveillance et sous surveillance.

2.4 Application de l'apprentissage profond

Parmi les nombreuses applications de l'apprentissage profond, nous pouvons citer [16] :

- **La reconnaissance faciale :** un algorithme de DL va apprendre à détecter les caractéristiques d'un visage sur une photo. Il s'agit en premier lieu, de donner un certain nombre d'images à l'algorithme, puis à force d'entraînement, l'algorithme va être en mesure de détecter un visage sur une image.
- **Voitures autonomes :** les entreprises qui construisent de tels types de services d'aide à la conduite, ainsi que des voitures autonomes telles que Google, doivent apprendre à un ordinateur à maîtriser certaines parties essentielles de la conduite, à l'aide de systèmes de capteurs numériques au lieu de l'esprit humain.
- **Reconnaissance d'image :** Un autre domaine populaire en matière de DL est la reconnaissance d'images. Son objectif, est de reconnaître et d'identifier les personnes et les objets dans les images, ainsi que de comprendre le contenu et le contexte.
- **Génération automatique de texte :** C'est une tâche intéressante, où un corpus de texte est appris et à partir de ce modèle, un nouveau texte est généré, mot par mot ou caractère par caractère.

2.5 Types des réseaux de neurones profonds

Les types des réseaux de neurones profonds les plus utilisés, dans le procédé de l'apprentissage profond, sont les CNN :

2.5.1 Réseaux de neurones Convolutionnels CNN

Les réseaux de neurones convolutionnels sont parmi les types les plus utilisés, pour imiter le comportement du cerveau humain.Ils sont inspirés par les travaux de Hubel et Wiesel sur le cortex visuel chez les mammifères.[17]

Pendant les années 1990, ces réseaux se sont popularisés avec les travaux de Y. LeCun et al. sur la reconnaissance de caractères.[18]

2.5.1.1 *Les types de couches dans les CNN :*

Nous allons nous focaliser sur les couches les plus classiques : la couche de convolution (Conv), la couche entièrement connectée (FC) et la couche d'union (pooling).

La couche de convolution (Conv)

Elle est utilisée pour la détection de motifs avec une invariance en translation, ensemble de filtres qui sont appris au cours de l'entraînement. La taille et le nombre de ces filtres sont définis à priori, ils parcourent la totalité de l'image, le plus souvent avec une fonction d'activation Relu. Elle représente la réponse de ce filtre à chaque position spatiale. Ces cartes d'activations sont concaténées le long de la dimension de l'image, pour former un nouveau tenseur.La figure (2.3) montre le traitement d'une couche de convolution [1].

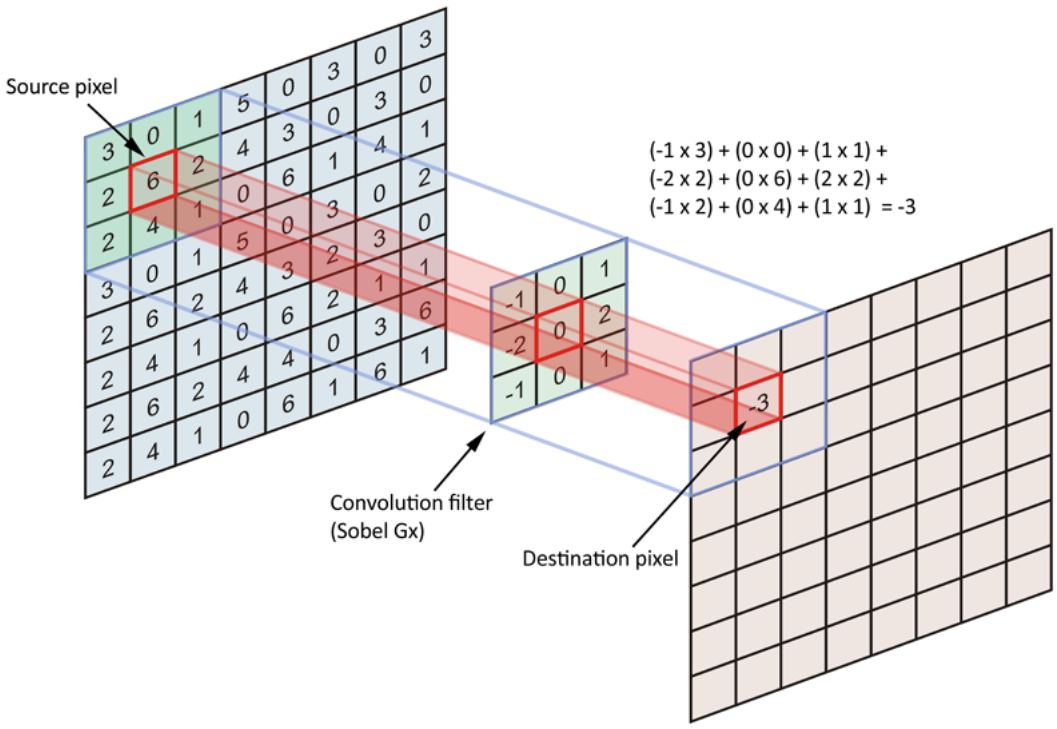


FIGURE 2.3: Couche de convolution

La couche d'union (Pooling)

Elle est utilisée pour réduire la détection des patterns ambigus, après une couche de convolution.

1. https://thigiacmaytinh.com/wp-content/uploads/2018/05/kernel.png

Ces fonctions sont prédéfinies et réduisent, pour les couches ultérieures, le nombre de paramètres à apprendre, tout en élargissant le champ de réception (receptive field) [19].

Elles opèrent indépendamment, des profondeurs du réseau et ne nécessitent pas de poids à entraîner.

L'idée derrière l'opération de "pooling" est de retirer une certaine information dans un même voisinage pour l'entrainement [20]. La figure (2.4) illustre ce procédé.

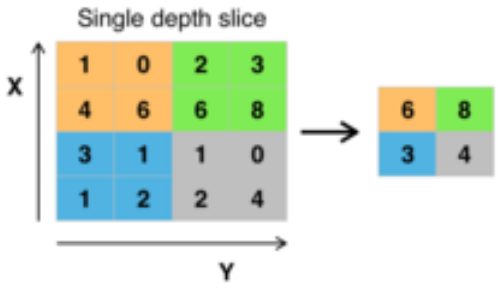


FIGURE 2.4: Couche de pooling [19]

La couche entièrement connectée (FC)

Elle est utilisée pour associer les différents motifs afin d'en déduire la classe. Ce type de couches reçoit un vecteur en entrée et produit un nouveau vecteur en sortie.

Pour cela, cette couche applique une combinaison linéaire puis, éventuellement, une fonction d'activation aux valeurs reçues en entrée.

2.5.1.2 *Schéma Général d'un CNN :*

Dans la figure (2.5) nous illustrons le schéma général d'un CNN, qui est constitué généralement d'un nombre de couches de convolution, certaines d'entre elles sont suivies par des couches de pooling, et finalement, par des couches entièrement connectées.

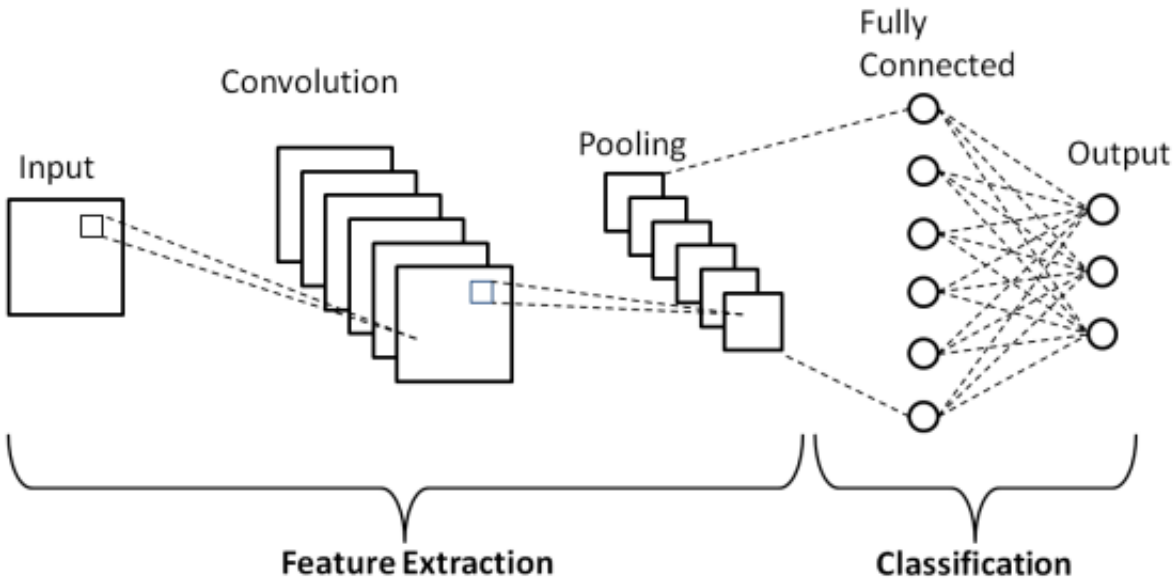


FIGURE 2.5: Schéma général d'un CNN [21]

2.6 L'architecture des réseaux de neurones Convolutionnels (CNN)

Dans cette section, nous allons montrer les principales architectures des réseaux de neurones convolutionnels CNN :

2.6.1 LeNet

LeNet est une architecture de CNN classique proposée par Yann LeCun en 1998. Cette architecture a été conçue pour reconnaître les patterns visuels, directement à partir de l'image.

Elle permet de reconnaître les patterns tels que les caractères manuscrits avec une robustesse aux distorsions et aux transformations géométriques [22]. Le schéma de l'architecture LeNet est illustée dans les figures (2.6) et (2.7).

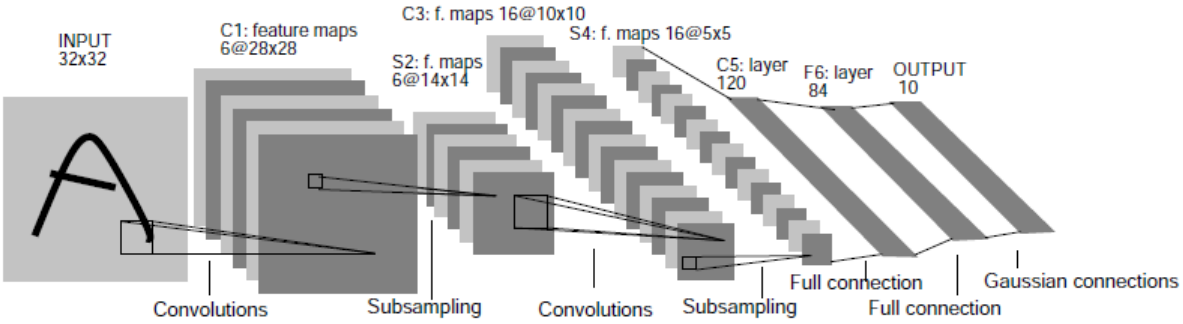


FIGURE 2.6: Schéma de l'architecture LeNet [22]

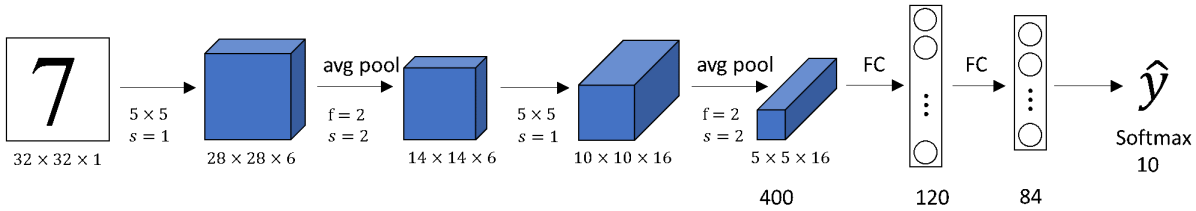


FIGURE 2.7: Schéma de l'architecture détaillée de LeNet [22]

2.6.2 AlexNet

AlexNet est un CNN conçu par Alex Krizhevsky publié avec Ilya Sutskever et son chef de doctorat Geoffrey en 2012.

Il contient huit couches : les cinq premières étaient des couches de convolution, certaines d'entre elles sont suivies par des couches de pooling, les trois dernières étaient des couches entièrement connectées (FC). Ces couches utilisent une fonction d'activation ReLu [23]. La figure (2.8) illustre le schéma d'AlexNet.

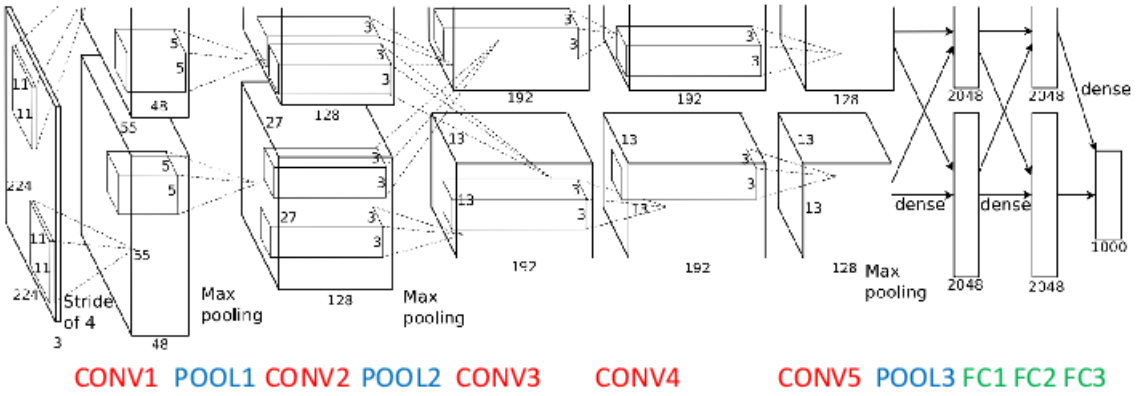


FIGURE 2.8: Schéma de l'architecture AlexNet [23]

AlexNet est considéré comme l'une des architectures les plus influentes publiées dans le domaine de la vision par ordinateur et de l'apprentissage profond.

2.6.3 VGGNet

VGGNet est un réseau neuronal convolutionnel de profondeur de 16 couches, présenté par les chercheurs du Visual Graphics Goup à l'université d'Oxford en 2014,il se caractérise par sa forme pyramidale [24]. La figure (2.9) montre cette architecture.

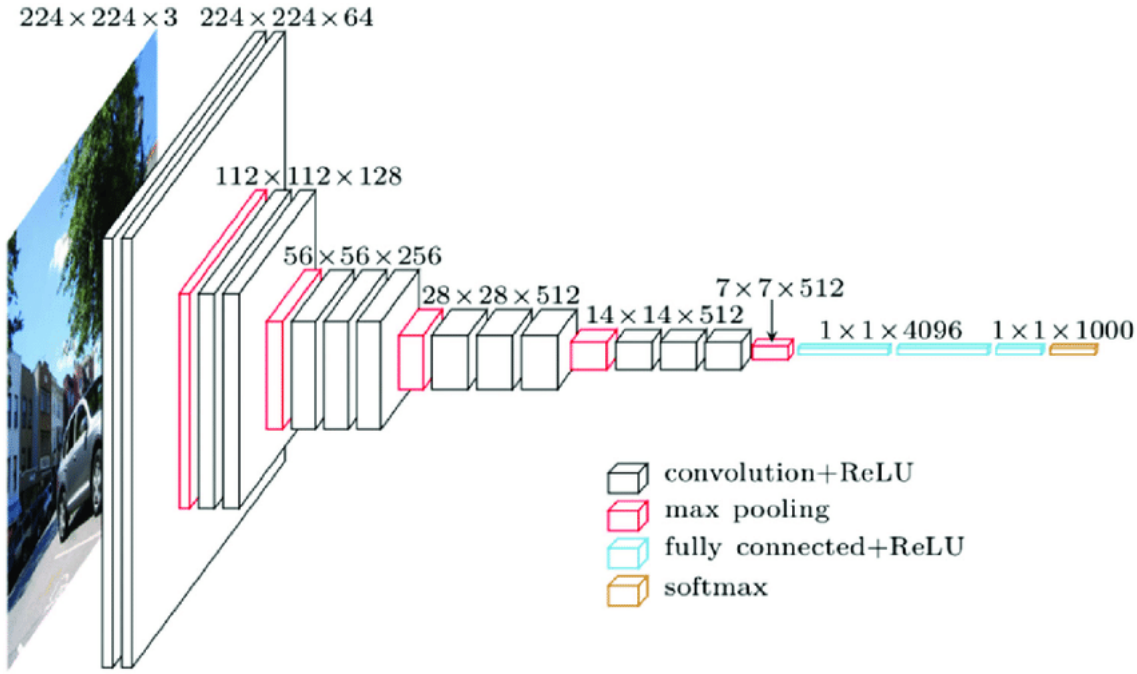


FIGURE 2.9: Schéma de l'architecture VGG [24]

2.6.4 ResNet

Le réseau de neurone Résiduel ResNet est un réseau qui utilise une architecture nouvelle avec « connexion par saut ». Ces connexions sont connues sous le nom d'unités récurrentes fermées [25].

Elles ressemblent fortement aux RNN, ces unités sont appelées « Residual Block » [26]. Le schéma de ces unités est présenté dans la figures (2.10)

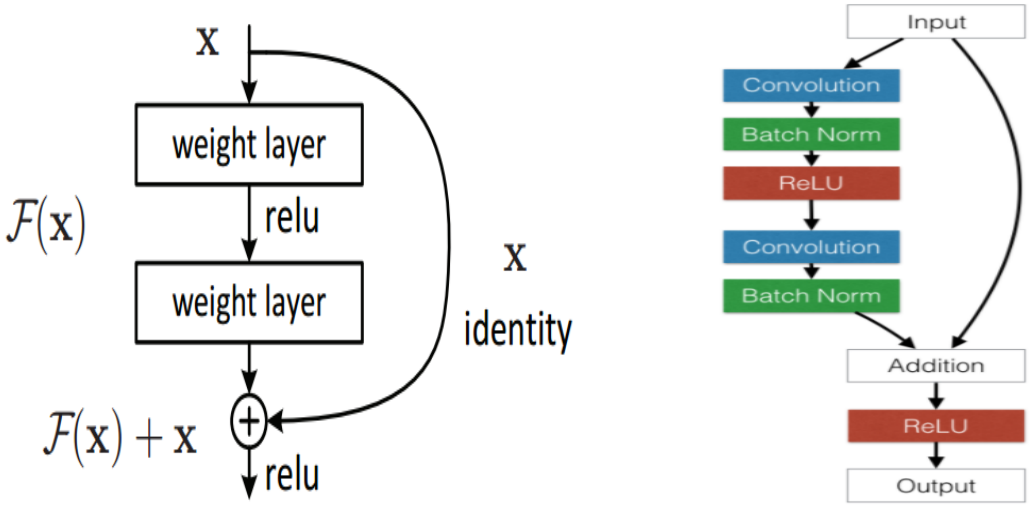


FIGURE 2.10: Schéma du "Residual Block" [25]

2.6.5 GoogleNet

GoogLeNet est le nom de la nouvelle architecture proposée par les chercheurs de Google, pour participer à la compétition ILSVRC en 2014. La contribution apportée par cette architecture, est une unité de base appelée « Inception Module » [27].

Ce modèle utilise « Average Pooling » au lieu des couches entièrement connectées. Cela réduit un grand nombre de paramètres comparé à AlexNet [28]. Les figures (2.11) et (2.12) montrent l'architecture du modèle "GoogLeNet" et de "Inception Module".

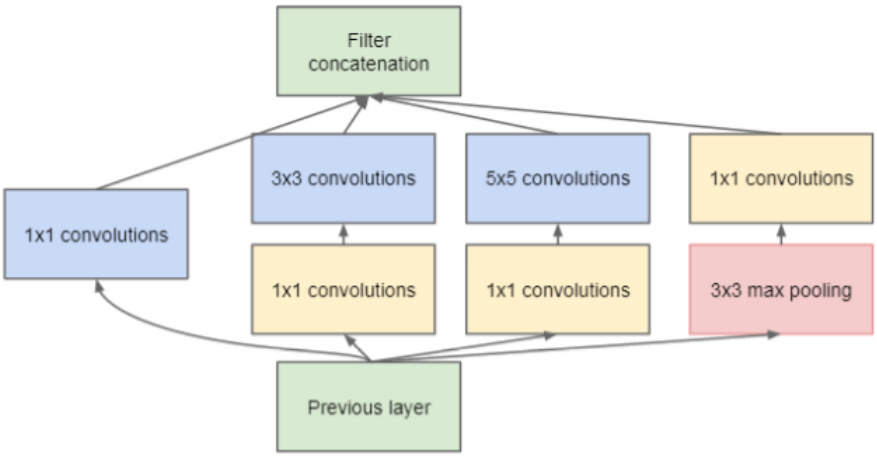


FIGURE 2.11: Schéma de "Inception Module" [27]

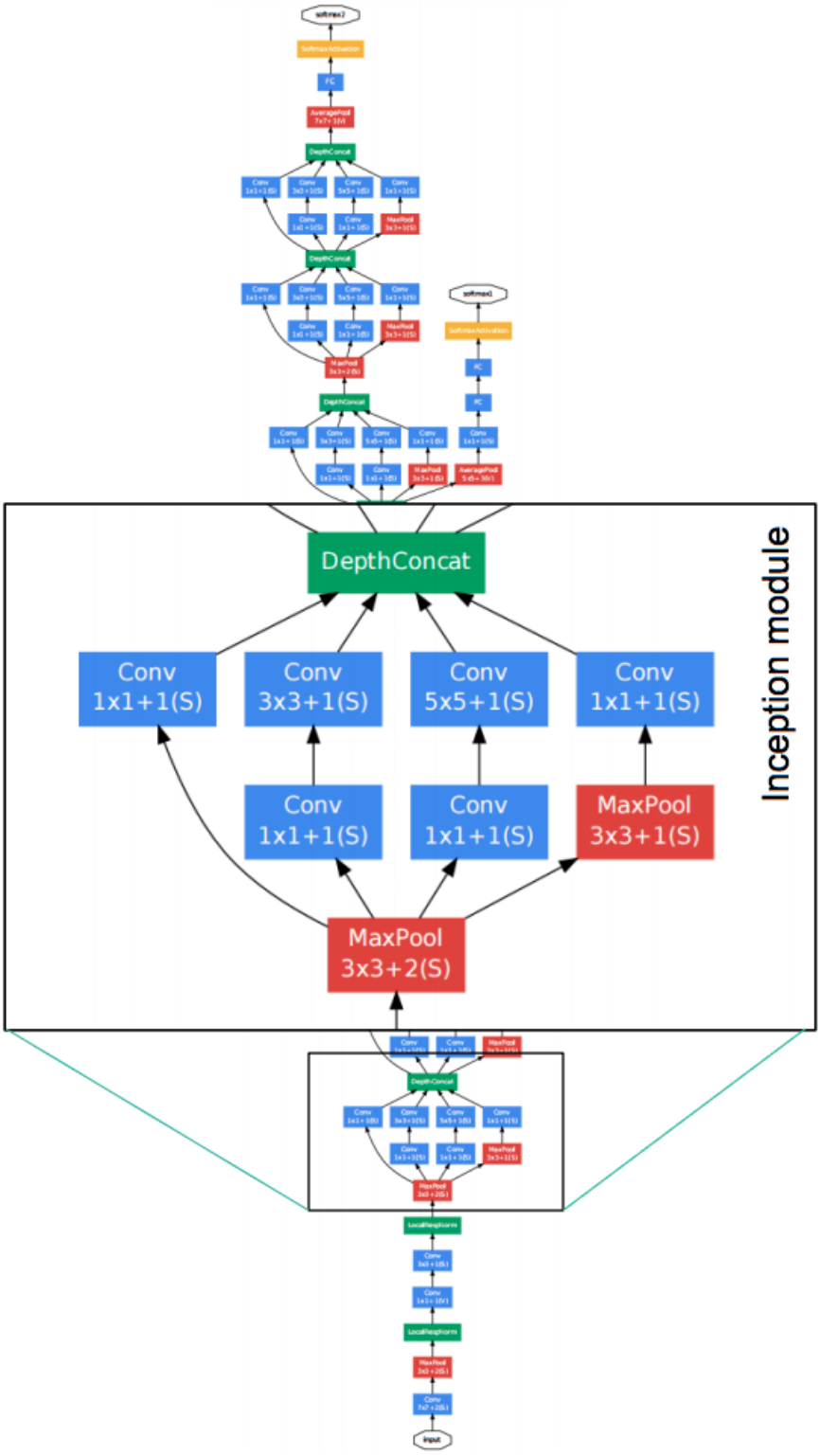


FIGURE 2.12: Schéma de l'architecture GoogleNet [28]

2.7 Conclusion

Dans ce chapitre nous avons présenté le domaine de l'apprentissage profond. Nous avons introduit puis expliqué les réseaux de neurones profonds (particulièrement les réseaux convolutionnels), qui seront essentiels pour la reconnaissance des chiffres nécessaire à notre problématique. Finalement, nous clôturons ce chapitre par l'illustration des différentes architectures des réseaux de neurones convolutionnels. Dans le chapitre qui suit, nous allons nous intéresser à l'utilisation des réseaux de neurones profonds dans le contexte d'une reconnaissance de caractères. Le cas des plaques minéralogiques a été choisi pour sa proximité avec notre problématique.

Chapitre 3

Etat de l'art de la Reconnaissance des chiffres

3.1 Introduction

La reconnaissance des caractères est souvent considérée comme l'un des principaux problèmes d'apprentissage du réseau neuronal. Ce chapitre sera totalement consacré à la présentation de la reconnaissance des chiffres qui est la base de notre travail.

Comme nous n'avons pas pu trouver des travaux identiques à notre problématique, de reconnaissance des chiffres des compteurs électroniques de tours présents dans les administrations algériennes, nous nous focaliserons sur des travaux similaires comme la reconnaissance des chiffres des plaques minéralogiques.

3.2 Détection et reconnaissance des plaques minéralogiques

La lecture automatique de plaques minéralogiques ou Lecture automatisée de plaques d'immatriculation, consiste à mettre en place une combinaison de moyens électroniques et informatiques, en utilisant des techniques de traitement d'images et de vision par ordinateur, pour extraire le numéro d'identification d'après l'image de la plaque, sous format de caractères. La figure (3.1) illustre un exemple d'un système de reconnaissance [1].

1. https://www.alphanumeric-vision.com

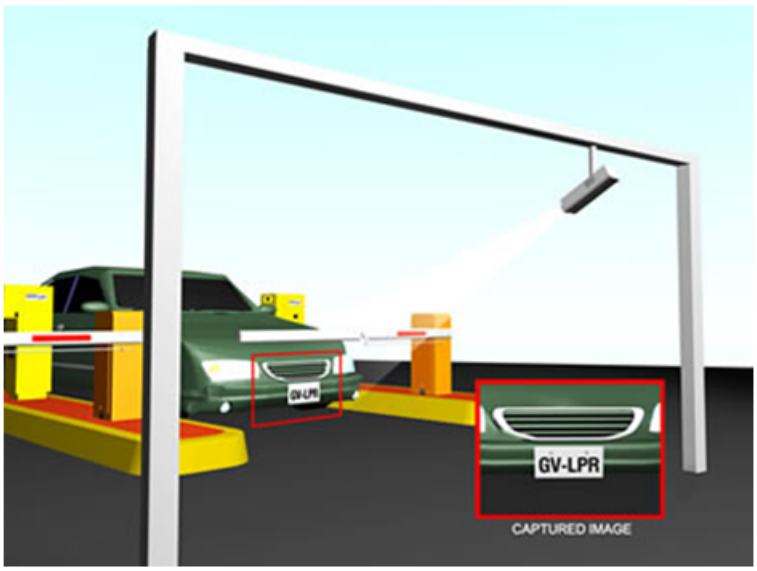


FIGURE 3.1: Schéma du système de reconnaissance

3.3 ÉLÉMENTS D'UN SYSTÈME TYPIQUE DE LECTURE AUTOMATIQUE DE PLAQUES MINÉRALOGIQUES

3.3.1 Matériel

Camera

Elle sert à prendre une image du véhicule [29].

Éclairage

C'est une lumière projetée sur la scène, pour éclairer la plaque. Ce système permet une utilisation du jour comme de la nuit du système de reconnaissance [29].

Ordinateur

Un ordinateur qui exécute le système de reconnaissance, lit l'image, identifie et détecte la plaque dans l'image [29].

3.3.2 Logiciel

Frame Grabber

C'est une carte d'interface entre le pc et la camera, qui permet le transfert des images vers le Pc [29].

Base de données

Elle inclut les données et les résultats de la reconnaissance [29].

3.4 Domaines d'application du système de lecture automatique de plaques minéralogiques

Il existe de nombreuses applications qui utilisent de tels systèmes de reconnaissance, dont certaines sont décrites ci-dessous :

3.4.1 Contrôle de vitesse

Le numéro d'immatriculation est utilisé pour imposer une amende aux véhicules qui roulent à grande vitesse, à l'utilisation illégale des voies et à la détection de véhicules volées ou recherchées [30]. La figure (3.2) montre un exemple de voiture capturée

FIGURE 3.2: Application du système dans le contrôle de vitesse [30]

3.4.2 Contrôle d'accès au parking

Le système est utilisé pour l'entrée automatique au parking, pour les membres déjà enregistrés, ou dans certains cas, pour le calcul des frais de stationnement des voitures. Lorsque une voiture entre, la plaque d'immatriculation est reconnue et marquée. A la sortie, elle est relue et la durée de stationnement est facturée pour le conducteur [30]. La figure (3.3) illustre un parking contrôlé par une application.

FIGURE 3.3: Système de contrôle d'accès au parking [30]

3.4.3 Contrôle Passage aux frontières

Cette application aide le registre des entrées ou des sorties dans un pays et peut être utilisée, pour surveiller les postes frontières [30]. Cette installation est illustrée à la figure (3.4). Chaque information de véhicule est enregistrée dans une base de données.

FIGURE 3.4: Frontière franco-allemande [30]

3.5 Étapes du processus de lecture automatique de plaques minéralogiques

Chaque système de reconnaissance automatique de plaques d'immatriculation est constitué des étapes suivantes :

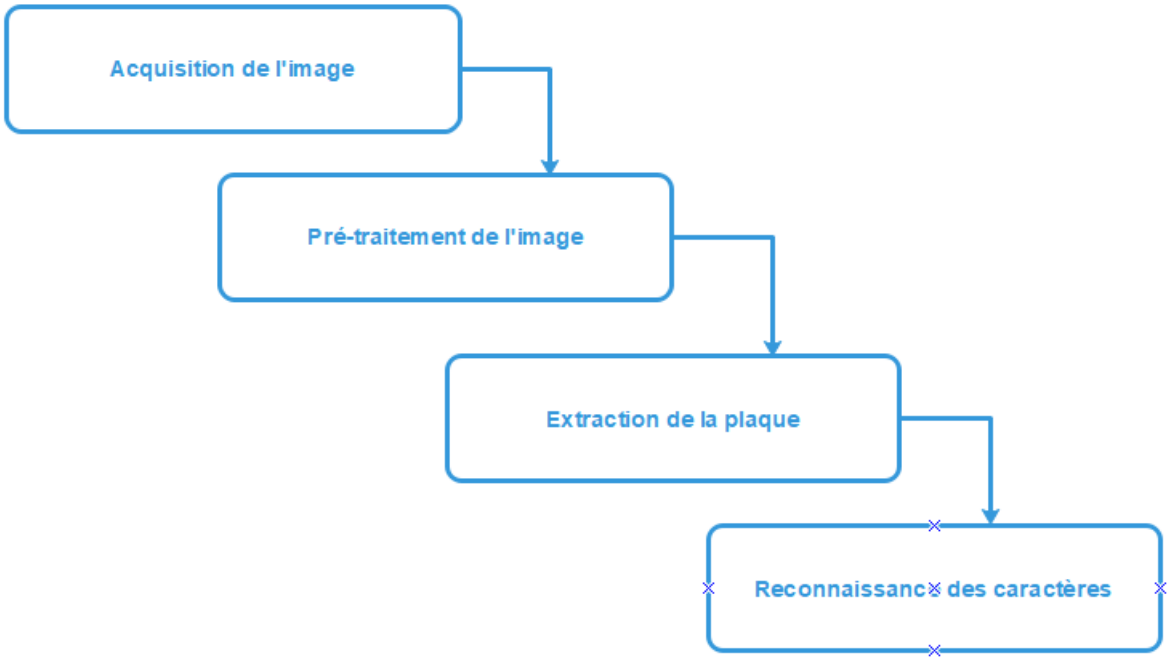


Figure 3.5: Étapes du processus de lecture de plaques

3.5.1 Acquisition de l'image

La première étape consiste à acquérir l'image ou tout simplement, à capturer des images et des vidéos du véhicule lors de son passage.

3.5.2 Pré-traitement d'image

Cette étape consiste à traiter l'image, principalement, la convertir en niveau de gris pour que la phase du Deep Learning soit plus rapide. Une image en niveaux de gris de M pixels de hauteur et de N pixels de largeur est représentée par une matrice de type de

données double de taille M * N, dont les valeurs ont été mises à l'échelle pour représenter les intensités. Chaque valeur de pixel des images en niveaux de gris, est comprise entre 0 (noir) et 255 (blanc), Voici deux méthodes pour convertir une image en niveau de gris :

Méthode 1 : moyenne

$$Grayscale = (B + G + R)/3$$

Méthode 2 :

$$Grayscale = (B * 0.3) + (G * 0.59) + (R * 0.11)$$

La figure (3.6) représente une image avant et après conversion

(a) Image Originale (b) Image en niveau de gris

FIGURE 3.6: Image d'une voiture avant et après conversion en niveau de gris [31]

3.5.3 Extraction de la plaque d'immatriculation

L'étape suivante consiste à extraire la plaque d'immatriculation de l'image. Pour cela, il existe différentes méthodes, qui se basent sur la recherche des caractéristiques de la plaque dans l'image comme la couleur, la forme etc [32]. Pour détecter la position de la plaque dans l'image, il existe différentes méthodes pour l'extraction de la plaque qui peuvent être classées en différentes catégories :

3.5.3.1 *Extraction basée sur le détecteur YOLO*

YOLO (You Only Look Once) est l'un des algorithmes les plus rapides pour la détection d'objet qui est basé sur un réseau de neurone profond [33], son principe est simple :

il faut entraîner ce réseau sur un ensemble d'images annotées à la main, c'est-à-dire déterminer la position de l'objet à extraire, puis après l'entraînement l'algorithme va détecter automatiquement la position de l'objet. Un exemple est illustré dans la figure (3.7) :

FIGURE 3.7: Plaque détectée par YOLO [34]

3.5.3.2 *Extraction basée sur le contour*

Étant donné, que la plaque d'immatriculation a normalement une forme rectangulaire avec un rapport d'aspect connu, elle peut être extraite en trouvant tous les rectangles possibles dans l'image.

Les méthodes de détection des contours sont couramment utilisées pour trouver ces rectangles. Le filtre Sobel est utilisé pour détecter les bords. En raison de la transition de couleur entre la plaque d'immatriculation et la carrosserie, la limite de la plaque d'immatriculation est représentée par des bords dans l'image. Les bords des plaques d'immatriculation doivent être continus [35].

3.5.3.3 *Extraction basée sur les caractéristiques globales de l'image*

L'analyse des composants connectés (CCA) est une technique importante dans le traitement d'images binaires. Elle scanne une image binaire et étiquette ses pixels en composants, en fonction de la connectivité des pixels (Mesures spatiales), telles que l'aire et le rapport d'aspect, sont l'utilisant couramment pour l'extraction de plaques d'immatriculation

Les objets connectés, qui ont les mêmes caractéristiques géométriques que la plaque, sont choisis pour être candidats.

Cet algorithme peut échouer dans le cas d'images de mauvaise qualité, ce qui entraîne des contours déformés.

La corrélation croisée 2D est utilisée pour trouver les plaques d'immatriculation.

La corrélation croisée 2D avec un modèle de plaque d'immatriculation pré-enregistrée est effectuée sur toute l'image pour localiser la zone de plaque d'immatriculation la plus probable.
L'extraction de plaques d'immatriculation à l'aide de la corrélation avec un modèle est indépendante de la position de la plaque d'immatriculation dans l'image. Cependant, la corrélation croisée 2D prend du temps [35].

3.5.3.4 *Extraction basée sur la texture*

Ce type de méthode dépend de la présence des caractères dans la plaque d'immatriculation, ce qui entraîne un changement dans le niveau de gris entre la couleur des caractères et la couleur de fond de plaque. Il en résulte également, une zone de densité de bord élevée en raison de la transition de couleur.

Les filtres de Gabor sont l'un des principaux outils de l'analyse de texture. Cette technique a l'avantage d'analyser la texture dans des orientations et des échelles illimitées. Cependant, cette méthode prend du temps.

La fréquence spatiale est identifiée en utilisant la transformée de Fourier discrète (DFT) et la transformée en ondelettes (DWT), car elles produisent des harmoniques qui sont détectées dans l'analyse du spectre. La DFT et la DWT sont utilisées, de manière rangée, pour détecter la position horizontale de la plaque et de manière colonne pour détecter la position verticale.

Toutes les méthodes basées sur la texture ont l'avantage de détecter la plaque d'immatriculation même si sa limite est déformée. Cependant, ces méthodes sont complexes sur le plan informatique, en particulier lorsqu'il y a de nombreux bords, comme dans le cas d'un fond complexe ou dans des conditions d'éclairage différentes [35].

3.5.3.5 *Extraction basée sur les caractéristiques de la couleur*

Étant donné que certains pays ont des couleurs spécifiques pour leurs plaques d'immatriculation, certains travaux impliquent l'extraction de plaques d'immatriculation en localisant leurs couleurs dans l'image.

L'idée de base est que la combinaison de couleurs d'une plaque est unique, et cette combinaison se produit presque uniquement dans une région de plaque [35].

Comparaison entre ces méthodes

Méthode	Avantage	Inconvénient
Extraction basée sur le contour	Simple, rapide et direct	Mise en oeuvre difficile pour les images complexe car elle est trop sensible aux bords indésirables
Extraction basée sur les caractéristiques globales de l'image	Direct, indépendant de la position de la plaque d'immatriculation.	Peut générer des objets déformés.
Extraction basée sur la texture	Être capable de détecter même si les bornes sont déformées.	
Extraction basée sur les caractéristiques de la couleur	Être capable de détecter les plaques d'immatriculation inclinées et déformées.	RGB est limité aux conditions d'éclairage, HLS est sensible aux bruit

TABLE 3.1: Comparaison entre les méthodes extraction de la plaque d'immatriculation [35]

3.5.4 Reconnaissance des caractères

Après l'extraction de la plaque d'immatriculation, la dernière étape du processus de la lecture automatique des plaques minéralogiques est la reconnaissance des chiffres de la plaque minéralogique. Pour cela, nous allons présenter quelques modèles de la littérature concernant cette problématique.

3.5.4.1 *Modèle de Selmi et al*

Configuration du modèle

Le tableau (3.2) illustre l'architecture du modèle proposé [36].

Couche	Paramètre
Entré	Image en niveau de gris 32X32 Pixel
Couche de Convolution (ReLu)	Profondeur : 32 Fenêtre : 5X5 Pas : 1
Couche de Pooling	Fenêtre : 5X5
Couche de Convolution (ReLu)	Profondeur : 64 Fenêtre : 3X3 Pas : 1
Couche de Pooling	Fenêtre : 5X5
Couche de Convolution (ReLu)	Profondeur : 128 Fenêtre : 3X3 Pas : 1
Couche de Convolution (ReLu)	Profondeur : 256 Fenêtre : 3X3 Pas : 1
Couche de Pooling	Fenêtre : 5X5
Couche Entièrement Connecté FC (ReLu)	Taille : 1024 neurones
Couche de Dropout	0.5
Couche Entièrement Connecté FC (Softmax)	Taille : 37 neurones

TABLE 3.2: Configuration du Modèle Selmi [36]

Description du modèle

Le modèle proposé par Selmi et al Dans [36] en 2017 est un réseau de neurone convolutionnel (CNN). L'entrée du modèle est une image de taille 32X32 pixel. Il est constitué principalement de dix couches, huit couches de convolution activées par une fonction d'activation ReLu, trois couches de pooling, une couche Dropout et deux couches entièrement connectées. Une d'entre elles est la sortie composée de 37 neurones c'est à dire 37 classes (10 chiffres de 0 à 9 et 26 lettres et la catégorie négative). Cette couche est activée par la fonction Softmax.

Selmi a testé son modèle sur deux datasets, le premier est le dataset AOLP qui contient 2049 images de plaques minéralogiques originaires de Taiwan. Elle est divisée en trois parties : Patrouille de route (RP) avec 611 images, Contrôle d'accès (AC) avec 681 images et Law Enforcement (LE) avec 757 images.

Le deuxième dataset est issu de PKU (Pekin University). Il contient 3977 images de plaques chinoises [36].

3.5.4.2 *Modèle de Wu et al*

Configuration du modèle

Le tableau (3.3) illustre l'architecture du modèle proposé [37].

Couche	Paramètre
Entré	Image 136X36 Pixel
Bloc de convolution (Couche de convolution Relu +Batch Normalisation Max Pooling)	Profondeur : 32 Fenêtre : 3X3 Pas 1 Fenêtre : 2X2
8 Dense Block (Couche de convolution Relu +Batch Normalisation)	Profondeur : 128 Fenêtre : 3X3
Bloc de transition (Couche de convolution Relu +Batch Normalisation Moyenne Pooling)	Profondeur : 128 Fenêtre : 1X1
8 Dense Block (Couche de convolution Relu +Batch Normalisation)	Profondeur : 192 Fenêtre : 3X3
Bloc de transition (Couche de convolution Relu +Batch Normalisation Moyenne Pooling)	Profondeur : 128 Fenêtre : 1X1
8 Dense Block (Couche de convolution Relu +Batch Normalisation)	Profondeur : 192 Fenêtre : 3X3
Couche Entièrement Connecté FC (Softmax)	Taille : 68 neurones

TABLE 3.3: Configuration du Modèle Wu [37]

Description du modèle

Ce modèle est proposé par l'équipe de chercheurs Wu et al en 2018 [37]. Il est basé sur DenseNet avec une sortie 68 neurones donc 68 classes (31 caractères chinois, dix chiffres et 26 lettres).

Principalement, il est constitué d'un bloc de convolution activé avec une fonction ReLu, 3 dense Block, 2 bloc de transition et une couche entièrement connectée (sortie) activée avec la fonction d'activation Softmax.L'entré du modèle est une image de taille 136X36.

Le modèle est testé sur deux ensembles d'images :le premier Dataset-1[ref] qui est constitué de 203774 images et le second est le dataset AOLP utilisé dans le modèle de Selmi [37].

3.5.4.3 *Modèle de Spahel et al*

Configuration du modèle

Le tableau (3.4) illustre l'architecture du modèle proposé [38].

Couche	Paramètre
Entré	Image en niveau de gris 200X40 Pixel
3 Bloc de convolution (Couche de convolution Relu +Batch Normalisation)	Profondeur : 32 Fenêtre : 5X5 Pas : 1
Max Pooling	Fenêtre : 2X2
3 Bloc de convolution (Couche de convolution Relu +Batch Normalisation)	Profondeur : 64 Fenêtre : 3X3 Pas : 1
Max Pooling	Fenêtre : 2X2
3 Bloc de convolution (Couche de convolution Relu +Batch Normalisation)	Profondeur : 128 Fenêtre : 3X3 Pas : 1
Max Pooling	Fenêtre : 2X2
8 Branche de sortie (2 X Couche Entièrement Connecté (Softmax))	Taille : 37 neurones

TABLE 3.4: Configuration du Modèle Spahel [38]

Description du modèle

C'est un modèle basé sur un réseau de neurone convolutionnel (CNN) proposé par Spahle et al en 2017 [38]. Le modèle proposé a une architecture spéciale au niveau des couches entièrement connectées, qui sont représentées par plusieurs branches.

Principalement, le modèle est constitué de trois couches de convolution avec des fonctions d'activation ReLu, trois couches de pooling puis huit branches. Pour chaque branche, elle contient deux couches entièrement connectées (FC) avec la fonction Softmax, l'entrée du modèle est une image de dimension 200X40 Pixels [38].

Ce modèle est testé sur trois Datasets différents : ReId, HDR et Svoboda [39].

3.6 Conclusion

Dans ce chapitre, nous avons introduit la reconnaissance des caractères, précisément la reconnaissance des chiffres. Puis nous avons présenté la problématique de détection et reconnaissance des plaques minéralogique, qui est une problématique assez proche de la nôtre, en détaillant toutes les étapes du processus de lecture automatique de plaques minéralogiques. Enfin, nous avons présenté certains modèles de la littérature en ce qui concerne la reconnaissance des plaques minéralogiques. Dans le chapitre suivant, nous allons présenter notre modèle qui permettra la reconnaissance des numéros de tours sur un afficheur sept segments.

Chapitre 4

Modèle pour la gestion des tours dans les administrations Algériennes

4.1 Introduction

Après avoir détailler dans le chapitre précédent, l'état de l'art de la reconnaissance des chiffres, nous allons présenter notre contribution pour gérer les tours dans les administrations algériennes, et ainsi régler le problème de surcharge au niveau des files d'attentes notamment, en période de COVID 19.

4.2 Problématique étudiée

Nous allons nous intéresser à la problématique de la gestion des tours dans les administrations algériennes, spécialement dans les mairies et les différentes annexes de la commune de Jijel.

Pour cela, nous allons développer un système de lecture automatique de la plaque de compteur électronique des tours, et une application mobile destinée à informer le citoyen de l'imminence de l'arrivée de son tour.

4.3 Architecture du système proposé

Notre projet, porte sur la tâche de développement d'un système basé sur l'apprentissage profond pour la gestion des tours dans les administrations, ce dernier est devisé en deux parties :

4.3.1 Partie reconnaissance des chiffres de la plaque 7 segments

Cette partie consiste à élaborer un lecteur automatique des plaques de compteur électronique des tours, basé sur l'apprentissage profond. Pour atteindre cet objectif, il nous faut une caméra située au niveau des administrations pour la captation des images de la plaque afin de les envoyer à un serveur (Firebase de Google). Le serveur Firebase va les transmettre à un logiciel [40], hébergé par un noeud de calcul, qui va reconnaitre le numéro présent sur la plaque. La figure (4,1) illustre le processus de cette partie :

FIGURE 4.1: Schéma explicatif de la partie de reconnaissance

Le pseudo-algorithme (2) montre les étapes du processus de la lecture automatique illustré dans la figure (4.1)

Algorithm 2 pseudo-algorithme lecture automatique des plaques

1-Une application mobile qui capture l'image et l'envoie au serveur, elle joue le rôle de la caméra dans l'administration

2-Le serveur "Firebase" reçoit l'image et la transmet au programme python

3-Le programme python exécute le détecteur Yolov3 qui va extraire les chiffres de la plaque

4-Découper le résultat de l'étape 3

5-Faire la prédiction sur l'image résultante.

Dans la prochaine section nous allons présenté la deuxième partie du système, consacré à l'application mobile.

4.3.2 Partie de l'application mobile

Grâce aux résultats de la partie précédente,la prédiction du numéro actuel de la plaque de l'administration sera disponible dans le programme. Dés lors, le programme pourra l'envoyer à nouveau au serveur Firebase (étape 1 dans la figure (4.2)). Ce dernier s'occupera de l'envoie des notifications push aux citoyens, via internet vers une application mobile de notre conception (étape 2 dans la figure (4.2)).La figure (4.2) présente un schéma général explicatif de cette partie.

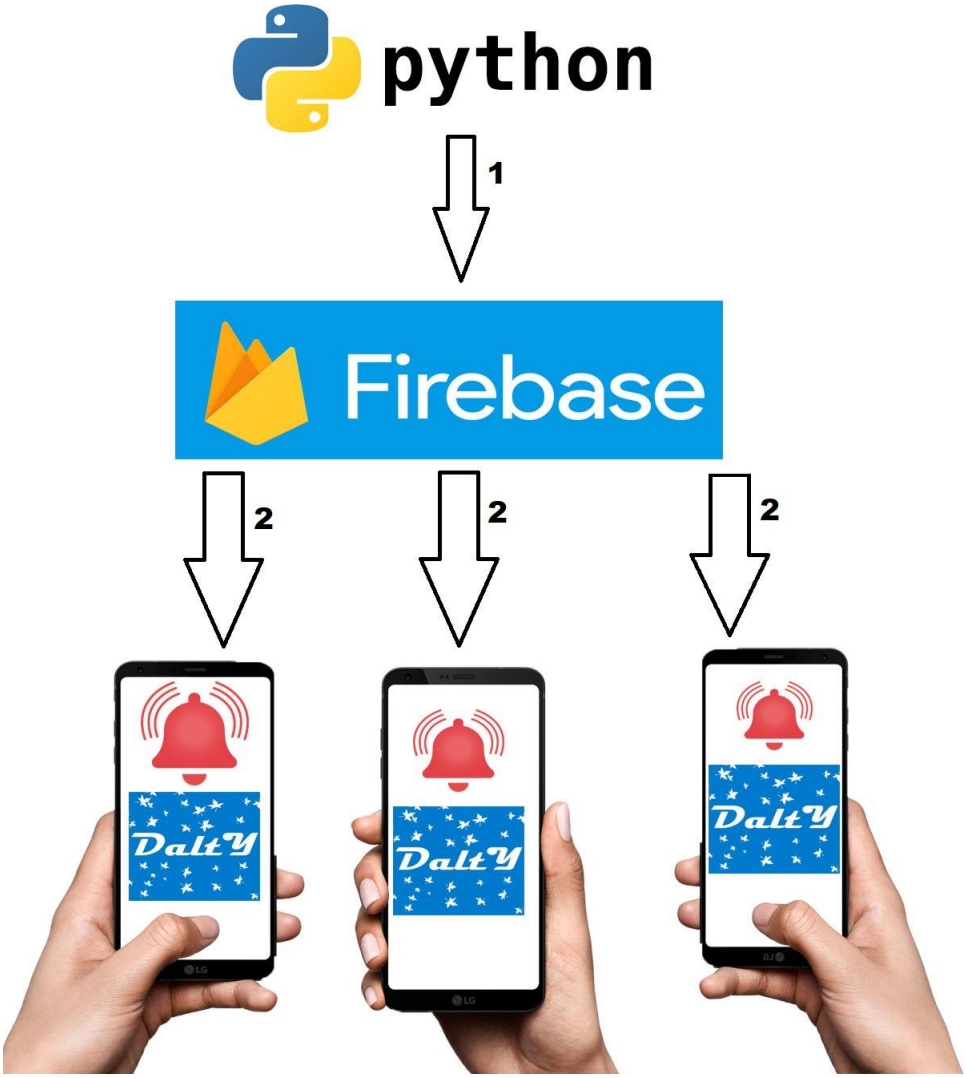


FIGURE 4.2: Processus de distribution des notifications pushs

Le schéma général de notre système est présenté dans la figure (4.3)

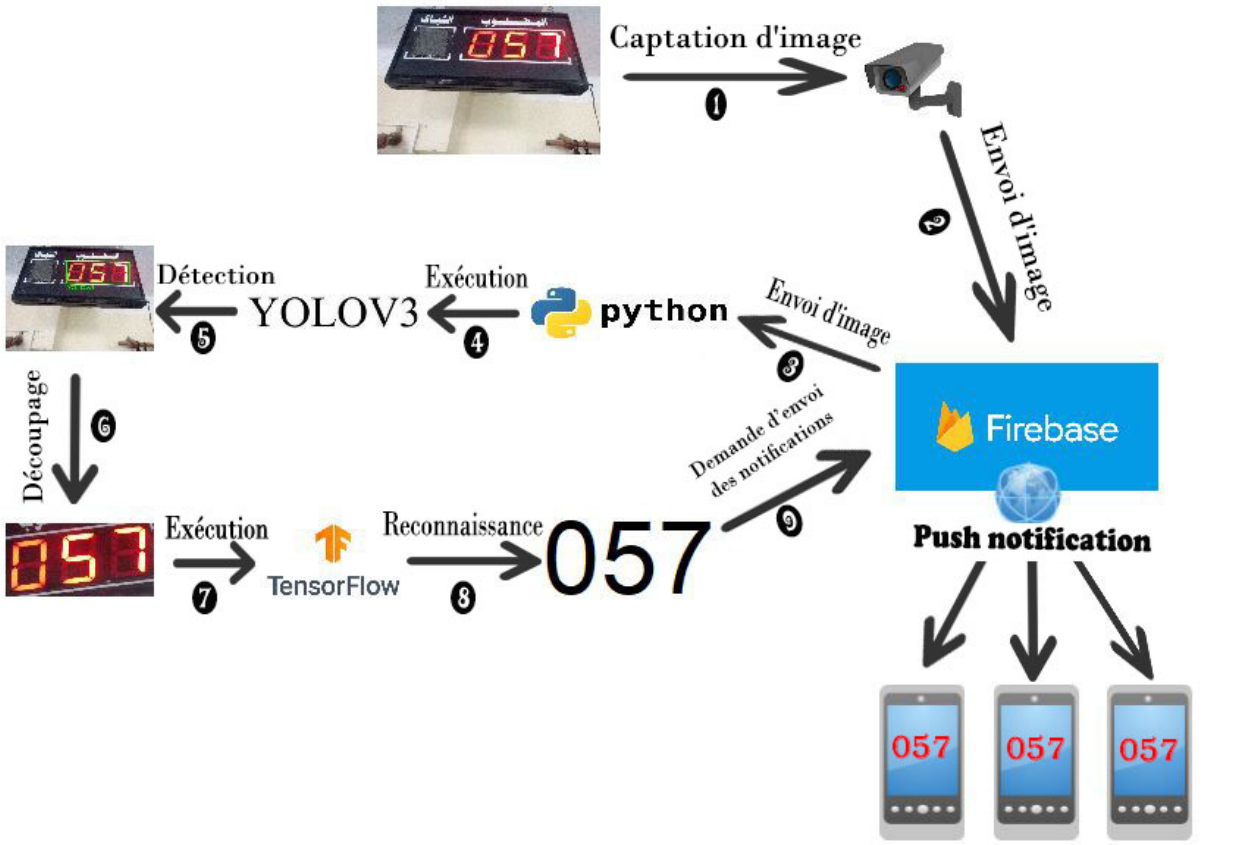


FIGURE 4.3: Processus général du système

4.4 LA DÉTECTION DES CHIFFRES DES PLAQUES DE TOURS DANS LES ADMINISTRATIONS ALGÉRIENNES

Pour la détection des chiffres de la plaque de compteur des tours, il existe plusieurs algorithmes parmi lesquels :

4.4.1 YOLOv3 (you look only once)

YOLOv3 est l'un des algorithmes les plus rapides pour la détection d'objets [41], qui peut être considéré comme une combinaison d'un localisateur et d'un identificateur d'objets. Il ne transmet l'image entière qu'une seule fois via le réseau. Il divise l'image en cellules ou boites, pour chaque boîte englobante, le réseau prédit également la confiance

qu'il y a réellement un objet et la probabilité que l'objet enfermé appartienne à une classe particulière [42]. Le paramétrage et le dataset d'apprentissage de cet algorithme sont détaillés dans le chapitre 5. La figure (4.4) illustre les résultats de l'exécution du détecteur YOLOv3.

FIGURE 4.4: Résultats de l'exécution du détecteur YOLO

4.5 La reconnaissance des chiffres des plaques de tours dans les administrations algériennes

Pour la reconnaissance des chiffres des plaques de tours, nous avons opté pour les réseaux de neurones convolutionnels . La section prochaine présente la configuration du modèle proposé.

4.5.1 Architecture du Classifieur

Pour la phase de la reconnaissance des chiffres de la plaque de compteur, nous avons proposé l'architecture du modèle de classification présentée dans le tableau (4.1) :

Couche	**Paramètre**
Entrée	Image en niveau de gris 64X64 Pixel
Couche de Convolution (ReLu)	Profondeur : 32 Fenêtre : 3X3 Pas : 1
Couche de Pooling	Fenêtre : 2X2
Couche de Convolution (ReLu)	Profondeur : 64 Fenêtre : 3X3 Pas : 1
Couche de Pooling	Fenêtre : 2X2
Couche de Convolution (ReLu)	Profondeur : 96 Fenêtre : 3X3 Pas : 1
Couche de Convolution (ReLu)	Profondeur : 128 Fenêtre : 3X3 Pas : 1
Couche de Pooling	Fenêtre : 2X2
Couche de Convolution (ReLu)	Profondeur : 196 Fenêtre : 3X3 Pas : 1
Couche de Convolution (ReLu)	Profondeur : 256 Fenêtre : 3X3 Pas : 1
Couche de Pooling	Fenêtre : 2X2
Flatten	/
Couche Entièrement Connecté FC (ReLu)	Taille : 2048 neurones
Couche Entièrement Connecté FC (Sigmoid)	Taille : 30 neurones

Table 4.1: Architecture du Modèle Proposé

Le compteur de tours se présente sous la forme d'un afficheur 7 segments qui énumère les numéros sur trois positions : les unités, les dizaines et les centaines. L'entrée du classifieur sera une image carrée de 64 pixels de côtés et la sortie est représentée par 30 neurones : un neurone par chiffre et par position.

4.5.2 Présentation du Modèle

L'architecture du modèle proposé, se constitue de douze couches : six couches de convolution et quatre couches de Pooling plus deux couches entièrement connectées. L'une d'entre elles est la couche de sortie du système.

L'entrée de notre système est une image de taille 64X64 Pixels en canal rouge, cette dernière passe par la couche de convolution de 32 filtres 3X3 et un pas de 1 qui est activé par une fonction d'activation ReLu, pour donner des valeurs positives au neurone. Ensuite, nous appliquons la couche de Pooling avec un filtre 2X2, pour diminuer la taille de l'image et le nombre de paramètres. Par la suite, nous répétons les mêmes étapes avec, cependant, un changement dans le nombre de filtres de la couche de convolution à 64 filtres, cette partie a été inspirer de l'architecture du modèle AlexNet [23].

Dans l'étape suivante, nous passons par deux couches de Convolution successives avec 96 et 128 filtres de 3X3 respectivement, suivie d'une couche de Pooling d'un filtre de 2X2.Nous répétons le même procédé en changeant le nombre de filtres à 196 et 256 filtres suivis toujours d'une couche de Pooling. Elle correspond à l'architecture VGGNet [24].

Maintenant, que l'extraction des caractéristiques de l'image est faite, nous alimentons les couches entièrement connectées, précisément, nous proposons deux couches : une avec 2048 neurones qui s'active avec une fonction d'activation ReLu et une dernière couche de sortie du système constituée de 30 neurones (10 neurones pour chaque chiffre de 0 jusqu'à 9 pour chaque position) activée par une fonction d'activation Sigmoid. La figure (4.5) illustre l'architecture du classifeur proposé.

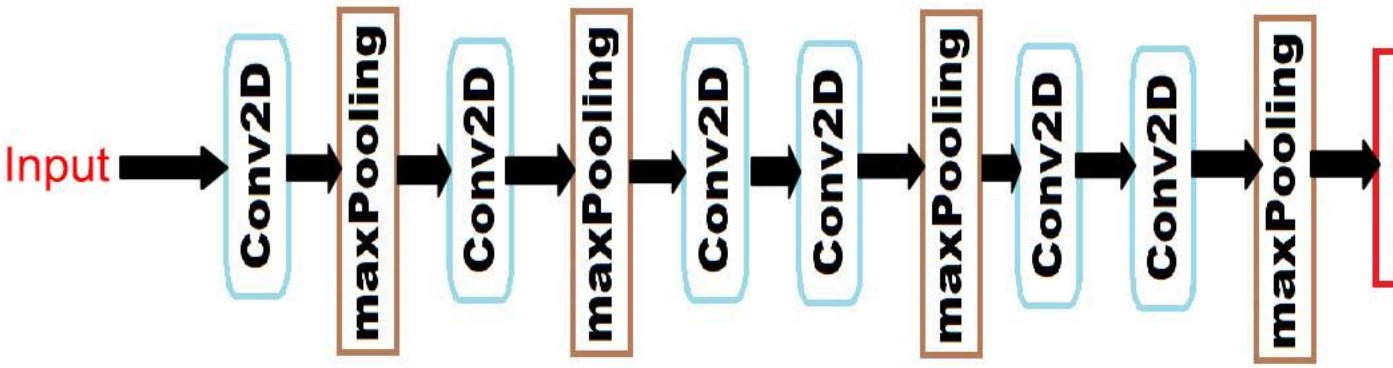


FIGURE 4.5: Architecture du modèle proposé

4.6 CONCLUSION

Dans ce chapitre, nous avons présenté notre contribution pour gérer les tours dans les administrations algériennes. Dans le chapitre suivant nous présenterons l'implémentation de ce système et les expérimentations effectuées sur notre modèle, afin de prouver sa robustesse.

Chapitre 5

Implémentation et résultats

5.1 Introduction

Après avoir présenté dans les chapitres précédents, l'état de l'art de la reconnaissance des chiffres, en commençant par les réseaux de neurones puis l'apprentissage profond, puis après avoir présenté notre modèle théorique pour la reconnaissance des chiffres d'un afficheur de tour sept segments, ce chapitre sera consacré à l'étape de l'implémentation et de la réalisation. Nous commençons d'abord, par une brève illustration de l'environnement de travail ainsi que l'ensemble des logiciels, utilisés dans la réalisation du système. Après cette illustration, nous allons présenté plusieurs tests paramétriques effectués sur notre modèle de reconnaissance, puis nous passons à un aperçu des interfaces les plus importantes de notre application.

5.2 Présentation des outils utilisés

Cette section présente les différents outils matériels et logiciels utilisés, tout le long de l'implémentation du projet.

5.2.1 Outils matériels

5.2.1.1 *Outils matériels distants*

Nous avons utilisé l'outil Google Collaboratory [1], souvent raccourci « Colab », qui est un produit de Google Research. Colab permet, à tout utilisateur, d'écrire et d'exécuter le code Python de son choix par le biais du navigateur. C'est un environnement, particulièrement adapté aux Deep Learning et à l'analyse des données. En termes plus techniques, Colab est un service hébergé de notebook Jupyter qui ne nécessite aucune configuration et permet d'accéder gratuitement à des ressources informatiques (GPU ...), de plus, l'utilisation du colab est gratuite. Les notebooks Colab sont stockés dans Google Drive et peuvent aussi être chargés depuis GitHub et partagés avec d'autres utilisateurs comme des documents Google Docs ou Sheets [47] La configuration matérielle de Google colab utilisée pour l'entrainement est la suivante :

– Processeur Intel core Xeon CPU @2.3 Ghz 45 MB de cache.

1. https://Colab.research.google.com/

- Processeur graphique (GPU) NVIDIA Tesla K80 ayant 2496 coeurs CUDA, Compute 3.7, 12 Go GDDR5 VRAM.
- Mémoire vive de 12.6 Go.
- Disque dure de capacité 320 Go.

La figure (5.1) présente le logo de Google Colab.

FIGURE 5.1: Icone Colab

5.2.1.2 *Outils matériels locaux*

Dans la réalisation de notre projet, nous avons utilisé nos deux ordinateurs portables personnels et nos smart-phones dont les caractéristiques sont :

- **Pc 1 :**
  - Processeur Intel core I3-6100U cpu @ 2.30Ghz.
  - Processeur graphique (GPU) Intel HD graphics 520.
  - Mémoire vive de 4 Go.
  - Disque dure de capacité 500 Go.
  - Système d'exploitation Windows 10 X64 bits.
- **Pc 2 :**
  - Processeur Intel core I5-5200U cpu @ 2.20Ghz.
  - Processeur graphique (GPU) Intel HD graphics 5500.
  - Mémoire vive de 4 Go.
  - Disque dure de capacité 500 Go.
  - Système d'exploitation Windows 10 X64 bits.
- **Téléphone 1 :**
  - Processeur Hisilicon kirin 659 2.36Ghz.
  - Mémoire vive de 4 Go.

- Disque dure de capacité 64 Go.
- Système d'exploitation Android 9 Pie.

– **Téléphone 2 :**
- Processeur MT6737v ARM Cortex-A53 1.3Ghz.
- Mémoire vive de 2 Go.
- Disque dure de capacité 16 Go.
- Système d'exploitation Android 7 Nougat.

5.2.2 Outils logiciels

5.2.2.1 *Python*

Python[2] est un langage de programmation interprété, interactif et orienté objet. Il comprend des modules, des classes, des exceptions, un typage de données dynamiques de très haut niveau. Il prend en charge plusieurs paradigmes de programmation au-delà de la programmation orientée objet, tels que la programmation procédurale et fonctionnelle.

Python combine une puissance remarquable avec une syntaxe très claire. Il a des interfaces avec de nombreux appels système et bibliothèques, ainsi qu'avec divers systèmes de fenêtres, et il est extensible en C ou C ++. Il est également utilisable comme langage d'extension pour les applications qui nécessitent une interface programmable. Enfin, Python est portable : il fonctionne sur de nombreux systèmes, y compris Linux, Mac OS et Windows [44]. La figure (5.2) présente le logo de Python.

FIGURE 5.2: Icone Python

2. https://www.Python.org/

5.2.2.2 *PyCharm IDE*

PyCharm [3] est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciels de gestion de versions, et supporte le développement web avec Django. Il est Développé par l'entreprise tchèque JetBrains.

C'est un logiciel multi-plateformes qui fonctionne sous Windows, Mac OS et Linux. Il est décliné en édition professionnelle, diffusée sous licence propriétaire, et en édition communautaire diffusée sous licence Apache. La figure (5.3) présente le logo de Pycharm IDE.

FIGURE 5.3: Icone Pycharm IDE

5.2.2.3 *OpenCV*

OpenCV (Open Source Computer Vision Library) [4] est une bibliothèque de logiciels, open source de vision par ordinateur et d'apprentissage automatique. OpenCV a été conçu pour fournir une infrastructure commune pour les applications de vision par ordinateur. Étant un produit sous licence BSD, OpenCV facilite l'utilisation et la modification du code par les entreprises et les chercheurs.

La bibliothèque possède plus de 2500 algorithmes optimisés, qui comprennent un ensemble complet d'algorithmes de vision par ordinateur et d'apprentissage automatique. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, identifier des objets, classer les actions humaines dans des vidéos, suivre les mouvements de caméra, suivre des objets en mouvement, extraire des modèles 3D d'objets, produire des nuages de points 3D à partir de caméras stéréo et assembler des images, pour produire une haute

3. https://www.Jetbrain.com/Pycharm/
4. https://www.OpenCV.org/

résolution de l'image d'une scène entière, etc. OpenCV compte plus de 47 000 utilisateurs communautés et un grand nombre de téléchargements, dépassant les 18 millions. La bibliothèque est largement utilisée dans les entreprises, les groupes de recherche et les organismes gouvernementaux.

Elle possède des interfaces C++, Python, Java et MATLAB et prend en charge Windows, Linux, Android et Mac OS. OpenCV est écrite nativement en C++ et possède une interface basée sur des modèles qui fonctionnent de manière transparente avec les conteneurs STL. La figure (5.4) présente le logo d'OpenCV.

FIGURE 5.4: Icone OpenCV

5.2.2.4 *Tensorflow*

TensorFlow [5] est un outil open source d'apprentissage automatique développé par Google. Le code source a été ouvert le 9 novembre 2015 par Google et publié sous licence Apache. Il est fondé sur l'infrastructure DistBelief, initiée par Google en 2011, et il est doté d'une interface pour Python et Julia.

TensorFlow est l'un des outils les plus utilisés en Intelligence artificielle dans le domaine de l'apprentissage automatique. [45]. La figure (5.5) présente le logo de Tensorflow.

5. https://www.tensorflow.org/

FIGURE 5.5: Icone Tensorflow

5.2.2.5 *Keras*

Keras [6] est une API d'apprentissage profond (Deep Learning), écrite en Python, exécutée au-dessus de la plateforme TensorFlow. Elle a été développé dans le but de permettre une expérimentation rapide, pour pouvoir passer de l'idée au résultat le plus rapidement possible.

Keras est l'API de haut niveau de TensorFlow 2.0 : une interface approchable et hautement productive pour résoudre les problèmes d'apprentissage automatique. Elle fournit des abstractions et des blocs de construction essentiels pour développer et expédier des solutions d'apprentissage automatique à grande vitesse d'itération. Elle a été développé par François chollet dans le cadre du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) [46].

La figure (5.6) présente le logo de Keras.

FIGURE 5.6: Icone Keras

6. https://www.keras.io/

5.2.2.6 *Pickle*

Pickle est une bibliothèque sous python, qui implémente des protocoles binaires pour la sérialisation et la désérialisation d'une structure d'objet python.

- **Sérialisation** : En anglais « **Pickling** » est le processus pour lequel, une hiérarchie d'objets python est convertie en un flux d'octets.
- **Désérialisation** : En anglais « **Unpickling** » est l'opération inverse, par laquelle un flux d'octets (à partir d'un fichier binaire ou d'un objet de type octets) est reconverti en une hiérarchie d'objets.

5.2.2.7 *Nativescript*

NativeScript [7] est un framework JavaScript multi-plateformes qui permet de développer des applications IOS et Android natives, à partir d'une seule base de code. Le framework fournit un accès JavaScript aux API natives, aux interfaces utilisateurs et aux moteurs d'iOS et d'Android.

En utilisant JavaScript ou TypeScript, il est possible de créer un projet qui s'intègre dans une application IOS ou Android avec une expérience utilisateur entièrement native [8]. La figure (5.7) présente le logo de Nativescript.

FIGURE 5.7: Icone Nativescript

5.2.2.8 *Angular*

Angular [9] est une plateforme pour la création d'applications client à l'aide de HTML et TypeScript. Angular est écrite en TypeScript, elle implémente les fonctionnalités de base et celles facultatives, sous la forme d'un ensemble de bibliothèques, importées dans les applications. La figure (5.8) présente le logo d'Angular.

7. https://www.nativeScript.org/about/
8. https://www.nativescript.org/faq/whatisnativescript
9. https://www.angular.io/docs/

FIGURE 5.8: Icone Angular

5.2.2.9 *Firebase*

Firebase[10] est une plateforme mobile de Google qui facilite la création de back-end, elle permet de développer rapidement des applications pour mobiles et pour le web. Elle est créée en 2011 par James Tamplin et Andrew Lee.

Firebase n'est pas seulement une base de données, Il n'y a vraiment aucune limite aux types d'applications qui peuvent être aidées par les produits Firebase. Il n'y a que des limites aux plates-formes, sur lesquelles elle peut être utilisée. IOS et Android sont les principales cibles des SDK Firebase, et il existe une prise en charge croissante du Web , Flutter , Unity et C ++. La figure (5.9) présente le logo de Firebase.

FIGURE 5.9: Icone Firebase

10. https://www.meduim.com/firebase-developers/what-is-firebase

5.3 Construction du dataset

La partie suivante représente l'ensemble des traitements effectués pour avoir une base d'images pré-traitées. Cette base est utilisée comme dataset pour l'entrainement du modèle de reconnaissance.

5.3.1 Dataset proposé pour la détection des numéros du tour

Afin de construire notre dataset, nous commençons d'abord, par la captation d'images brutes, depuis l'environnement considéré. Nous avons opté pour la prise des photos directes capturées dans différentes mairies et annexes de la commune de Jijel. Elles ont été prises sous différents angles et éclairages. Ce dataset est constitué de 2000 images annotées à la main. L'image (5.10) illustre ce dataset.

Figure 5.10: Illustration du dataset YOLO

5.3.2 YOLOv3 (You Only Look Once)

Afin de permettre la reproduction de l'étape de détection d'objets avec l'algorithme YOLOv3, déjà présenté dans le chapitre 4, nous présentons ci-dessous les étapes de la configuration et la mise en oeuvre de ce dernier.

5.3.2.1 *Étapes du YOLOv3*

- Créer le fichier yolo-obj.cfg avec le même contenu que dans yolov3.cfg et :
  - Changer la ligne Batch vers Batch= 64
  - Changer la ligne subdivision vers subdivision= 16
  - Changer la ligne max_batches en (classes * 2000) par exemple max_batches = 6000 si vous entraînez pour 3 classes
  - Changer la ligne steps à 80 et 90 de max_batches, par exemple steps = 4800,5400 si max_batches=6000
  - Changer la ligne classes = 80 avec votre nombre d'objets dans chacune des 3 couches [yolo] :
  - Changer la ligne filters = 255 en filters = (classes + 5) x3 dans les 3 [convolutionnels] avant chaque couche [yolo] Donc, si classes = 1 doivent être filtres = 18. Si classes = 2, alors écrivez filters = 21.
- Créer le fichier obj.names dans le répertoire build\darknet\x64\data\avec les noms des objets, chacun dans une nouvelle ligne
- Créer le fichier obj.data dans le répertoire build\darknet\x64\data\contenant par exemple :
  classes= 2
  train = data\train.txt
  valid = data\test.txt
  names = data\obj.names
  backup = backup\

- Placer les fichiers image des objets dans le répertoire build\darknet\x64\data\obj\
  Il est nécessaire d'étiqueter chaque objet sur les images du jeu de données (en marquant des boîtes délimitées d'objets), YOLO créera un fichier .txt pour chaque fichier image dans le même répertoire et avec le même nom, mais avec l'extension .txt, et mettra dans le fichier : numéro d'objet et coordonnées d'objet sur cette image, pour chaque objet dans une nouvelle ligne :
  <object-class><x_center><y_center><width><height>

Où :
<object-class> : numéro d'objet entier de 0 à (classes-1)
<x_center><y_center><width><height> : valeurs flottantes par rapport à la largeur et la hauteur de l'image, elles varient entre [0,1]

- Créer le fichier train.txt dans le répertoire build\darknet\x64\data\, avec les noms de fichiers des images, chaque nom de fichier sur une nouvelle ligne par exemple :
  data\obj \img1.jpg
  data\obj \img2.jpg

- Télécharger pre-trained weights pour les couches convolutives et le placer dans le répertoire build\darknet\x64\
- Commencer l'entrainement en utilisant la ligne de commande :
  « darknet.exe detector train data\obj.data yolo-obj.cfg yolov4.conv.137 »

- Une fois l'entraînement terminé le résultat sera généré dans le répertoire :
  \darknet\x64\backup\Après chaque 100 itérations, il est possible d'arrêter puis recommencer l'entraînement à partir de ce point. Par exemple, après 2000 itérations, il est possible d'arrêter la formation et, plus tard, simplement commencer la formation en utilisant : «darknet.exe detector train data\obj.data yolo-obj.cfg backup\yolo-obj_2000.weights» [47].

5.3.3 Annotation

L'une des étapes incontournable pour faire le YOLO est l'annotation des images à la main. Pour l'annotation des images, nous avons utilisé l'outil **BBox Label Tool** fournit par [48].

5.3.3.1 *Définition BBox Label Tool*

C'est un outil simple pour étiqueter les boites de délimitation d'objets dans les images, implémenté avec python TKinter [48]. L'interface principale du BBox Label Tool est représentée dans la figure (5.11)

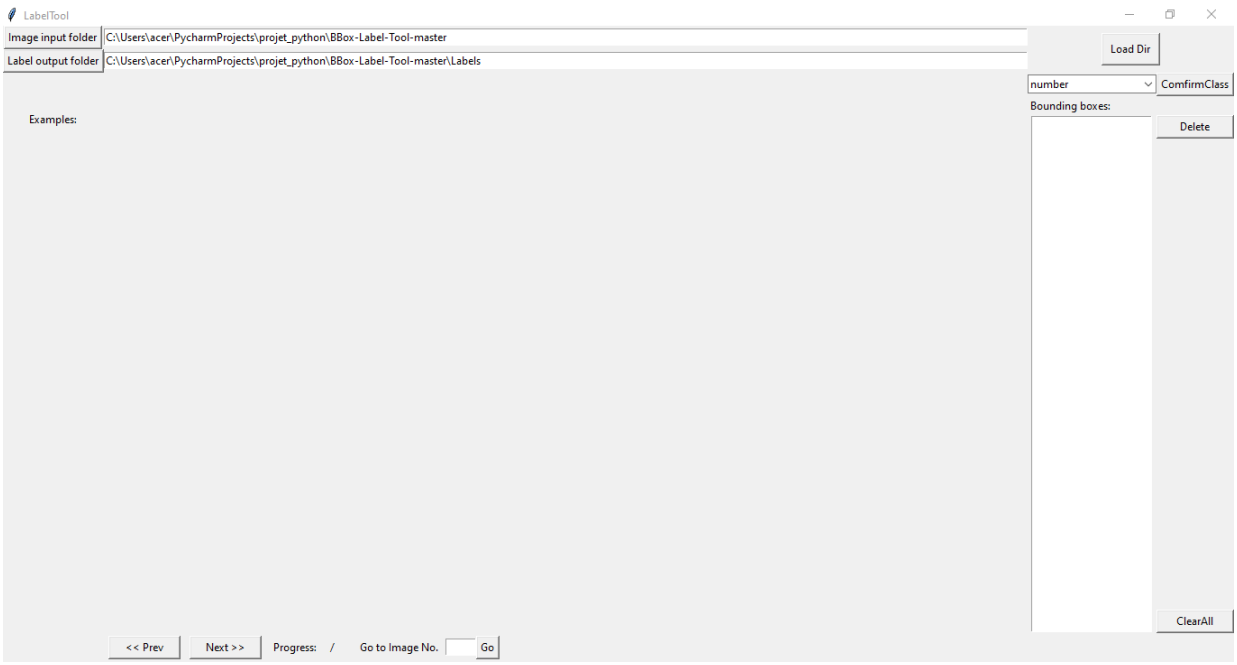


FIGURE 5.11: Interface principale BBox Label Tool [48]

La figure (5.12) montre un exemple de la phase d'annotation, effectuée par nous mêmes.

FIGURE 5.12: Utilisation BBox Label Tool sur notre dataset

La figure (5.13) montre le résultat de l'algorithme YOLOV3 sur notre plaque

FIGURE 5.13: Résultats du détecteur YOLO

5.3.4 Dataset de reconnaissance

Nous avons collecté les images, issues du détecteur YOLOv3. Ainsi, le dataset est constitué de 1071 images réelles. Ces images ont été augmentées à 6304 images par des transformations géométriques (transformations affines, translations, rotations, cisaillements). Des exemples sont présentés dans la figure (5.14) :

FIGURE 5.14: Exemples du dataset de reconnaissance

D'autre part, nous avons procédé à une autre augmentation par des plaques de compteur électronique de tour générées automatiquement.

5.3.5 Génération des plaques automatique

Il est difficile d'obtenir un grand nombre d'images réelles étiquetées. Pour cela, le rôle de la génération artificielle de données est très important et permet de pallier au problème du manque de données d'entraînement [49].

Nous avons proposé un algorithme pour générer des plaques, assez proches d'une plaque de compteur électronique de tours présente dans les mairies. Voici les étapes de l'algorithme

Algorithm 3 Algorithme de génération proposé

Choisir un fond de plaque aléatoire.
Choisir un nombre de 3 chiffres aléatoires.
Mettre le nombre sous la forme de sept segments.
Générer la plaque.
Appliquer un degré aléatoire de luminosité sur la plaque.
Augmenter avec des transformations géométriques.
Ajouter des bruits et des flous.
Retourner la plaque.

Dans ce qui suit, nous allons détailler chaque étape de l'algorithme :

5.3.5.1 *Choisir un fond de plaque aléatoire*

Nous avons choisi un des 3 fonds de plaques de compteur électronique de tour. La figure (5.15) illustre les 3 fonds disponibles

FIGURE 5.15: Fonds aléatoires

5.3.5.2 *Choisir un nombre de 3 chiffres aléatoires*

Dans cette étape, nous avons choisi 3 chiffres aléatoires, qui constituent un nombre entre 000 et 999

5.3.5.3 *Mettre le nombre sous la forme Sept Segments*

Comme son nom l'indique, cet afficheur possède 7 segments. Chaque segment est une portion de l'afficheur, qui est allumée ou éteinte pour réaliser l'affichage. Un schéma de cet afficheur est représenté dans la figure (5.16)

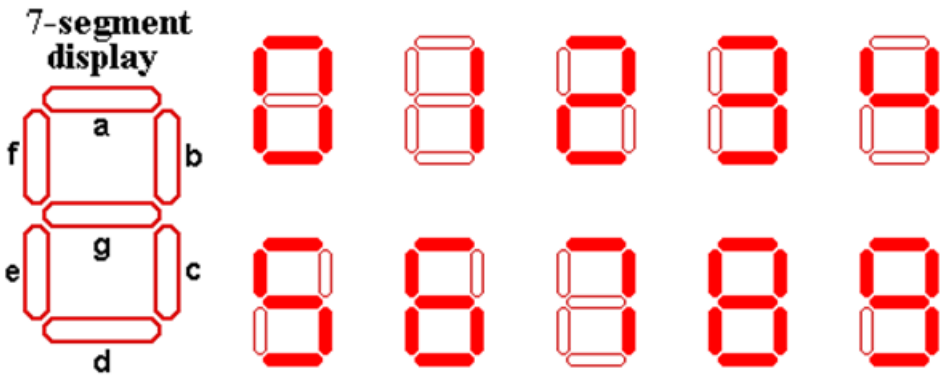


FIGURE 5.16: Format sept segments

Nous avons implémenté cet algorithme avec python TKinter de telle façon qu'un chiffre est représenté par sept champs correspondants aux sept segments de l'afficheur.

Par exemple : Le chiffre 1 est représenté par (0, 1, 1, 0, 0, 0, 0) et le chiffre 5 par (1, 0, 1, 1, 0, 1, 1)

5.3.5.4 *Générer la plaque*

Le traitement de toutes les étapes précédentes permet de former la plaque. Voici un exemple d'une plaque générée basique dans la figure (5.17)

FIGURE 5.17: Plaque générée basique

5.3.5.5 *Appliquer le degré de luminosité*

Afin d'éclaircir ou assombrir l'image originale, un filtre aléatoire est appliqué.

5.3.5.6 *Augmenter avec des transformations géométriques*

Nous appliquons à notre plaque des transformations géométriques comme la rotations, la translations et/ou le cisaillements.

Tous les traitements ont été rassemblés, en ajoutant un bruit et un flou aux images résultantes

La figure (5.18) montre des exemples de plaques générées avec notre algorithme

FIGURE 5.18: Exemple de plaques générées finales

5.4 IMPLÉMENTATION ET MISE EN OEUVRE

L'étape de l'implémentation est constituée de trois parties détaillées ci-dessous :

5.4.1 Apprentissage profond

5.4.1.1 *Pré-traitement*

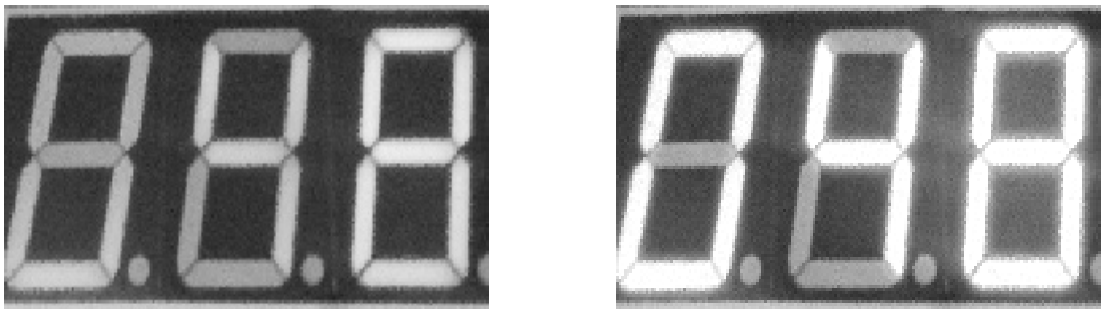
C'est l'une des étapes incontournables pour la préparation de l'entraînement du modèle.

Lecture d'images

Les images résultantes de l'exécution du détecteur YOLO seront lues, puis redimensionnées à 64X64 Pixels, ce qui convient à l'entrée du modèle en utilisant la bibliothèque OpenCV déjà présentée.

Conversion de l'espace de couleur

Cette étape consiste à traiter l'image, principalement, la convertir en niveau de gris pour que la phase du Deep Learning soit plus rapide. Dans notre cas, les informations présentes sur l'image ne seront pas clairement visibles. Pour cela, nous avons opté pour la conversion en canal rouge car le rouge étant la couleur d'émission des tubes de l'afficheur sept segments, ces conversion ont été réalisées à l'aide de la bibliothèque OpenCV déjà présenté précédemment. La figure (5.19) représente une image en niveau de gris corrompue et une autre en canal rouge qui met en évidence les caractéristiques pertinentes (selon notre besoin) de l'image.



(a) Image en niveau de gris (b) Image en canal rouge

FIGURE 5.19: Conversion de l'image

Sérialisation

C'est le processus pour lequel nous allons convertir notre dataset en flux d'octets, en utilisant la bibliothèque Pickle déjà présenté précédemment, pour nous permettre de l'utiliser dans l'entrainement de notre modèle.

Codage du numéro de la plaque

Dans cette étape, nous allons coder le numéro présent sur l'image, de sorte que le réseau puisse l'assimiler. Nous allons coder les trois chiffres de la plaque en vecteur de

trente chiffres, dix pour chaque chiffre. La figure (5.20) illustre ce processus.

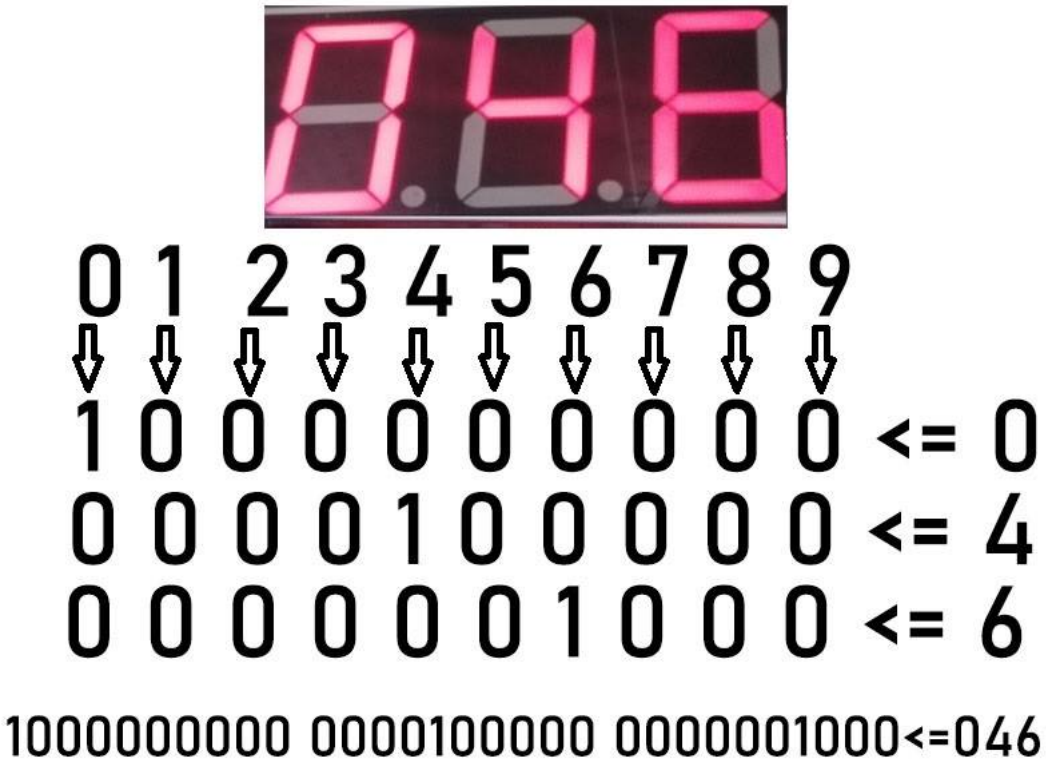


FIGURE 5.20: Codage du numéro

5.4.1.2 *Expérimentations paramétriques*

Dans ce qui suit, nous allons présenter les expérimentations avec plusieurs tests paramétriques, effectués sur le modèle proposé et présenté précédemment dans le chapitre 4.

Expérimentation entre VGG et Alexnet

Dans cette section, nous allons montrer les résultats de l'entrainement des modèles VGG [24] et AlexNet [23] sur notre Dataset. Après l'entrainement, ces modèles ont été testés sur une base de tests constitués de 300 images. Les résultats sont présentés dans le tableau (5.1).

	Dataset	Taille du lot	Nombre d'époque	Taux du réseau
VGG	Notre dataset	32 images	30	85,6%
Alexnet	Notre dataset	32 images	30	87%

TABLE 5.1: Comparaison entre les modèles VGG et AlexNet

D'âpres le tableau (5.1), nous remarquons que la précision des deux modèles est quasi-identique. Nous avons donc décidé de faire une hybridation entre ces deux modèles pour construire l'architecture de notre modèle.

Expérimentation par rapport à la taille de la couche entièrement connecté (FC)

Après avoir décider l'architecture principale de notre modèle, il nous reste à déterminer la taille de la couche entièrement connectée. Trois tests ont été effectués avec 512, 2048 et 4096 neurones comme taille de la couche. Le tableau (5.2) illustre les résultats obtenus.

	Dataset	Taille du lot	Nombre d'époque	Taux du réseau
512	Notre dataset	32 images	30	74.3%
2048	Notre dataset	32 images	30	95.3%
4096	Notre dataset	32 images	30	89.3%

TABLE 5.2: Comparaison entre les modèles avec 512, 2048 et 4096 neurones

D'après le tableau ci dessus, nous remarquons que le modèle avec 2048 neurones dans la couche entièrement connectée, apporte une précision remarquable de 95.3 % contrairement aux les autres modèles avec 512 et 4096 neurones.

Choix de l'optimisateur

Les optimisateurs les plus utilisés dans l'apprentissage profond, sont : ADAM et SGD [50]. Nous avons testé notre modèle avec les deux optimisateurs afin de décider lequel est le plus compatible à notre architecture. Le tableau (5.3) illustre les résultats.

	Dataset	Taille du lot	Nombre d'époque	Taux du réseau
SGD	Notre dataset	32 images	30	95.3%
ADAM	Notre dataset	32 images	30	79.6%

TABLE 5.3: Comparaison entre les modèles avec SGD et ADAM

Nous constatons que le modèle avec l'optimisateur SGD apporte une précision plus importante que celle du modèle avec ADAM. Pour cela, nous avons décidé d'utiliser cet optimisateur.

5.4.1.3 *Présentation du code source d'entrainement du modèle proposé*

Dans cette partie, nous allons présenter le code source de l'entrainement du modèle proposé. La figure (5.21) présente le code source du modèle proposé sous le langage Python.

```
x = pickle.load(open("/content/gdrive/My Drive/x_brute", "rb"))
y = pickle.load(open("/content/gdrive/My Drive/y_brute", "rb"))

x, x_test, y, y_test = train_test_split(x, y, test_size=0.005,random_state=42)
x, x_val, y, y_val = train_test_split(x, y, test_size=0.1,random_state=42)

model = Sequential()
model.add(Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(64, 64, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(96, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(196, (3,3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(196, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(2048, activation='relu'))
model.add(Dense(30, activation='sigmoid'))
```

FIGURE 5.21: Code source du modèle d'entrainement

5.4.1.4 *Présentation des résultats d'entrainement du modèle proposé*

La précision et la perte de l'entrainement du modèle proposé sont illustré dans la figure (5.22).

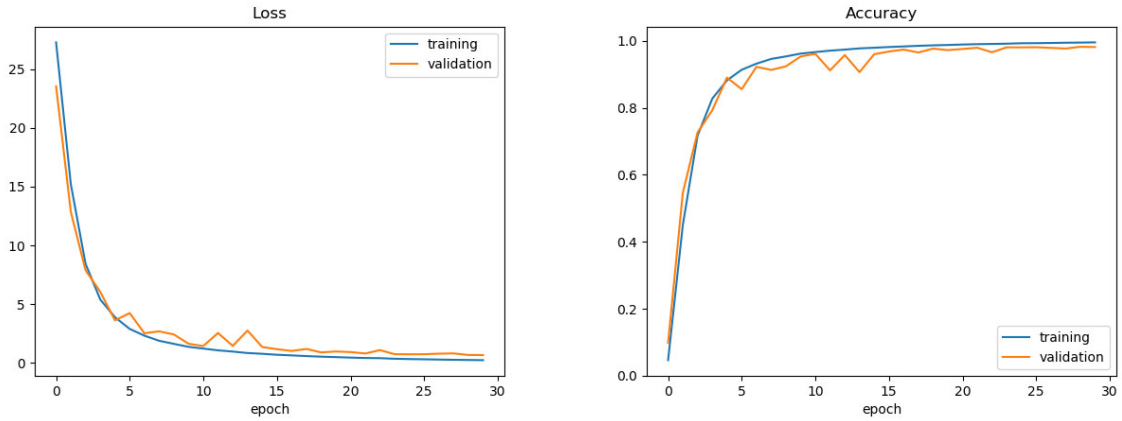


FIGURE 5.22: Illustration de la précision et la perte du modèle proposé

5.4.2 Application Serveur

Dans cette section, nous présentons l'application développée, qui va interagir avec l'API Firebase et exécuter le modèle de l'apprentissage profond et finalement, extraire le numéro.

Premièrement, l'application serveur récupère l'image depuis Firebase. L'apprentissage profond prédit le numéro, cette prédiction sera stocké dans le "Real Time Database" de Firebase. Une notification push est alors déclenchée et envoyée aux citoyens concernés. Les figures (5.23) et (5.24) présentent les interfaces de notre application.

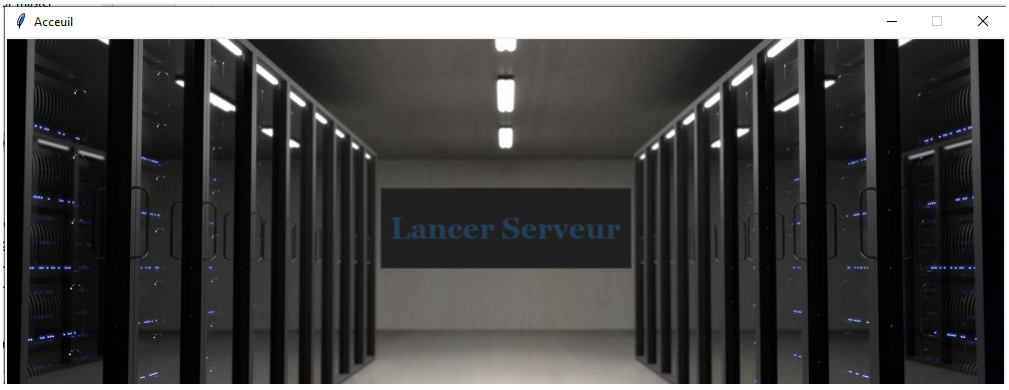


FIGURE 5.23: Présentation de l'interface d'accueil Python

FIGURE 5.24: Présentation de l'interface du traitement Python

5.4.3 Application Mobile

Dans ce qui suit, nous allons détailler le traitement et les technologies utilisées à la création de notre application mobile et ses interfaces, développé sous les plateformes déjà présentées précédemment, NativeScript combiné avec Angular parce qu'ils ont plus de choix et de flexibilité, ils Permettent de déployer l'application sur différentes Plateformes comme IOS et Android.

5.4.3.1 *Les technologies utilisées*

Notification push

Avec le développement de l'Internet, les appareils mobiles sont désormais utilisés de manière omniprésente. Les notifications push en temps réel deviennent une partie très importante de l'utilisation des Smartphones.[51]

Ces alertes ont été initialement créées et popularisées avec les appareils mobiles BlackBerry il y a des années. Des notifications étaient envoyées, lorsque les utilisateurs recevaient de nouveaux e-mails dans leur boîte de réception. Cependant, de nos jours, les

alertes sont moins fréquemment utilisées pour les e-mails.

Une notification push est un bref message ou une alerte qui est envoyée via une application déjà installée si la réception de ces messages a été déjà activée. Que se soit un IPhone, un Android ou toute autre marque de téléphone, l'utilisateur, peut toujours recevoir ces notifications. Pour offrir plus d'accessibilité, même si l'application n'est pas ouverte au moment de la notification, le message peut être visible. Cela permet d'atteindre un large nombre de personnes en diffusant le message à un groupe entier en même temps [52].

Code QR

QR code est un code noir et blanc, composé de trois carrés dans les coins, et de petits carrés à l'intérieur. QR est l'abréviation de « Quick Response » en anglais, elle signifie que le contenu du code peut être décodé rapidement, après avoir été lu, c'est une méthode de collecte de données rapide, facile, précise et automatique [54].

Le code QR a été développé pour la première fois en 1994 par Denso Wave au Japon. À partir de ce moment, il est devenu largement utilisé comme marque d'identification pour toutes sortes de produits commerciaux, publicités et autres annonces publiques.

Ce code a plusieurs utilisations comme le Passage d'un appel téléphonique, envoie d'un sms, prise d'un rendez-vous, réservation des billets de transport et comme dans notre cas, l'inscription dans une application [53].

Nous avons proposé un nouveau concept de tickets dans l'administration algérienne, un ticket doté d'un code QR qui permet l'inscription directe à l'application, destiné aux citoyens, sans remplir les informations manuellement. La figure (5.25) illustre le ticket proposé.

FIGURE 5.25: Exemples de tickets proposés

5.4.3.2 *Présentation des interfaces*

Notre application se compose principalement de quatre interfaces.

Fenêtre d'authentification

Toute application doit avoir un système d'authentification afin d'en sécuriser l'accès. Pour notre cas, nous avons utilisé le système d'authentification de Firebase. Pour faciliter la tâche aux citoyens. Il suffit que l'application soit connectée à internet et que l'utilisateur saisisse une adresse e-mail et un mot de passe valide déjà inscrit chez Firebase et cliquer sur « Connexion », alors l'utilisateur aura l'accès à l'application et les informations du client seront sauvegardées dans la base de données. La figure (5.26) représente l'interface d'authentification des utilisateurs tel que :

- **La fenêtre à gauche** : est la fenêtre qui s'affiche, lorsque l'application se lance.
- **La fenêtre au milieu** : affichage du message d'erreur si le Smartphone n'est pas

connecté à internet.

– **La fenêtre à droite** : vérification si les données saisies et accès au système.

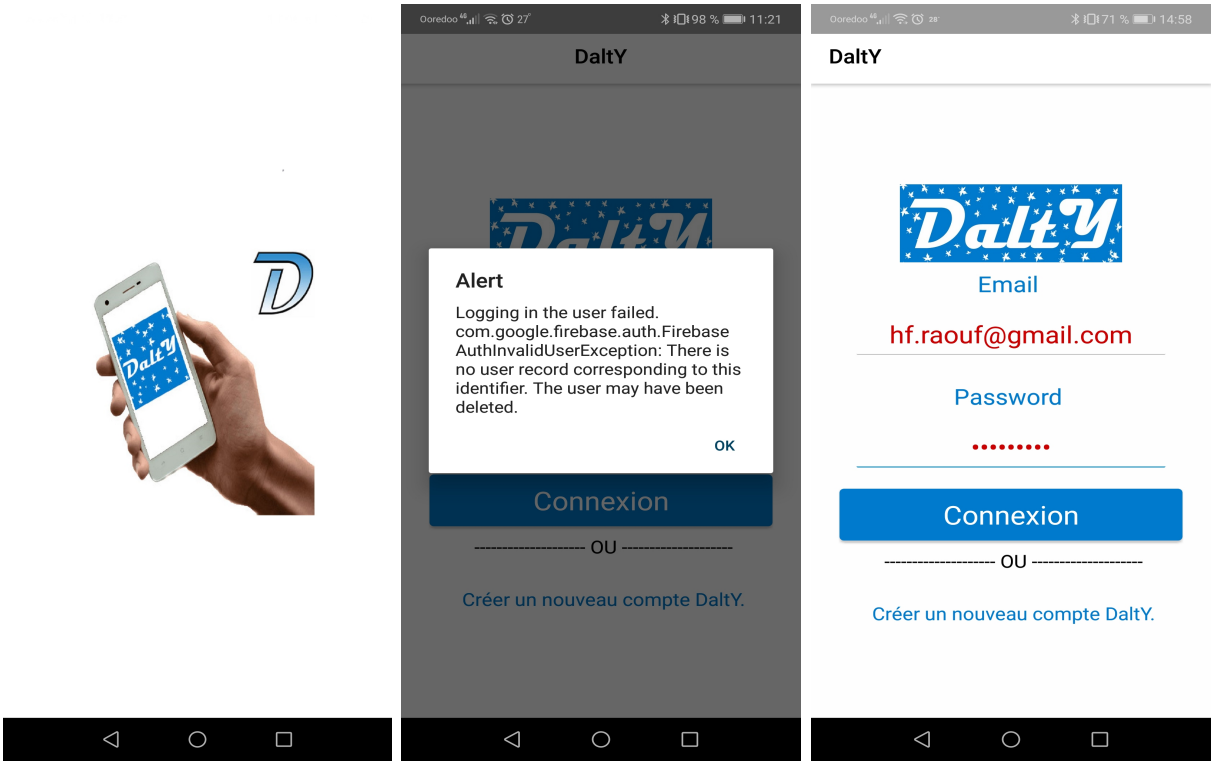


FIGURE 5.26: Fenêtre d'authentification

Fenêtre d'inscription

La fenêtre d'inscription offre aux utilisateurs la possibilité de s'inscrire à notre système via leur email et mot de passe. Ces informations seront automatiquement enregistrées dans la base de données des clients. La figure (5.27) illustre ce traitement.

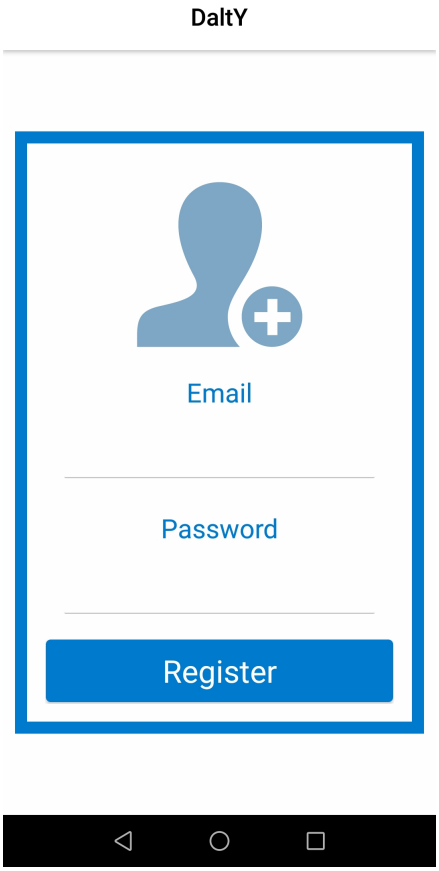


FIGURE 5.27: Fenêtre d'inscription

Fenêtre de formulaire

Dans cette fenêtre, l'utilisateur a deux possibilités : il peut soit saisir le numéro de son ticket manuellement, comme il peut utiliser le bouton « scanner » pour scanner le code QR associé au ticket, et le formulaire se remplit alors automatiquement. Ensuite, l'utilisateur devra entrer le seuil de notification c'est-à-dire combien de personnes restantes dans la file d'attente sont nécessaires pour déclencher l'envoi des notifications.Ce procédé est illustré dans la figure (5.28).

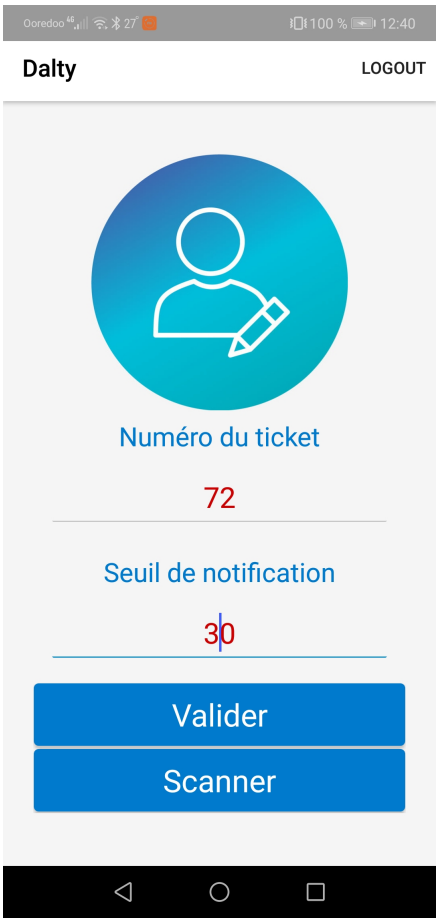


FIGURE 5.28: Fenêtre de formulaire

Page d'accueil

Dans cette fenêtre, l'utilisateur pourra connaître le numéro actuel en traitement dans l'administration ainsi que, le nombre de personnes en attente avant que sont tours n'arrive. La figure (5.29) est une capture de cette fenêtre.

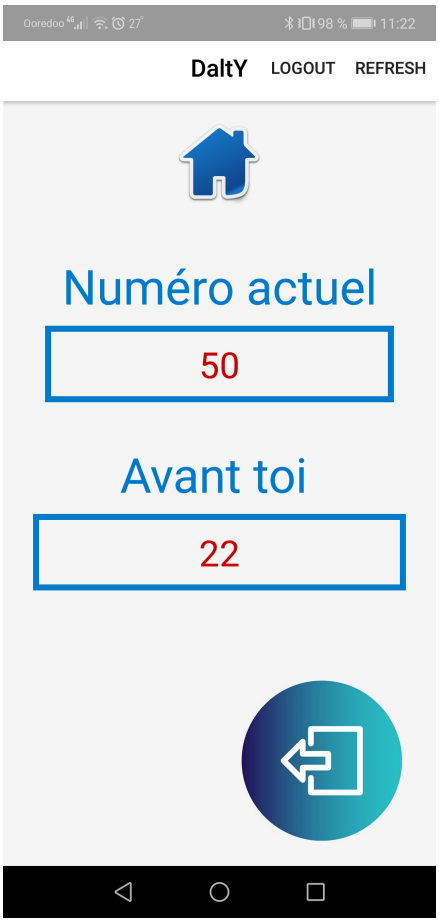


FIGURE 5.29: Page d'accueil

La figure (5.30) illustre un reçoit de notification push.

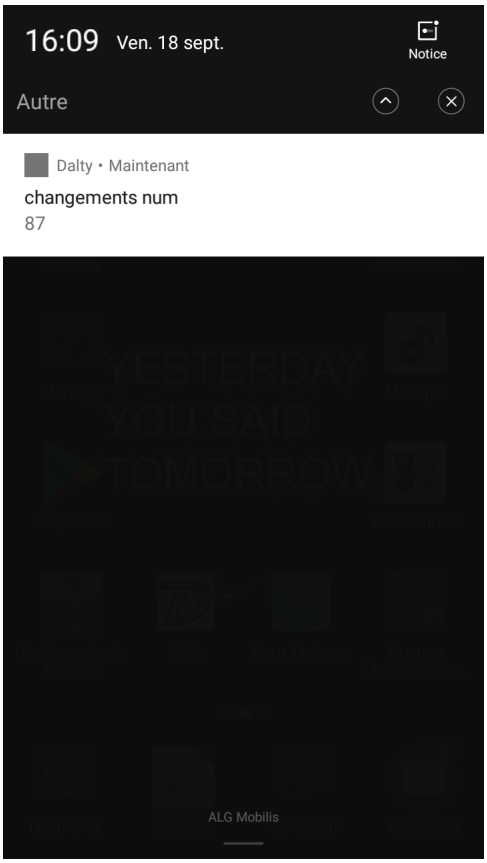


FIGURE 5.30: Illustration du reçoit de notification

5.5 CONCLUSION

Dans ce chapitre, nous avons détaillé l'implémentation de notre application tout en présentant, les outils logiciels et matériels utilisés. Nous avons expliqué notre cheminement expérimental afin de fixer les différents paramètres de notre modèle. Enfin nous avons illustré le traitement et les interfaces de l'application sur les différents systèmes que nous avons créés.

Conclusion et perspectives

Le cadre de ce projet de fin d'étude, a pour objectif de développé un système basé sur l'apprentissage profond, pour la gestion des tours dans les administrations algériennes.

Notre système a permis de cerner les différents problèmes rencontrés au début du projet, par la proposition d'un modèle CNN pour reconnaître les chiffres de la plaque du compteur électronique. Pour l'entraînement de notre modèle, nous avons construit un dataset inédit, capturé par nos soins aux différentes mairies. Après cette étape, ces chiffres des plaques ont été détectés en utilisant le détecteur YoloV3. Ensuite, nous avons entraîné notre modèle sur ce dataset, les résultats de la reconnaissance ont atteint une précision de 95 %.

En plus, nous avons développé une application mobile destinée aux citoyens, dans laquelle, ils vont se connecter et saisir le numéro du ticket et le seuil de notification. L'application va afficher le numéro actuel traiter dans l'administration et elle va leur permettre de recevoir des notifications lors du changement du numéro au niveau du guichet de l'administration.

En parcourant les différents chapitres de ce mémoire, nous avons présenté les notions fondamentales des réseaux de neurones artificiels et leur mise en oeuvre dans le procédé d'apprentissage profond, tout en détaillant, l'architecture des réseaux de neurones convolutionnels. Enfin, dans le dernier chapitre nous avons détaillé notre contribution à la résolution de la problématique de gestion des tours dans les administrations algériennes.

Dans le futur et comme perspectives, nous proposons, dans la continuité de ce projet,

de procéder à la collecte d'un grand nombre d'images de la plaque de compteur électronique de tours pour bien alimenter l'apprentissage du modèle proposé. Cela, permettra d'améliorer, certainement, la précision de reconnaissance, et améliorer cette précision avec un modèle prédictif qui permet de minimiser le taux d'erreur, par la prédiction du numéro actuel en se basant sur le numéro précédent.

Enfin, nous souhaitons que notre système soit bénéfique, puisse minimiser l'impact du COVID-19, et sera utilisé réellement, dans les administrations pour qu'il soit un premier pas pour une e-gouvernance des administrations algériennes.

Bibliographie

[1] Jean Philippe Renard, *Réseaux neuronaux,une introduction accompagnée d'un modèle Java* Vuibert, 2006.

[2] Claude Touzet, *Les réseaux de neurones Artificiels, Introduction au connexionnisme*. EC2, Collection de l'EERIE,N. Giambiasi, 1992.

[3] Franck Rosenblatt *The perceptron : a probabilistic model for information storage and organization in the brain*, Psychological review, Volume 36, Numéro 6, 1958.

[4] Warren, S.McCulloch, Walter Pitts *A logical calculus of the ideas immanent in nervous activity.*Bulletin of Mathematical Biology, Volume 52, No 1/2, Pages 99-115, 1990

[5] Youcef Djeriri *Les réseaux de neurones artificiels*, 2017.

[6] S. I. Gallant,*Perceptron-based learning algorithms*, IEEE Transactions on Neural Networks, Volume 1, No 2, Pages 179-191, 1990.

[7] Brain K.Spears *contemporary machine learning : a guide for practioners in the physical sciences*, 2017.

[8] Sushmita Mitra, Sankar K.Pal *Fuzzy multi-layer perceptron, inferencing and rule generation*, IEEE Transactions on Neural Networks, Volume 6, Numéro 1, pages 51-63, 1995.

[9] Marc Parizeau *Le perceptron multi couche* Departement génie electrique et de geni informatique université LAVAL 2004

[10] Marc Parizeau *Algorithme de propagation et rétropropagation des erreurs* Departement génie electrique et de geni informatique université LAVAL 2004

[11] Yan LeCun, Yoshua Bengio, Geoffrey Hinton *Deep Learning*, Nature, Volume 521, Pages 44-436,2015

[12] Ivan Vasilev, Daniel Slater, Gianmario Spacagna, Peter Roelants, Valentino Zocca *Python Deep Learning*, Packt, 2019

[13] Zhang Lu, Tan Jianjun, Han Dan, Zhu Hao *From machine learning to deep learning : Progress in machine intelligence*, Drug Discovery Today, Volume 22, 2019.

[14] Antonio Guli *Deep Learning with keras* Packt, 2017

[15] Haohan Wang, Bhiksha Raj *on the origin of Deep Learning*

[16] Li Dong, Dong Yu *Deep Learning : Methods And Applications*, Fondation and trends in signal processing, Volume 7, 2013.

[17] David Hunter Hubel, Torsten Nils Wiesel *Receptive Fields Binocilar Interaction And Functional Archtecture In Cat's Visual Cortex*, ScienceDirect, Volume 38, Pages 103-114, 2003.

[18] Yan LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W.Hubbard, L.D. Jackel *Handwritten Digit Recognition with a Back-Propagation Network*, Advances in neural information processing systems, Volume 2, 1990.

[19] Vincent Lorrain *Etude et conception de circuits innovants exploitant les caractéristiques des nouvelles technologies mémoires résistives*, Thèse de doctorat, Université de Paris-Saclay, 2018.

[20] Paul Blanc-Durand *Réseaux de neurones convolutifs en médecine nucléaire : applications à la segmentation automatique des tumeurs gliales et à la correction d'atténuation en TEP/IRM*, Thèse de doctorat, Université de Paris Descartes, 2018.

[21] Phung, Rhee *A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets*, Applied Sciences, Volume 9, Pages 319-345, 2019.

[22] Yan LeCun, Patrick Haffner, Léo Bottou, Yoshua Bengio *Object Recognition with Gradient-Based Learning*, Shape Contour and groupping in ComputerVision, Volume 1681, Pages 319-345, 2000.

[23] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton *ImageNet Classification with Deep Convolutional Neural Networks*, Neural Information Processing systems, Volume 25, 2012.

[24] Karen Simonyan, Andrew Zisserman *Very Deep Convolutional Network s For Large-Scale Image Recognition*, arXiv 1409, 2014

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Deep Residual Learning for Image Recognition*, IEEE Conference On computerVision and Pattern Recognition CVPR, Pages 770-778, 2016.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Identity Mapping in Deep Residual Network*, Volume 9908, Pages 630-645, 2016.

[27] Christian Szegedy, Wei Liu, Vincent Vanhoucke, and al *Going Deeper With Convolution* , IEEE Conference On computerVision and Pattern Recognition CVPR, Pages 1-9, 2015.

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shelens *Rethinking The Inception Architecture For Computer Vision*, 2016.

[29] Atanas Atanassov *Advanced Software Architecture of an Automatic Vehicle Number Plate Recognition System*, Journal of the university of chemical Technology and Metallurgy, Volume 47, Pages 77-82, 2012.

[30] Sarfraz Muhammad, Ahmed Mohammed Jameel, Ghazi *License Plate Recognition System : Saudi Arabian Case*, Computer-Aided Intelligent Recognition Techniques and Applications, Volume 10, Pages 36-41, 2003.

[31] Shreeja Chakraborty, Ranjan Parekh *An Improved Template Matching Algorithm for Car License Plate Recognition*, International Journal of Computer Applications, Volume 118, Numéro 25, 2015.

[32] Hanifi Majdoulayne *Extraction de caractéristiques de texture pour la classification d'images satellites*, Thèse de doctorat, Université de Toulouse III-Paul Sabatier, 2009.

[33] Bilel Benjdira, Taha Khursheed, Anis Koubaa, Adel Ammar, Kais Ouni *Car Detection using Unmanned Aerial Vehicles : Comparison between Faster R-CNN and YOLOv3*, 1st International Conference on Unmanned Vehicle Systems-Oman (UVS), Pages 1-6, 2019.

[34] Sérgio Montazzolli, Claudio Jung *Real-Time Brazilian License Plate Detection and Recognition Using Deep Convolutional Neural Networks*, 30th SIBGRAPI Conference on Graphics, Patterns and Images, Pages 55-62, 2017.

[35] Shan Du, Mahmoud Ibrahim, Mohamed Shehata, Wael Badawy *Automatic licence plate recognition ALPR, A state-of-the Art Review*, IEEE Transaction on circuitsD and systems for video technology, Volume 23, Number 2, Pages 311-325, 2013.

[36] Ziad Selmi, Mohamed Ben Halima, Umapada Pal, Adel Alimi *DELP-DAR System for Licence Plate detection and recognition*, Pattern Recognition Letters, Volume 129, 2019.

[37] Changhao Wu, Shugong Xu, Guocong Song, Shunqing Zhang *How Many Labeled Licence Plate are needed*, Pattern Recognition and Computer Vision, 2018.

[38] Jakub Spanhel, Jakub Sochor, Roman Juranek, Adam Herout, Lukas Marsik, Pavel Zemcik *Holistic recognition of low quality licence plate by CNN using track anotated data*, IEEE International Conference on Advanced Video and signal Based Surveillance , Pages 1-6, 2017.

[39] Pavel Svoboda, Michael Hradis, Lukas Marsik, Pavel Zenck *CNN for Licence Plate motion deblluring*, IEEE Conference on Image Processing ICIP, Pages 3832-3836, 2016.

[40] Chunnu Khawas, Pritam Shah *Application of Firebase in Android App Development-AStudy*, International Journal of Computer Applications, Volume 179, Numéro 46, 2018.

[41] Jiwoong Choi, Dayoung Chun, Hyun Kim, Hyuk-Jae Lee *Gaussian YOLOv3 : An Accurate and Fast Object Detector Using Localization Uncertainty for Autonomous Driving* , Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Pages 502-511, 2019.

[42] Joseph Redmon, Ali Farhadi *YOLOv3 : An Incremental Improvement*, 2014.

[43] Ekaba Bisong *Google Colaboratory*, Building Machine Learning and Deep Learning Models on Google Cloud Platform, Pages 59-64, 2019.

[44] Mark Lutz *Programming python*, O'Reilly Media, 2001.

[45] Sanjay Kumar, Manish Kumar *A Study on the Image Detection Using Convolution Neural Networks and TenserFlow*, 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Pages 1080-1083, 2018.

[46] Sanjay Kumar, Manish Kumar *Introduction to Keras*, Deep Learning with Python, Pages 97-111, 2017.

[47] Alexey AB, Darknet, *https://www.github.com/AlexeyAB/Darknet* Consulté le 15/06/2020.

[48] Xiaqunfeng, BBox-Label-Tool *https://www.github.com/Xiaqunfeng/BBox-Label-Tool* Consulté le 16/06/2020.

[49] Ashish Shrivstava, Tomas Pfisten, Oncel Tuzel, Joshua Sysskind, Wenda Wang and Russell Webb *Learning From Simulated And Unsupervised Images Though Adversarial Training*, In Proceeding of IEEE Conference On Computer Vision And Patterns Recognition CVPR, Pages 2107-2116, 2017.

[50] Nitish Shirish Keskar, Richard Socher,*Improving generalization performance by switching from adam to sgd*, arXiv preprint, 2017.

[51] Zhanlin Ji, Ivan Ganchev, Mairtin O'Droma, Qingjuan Zhao *A Push-Notification Service for Use in the UCWW*, International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Pages 318-322, 2014.

[52] Isikligil Emre, Samakay Semih, Kilinc Deniz *A Prototype Framework for High Performance Push Notifications*, International Journal of Computer Applications, Volume 166, Pages 8-11, 2017.

[53] Jae Hwa Chang *An Introduction To Using QR codes in Scholarly journals*, Science Editing, Volume 1, pages 113-117, 2014.

[54] Yue Liu, Ju Yang, Mingjun Liu *Recognition of QR code with Mobile phones*, 2008 Chinese Control and Decision Conference, Pages 203-206, 2008.

Résumé

Dans ce projet, nous avons mis au point une amélioration du système de gestion des tours dans les administrations algériennes. Notre système est composé de deux parties. Premièrement, d'une API serveur dotée d'un apprentissage profond des chiffres et des plaques électroniques, passé par un modèle convolutionnel. Deuxièmement, d'une application mobile sous réseau pour les clients afin de recevoir des notifications et les informer du numéro affiché actuellement dans l'administration. Les résultats des expériences atteignent 95% de reconnaissances correctes et tout le système est passé sous une expérimentation pratique réelle dans la mairie de la ville de Jijel.

Mots-Clés: Apprentissage profond, vision par ordinateur, réseaux de neurones profonds convolutionnels CNN.

Abstract

In this project, we have developed an improvement for the turn management system in the Algerian administration. Our system is composed of two parts. The first one is an API server endowed with a deep learning classifier, for the numbers of the electronic plate, going through a convolutional model. The second part is a networked mobile application for customers to receive notifications and inform them of the number currently displayed in the administration electronic plate. The results of the experiments reach 95% correct recognition and the whole system went under a real practical experimentation in the town hall of the city of Jijel.

Key-words: Deep Learning, Computer Vision, Deep Convolutional Neuronal Network CNN.

الملخص

في هذا المشروع ، قمنا بتطوير نظام تسيير الأدوار في الإدارة الجزائرية ، ويتكون نظامنا من جزأين : الأول عبارة عن خادم مزود بمعرفة عميقة لأرقام اللوحات الإلكترونية التي تمر عبر نموذج تلافيفي **(CNN)**وثانيًا ، تطبيق جوال متصل بالشبكة للعملاء لتلقي الإخطارات وإبلاغهم بالرقم المعروض حاليًا في الإدارة . وصلت نتائج التجارب إلى 95٪ من التعرف الصحيح وخضع النظام بأكمله لتجربة عملية حقيقية في دار البلدية بمدينة جيجل.

كلمات مفتاحية: التعرف العميق , الرؤية الحاسوبية , الشبكات العصبية العميقة ....