REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA



Faculté de **T**echnologie

Mémoire de fin d'études de Master

Département : Génie électrique
Filière : Télécommunication
Spécialité : Système de télécommunication

Implémentation d'un réseau de neurone covolutionnel CNN pour la reconnaissance automatique des mots manuscrits en arabe

---

Présenté par :

MAHDJOUBI Abderrahmane

HALBIGUE Amira

Proposé et Dirigé par :

Dr. KARA REDA

Dr. BENYEZZA HAMZA

Promotion 2019

Dédicaces

Je dédie ce modeste travail :
Les deux êtres les plus chères au monde, les bougies
Qui m'ont toujours guidé vers le bon chemin :

Mon père, à lui, je dois offrir tout le respect et l'amour pour son soutien et sa tendresse.

Ma mère, la lumière de ma vie, qui a tout fait pour mon réussite et mon bonheur.

A mes frères ISMAIL, SLIMANE et ma sœur IMANE à mon binôme ABDERRAHMEN, à mes chères sœurs a tous les membres de ma famille pour leur aide et courage.

A tous les enseignants durant mes années d'études avec lesquels j'ai beaucoup appris,

Ainsi qu'à toutes les personnes que j'ai connues, qui m'ont aidées, soutenues et encouragées

AMIRA

Dédicaces

Je dédie ce modeste travail :
Les deux êtres les plus chères au monde, les bougies
Qui m'ont toujours guidé vers le bon chemin :

Mon père, à lui, je dois offrir tout le respect et l'amour pour son soutien et sa tendresse.

Ma mère, la lumière de ma vie, qui a tout fait pour mon réussite et mon bonheur.

A mes frères ABDELLAH, BENHAOUA, ZAKARIA, MOHAMED, OUSSAMA et ABDELHAk et mes sœur NOUSSAIBA et ROUMAISSA et ma chère tante FOUZIA à ma binôme AMIRA, à mes amies a tous les membres de ma famille pour leur aide et courage.

A tous les enseignants durant mes années d'études avec lesquels j'ai beaucoup appris,

Ainsi qu'à toutes les personnes que j'ai connues, qui m'ont aidées, soutenues et encouragées

ABDERRAHMANE

Remerciements

*Je remercie en premier lieu mon **DIEU** de m'avoir aidé à défier tous les obstacles, afin de compléter cette thèse.*

C'est avec un immense plaisir que j'exprime mes remerciements ; les plus chaleureux à notre encadreur
Monsieur KARA REDA, qui a toujours suivi ce travail avec intérêt et qui l'a témoigné par des encouragements de toutes sortes.
Nous adressons aussi nos vifs remerciements à Mr HANTABLI HAMZA pour le temps qu'il a consacré et pour les précieuses informations qu'ils nous ont prodigués avec intérêt et compréhension.
Mes sincères remerciements vont à notre Co-encadreur Mr BENYEZZA pour son aide.
Je remercie très vivement les membres du jury Mr BENGHARBIA et Mr BOUHADDA qui ont accepté de juger ce modeste travail.

Nos remerciements s'adressent aussi à nos amis FARADJI MOHAMED et ZERDALI ABDELHAK pour leur aide précieuse durant notre projet,

Enfin, pour tous ceux qui, de près ou de loin, ont rendu ce travail possible, je dis pour la nième fois merci à Mr BENGHARBIA BILLAL.

ملخص:

هناك أنواع كثيرة من الشبكات العصبية يتم تطوير كل نوع لغرض معين، من بين هذه الأنواع اخترنا الشبكات عصبية تلافيفيه الاصطناعية، وقمنا باستخدام التعلم العميق للتعرف على الكلمات المكتوبة بخط اليد وتصنيفها باللغة العربية، بحيث استعملنا في البداية برنامج "ماتلاب" لفهم الشبكات العصبية التلافيفية من خلال قاعدة بيانات الأرقام المكتوبة بخط اليد وبعدها استعملنا لغة البرمجة "بايثون" لقاعدة بيانات الكلمات المكتوبة بخط اليد باللغة العربية، لذلك اقترحنا بنيات مختلفة.

أظهرت النتائج التي تم الحصول عليها أن اختيار الهندسة وعدد التكرارات وحجم المصفاة، لها تأثير كبير لتحقيق نتائج أفضل ومن أجل التحقق من النتائج قمنا باختبار للتصنيف.

كلمات مفتاحية:

الشبكات العصبية، والشبكات العصبية التلافيفية، التعلم العميق، الكلمات المكتوبة بخط اليد باللغة العربية.

Résumer :

Il existe beaucoup de types de réseaux de neurones chaque type étant développé pour un objectif particulier parmi ces types on a choisi les réseaux de neurones convolutionnels (CNN) et on a utilisé l'apprentissage profondeur pour la reconnaissance et la classification de mots manuscrits en langue arabe ou début on a utilisé MATLAB pour bien comprendre le CNN avec la base de donnée de chiffres manuscrits et après on a utilisé le langage Python pour la base de donnée de mots manuscrits en langue arabe, alors on a proposé des différents architectures. Les résultats obtenus ont montré sue le choix de l'architecture, la taille de filtres et le nombre d'époque ont une grande influence pour avoir des meilleurs résultats. Afin de vérifier les résultats nous avons fait un test de classification.

Mots clés :

Les réseaux de neurones, les réseaux de neurones convolutionnels (CNN), l'apprentissage profondeur, mots manuscrits en langue arabe.

Abstract :

There are many types of neural networks each type being developed for a particular purpose among these types we have chosen convolutional neural networks (CNN) and we have used deep learning for the recognition and classification of handwritten words in Arabic or early we used MATLAB to understand CNN with the database of handwritten digits and after we used the Python language for the database of handwritten words in Arabic, so we proposed different architectures. The results obtained showed the choice of architecture, the size of filters and the number of epochs have a great influence for better results. In order to verify the results we did a classification test.

Key words :

Neural networks, CNN, deep learning, handwritten words in Arabic language.

Liste des figures

Chapitre I

Figure I.1 Représentation d'un neurone réel 4
Figure I.2 Modèle de base d'un neurone formel 6
Figure I.3 Le Perceptron : structure et comportement 10
Figure I.4 Modèle du Perceptron Multi–Couches 11

Chapitre II

Figure II.1 Processus de convolution et de sous-échantillonnage 15
Figure II.2 Exemple de cas justifiant le Padding 17
Figure II.3 Exemple d'architecture CNN 18
Figure II.4 Une couche du CNN en 3 dimensions. (Vert : volume d'entrée, bleu : volume du champ récepteur, gris : couche de CNN, cercles : neurones artificiels indépendants) 19
Figure II.5 Ensemble de neurones (cercles) créant la profondeur d'une couche de convolution (bleu). Ils sont liés à un même champ récepteur (rouge). 21
Figure II.6 convolution avec un filtre 3*3 et un pas de 1 22
Figure II.7 Pooling avec un filtre 2*2 et un pas de 2. 23
Figure II.8 Description du flux de données à travers un réseau .....
Neuronal profond et met en évidence le flux de données à travers la couche 25
Figure II.9 Exemples de modèles de CNN 27
Figure II.10 Exemple de max pooling (2*2). 28
Figure II.11 Le Perceptron : structure et comportement 32

Chapitre III

Figure III.1 L'environnement Spyder 36
Figure III.2 L'organigramme des étapes de notre travaille 38
Figure III.3 La Base de données de mots manuscrits arabe 39
Figure III.4 La précision de validation d'une seule couche de convolution 41
Figure III.5 La précision de validation de deux couches de Convolution 41
Figure III.6 La précision de validation de trois couches de convolution 42
Figure III.7 La précision avec une seule couche de maxpooling 44
Figure III.8 La précision avec deux couches de maxpooling 45
Figure III.9 La précision avec un filtre de taille 3*3 47
Figure III.10 La précision avec un filtre de taille 5*5 47

Figure III.11 La précision avec un filtre de taille 7*7 48
Figure III.12 La précision avec un filtre de taille 9*9 48
Figure III.13 La précision du système intelligent 50
Figure III.14 La première image de test 51
Figure III.15 La deuxième image de test 52
Figure III.16 résultat obtenus 52
Figure III.17 résultat obtenus 53
Figure III.18 résultat de classification de la première image 53
Figure III.19 résultat de classification de la deuxième image 54

Liste des tableaux

Chapitre II

Tableau II.1 Les performances de la taille de filtre et le nombre d'epoch sur un réseau cnn . 33

Tableau II.2 Les performances des coefficients d'apprentissage sur un réseau cnn 34

Chapitre III

Tableau III.1 L'influence de la couche de convolution 42

Tableau III.2 L'influence de la couche de maxpooling 44

Tableau III.3 L'influence de la taille de filtre. 46

Tableau III.4 L'influence des architectures par bloque 50

Liste des Abréviations

API : Application Programing Interface.

CNN : Convolutionnel Neural Network.

Conv : Convolution.

FC : Fully Connected.

GPU : Graphics Processing Unit.

MATLAB : Matrix Laboratory.

MLP : Multi Couche Perception.

MNIST : Mixed National Institu of Standard ans Technology.

NN : Neural Network.

ONEIROS : Open_ended Neuro_Electronic Intelligent Robot Operating System.

PMC : Perception Multi Couche.

Pool : Pooling.

ReLU : Unités Rectifié Linéaire.

RGB (RVB) : Rouge Vert Bleu.

RNA : Réseaux Neurones Artificiels.

Table des matières

Liste des figures **I**

Liste des tableaux **III**

Abréviations **IV**

Introduction générale : 1

Chapitre I

Les reseaux de neurones

I.1-Introduction:.....3

I.2-Le neurone : 4

I.3-Bases Biologiques :.....4

I.4-Neurone formel (Artificiel) :.....5

I.4-1-Définition :..... 5

I.4-2-Les paramètres :..... 5

I.5-Principe de fonctionnement : 7

I.6-Architecture des réseaux de neurones :.....7

I.6-1- Réseaux statiques :..... 7

I.6-2- Réseaux dynamiques : 8

I.6-3- Réseaux auto-organisés : 8

I.7-Apprentissage des neurones artificiels :.....8

I.7-1-Définition :..... 9

I.7-2-Apprentissage supervisé : 9

I.7-3-Apprentissage non supervisé : 9

I.8-Exemples de deux réseaux de neurones les plus utilisés: 9

I.8-1-Perceptron mono couche : 9

I.8-2-Perceptron multicouche :..... 10

I.8-2-1-Rétro propagation du gradient pour un Perceptron Multi–Couches :..... 11

I.8-2-2-Mise à jour des poids :..... 12

I.10-conclusion :..... 12

Chapitre2

Les réseaux de neurones convolutionnels :

Application sur la classification des chiffres manuscrits

II.1 Introduction : 14
II.2 Les réseaux de neurones convolutionnels: 14
II.3 Architecture de réseaux de neurones convolutionnels : 18
II.4 L'entrainement d'un nouveau CNN: 20
II.4.1 Définition des couches d'apprentissage en profondeur : 20
II.4.2 Couche d'image d'entrée (Image Input Layer) : 20
II.4.3 La couche de convolution (CONV) : 21
II.4.4 La couche de pooling (POOL) : 22
II.4.5 La couche de correction (Relu) : 24
II.4.6 La couche entièrement connectée (FC) : 24
II.4.7 La couche de perte (LOSS) : 25
II.5 Architecture de couche (layer): 25
II.6 Exemples de modèles de CNN : 26
II.7 Choix des hyper paramètres : 27
II.7.1 Nombre de filtres : 27
II.7.2 Forme de filtres : 27
II.7.3 Forme de Max pooling : 28
II.8 Avantages de CNNs : 28
II.9 Méthode de régulation : 29
II.9.1 Empirique : 29
II.9.1.1 Dropout : 29
II.9.1.2 DropConnect : 29
II.9.1.3 Pooling stochastique : 29
II.9.2 Explicite : 30
II.9.2.1 Taille du réseau : 30
II.9.2.2 Dégradation du poids : 30
II.10 Exemple d'application avec un réseau de neurones CNN: 30
II.10.1 Présentation de l'application : 31
II.10.2 Résultats de la simulation de la reconnaissance de nombres manuscrits : 32
II.11 Conclusion : 34

Chapitre3

L'application du CNN pour la classification et la reconnaissance automatique des mots manuscrites en arabe

III.1 Introduction : 35
III.2 Présentation des outils de développement : 35
III.2.1 Logiciels et librairies utilisés dans la simulation : 35
III.2.1.1 Python : 35
III.2.1.2 Spyder : 35
III.2.1.3 TensorFlow : 36
III.2.1.4 Keras : 37
III.2.1.5 Scikit-learn : 37
III.2.1.6 Pandas : 37
III.2.2 Configuration utilisé dans la simulation : 38
III.3 Les différentes étapes de notre expérimentation : 38
III.4 Notre système de reconnaissance de mots arabe avec un CNN: 40
III.4.1 L'influence du nombre de couche de convolution : 41
III.4.2 L'influence du nombre de la couche de maxpooling: 43
III.4.3 L'influence de la taille du filtre : 46
III.4.4 L'architecture du CNN la plus performante : 48
III.5 Exemples de tests : 51
III.6 Conclusion : 54
Conclusion générale : 55
Bibliographie: 56

Introduction générale :

La reconnaissance est un domaine qui couvre divers aspects tels que la reconnaissance de visage, la reconnaissance de chiffres, la reconnaissance d'empreinte digitale, la reconnaissance des images et la reconnaissance de mots manuscrits en arabe etc.…

Le système de reconnaissance de mots manuscrits est un système intelligent capable de classifier des mots manuscrits à la manière de l'homme. Cette classification joue un rôle important dans l'automatisation de processus tels le traitement automatique de chèques bancaires.

La reconnaissance de mots manuscrits en arabe est très importante pour les pays dont la langue écrite est l'arabe. Tous les systèmes existants concernent essentiellement les textes écrits en caractères latins. Il est donc de notre devoir de concevoir nos propres systèmes automatiques et intelligents. Ceci étant, cette tâche n'est pas aisée du fait des variations illimitées de l'écriture humaine arabe et de l'indisponibilité de bases de données exhaustives.

Les réseaux neurones convolutionnels (CNN) sont un type de réseaux neurones artificiels dont le fonctionnement est inspiré par les processus biologique. Le CNN est l'une des structures de réseaux les plus représentatives de la technologie d'apprentissage profond (deep Learning). Ce concept est de nos jours utilisé avec beaucoup de succès dans le domaine de la reconnaissance.

Notre travail s'inscrit dans cette thématique. Notre objectif est de concevoir un système automatique de reconnaissance de mots manuscrits en arabe utilisant les réseaux neurones convolutionnels.

Pour ce faire, nous avons structuré notre mémoire en trois chapitres :

- dans le premier nous présenterons une vision globale sur les réseaux neurones artificiels, les bases biologiques et leurs différentes architectures. Nous avons également exposé les différentes méthodes d'apprentissage des réseaux de neurones.
- Dans le deuxième chapitre, nous décrirons en détail les réseaux neurones convolutionnels objet de notre travail. Nous illustrons à travers une application de reconnaissance automatique de nombre manuscrits l'influence de sa structure et des divers paramètres sur sa précision et sa rapidité d'exécution.

- Le troisième chapitre concernera le détail de notre réalisation. nous présentons notre la conception du système de reconnaissance automatique des mots manuscrits en arabe. Nous détaillerons également les différentes étapes nécessaires à son implémentation. De plus, plusieurs expérimentations seront exposées pour obtenir un système intelligent qui reconnaisse un maximum de mots manuscrits en arabes.

Enfin, une conclusion nous permettra de résumer l'ensemble des travaux réalisés. Nous terminerons ce mémoire par quelques perspectives envisageables dans un futur proche ou plus lointain.

I.1-Introduction :

Dans le monde qui nous entoure, il existe plusieurs technologies pour développer et perfectionner des méthodes de traitements et d'analyse telles que prédictions météorologiques, prévision de la consommation d'eau, reconnaissance du code postal, compression de données, filtrage, classification,… Parmi ces techniques il existe celles qui utilisent les réseaux de neurones artificiels (RNA).

Les réseaux de neurones artificiels sont présentés comme la vague du futur en informatique. Ce sont en effet des mécanismes d'autoapprentissage qui ne nécessitent pas les compétences traditionnelles d'un programmeur.

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau. [1]

Dans ce chapitre nous débuterons par une brève introduction, suivie du détail du modèle du neurone biologique et du neurone formel. Cette partie sera suivie de quelques définitions de réseaux de neurones artificiels. Ensuite, nous étudierons les différentes architectures qui différencient ces réseaux, nous terminerons par les types d'apprentissage les plus utilisés (perceptron multicouche (PMC)).

I.2-Le neurone :

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. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone (unique) pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de quelques dizaines d'Angströms (10-9 m) entre l'axone du neurone afférent et les dendrites (on dit une dendrite) du neurone efférent. La jonction entre deux neurones est appelée la synapse. [1]

I.3-Bases Biologiques :

Le neurone est le processeur élémentaire du cerveau. Chaque neurone traite l'information qui lui parvient localement, puis transmet aux autres neurones qui lui sont connectés l'information qu'il a traitée (voir la figure I.1). Ces cellules peuvent apprendre en changeant l'intensité de leurs connexions avec d'autres cellules ou détruire ou même créer de nouvelles connexions. Elles peuvent aussi changer leur(s) règle(s) de traitement de l'information. Ce processus de changement est appelé apprentissage et joue un rôle fondamental dans le comportement du neurone. [2]

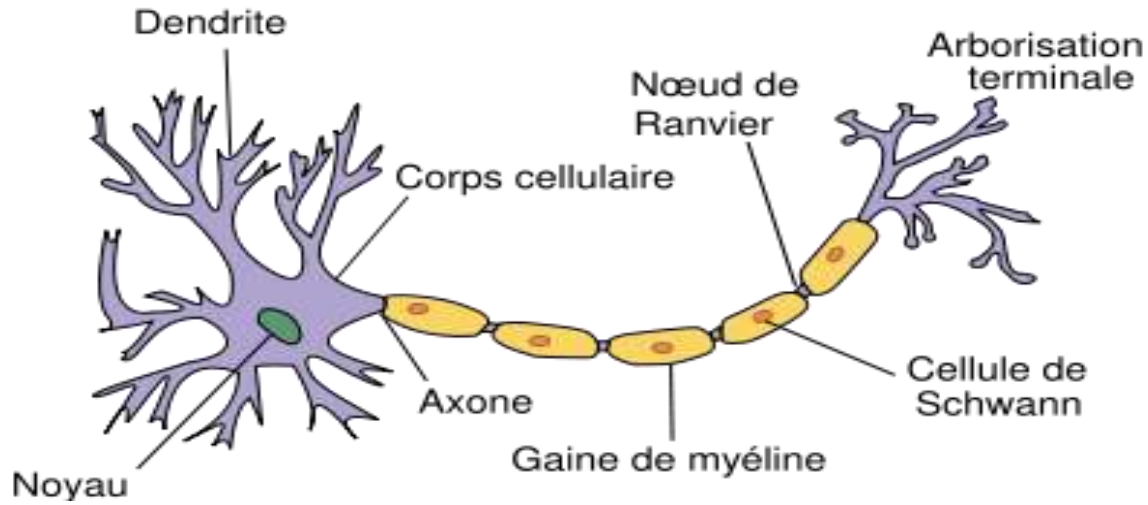


Figure I.1 Représentation d'un neurone réel

La figure 1 montre la représentation graphique d'un neurone. D'après la figure 1, vous pouvez constater que le neurone est composé des parties suivantes :

- **Corps cellulaire :** Noyau du neurone.
- **Dendrites :** Récepteurs principaux du neurone.
- **Axone :** Fibre nerveuse de transport pour les signaux émis par le neurone.
- **Synapse :** Connexion entre la dendrite et l'axone de deux neurones [2]

I.4-Neurone formel (Artificiel) :

I.4-1-Définition :

Neurone formel est un processeur élémentaire qui réalise une somme pondérée des signaux qui lui parviennent. La valeur de cette sommation est comparée à un seuil et la sortie du neurone est une fonction non linéaire. [3]

Un neurone est une fonction paramétrée non linéaire de ses variables d'entrée. Naturellement, un réseau de neurones est la composition des fonctions non linéaires de deux neurones ou plus. », [4]

Les variables du neurone sont souvent appelées entrées du neurone et sa valeur est sa sortie.

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit [1]

I.4-2-Les paramètres :

Les paramètres sont affectés aux entrées des neurones ; la sortie du neurone est une combinaison non linéaire des entrées {xi}, pondérées par les paramètres {Wi}, souvent appelées poids.

Le potentiel v le plus fréquemment utilisé est la somme pondérée des entrées, avec un terme constant supplémentaire appelé «biais» (voir la figure I.2), [4]

Soient :

- d : la dimension de l'espace des entrées.

- Ei : un vecteur d'entrée.

- w : un vecteur de poids.

- b : le biais.

- v : un coefficient (< 0.05).

- e : l'erreur.

On peut définir la sortie du réseau ainsi :

$$\mathbf{Y\ (pi)} = \sum_{j=1}^{d}(\boldsymbol{Wj} \cdot \boldsymbol{Ei,j}) + \boldsymbol{b} = \mathbf{W} \cdot \mathbf{Ei} + \mathbf{b} \qquad \textbf{Equ I.1}$$

$$\mathbf{L'erreur} = \mathbf{la\ cible(n)} - \mathbf{la\ sortie(n)} \qquad \textbf{Equ I.2}$$

Les formules à appliquer pour mettre à jour les poids et biais lorsque l'on entre un individu dans le réseau sont donc :

$$\mathbf{W\ new} = \mathbf{W\ old} + 2 \cdot \boldsymbol{\nu} \cdot \mathbf{e}\ (\mathbf{Ei}) \cdot \mathbf{Ei} \qquad \textbf{Equ I.3}$$

$$\mathbf{P\ new} = \mathbf{P\ old} + 2 \cdot \boldsymbol{\nu} \cdot \mathbf{e}\ (\mathbf{Ei}) \qquad \textbf{Equ I.4}$$

Les fonctions d'activation :

- Fonction logistique
- Fonction hypebolique
- Fonction a seuil (sigmoid)
- Fonction Gaussienne
- Softmax

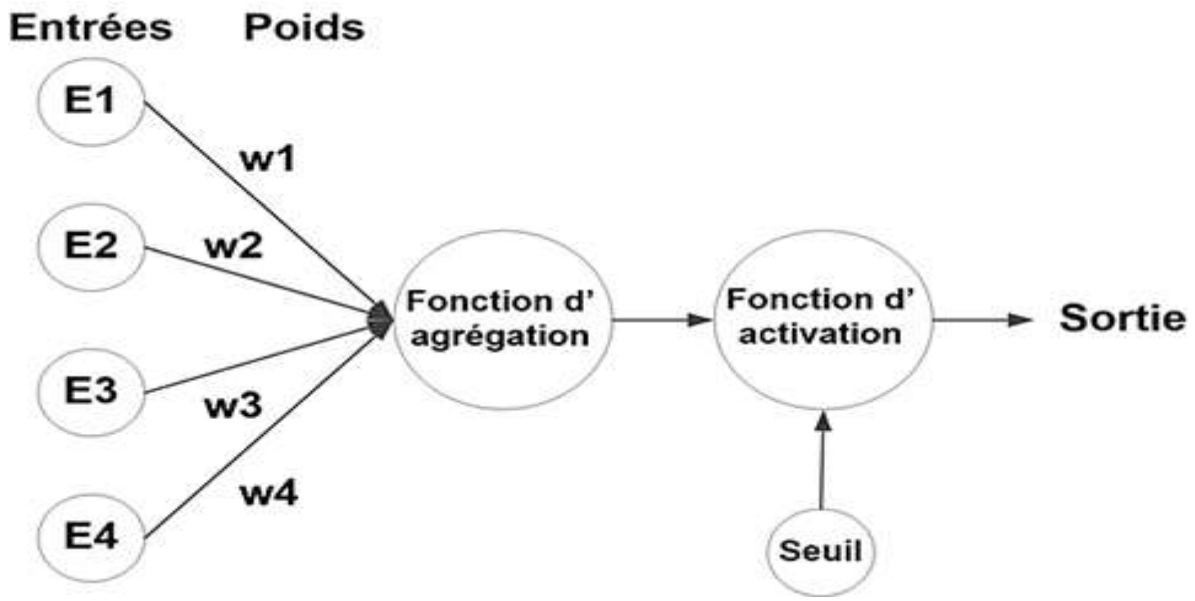


Figure I.2 Modèle de base d'un neurone formel.

Nous pouvons définir un neurone par les trois éléments suivants [2]:

- La fonction d'entrée totale qui définit le prétraitement effectué sur les entrées.
- La fonction d'activation (ou d'état) du neurone qui définit son état interne en fonction de son entrée totale.
- La fonction de sortie qui calcule la sortie du neurone en fonction de son état d'activation.

I.5-Principe de fonctionnement :

Un neurone reçoit des entrées et fournit une sortie, grâce à différentes caractéristiques :

- Des **poids** accordés à chacune des entrées, permettant de modifier l'importance de certaines par rapport aux autres.
- Une **fonction d'agrégation**, qui permet de calculer une unique valeur à partir des entrées et des poids correspondants.
- Un **seuil** (ou biais), permettant d'indiquer quand le neurone doit agir.
- Une **fonction d'activation**, qui associe à chaque valeur agrégée une unique valeur de sortie dépendant du seuil. [5]

Principes :

- pas de notion temporelle
- coefficient synaptique : coefficient réel
- sommation des signaux arrivant au neurone
- sortie obtenue après application d'une fonction de transfert [6].

I.6-Architecture des réseaux de neurones :

L'architecture d'un réseau de neurones définit son fonctionnement et joue un rôle important dans son comportement. Elle est en fonction de ses couches et de la structure des connexions de ses neurones ; ces paramètres permettent de distinguer les différentes classes et/ou types d'architecture neuronales. [7]

Donc Il existe trois grands types d'architectures de réseaux de neurones :

I.6-1- Réseaux statiques :

Ce type de réseaux est organisé généralement en couches de neurones. Chaque neurone d'une couche reçoit ses entrées à partir des neurones de la couche précédente ou tout simplement de l'entrée du réseau. Dans tels réseaux il n'existe pas de feed-back (boucles de retour d'informations). Le traitement des données se fait en sens unique et le flux d'informations circule directement de la couche d'entrée à la couche de sortie ; le traitement est donc réalisé en boucle ouverte. Ces réseaux peuvent être utilisés pour les problèmes de classification ou d'approximation des fonctions.

I.6-2- Réseaux dynamiques :

Les réseaux dynamiques ou bien récurrents, sont les réseaux pouvant comporter des boucles (feed-back) entre les neurones. En général, la sortie de chaque neurone peut être envoyée vers l'entrée de tous les autres neurones du réseau. Ainsi, ces boucles ramènent l'information en provenance de la couche de sortie sur la couche d'entrée simultanément avec le signal d'entrée présent au même instant.

Un réseau dynamique peut donner une sortie différente en lui présentant la même entrée à des instants différents, contrairement aux réseaux statiques qui donnent la même sortie pour une même entrée. Au début, ces réseaux étaient souvent utilisés pour les problèmes de classification et de mémorisation. Actuellement, dans plusieurs travaux la structure interne dynamique de ces réseaux est exploitée pour l'identification ou la commande des systèmes dynamiques.

I.6-3- Réseaux auto-organisés :

Les réseaux de neurones auto-organisés sont des réseaux qui changent leurs structures internes pendant l'apprentissage. Ainsi les neurones se regroupent typologiquement suivant la représentation des exemples. Ces réseaux sont des dérivées des modèles de Kohonen. [7]

I.7-Apprentissage des neurones artificiels :

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. Elle ne concerne cependant pas tous les modèles, mais les plus utilisés.

Une fois qu'un réseau a été structuré pour une application particulière, ce réseau est prêt à être formé. Pour démarrer ce processus, les poids initiaux sont choisis de manière aléatoire. Ensuite, l'apprentissage commence.

I.7-1-Définition :

L'apprentissage est la modification des poids du réseau dans l'optique d'accorder la réponse du réseau aux exemples et à l'expérience. Il est souvent impossible de décider à priori des valeurs des poids des connexions d'un réseau pour une application donnée. A l'issu de l'apprentissage, les poids sont fixés : c'est alors la phase d'utilisation.

Au niveau des algorithmes d'apprentissage, Il existe deux approches d'apprentissage :

I.7-2-Apprentissage supervisé :

Apprentissage supervisé implique un mécanisme fournissant au réseau le résultat souhaité, soit en "évaluant" manuellement les performances du réseau, soit en fournissant les résultats souhaités avec les entrées. La grande majorité des réseaux utilisent l'apprentissage supervisée.

Les coefficients synaptiques sont évalués en minimisant l'erreur (entre sortie souhaitée et sortie obtenue) sur une base d'apprentissage. [6]

I.7-3-Apprentissage non supervisé :

Apprentissage non supervisé est l'endroit où le réseau doit donner un sens aux entrées sans aide extérieure, il est utilisé pour effectuer une caractérisation initiale des entrées.

On ne dispose pas de base d'apprentissage. Les coefficients synaptiques sont déterminés par rapport à des critères de conformité : spécifications générales. [6]

I.8-Exemples de deux réseaux de neurones les plus utilisés :

I.8-1-Perceptron mono couche :

Avant d'aborder le comportement collectif d'un ensemble de neurones, nous allons présenter le Perceptron (un seul neurone) en phase d'utilisation. L'apprentissage ayant été réalisé, les poids sont fixes. Le neurone de la figure 3 réalise une simple somme pondérée de ses entrées, compare une valeur de seuil, et fourni une réponse binaire en sortie (voir la figure I.3).

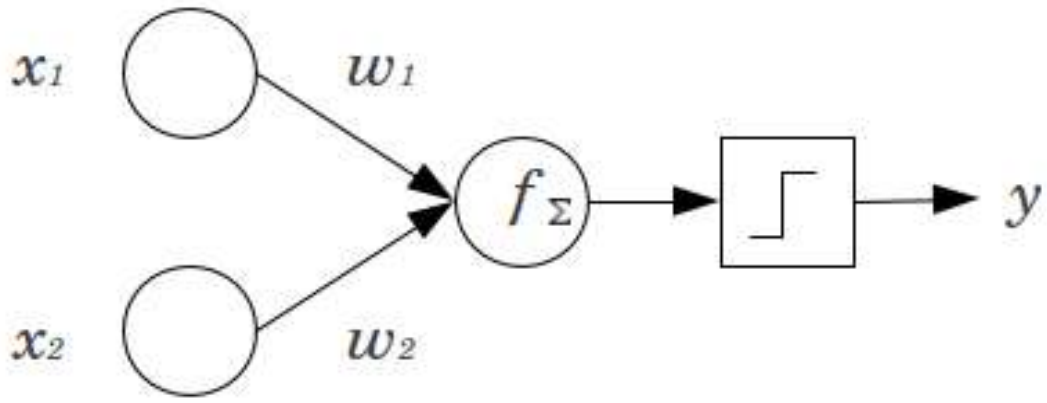


Figure I.3 Le Perceptron : structure et comportement.

(Les connexions des deux entrées x1 et x2 au neurone sont pondérées par les poids w1 et w2).

I.8-2-Perceptron multicouche :

D'une façon générale, un perceptron multicouche (Multi Layer Perceptron) contient une couche d'entrée, une ou plusieurs couches cachées, chargées des calculs possédant une fonction d'activation et une couche de sortie. La couche d'entrée représente toujours une couche virtuelle associée aux entrées du système, elle ne contient aucun neurone tandis que les couches suivantes sont des couches de neurones. La couche de sortie correspond aux sorties du système. [8]

Dans le modèle du Perceptron Multi–Couches, les perceptrons sont organisés en couches. Les perceptrons multi–couches sont capables de traiter des données qui ne sont pas linéairement séparables Avec l'arrivée des algorithmes de rétro propagation, ils deviennent le type de réseaux de neurones le plus utilisé.

L'utilité de plusieurs couches cachées n'a pas été démontrée. La Figure I.4 illustre la structure d'un MLP présentant quatre neurones en entrée, trois neurones sur la couche cachée et deux en sortie. [9]

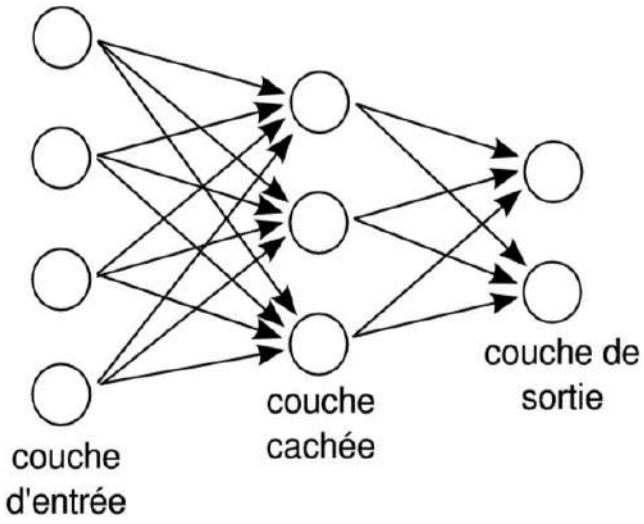


Figure I.4 Modèle du Perceptron Multi–Couches.

Lorsque tous les neurones d'une couche sont connectés aux neurones de la couche suivante, on parle alors de couches complètement connectées. Les équations de propagation décrites plus haut s'appliquent à tous les neurones.

Cependant, le passage d'une couche à l'autre peut être formalisé sous forme matricielle. Soit un MLP dont le nombre de neurones sur la couche d'entrée est n0, n1 sur la couche cachée et n2 sur la couche de sortie. Les poids de la couche cachée peuvent s'écrire sous la forme d'une matrice $W \in Rn1 \times n0$.

Pour un vecteur $X \in Rn0$ présenté en entrée, le vecteur potentiel V et le vecteur de sortie Y pour la couche cachée s'écrivent donc :

$$V = WX$$

$$Y = \Phi(V)$$

Avec Φ une fonction d'activation. La sortie de la couche cachée devient ensuite l'entrée de la dernière couche [9]

I.8-2-1-Rétro propagation du gradient pour un Perceptron Multi–Couches :

La rétro propagation du gradient s'applique de manière récursive de la couche de sortie à la couche d'entrée. L'initialisation se fait donc par le calcul de l'erreur à la sortie de la dernière couche. Soit l'échantillon d'apprentissage $X \in Rn0$, et le vecteur de sortie désiré $D \in Rn2$. L'erreur sur la couche de sortie peut être définie comme précédemment :

$$L = 1/2 \| Y2 - D \|^2$$

Où le vecteur Y2 représente la sortie du MLP (i.e. la sortie de la couche 2). De manière immédiate, le gradient de l'erreur en sortie peut être calculé :

$$\partial L/\partial Y2 = Y2 - D$$

La rétro propagation du gradient s'applique alors récursivement de la couche de sortie à la couche d'entrée. Pour une couche dont X est le vecteur d'entrée, V le potentiel, W la matrice de poids et Y la sortie, la rétro propagation s'écrit donc :

$$\begin{aligned} \frac{\partial L}{\partial V} &= \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial V} \\ &= \Phi'(V)\frac{\partial L}{\partial Y} \\ \frac{\partial L}{\partial X} &= \frac{\partial L}{\partial V}\frac{\partial V}{\partial X} \\ &= W^T\frac{\partial L}{\partial V} \end{aligned}$$

I.8-2-2-Mise à jour des poids :

Une fois la rétro propagation du gradient effectuée pour toutes les couches, les matrices de poids sont mises à jour :

$$W \leftarrow W - \lambda dW$$

Où λ est le taux d'apprentissage. [9]

I.10-conclusion :

A partir du fonctionnement du cerveau humain et du modèle neuronal biologique simple, les chercheurs ont pu développer et mettre aux points des modèles neuronaux artificiels complexes.

Les réseaux de neurones artificiels, sont le cœur de plusieurs applications réelles. C'est un outil indispensable qui a fait ses preuves dans des applications pratiques du fait de leurs caractéristiques de généralisation et de robustesses face au bruit.

Dans ce chapitre nous avons décrit les réseaux de neurones artificiels, les bases biologiques et leurs différentes architectures. Nous avons également exposé les différentes méthodes d'apprentissage des réseaux de neurones.

Dans le prochain chapitre, nous allons présenter l'architecture et le fonctionnement du réseau de neurones convolutionnel que nous avons choisi pour réaliser la reconnaissance automatique de caractères manuscrits.

Chapitre 2

Les réseaux de neurones convolutionnels

II.1 Introduction :

Après avoir présenté dans le chapitre précédent les réseaux neurones artificiels et leurs types, nous présentons dans cette section les réseaux de neurones convolutionnels CNN (Convolutionel Neural Network) objet de notre projet. En effet, ces réseaux ont démontré leurs performances dans des applications concrètes de classifications d'objets. Cela rejoint notre travail de recherche qui consiste à concevoir un système automatique de reconnaissance de mots manuscrits arabes.

Par conséquent, nous présentons dans ce chapitre une étude détaillée sur l'architecture et l'apprentissage des réseaux neurones convolutionnels. Nous illustrons à travers une application de reconnaissance automatique de nombre manuscrits l'influence des dimensions des filtres et des divers paramètres sur la précision et la rapidité d'exécution du réseau de neurone convolutionnel.

II.2 Les réseaux de neurones convolutionnels:

Parmi les réseaux de neurones à apprentissage profond, ceux qui sont le plus largement utilisés dans les problèmes de vision par ordinateur sont les réseaux de neurones convolutionnels. Ceux-ci sont basés sur l'architecture d'un perceptron multicouche MLP (Multi Layer Perceptron). Le fonctionnement des réseaux CNN repose sur les mêmes principes que le fonctionnement du cortex visuel. Cela consiste à assigner des neurones spécifiques qui ne traitent que de petites parties de l'image d'entrée (aussi appelée champ visuel). Ces neurones sont chargés de reconnaître les motifs pertinents dans l'image et ils sont empilés dans des structures similaires à des couches. Cette architecture permet d'élaborer des modèles de plus en plus complexes. Le nombre de poids à apprendre pour une image est exceptionnellement élevé. Ceci est généralement impossible à traiter si on est confronté à une MLP (multi layer perceptron : perceptron multicouche) normale. Pire encore, cette connectivité complète entre les couches successives est, dans certains cas, inutile, car la position spatiale n'est pas prise en compte. Les CNN, quant à eux, se basent uniquement sur la corrélation locale, chaque neurone ne considère qu'une petite partie de son entrée dans son ensemble, et néglige le reste économisant ainsi beaucoup d'arêtes et prenant en compte la relation entre pixels proches. De plus, les éléments reconnaissables sont les mêmes indépendamment de leur position dans l'image, il est donc logique de chercher des bords et des coins autour de toute l'image. Par conséquent, chaque neurone recherchera les mêmes caractéristiques que le reste des neurones dans sa couche, mais

dans des endroits différents. En tant que tel, le concept de poids partagés qu'ils utilisent consiste en ce qu'il n'y a qu'un seul ensemble de poids pour tous les neurones dans une couche.

Dans les réseaux CNN, de petites parties de l'image (appelées un champ réceptif local) sont traitées comme des entrées de la couche la plus basse de la structure hiérarchique. L'information se propage généralement à travers les différentes couches du réseau où, à chaque couche, un filtrage numérique est appliqué afin d'obtenir des caractéristiques saillantes des données observées. Le procédé fournit un niveau d'invariance au déplacement, à l'échelle et à la rotation lorsque le champ réceptif local permet aux neurones ou à l'unité de traitement d'accéder à des éléments élémentaires tels que des arêtes ou des coins orientés. Essentiellement, l'image d'entrée est convolutionnée avec un ensemble de N petits filtres dont les coefficients sont soit formés, soit prédéterminés en utilisant certains critères. Ainsi, la première couche (ou la couche la plus basse) du réseau est constituée de «cartes de caractéristiques» qui sont le résultat des processus de convolution, avec un biais additif et éventuellement une compression ou une normalisation des caractéristiques. Cette étape initiale est suivie d'un sous-échantillonnage (généralement une opération de moyennage de 2 3 2) qui réduit encore la dimensionnalité et offre une certaine robustesse aux déplacements spatiaux (voir la figure II 1). La carte des caractéristiques sous-échantillonnées reçoit ensuite un biais de pondération et de formation et se propage finalement à travers une fonction d'activation. Certaines variantes existent avec seulement une carte par couche ou des sommations de plusieurs cartes.

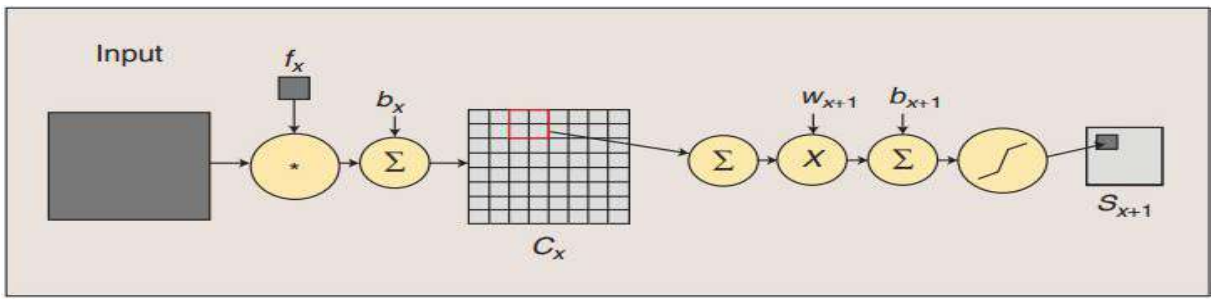


Figure II.1 Processus de convolution et de sous-échantillonnage

Le processus de convolution consiste à convoler une entrée (image pour la première étape ou carte de caractéristiques pour les étapes ultérieures) avec un filtre formable fx puis ajouter une polarisation bx pour former la couche de convolution Cx. Le sous-échantillonnage consiste à sommer un voisinage (quatre pixels), pondéré par w_{x+1}, ajoutant un biais d'apprentissage b_{x+1}, et passant par une fonction sigmoïde pour produire une carte de caractéristiques plus ou moins deux fois S_{x+1}.[12]

La partie de l'espace d'entrée à laquelle un neurone est connecté est appelée champ réceptif et peut se chevaucher avec celle d'autres neurones. Chaque champ réceptif est un espace 3D avec la largeur et la hauteur de celui-ci, et le nombre de canaux d'entrée. Par conséquent, pour chaque couche, il n'y a qu'un poids par valeur dans le champ réceptif. En tant que tel, si le champ réceptif est 5*5*3, il n'y aura que 75 poids dans cette couche, utilisés par tous ses neurones. Cela permet de former de grands réseaux, le nombre de poids restant relativement faible. Cependant, seules ce type de couches appelées couche de convolution (Convolutional Layer) répondent à ces règles. Les autres répondent à d'autres règles que nous expliquerons plus loin dans cette section. Cependant, l'utilisation des couches convolutionnelles permet de réduire considérablement le nombre de poids dans le réseau, et sont les plus emblématiques de CNN. L'ensemble des poids de chaque couche est appelé le noyau, ou filtre, de cette couche. La raison en est que, lorsqu'elles sont partagées, une étape en avant de la couche peut être interprétée comme la convolution des poids et l'entrée. La fonction d'activation la plus couramment utilisée est l'unité Linéaire Redresseur (Relu), qui applique la fonction suivante à la sortie du neurone: f (x) = max (0; x). Des variations sur cette fonction sont également utilisées, telles que Noise Relu. Les couches de neurones sont empilées, et leurs sorties forment des volumes 3D. L'entrée de la première couche, c'est-à-dire l'image elle-même, peut également être considérée comme un volume 3D : Largeur × Hauteur × canal. Chaque pile de couches utilise une configuration de noyau différente et toutes les couches qui la composent sont connectées à toutes les couches de la pile précédente. Les principaux paramètres à prendre en compte dans les couches CNN sont :

- ***Taille du noyau :*** la largeur et la hauteur du champ réceptif pour les neurones de cette pile. Les deux côtés sont généralement de taille égale.
- ***Stride :*** chaque neurone traite une région de l'espace d'entrée. Comme ces régions peuvent se chevaucher, la pooling indique la distance entre leurs centres. En tant que tel, une pooling de 1 signifie que chaque neurone traite la même région que son voisin à l'exception d'une colonne. Plus le pooling est grand, plus la largeur et la hauteur de sortie de cette couche sont petites.
- ***Padding :*** dans certains cas, les neurones situés dans les limites de la couche ne peuvent pas traiter tout un champ réceptif.

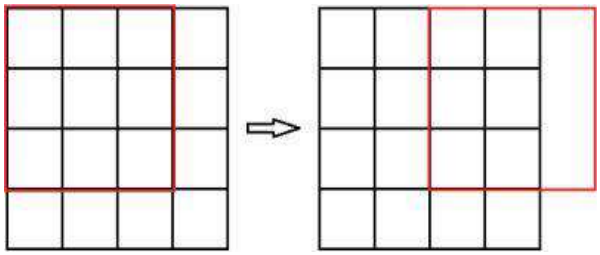


Figure II.2 Exemple de cas justifiant le Padding

La figure II.2 illustre l'exemple d'une image 4*4 traitée par un noyau 3*3 avec un pooling de 2. Comme la différence entre les champs réceptifs est de 2 pixels, la dernière colonne du champ réceptif du second neurone tombe en dehors de l'image et n'est pas traitée. Afin de résoudre ce problème, une possibilité est d'ajouter une "bordure" autour de l'image de 0. De cette façon, nous garantissons que tous les neurones traitent un champ réceptif. Il s'agit du padding ou le rembourrage, s'il est utilisé, est généralement 1 ou 2.

- ***Type de couche :*** Ceci sera expliqué plus en détail par la suite, mais ici nous disons simplement que toutes les couches d'une pile appartiennent au même type. Les plus utilisés sont les couches de regroupement, de convolution et de connexion complète. Ce sont les paramètres les plus utilisés, bien que tous les types de réseaux ne les utilisent pas tous. Ces paramètres ont une conséquence intéressante : même si dans les réseaux de neurones classiques, le nombre de neurones de chaque couche est spécifié, il ne l'est pas dans les CNN. Au lieu de cela, il est déduit des paramètres de la couche, et consiste en le nombre de neurones nécessaires pour traiter toute l'image. Par conséquent, la taille du côté de la couche est obtenue en utilisant l'équation 2.1

$$H = \left(\frac{WF+2*P}{S}\right) + 1 \qquad \textbf{Equ II.1}$$

Où **W** est le côté de l'espace d'entrée, **F** le côté du noyau utilisé, **P** correspond au remplissage et **S** à la chaîne. Par conséquent, la sortie de toutes les couches d'une pile sera la même. Ceci, avec le nombre de couches empilées, indique la taille de la sortie de cette pile. Cela étant dit, la sortie d'une couche ne sera jamais plus grande en largeur et en hauteur que son entrée. Le paramètre de pooling, en particulier, a le potentiel d'effectuer des réductions importantes, car un pooling de 2 réduira de moitié la taille du côté de l'image. Par conséquent, lorsque l'image

est propagée à travers le réseau, elle devient plus petite (figure. II.3). Cela signifie que chaque neurone des dernières couches traitera des taches plus grandes de l'image originale.

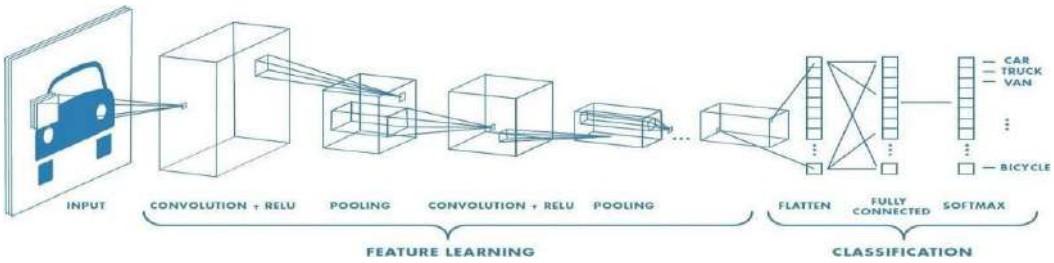


Figure II.3 Exemple d'architecture CNN

Dans la figue ci-dessus, nous pouvons voir chaque pile de couches et comment chaque neurone ne traite qu'une partie de son entrée. La taille de l'image continue à diminuer jusqu'à atteindre les couches profondes entièrement connectées. **[13]**

II.3 Architecture de réseaux de neurones convolutionnels :

Les réseaux de neurones perceptron multicouche ont beaucoup de mal à gérer des images de grande taille. Ceci est dû à la croissance exponentielle du nombre de connexions (poids) avec la taille de l'image. A contrario, la conception des réseaux de neurones convolutionnels est certes conçues autour d'un MLP mais a été modifiée pour s'inspirés du comportement du cortex visuel d'animaux. Ces modifications rendent les CNN très efficaces dans le traitement massif du nombre important de données fournies par un système de vision.

Par exemple, si on prend une image de taille 32*32*3 (32 de large, 32 de haut, 3 canaux de couleur), un seul neurone entièrement connecté dans la première couche cachée du MLP aurait 3072 entrées (32*32*3). Une image 200*200 conduirait ainsi à traiter 120 000 entrées par neurone ce qui, multiplié par le nombre de neurones, devient énorme.

Les CNN visent à limiter le nombre d'entrées tout en conservant la forte corrélation « spatialement locale » des images naturelles. Par opposition aux MLP, les CNN disposent de caractéristiques spécifiques suivantes :

1. ***Volumes 3D de neurones* :** la couche de neurones n'est plus simplement une surface (perceptron), mais devient un volume avec une profondeur. Si on considère un seul champ récepteur du CNN, les n neurones associés (sur la profondeur) forment

l'équivalent de la première couche d'un MLP.

2. ***Connectivité* locale :** grâce au champ récepteur qui limite le nombre d'entrées du neurone, tout en conservant l'architecture MLP, les CNN assurent ainsi que les «filtres» produisent la réponse la plus forte à un motif d'entrée spatialement localisé, ce qui conduit à une représentation parcimonieuse de l'entrée. Une telle représentation occupe moins d'espace en mémoire. De plus, le nombre de paramètres à estimer étant réduit, leur estimation (statistique) est plus robuste pour un volume de données fixé (comparé à un MLP).
3. ***Poids partagés* :** dans les CNNs, les paramètres de filtrage d'un neurone (pour un champ récepteur donné) sont identiques pour tous les autres neurones d'un même noyau (traitant tous les autres champs récepteurs de l'image). Ce paramétrage (vecteur de poids et biais) est défini dans une « carte de fonction ». Cela signifie que tous les neurones dans une couche de convolution donnée détectent exactement la même caractéristique. En multipliant les champs récepteurs, il devient possible de détecter des éléments indépendamment de leur position dans le champ visuel, ce qui induit une propriété d'invariance par translation. **[14]**

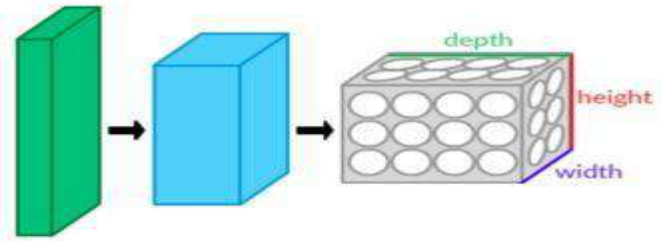


Figure II.4 Une couche du CNN en 3 dimensions. (Vert : volume d'entrée, bleu : volume du champ récepteur, gris : couche de CNN, cercles : neurones artificiels indépendants)

Une architecture CNN est formée par un empilement de couches de traitement indépendantes :

- La couche de convolution (CONV) qui traite les données d'un champ récepteur.
- La couche de pooling (POOL), qui permet de compresser l'information en réduisant la taille de l'image intermédiaire (souvent par sous-échantillonnage).
- La couche de correction (Relu), souvent appelée par abus 'Relu' en référence à la

fonction d'activation (Unité de rectification linéaire).

- La couche "entièrement connectée" (FC), qui est une couche de type perceptron.
- La couche de perte (LOSS).

II.4 L'entrainement d'un nouveau CNN:

Créer un nouveau réseau de neurones convolutif est nécessite une certaine expertise, du matériel informatique et une quantité de données annotées nécessaires.

Il s'agit d'abord de fixer l'architecture du réseau, c'est-à-dire le nombre de couches, leurs tailles et les opérations matricielles qui les connectent. L'entraînement consiste alors à optimiser les coefficients du réseau pour minimiser l'erreur de classification en sortie. Cet entraînement peut prendre plusieurs semaines pour les meilleurs CNN, avec de nombreux GPU travaillant sur des centaines de milliers d'images annotées.

Des équipes de recherche se spécialisent dans l'amélioration des CNN. Elles publient leurs innovations techniques, ainsi que le détail des réseaux entraînés sur des bases de données de références. **[15]**

II.4.1 Définition des couches d'apprentissage en profondeur :

On doit définir une couche d'apprentissage en profondeur personnalisée et spécifier des paramètres d'apprentissage facultatifs, des fonctions de transfert et une fonction de retour.

Les types et le nombre de couches incluses dépendent de l'application ou des données particulières. Par exemple, si on a des réponses catégoriques, on doit avoir une couche softmax et une couche de classification, alors que si la réponse est continue, on doit avoir une couche de régression à la fin du réseau. Un réseau plus petit avec seulement une ou deux couches de convolution peut être suffisant pour apprendre sur un petit nombre de données d'image en niveaux de gris. D'un autre côté, pour des données plus complexes avec des millions d'images colorées, vous aurez peut-être besoin d'un réseau plus complexe avec plusieurs couches convolutives et entièrement connectées. **[16]**

II.4.2 Couche d'image d'entrée (Image Input Layer) :

La couche d'entrée d'image définit la taille des images d'entrée d'un réseau de neurones convolutives et contient les valeurs de pixels brutes des images. On peut ajouter une couche

d'entrée à l'aide de la fonction ***image Input Layer*** et spécifier la taille de l'image à l'aide de l'argument ***input Size***. La taille d'une image correspond à la hauteur, la largeur et le nombre de canaux de couleur de cette image. **[16].**

Remarque : Pour une image en niveaux de gris, le nombre de canaux est 1, et pour une image couleur, il est 3.

II.4.3 La couche de convolution (CONV) :

Trois hyper paramètres permettent de dimensionner le volume de la couche de convolution (aussi appelé volume de sortie) : la ’profondeur’, le 'pas' et la ’marge’.

Profondeur' de la couche : nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur).

Le pas: contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.

La marge (à 0) ou 'zéro padding ' : parfois, il est commode de mettre des zéros à la frontière du volume d'entrée. La taille de ce 'zéro-padding' est le troisième hyper paramètre. Cette marge permet de contrôler la dimension spatiale du volume de sortie.

En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d’entrée. **[17]**

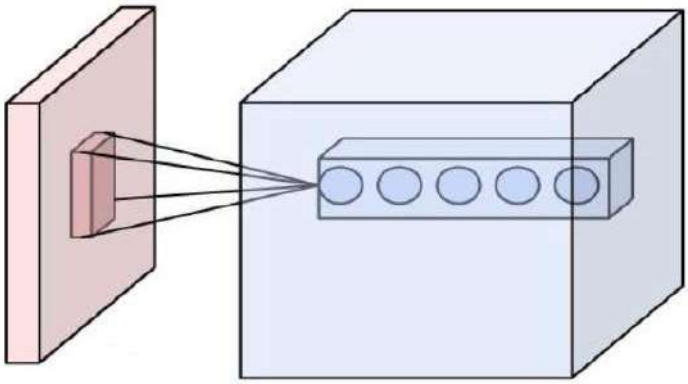


Figure II.5 Ensemble de neurones (cercles) créant la profondeur d'une couche de convolution (bleu). Ils sont liés à un même champ récepteur (rouge).

La couche de convolution est la composante clé des réseaux de neurones convolutifs, et constitue toujours au moins leur première couche.

Son but est de repérer la présence d'un ensemble de caractéristiques dites « features » dans les images reçues en entrée. Pour cela, on réalise un filtrage par convolution : le principe est de faire "glisser" une fenêtre représentant la feature sur l'image, et de calculer le produit de convolution entre la feature et chaque portion de l'image balayée. Une feature est alors vue comme un filtre : les deux termes sont équivalents dans ce contexte. **[24]**

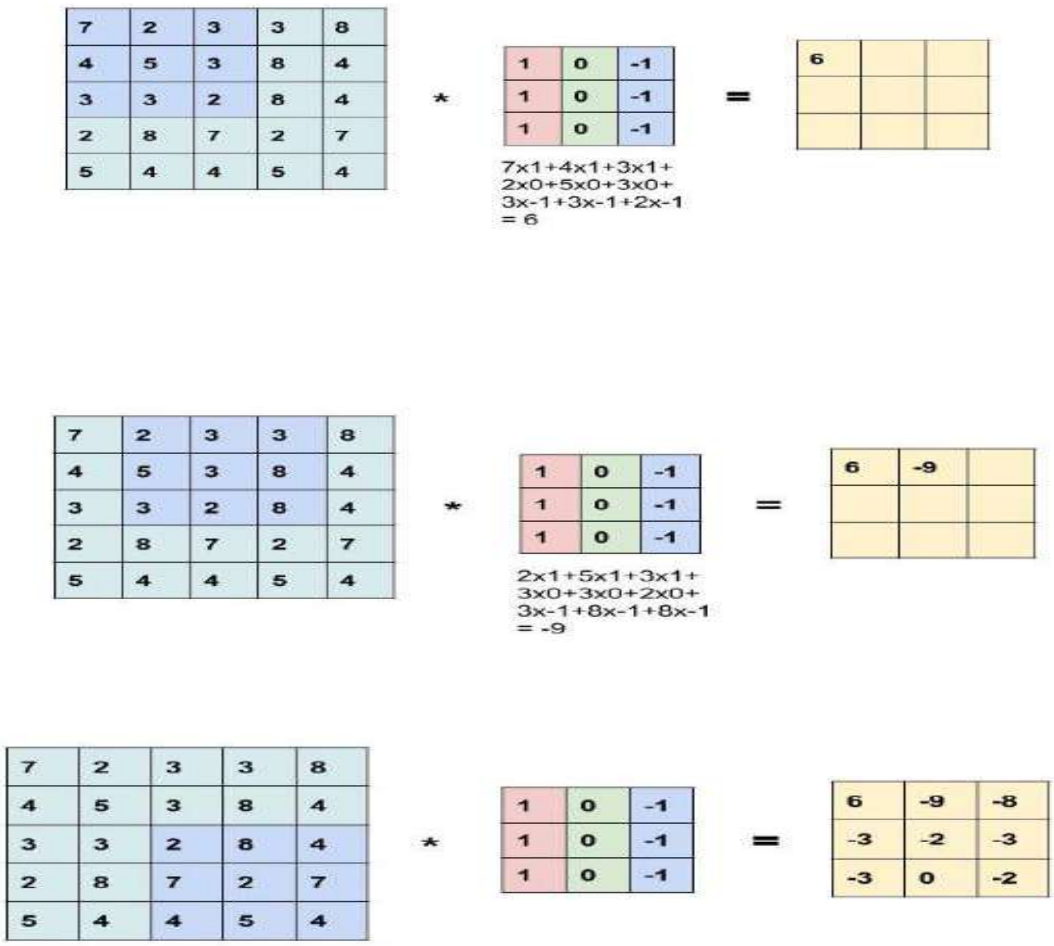


Figure II.6 convolution avec un filtre 3*3 et un pas de 1

II.4.4 La couche de pooling (POOL) :

Un autre concept important des CNNs est le pooling (« mise en commun »), ce qui est une forme de sous-échantillonnage de l'image. L'image d'entrée est découpée en une série de rectangles de n pixels de côté ne se chevauchant pas (pooling). Chaque rectangle peut être vu comme une tuile. Le signal en sortie de tuile est défini en fonction des valeurs prises par les différents pixels de la tuile.

Le pooling réduit la taille spatiale d'une image intermédiaire, réduisant ainsi la quantité de paramètres et de calcul dans le réseau. Il est donc fréquent d'insérer périodiquement une couche de pooling entre deux couches convolutives successives d'une architecture CNN pour contrôler l'overfitting (sur-apprentissage). L'opération de pooling créait aussi une forme d'invariance par translation.

La couche de pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée et la redimensionne uniquement au niveau de la surface. La forme la plus courante est une couche de mise en commun avec des tuiles de taille 2*2 (largeur/hauteur) et comme valeur de sortie la valeur maximale en entrée (cf. schéma). On parle dans ce cas de « Max- Pool 2*2 » (compression d'un facteur 4).

Il est possible d'utiliser d'autres fonctions de pooling que le maximum. On peut utiliser un « average pooling » (la sortie est la moyenne des valeurs du patch d'entrée), du « L2-norm pooling ». Dans les faits, même si initialement l'average pooling était souvent utilisé il s'est avéré que le max-pooling était plus efficace car celui-ci augmente plus significativement

L'importance des activations fortes. En d'autres circonstances, on pourra utiliser un pooling stochastique (voir « Méthodes de régularisation » plus bas dans ce document).

Le pooling permet de gros gains en puissance de calcul. Cependant, en raison de la réduction agressive de la taille de la représentation (et donc de la perte d'information associée), la tendance actuelle est d'utiliser de petits filtres [en] (type 2*2) (figure (II.7)). Il est aussi possible d'éviter la couche de pooling [en] mais cela implique un risque sur-apprentissage plus important. **[18]**

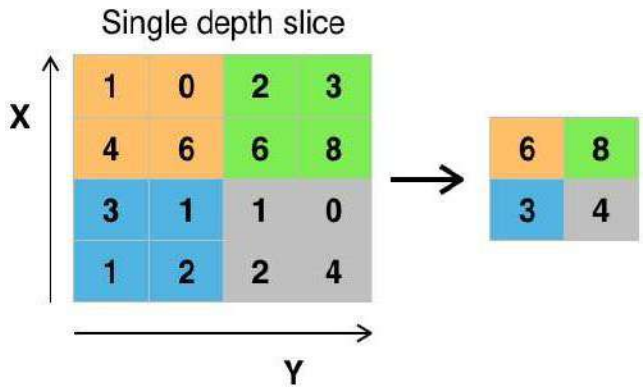


Figure II.7 Pooling avec un filtre 2*2 et un pas de 2.

II.4.5 La couche de correction (Relu) :

Souvent, il est possible d'améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie. On a notamment :

La correction Relu (abréviation d'Unités Rectifié linéaires) :

$f(x) = \max(0x)$ **Equ II.2**

Cette fonction, appelée aussi « fonction d'activation non saturante », augmente les propriétés non linéaires de la fonction de décision et de l'ensemble du réseau sans affecter les champs récepteurs de la couche de convolution.

La correction par tangente hyperbolique $f(x) = \tanh(x)$ **Equ II.3**

La correction par la tangente hyperbolique saturante :

$f(x) = |\tanh(x)|$ **Equ II.4**

La correction par la fonction sigmoïde. $f(x) = (1 + e^{-x})^{-1}$ **Equ II.5**

Souvent, la correction Relu est préférable, car il en résulte la formation de réseau neuronal plusieurs fois plus rapide, sans faire une différence significative à la généralisation de précision. **[14]**

II.4.6 La couche entièrement connectée (FC) :

Après plusieurs couches de convolution et de max-pooling, le raisonnement de haut niveau dans le réseau neuronal se fait via des couches entièrement connectées. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente (comme on le voit régulièrement dans les réseaux réguliers de neurones). Leurs fonctions d'activations peuvent donc être calculées avec une multiplication matricielle suivie d'un décalage de polarisation. Ces couches sont essentiellement des couches neurales reliées à tous les neurones de la couche précédente. Dans ce cas, elles n'utilisent aucun des paramètres introduits, en utilisant à la place le nombre de neurones. La sortie qu'ils produisent peut être comprise comme un vecteur de caractéristiques compact représentant l'image d'entrée. Elles sont également utilisées comme couches de sortie, avec un neurone par sortie, comme d'habitude. [**20]**

II.4.7 La couche de perte (LOSS) :

La couche de perte spécifie comment l'entrainement du réseau pénalise l'écart entre le signal prévu et réel. Elle est normalement la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La perte « **Softmax** » est utilisée pour prédire une seule classe parmi K classes mutuellement exclusives. La perte par entropie croisée sigmoïde est utilisée pour prédire K valeurs de probabilité indépendante dans [0,1]. La perte euclidienne est utilisée pour régresser vers des valeurs réelles. [13]

II.5 Architecture de couche (layer):

Une couche a deux composants principaux : le passage avant et le passage arrière. Pendant le passage avant d'un réseau, la couche prend la sortie x de la couche précédente, applique une fonction, puis émet (propage en avant) le résultat z vers la couche suivante. A la fin du passage avant, le réseau calcule la perte L entre les prédictions Y et la vraie cible T.

Lors du passage en arrière d'un réseau, chaque couche prend les dérivées de la perte par rapport à z, calcule les dérivées de la perte L par rapport à x, puis émet des résultats (se propage vers l'arrière) à la couche précédente. Si la couche a des paramètres apprenants, alors la couche calcule également les dérivées des pondérations de couche (paramètres apprenables) W.

La couche utilise les dérivées des poids pour mettre à jour les paramètres apprenables. (La figure 2.7) suivante décrit le flux de données à travers un réseau neuronal profond et met en évidence le flux de données à travers la couche. **[19]**

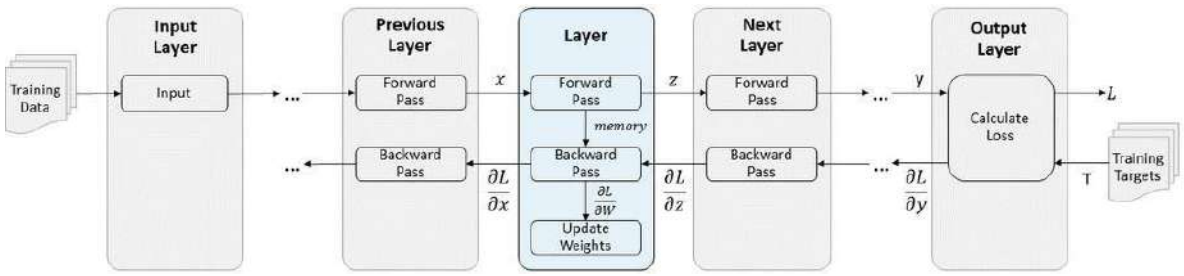


Figure II.8 Description du flux de données à travers un réseau

Neuronal profond et met en évidence le flux de données à travers la couche

Propriétés de couches:

Les propriétés de la classe sont définies en définissant la classe par défaut. Les couches définies par l'utilisateur contiennent trois propriétés:

Nom : le nom de la classe est défini comme un vecteur de caractères. Utiliser la propriété Name pour définir les couches et les indexer dans une grille. Si on ne spécifie pas le nom de la classe, le programme en affecte automatiquement un au moment de la formation.

Description : décrit une seule ligne de la couche et est définie comme un vecteur de caractères. Cette description apparaît lorsque l'on affiche la couche dans un tableau de couches. La Valeur par défaut est le nom de la classe de couche.

Type de couche : spécifiée en tant que chaine de caractères. Une valeur Type apparaît lorsque la couche est affichée dans un tableau de couche. La valeur par défaut est le nom de la classe de couche si la couche n'a pas d'autres propriétés, on peut supprimer la section des propriétés. **[20]**

II.6 Exemples de modèles de CNN :

La forme la plus commune d'une architecture CNN empile quelques couches Conv-ReLU, les suit avec des couches Pool, et répète ce schéma jusqu'à ce que l'entrée soit réduite dans un espace d'une taille suffisamment petite. À un moment, il est fréquent de placer des couches entièrement connectées (FC). La dernière couche entièrement connectée est reliée vers la sortie. Voici quelques architectures communes CNN qui suivent ce modèle :

INPUT -> CONV -> RELU -> FC

INPUT -> [CONV -> RELU -> POOL] * 2 -> FC -> RELU -> FC Ici, il y a une couche de CONV unique entre chaque couche POOL

INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL] * 3 -> [FC -> RELU] * 2 -> FC

Ici, il y a deux couches CONV empilées avant chaque couche POOL.

L'empilage des couches CONV avec de petits filtres de pooling (plutôt un grand filtre) permet un traitement plus puissant, avec moins de paramètres. Cependant, avec l'inconvénient de demander plus de puissance de calcul (pour contenir tous les résultats intermédiaires de la couche CONV). **[21]**

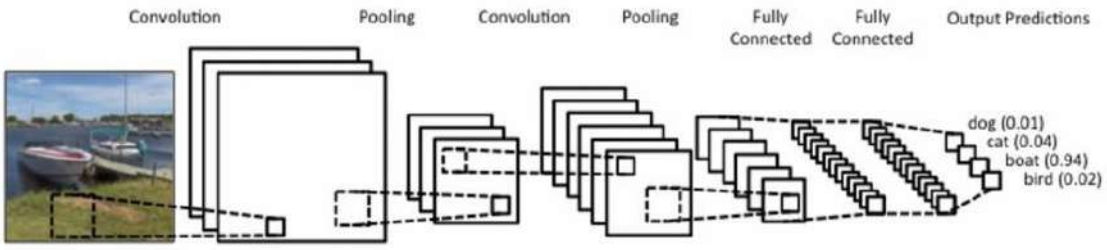


Figure II.9 Exemples de modèles de CNN

II.7 Choix des hyper paramètres :

Les CNNs utilisent plus d'hyper paramètres qu'un MLP standard. Même si les règles habituelles pour les taux d'apprentissage et des constantes de régularisation s'appliquent toujours, il faut prendre en considération les notions de nombre de filtres, leur forme et la forme du max pooling. **[21]**

II.7.1 Nombre de filtres :

Comme la taille des images intermédiaires diminue avec la profondeur du traitement, les couches proches de l'entrée ont tendance à avoir moins de filtres tandis que les couches plus proches de la sortie peuvent en avoir davantage. Pour égaliser le calcul à chaque couche, le produit du nombre de caractéristiques et le nombre de pixels traités est généralement choisi pour être à peu près constant à travers les couches. Pour préserver l'information en entrée, il faudrait maintenir le nombre de sorties intermédiaires (nombre d'images intermédiaire multiplié par le nombre de positions de pixel) pour être croissante (au sens large) d'une couche à l'autre.

Le nombre d'images intermédiaires contrôle directement la puissance du système, dépend du nombre d'exemples disponibles et la complexité du traitement. **[21]**

II.7.2 Forme de filtres :

Les formes de filtre varient grandement dans la littérature. Ils sont généralement choisis en fonction de l'ensemble de données. Les meilleurs résultats sur les images de MNIST (28*28) sont habituellement dans la gamme de 5*5 sur la première couche, tandis que les ensembles de données d'images naturelles (souvent avec des centaines de pixels dans chaque dimension) ont tendance à utiliser de plus grands filtres de première couche de 12*12, voire 15*15.

Le défi est donc de trouver le bon niveau de granularité de manière à créer des abstractions à l'échelle appropriée et adaptée à chaque cas. **[21]**

II.7.3 Forme de Max pooling :

Les valeurs typiques sont 2*2(figure II.10). De très grands volumes d'entrée peuvent justifier un pooling 4*4 dans les premières couches. Cependant, le choix de formes plus grandes va considérablement réduire la dimension du signal, et peut entraîner la perte de trop d'information. **[21]**

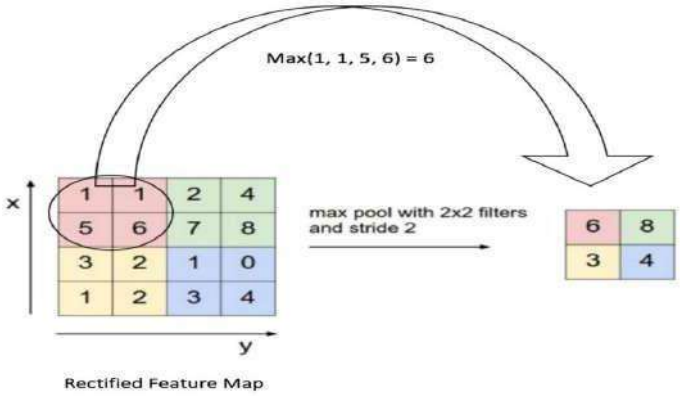


Figure II.10 Exemple de max pooling (2*2).

II.8 Avantages de CNNs :

Un avantage majeur des réseaux convolutifs est l'utilisation d'un poids unique associé aux signaux entrant dans tous les neurones d'un même noyau de convolution. Cette méthode réduit l'empreinte mémoire, améliore les performances et permet une invariance du traitement par translation. C'est le principal avantage du CNN par rapport au MLP, qui lui considère chaque neurone indépendant et donc affecte un poids différent à chaque signal entrant.

Lorsque le volume d'entrée varie dans le temps (vidéo ou son), il devient intéressant de rajouter un paramètre de temporisation (delay) dans le paramétrage des neurones. On parlera

Comparés à d'autres algorithmes de classification de l'image, les réseaux de neurones convolutifs utilisent relativement peu de prétraitement. Cela signifie que le réseau est responsable de faire évoluer tout seul ses propres filtres (apprentissage sans supervision), ce qui n'est pas le cas d'autres algorithmes plus traditionnels. L'absence de paramétrage initial et d'intervention humaine est un atout majeur des CNN. **[22]**

II.9 Méthode de régulation :

Pour ne pas tomber dans le problème de sur apprentissage il y a des méthodes de régularisation à utiliser**. [21]**

II.9.1 Empirique :

II.9.1.1 Dropout :

Les couches "FC" (Fully Connected) occupent la majeure partie de la mémoire du CNN. D'ailleurs le concept de FC crée un problème exponentiel de mémoire appelé "overfitting" ("sur-connexion" conduisant au sur-apprentissage) ralentissant le traitement de l'information. Pour prévenir cela, la méthode du dropout est utilisée pour "éteindre" les neurones aléatoirement (avec une probabilité prédéfinie, souvent un neurone sur deux) ainsi que les neurones périphériques. Ainsi, avec moins de neurones, le réseau est plus réactif et peut donc apprendre plus rapidement. À la fin de la séance d'apprentissage, les neurones "éteints" sont "rallumés" (avec leurs poids originaux). Plus la couche FC est proche de l'image source, moins on éteindra de neurones.

L'objectif est d'éteindre et rallumer les neurones aléatoirement, dans le cadre d'entraînements successifs. Une fois les séries d'entraînements terminées, on rallume tous les neurones et on utilise le réseau comme d'habitude. Cette technique a montré non seulement un gain dans la vitesse d'apprentissage, mais en déconnectant les neurones, on a aussi limité des effets marginaux, rendant le réseau plus robuste et capable de mieux généraliser les concepts appris. **[21]**

II.9.1.2 DropConnect :

Le DropConnect est une évolution du dropout, où on ne va non plus éteindre un neurone, mais une connexion (l'équivalent de la synapse), et ce de manière toujours aléatoire. Les résultats sont similaires (rapidité, capacité de généralisation de l'apprentissage), mais présentent une différence au niveau de l'évolution des poids des connexions. Une couche FC avec un DropConnect peut s'apparenter à une couche à connexion "diffuse". **[21]**

II.9.1.3 Pooling stochastique :

Le pooling stochastique reprend le même principe que le Max-pooling, mais la sortie choisie sera prise au hasard, selon une distribution multinomiale définie en fonction de l'activité de la zone adressée par le pool. Dans les faits, ce système s'apparente à faire du Max-pooling avec un grand nombre d'images similaires, qui ne varient que par des déformations localisées. On peut aussi considérer cette méthode comme une adaptation à des déformations élastiques de

l'image. C'est pourquoi cette méthode est très efficace sur les images MNIST (base de données d'images représentant des chiffres manuscrits). La force du pooling stochastique est de voir ses performances croître de manière exponentielle avec le nombre de couches du réseau. **[21]**

II.9.2 Explicite :

II.9.2.1 Taille du réseau :

La manière la plus simple de limiter le sur apprentissage est de limiter le nombre de couches du réseau et de libérer les paramètres libres (connexions) du réseau. Ceci réduit directement la puissance et le potentiel prédictif du réseau. C'est équivalent à avoir une "norme zéro". **[21]**

II.9.2.2 Dégradation du poids :

Le concept est de considérer le vecteur des poids d'un neurone (liste des poids associés aux signaux entrants), et de lui rajouter un vecteur d'erreur proportionnel à la somme des poids (norme 1) ou du carré des poids (norme 2 ou euclidienne). Ce vecteur d'erreur peut ensuite être multiplié par un coefficient de proportionnalité que l'on va augmenter pour pénaliser davantage les vecteurs de poids forts.

- La régularisation par norme 1 : La spécificité de cette régulation est de diminuer le poids des entrées aléatoires et faibles et d'augmenter le poids des entrées "importantes". Le système devient moins sensible au bruit.
- La régularisation par norme 2 : (norme euclidienne) La spécificité de cette régulation est de diminuer le poids des entrées fortes, et de forcer le neurone à plus prendre en compte les entrées de poids faible.

Les régularisations par norme 1 et norme 2 peuvent être combinées : c'est la "régularisation de réseau élastique" (Elastic net régulation). **[21]**

II.10 Exemple d'application avec un réseau de neurones CNN:

Pour illustrer l'architecture, le fonctionnement et l'interaction des différentes couches, nous avons mis en œuvre une application sous Matlab. Celle-ci concerne la reconnaissance automatique de nombre manuscrits allant de 0 jusqu'à 9 issus d'images. Cette application nous permettra par ailleurs de mettre en évidence l'influence des différents paramètres du réseau CNN sur la précision et le temps d'exécutions. Pour cela nous avons envisagé deux cas, dans

le premier, nous ferons varier la taille de filtre. Tandis que dans le deuxième, nous modifierons les taux Alpha et beta tout en gardant une taille de filtre constante.

II.10.1 Matériel utilisé :

- Un PC Accer i3-370M CPU 2.4 GHZ
- RAM de taille 3 Go
- Système d'exploitation Windows 7

II.10.2 Présentation de l'application :

Nous avons entrepris mettre en œuvre un réseau de neurones CNN pour réaliser la reconnaissance automatique de nombres manuscrits. Ce réseau devra détecter et classer, à chaque fois que l'on présentera une image avec un nombre écrit à la main. Pour atteindre notre objectif nous sommes passés par trois grandes étapes :

- Mise au point de la base de données.
- Création des classes.
- La classification et l'apprentissage.

Enfin, une fois ces étapes réalisés, nous pourrons évaluer les performances du réseau de neurones CNN en termes de précision lors de la classification d'images tests.

a) La base de données :

La base de données que nous avons utilisée contient les images des nombres allant de 0 à 9 écrits à la main de plusieurs façons (chaque nombre est écrit de plusieurs manières pour mimer au maximum la réalité). Ceci nous donne une base de données contenant 10.000 images. Ces images sont en niveau de gris avec une taille de 28*28 pixels.

Pour faire apprendre notre réseau de neurone nous utiliserons 8.000 images et nous garderons 2.000 pour tester les performances du CNN. Il faut préciser que nous avons repartis aléatoirement les images entre celle utilisées pour l'apprentissage et pour les tests.

b) Création des classes :

Pour pouvoir faire apprendre le réseau et lui permettre de classer les images présentées, nous avons dû préalablement créer 10 classes qui correspondent aux 10 nombres. La figure. II.11 illustre cette opération ou chaque ligne va correspondre à une classe.

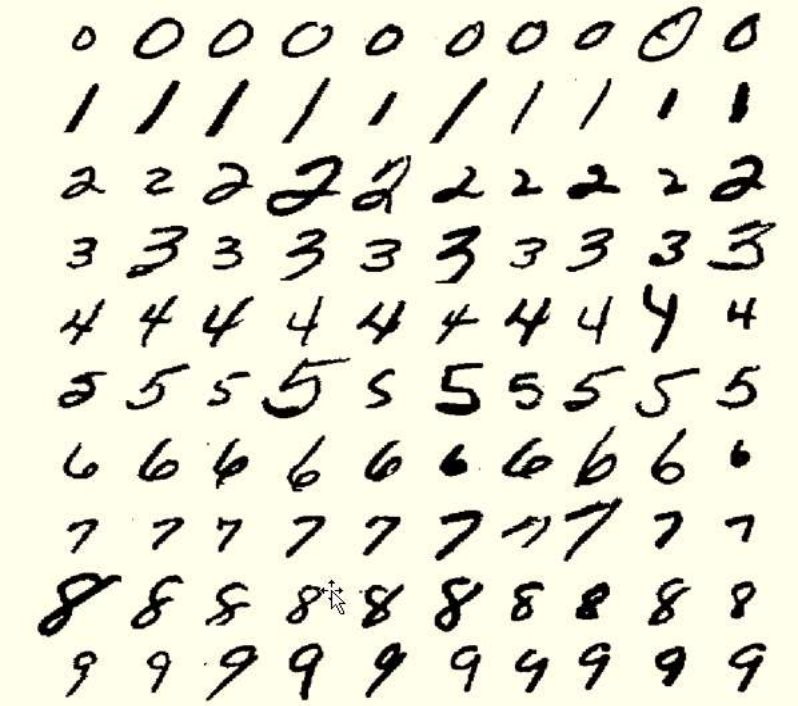


Figure II.11 Création des classes

II.10.3 Résultats de la simulation de la reconnaissance de nombres manuscrits :

Au début de l'exécution du réseau de neurone CNN, un premier prétraitement est effectué. Les images contenues dans la base de données sont redimensionnées en 28*28 pixels et transformées niveau de gris. Cette opération créera une matrice images – classes qui est sauvegardée dans un fichier MNIST.

Dans cette expérimentation, nous avons utilisé une seule architecture. Elle se compose d'une couche de convolution - une couche de Relu - une couche de max pooling - une couche full connected - une couche de Relu et enfin une couche de softmax.

En faisant varier la taille du filtre et avec les coefficients suivants fixes :

Alpha = 0.01

Beta = 0.95

Nous obtenons les résultats résumés dans les tableaux ci-dessous :

Tableau II-1 Les performances de la taille de filtre et le nombre d'epoch sur un réseau CNN

Taille de filtre	Itération (Epoch)	Précision	Temps d'exécution
Filtre 3*3	3	93.85%	4min 27s
	5	94.10%	7min 04s
Filtre 5*5	3	93.60%	3min 69s
	5	94.50%	6min 11s
Filtre 9*9	3	94.65%	2min 42s
	5	96.50%	5min 15s
Filtre 11*11	3	93.95%	2min 37s
	5	95.95%	4min 34s
Filtre 15*15	3	93.00%	2min 19s
	5	95.65%	3min 36s

Au vu de ces performances, nous pouvons déduire que le nombre d'itérations influe sur la qualité de l'apprentissage. Il augmente aussi considérablement le temps d'exécution. De plus, la taille de la fenêtre (filtre) à un impact considérable sur la précision. En effet, il faut choisir minutieusement ces dimensions pour obtenir des performances optimales. Un filtre trop grand diminue le temps d'exécution mais dégrade la précision. La taille optimale du filtre ne peut être obtenue qu'à travers plusieurs expériences.

Par ailleurs, l'influence des coefficients est démontrée par les résultats regroupés dans le tableau ci-dessous. La meilleure précision est obtenue avec un coefficient Alpha faible et un coefficient Beta proche de 1.

Tableau II-2 Les performances des coefficients d'apprentissage sur un réseau CNN

Filtre 9*9	***Alpha***	**0.01**	**0.2**	**0.9**	**1.5**	**0.01**	**0.01**
	Beta	0.95	0.59	0.95	0.95	0.01	1.5
	La Précision	94.65%	89.45%	39%	11.5%	87.9%	11.5%
	Temps D'exécution	2min 30s	2min 51s	2min 39s	2min 29s	3min 3s	4min 59s

II.11 Conclusion :

Ce chapitre a été pour nous l'occasion de présenter le réseau de neurones convolutionel CNN. Nous avons expliqué et illustré le fonctionnement et l'architecture de ce type de réseau de neurones. Nous avons également détaillé les différents paramètres et règles qui assurent l'apprentissage profond de ces réseaux. Pour expliciter ces concepts, nous avons implémenté un réseau CNN et nous l'avons utilisé dans une expérience de reconnaissance automatique de nombres manuscrits. Cette application a permis de mettre en exergue l'influence de plusieurs paramètres sur les performances du CNN. De plus, elle nous a permis de maitriser son architecture et les différents prétraitements nécessaires à l'obtention des meilleures performances. Ces expertises nous servirons à réalisation de notre projet en l'occurrence la reconnaissance automatique de mots manuscrits arabes. Les détails de cette application sont présentés dans le prochain chapitre.

Chapitre3 :

L'application du CNN pour la classification et la reconnaissance automatique des mots manuscrites en arabe

III.1 Introduction :

Dans ce chapitre nous présentons notre conception du système de reconnaissance automatique des mots manuscrits en arabe. Ce système est basé sur les réseaux de neurones convolutionnels CNN. L'implémentation de ce réseau de ce logiciel a été réalisée en langage Python avec les bibliothèques trensorflow_GPU, keras, Panda et Scikit-learn nécessaires à l'apprentissage et la classification d'images.

Comme le réseau CNN est le noyau de notre système, nous vous détaillerons les différentes étapes nécessaires à son implémentation. Par ailleurs, plusieurs expérimentations ont été réalisées pour obtenir un système intelligent qui reconnaisse un maximum de mots manuscrits en arabes que l'on lui présente.

III.2 Présentation des outils de développement :

III.2.1 Logiciels et librairies utilisés dans la simulation :

III.2.1.1 Python :

Python est un langage de programmation de haut niveau interprété (il n'y a pas d'étape de compilation) et orienté objet avec une sémantique dynamique. Il est très sollicité par une large communauté de développeurs et de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction du cout de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes. Python et ses bibliothèques sont disponibles (en source ou en binaires) sans charges pour la majorité des plateformes et peuvent être redistribués gratuitement. **[25]**

III.2.1.2 Spyder :

Spyder est un environnement scientifique puissant écrit en Python, pour Python, et conçu par et pour les scientifiques, les ingénieurs et les analystes de données. Il offre une combinaison unique des fonctionnalités avancées d'édition, d'analyse, de débogage et de profilage d'un outil de développement complet avec l'exploration de données, l'exécution interactive, l'inspection approfondie et les superbes capacités de visualisation d'un progiciel scientifique. Au-delà de ses nombreuses fonctionnalités intégrées, ses fonctionnalités peuvent être étendues davantage via son système de plug-in et son API. En outre, Spyder peut également être utilisé en tant que bibliothèque d'extensions PyQt5, permettant aux développeurs de tirer

parti de ses fonctionnalités et d'incorporer ses composants, tels que la console interactive, dans leur propre logiciel PyQt. **[26]**

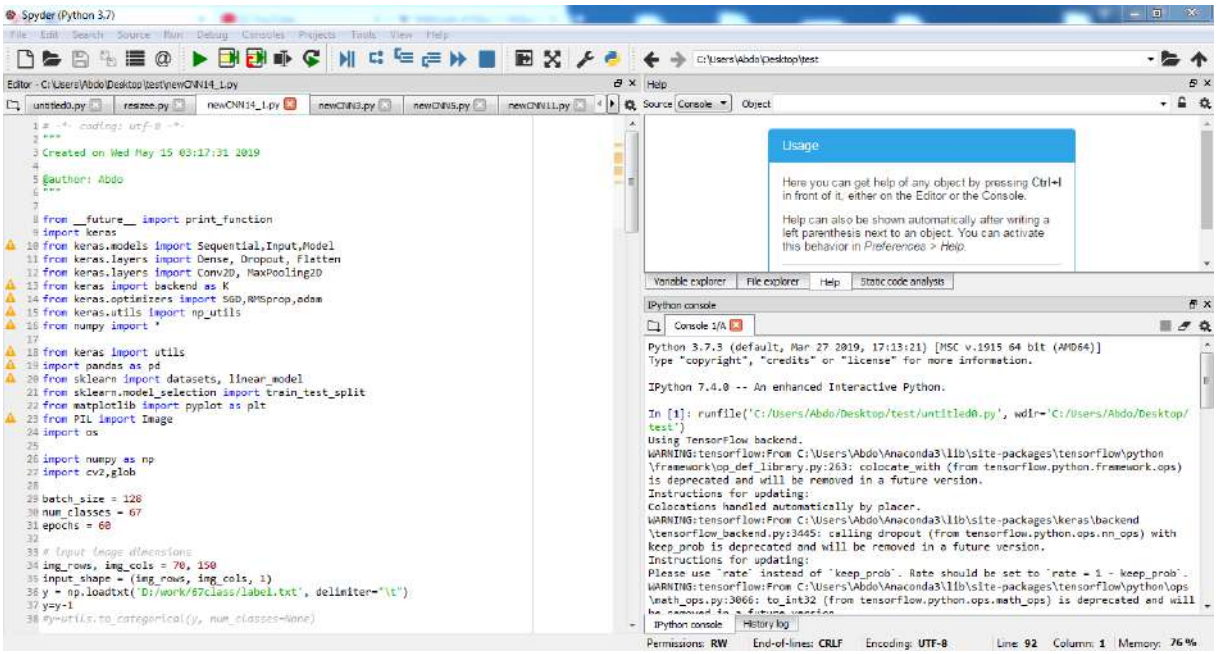


Figure III.1 L'environnement Spyder.

III.2.1.3 TensorFlow :

TensorFlow est un environnement « framework » de programmation pour le calcul numérique qui a été rendu Open Source par Google en Novembre 2015. Depuis son lancement, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning et donc les réseaux de neurones. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont principalement faites via des tables de données multi-dimensionnelles, appelées Tenseurs (Tensor). Un Tensor à deux dimensions est l'équivalent d'une matrice. Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow: Gmail, Google Photos, Reconnaissance de voix. **[27]**

III.2.1.4 Keras :

Keras est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur TensorFlow ou Theano. Il a été développé en mettant l'accent sur l'expérimentation rapide. Être capable d'aller de l'idée à un résultat avec le moins de délai possible est la clé pour faire de bonnes recherches. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), et son principal auteur et mainteneur est François Chollet, un ingénieur Google.

En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow. Chollet a expliqué que Keras a été conçue comme une interface plutôt que comme un cadre d'apprentissage end to end. Il présente un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilitent la configuration des réseaux neuronaux indépendamment de la bibliothèque informatique de backend. Microsoft travaille également à ajouter un backend CNTK à Keras aussi. **[28]**

III.2.1.5 Scikit-learn :

Scikit-learn est une bibliothèque libre Python dédiée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria et Télécom Paris Tech.

Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec des autres bibliothèques libre Python, notamment NumPy et SciPy. **[29]**

III.2.1.6 Pandas :

La librairie Pandas fait partie des librairies de base pour la data science en Python. Pandas fournit des structures de données puissantes et simples à utiliser, ainsi que les moyens d'opérer rapidement des opérations sur ces structures. Dans ce chapitre, nous verrons l'intérêt de la librairie Pandas, ainsi que les opérations basiques sur l'objet phare de cette librairie, le data frame **[30].** Nous l'avons utilisé pour :

- charger le fichier de données en tant que cadre de données Pandas et analyser les données.
- importer et manipuler, depuis **Sklearn**, le module de jeux de données. De plus, nous l'utiliserons pour réaliser une régression linéaire du modèle avec la fonction linear_model,
- tracer des graphiques à partir de la bibliothèque Matplotlib.

III.2.2 Configuration utilisé dans la simulation :

Lors des différents tests réalisés dans le chapitre précèdent, nous nous sommes aperçus que l'apprentissage des réseaux de neurones CNN nécessitait des ressources informatiques importantes tant en mémoire qu'en puissance du processeur graphique (GPU). Pour pallier à ce

problème, nous avons dû utiliser la configuration du matériel suivante :

- Un PC i5-4460 CPU 3.20 GHZ
- Carte graphique Nvidia GTX 960 Windforce 4Gb
- RAM de taille 8 Gb
- Système d'exploitation Windows 10

III.3 Les différentes étapes de notre expérimentation :

Pour illustrer le fonctionnement de notre logiciel de reconnaissance de mots manuscrits arabe, nous vous présentons l'organigramme ci-dessous.

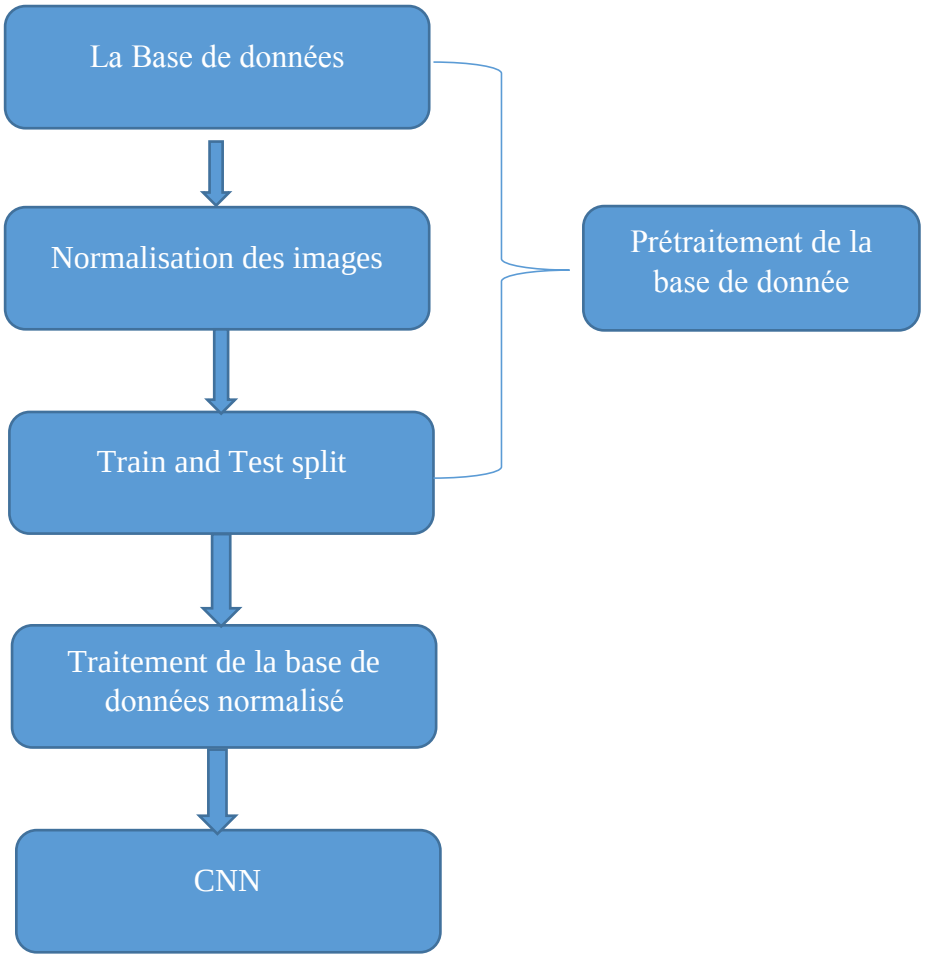


Figure III.2 L'organigramme des étapes de notre travaille.

- **La Base de données :**

La base de données que nous avons utilisée contient 7032 images. Celles-ci sont en niveau de gris, avec des tailles différentes se composant de 67 classes. La figure suivante représente certains exemples de mots contenus dans cette base de données.



Figure III.3 La Base de données de mots manuscrits arabe.

- **Normalisation des images :**

La normalisation et un processus qui modifie la plage des valeurs d'intensité de pixel. Cette opération de prétraitement est indispensable pour transformer et unifier les dimensions de toutes les images contenues dans la base. L'ensemble de ces images seront redimensionnées en une même taille 150*70 pixel.

- **Création de jeu de données d'apprentissage et test (Train et Test split) :**

Lorsqu'on travaille avec une seule base de données, nous sommes obligés de la scinder en 2 pour en tirer deux jeux de données distincts ; le premier servira à l'entrainement du réseau de neurones tandis que le deuxième sera utilisé pour évaluer ses performances. Dans notre cas, nous avons utilisé 4922 images pour le l'apprentissage, et 2110 images pour les tests d'évaluations.

- **Traitement de la base de données normalisé :**

Une dernière opération de prétraitement de la base avant l'utilisation du réseau consiste

à associer aux images une étiquette (Labels en anglais).

Comme l'intervalle de la probabilité est compris entre [0 1] et que l'intervalle de valeurs des pixels de l'image est compris entre [0 255], nous avons devisé les valeurs de chaque image par 255. Ceci nous permet d'obtenir une image normalisée dont les valeurs sont comprises dans l'intervalle de [0 1].

Enfin, nous précisons les classes de chaque image. Dans notre cas, nous disposons de 67 classes. De ce fait, nous créant vecteur de 67 éléments. La procédure pour créer ce vecteur binaire est simple. La classe 1 en codée comme suite : on met un à (1) le premier élément et les autres sont mis à zéros (0). Dans le cas de la classe 2, on met à un (1) le deuxième élément et les restes des éléments sont mis à zéros (0) et ainsi de suite.

Exemple :

67 éléments

Code de la classe 1 : 1 0 0 0 0 …… 0

Code de la classe 2 : 0 1 0 0 0 ……...0 0 0

Code de la classe 66 :0 0 0…….0 1 0

Code de la classe 67 :0 0 0…….0 0 1

III.4 Notre système de reconnaissance de mots arabe avec un CNN :

Nous consacrons cette partie pour décrire nos différents tests et expérimentations réalisées ainsi que les résultats obtenus lors de la reconnaissance de mots manuscrits arabes. Pour atteindre à un taux de précision le plus proche de 100%, nous avons effectué un nombre important de tests faisons varier le nombre de couches et la taille des filtres. Ces expériences nous ont permis d'une part de mettre en évidence encore une fois l'influences de ces paramètres sur l'efficacité du réseau et d'autre part de trouver ma configuration optimale pour réaliser notre tâche avec les meilleures performances. Dans un premier temps, nous allons démontrer l'influence du nombre de couches de convolution, ensuite celle de la couche de maxpooling, puis l'influence de la taille et du nombre de filtres et enfin, l'influence du bloque sur la précision

et les temps d'exécution. Notons que l'ensemble des expérimentations ont été menées sur la même base de données avec des d'images de mêmes tailles 150*70 pixels. Par ailleurs, nous avons fixé le nombre d'itérations à 20.

III.4.1 L'influence du nombre de couche de convolution :

Pour démontrer l'influence du nombre de couches de convolution sur la qualité d'apprentissage du réseau CNN, nous avons adopté trois architectures qui différents uniquement de par leurs nombre de couches de convolution (une seul couche puis deux et enfin trois couches). Ces trois architectures ainsi que les résultats obtenus sont regroupés dans le tableau et les figures ci-dessous.

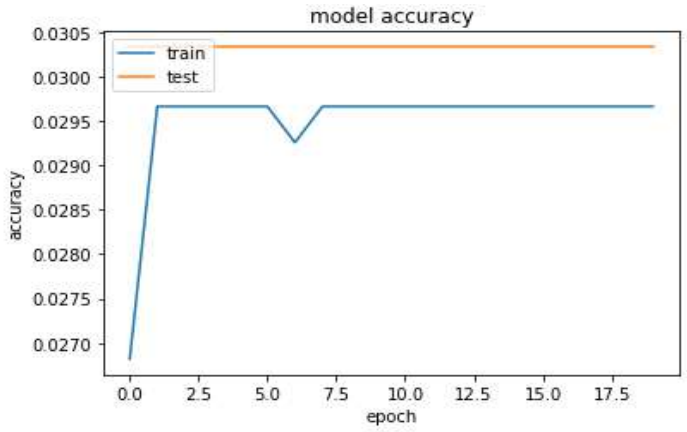


Figure III.4 La précision de validation d'une seule couche de convolution

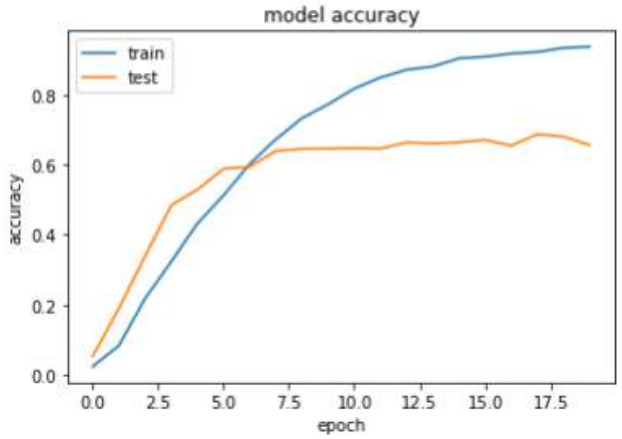


Figure III.5 La précision de validation de deux couches de Convolution.

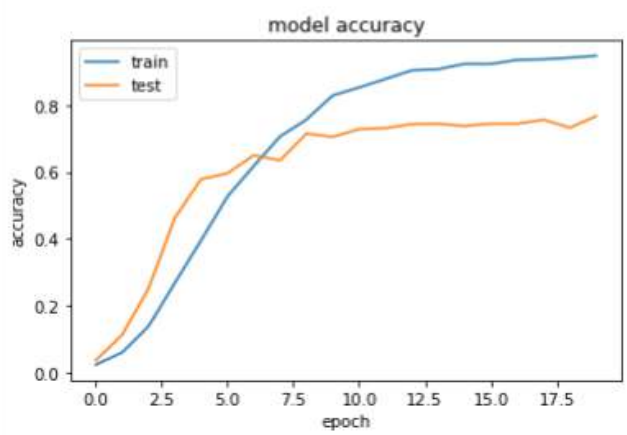


Figure III.6 La précision de validation de trois couches de convolution.

Tableau III.1 L'influence de la couche de convolution**.**

Architectures #1		**Architectures #2**		**Architectures #3**	
Convolution Filtre 7*7		Convolution Filtre 7*7 Convolution Filtre 7*7		Convolution Filtre 7*7 Convolution Filtre 7*7 Convolution Filtre 7*7	
MaxPooling		MaxPooling		MaxPooling	
Dropout		Dropout		Dropout	
Flatten		Flatten		Flatten	
Dense		Dense		Dense	
Dropout		Dropout		Dropout	
Sofmax		Sofmax		Sofmax	
Précision (%)	Temps d'exécution	Précision (%)	Temps d'exécution	Précision (%)	Temps d'exécution

3.03%	1min 21s	65.5%	3min 2s	76.7%	4min 22s

Nous avons vu dans le chapitre précédent que la couche de convolution est l'élément de base qui compose la structure des réseaux de neurones CNN.

D'après **les Figure (III.4, III.5, III.6)**, nous pouvons déduire les conclusions suivantes :

Quand nous utilisons un CNN avec deux couches de convolution, nous observons une augmentation importante de la précision (62.47%) mais en parallèle, nous constatons une multiplication par presque trois de la durée d'exécution. Quand nous utilisons trois couches de convolution nous observons une augmentation modérée de la précision (11.2%) par rapport à la configuration avec deux couches. De même, l'augmentation de la durée d'exécution reste beaucoup moins importante.

Depuis ces différents tests nous pouvons conclure qu'arriver à un certain nombre de couches de convolutions, les performances du réseau augmentent peu en comparaison avec les temps d'exécution. Ce qui nous permet de conclure qu'il faut trouver, en réalisant plusieurs tests, l'architecture optimale qui permet d'obtenir le meilleur compromis précision / temps d'exécution.

III.4.2 L'influence du nombre de la couche de maxpooling :

Pour appréhender l'influence du nombre de couche de maxpooling dans un réseau de neurones CNN, nous augmentons le nombre de couche au fur et à mesure. Cependant, nous avons fixé le nombre de couches de convolution à trois avec un filtre de taille 7*7. Les deux architectures ainsi que les résultats associés sont regroupés dans le tableau et les figures ci-dessous.

Tableau III.2 L'influence de la couche de maxpooling.

Architectures #1	Architectures #2
Convolution 7*7 convolution 7*7 convolution 7*7	Convolution 7*7 convolution 7*7 convolution 7*7

Maxpooling		Maxpooling	
		Convolution 7*7 Convolution 7*7 Convolution 7*7	
		Maxpooling	
Dropout		Dropout	
Flatten		Flatten	
Dense		Dense	
Dropout		Dropout	
Softmax		Softmax	
Précision (%)	Temps d'exécution	Précision (%)	Temps d'exécution
76.7%	4 min 22s	88.2%	4min 44s

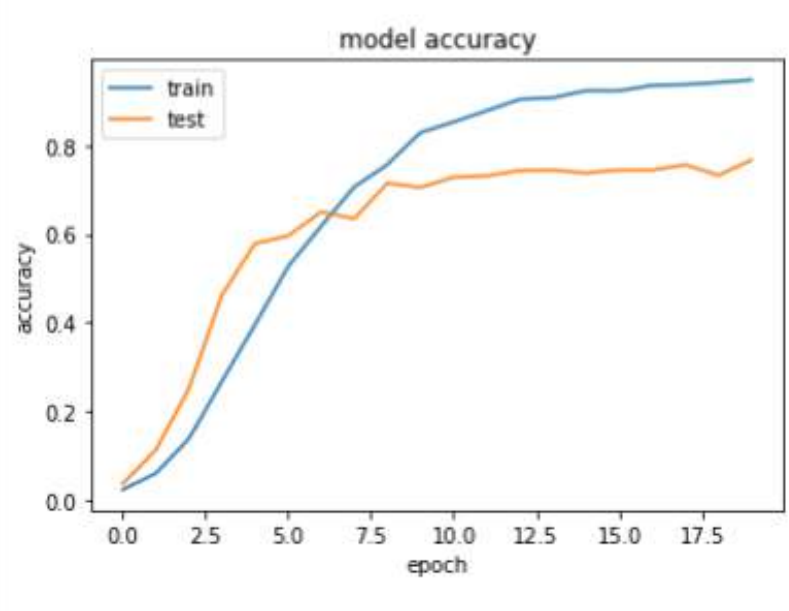


Figure III.7 La précision avec une seule couche de maxpooling.

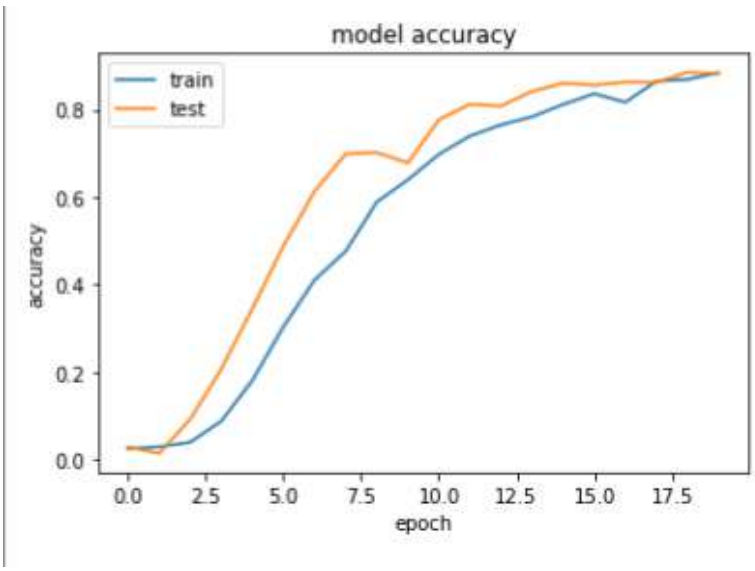


Figure III.8 La précision avec deux couches de maxpooling.

Nous l'avons constaté dans le chapitre précédent que la couche de maxpooling est une méthode permettant de prendre une large image et d'en réduire la taille tout en préservant les informations les plus importantes qu'elle contient. (Souvent par sous_échantillonage).

D'après **les Figure (III.7, III.8)**, nous pouvons déduire les remarques suivantes :

Quand on utilise deux couches de maxpooling on constate une augmentation de 11.5% de précision de validation par rapport à une seule couche. Cela est dû essentiellement à la

réduction de la taille des images traitées tout en gardant uniquement les informations importante. Quant à l'augmentation du temps de calcul de 22 secondes. Elle est due à cette phase supplémentaire de compression.

III.4.3 L'influence de la taille du filtre :

Dans cette partie, nous désirons mettre l'accent sur l'influence de la taille du filtre dans les performances du réseau CNN. Pour cela, nous avons choisis de vous présenter les résultats de trois tailles de filtres : 3*3, 5*5, 7*7 et 9*9. De plus, nous avons retenu la meilleure architecture depuis les expériences suivantes. Les résultats obtenus avec les trois nouvelles architectures sont présentés dans le tableau et les figures ci-dessous.

Tableau III.3 L'influence de la taille de filtre.

Architectures #1	Architectures #2	Architectures #3	Architectures #4
Convolution 3*3 Convolution 3*3	Convolution 5*5 Convolution 5*5	Convolution 7*7 Convolution 7*7	Convolution 9*9 Convolution 9*9
Maxpooling			
Convolution 3*3 Convolution 3*3	Convolution 5*5 Convolution 5*5	Convolution 7*7 Convolution 7*7	Convolution 9*9 Convolution 9*9
Maxpooling	Maxpooling	Maxpooling	Maxpooling
Dropout	Dropout	Dropout	Dropout
Flatten	Flatten	Flatten	Flatten
Dense	Dense	Dense	Dense
Dropout	Dropout	Dropout	Dropout

Softmax		Softmax		Softmax		Softmax	
Précision (%)	Temps d'exécution	Précision (%)	Temps d'exécution	Précision (%)	Temps d'exécution	Précision (%)	Temps d'exécution
73.6%	2min 24s	82.3%	3min 4s	86.5%	3min 52s	82.2%	3min 44s

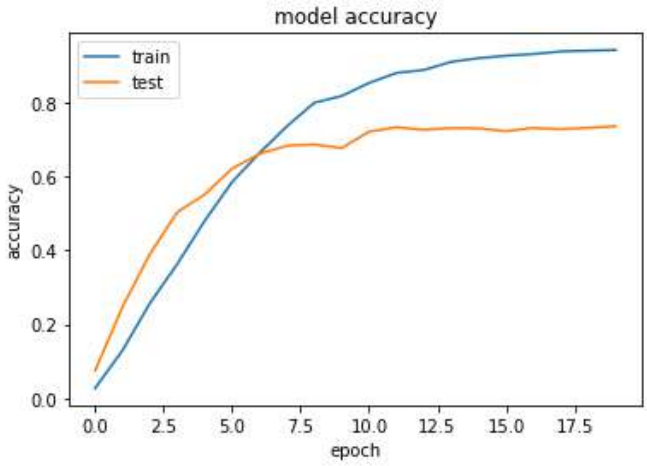


Figure III.9 La précision avec un filtre de taille 3*3.

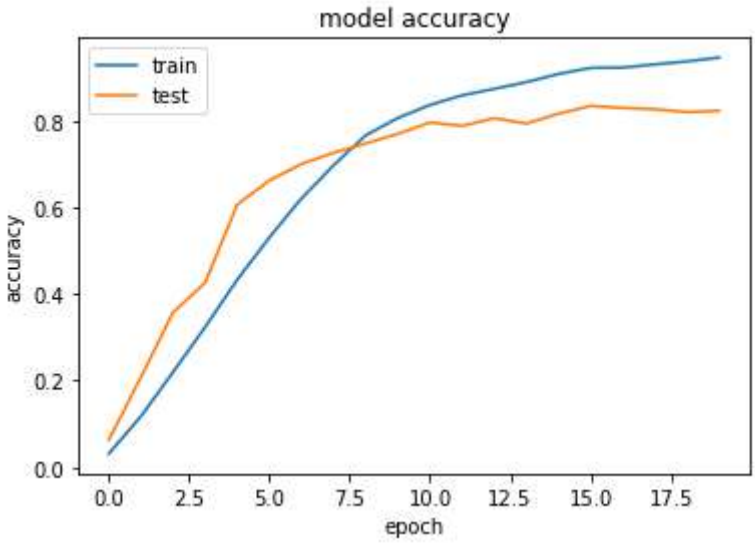


Figure III.10 La précision avec un filtre de taille 5*5

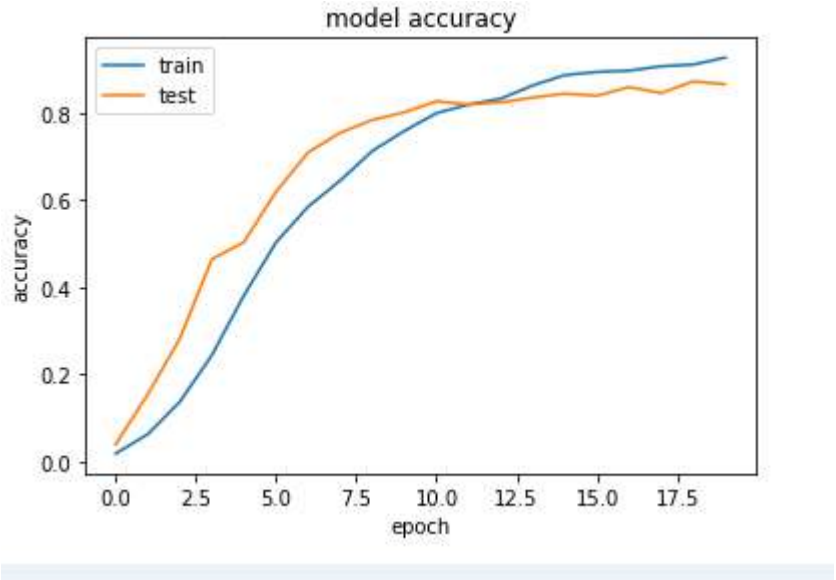


Figure III.11 La précision avec un filtre de taille 7*7

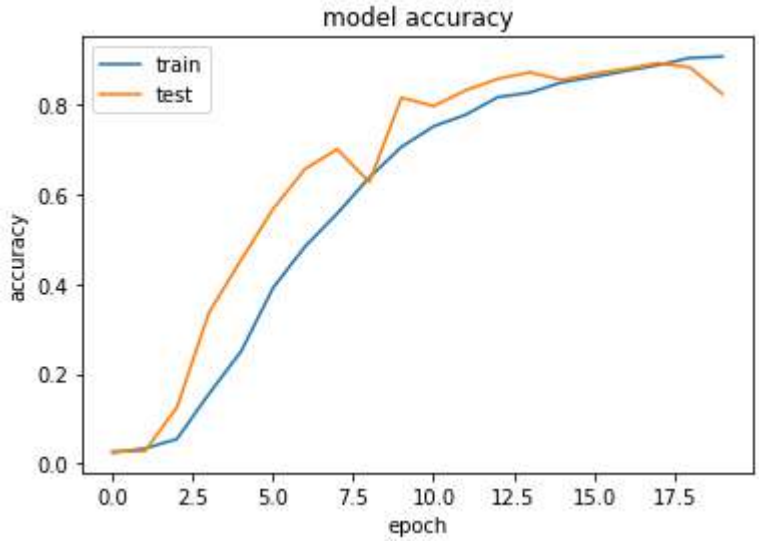


Figure III.12 La précision avec un filtre de taille 9*9

D'après **les Figure** (**III.9, III.10, III.11, III.12)** nous constatons que la meilleure taille de filtre est 7*7. DE ces tests, nous pouvons conclure logiquement que le sous-dimensionnement des filtres amoindrit la précision du CNN, mais le sur-dimensionnement provoque aussi une perte de performances.

III.4.4 L'architecture du CNN la plus performante :

Après avoir effectué les différents tests précédents et eu égard aux résultats obtenus, nous avons retenu deux configurations à comparer. Celles-ci sont présentées dans le tableau et

les figures ci-dessous.

Tableau III.4 L'influence des architectures par bloque.

Architectures #1	**Architectures #2**
Convolution 7*7 Convolution 7*7 Convolution 7*7	Convolution 7*7 32 Convolution 7*7 32
Maxpooling	
Convolution 7*7 Convolution 7*7 Convolution 7*7	Convolution 7*7 64 Convolution 7*7 64
Maxpooling	
	Convolution 7*7 128
	Maxpooling
Dropout	Dropout
Flatten	Flatten
Dense	Dense

Dropout		Dropout	
Softmax		Softmax	
Précision (%)	Temps d'exécution	Précision (%)	Temps d'exécution
88.2%	4min 44s	90.9%	4min 3s

Nous constatons qu'en utilisant l'architecture par bloque (la structure présentée dans la colonne de droite du tableau III.5), nous obtenons une augmentation de 2.7% de la précision et un gain de 41 seconds du temps d'exécution. Cette architecture est celle qui donne les meilleures performances. En effet, Le réseau de neurones CNN ainsi dimensionné peut apprendre rapidement les mots manuscrits arabes et les classés avec un taux d'erreurs très faible. C'est résultats sont remarquables du fait que notre système intelligent reconnait plus de 90% des mots manuscrits arabes que l'on lui présente. Bien entendu, nous avons réalisé un dernier test pour voir si on pouvait augmenter cette précision.

Nous avons dans ce test varié le nombre d'itérations lors de l'apprentissage du réseau CNN. Nous avons obtenu encore une augmentation de la précision en utilisant un nombre d'epoch égal à 60.

- La précision de la reconnaissance des mots tests est de 96.35%.
- Le temps d'exécution est de 8min 23s.

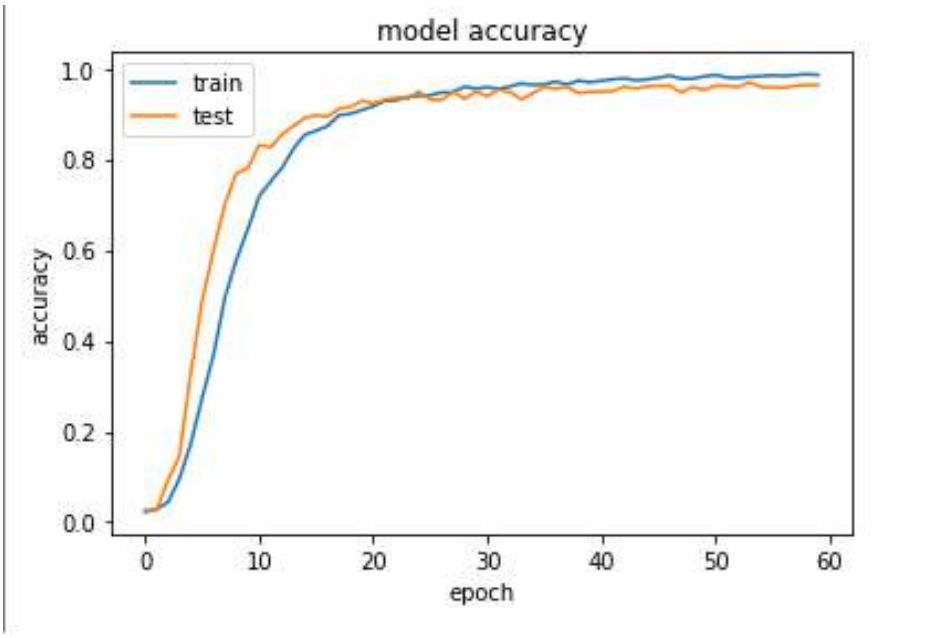


Figure III.13 La précision du système intelligent.

Cette dernière modification du nombre d'itérations nous a permis d'augmenter de 5.45% la précision du réseau de neurones CNN. Notre système intelligent de reconnaissance de mots manuscrits arabes reconnais plus de 96 mots sur 100 mots que (présentés après la phase d'apprentissage). Cependant, cette modification implique une augmentation importante du temps d'exécution.

III.5 Exemples de tests :

Pour vous illustrer les capacités de notre système intelligent à reconnaître des mots manuscrits arabes, nous présentons ci-dessous quelque exemple que nous avons réalisé avec de mots écrits à la main.

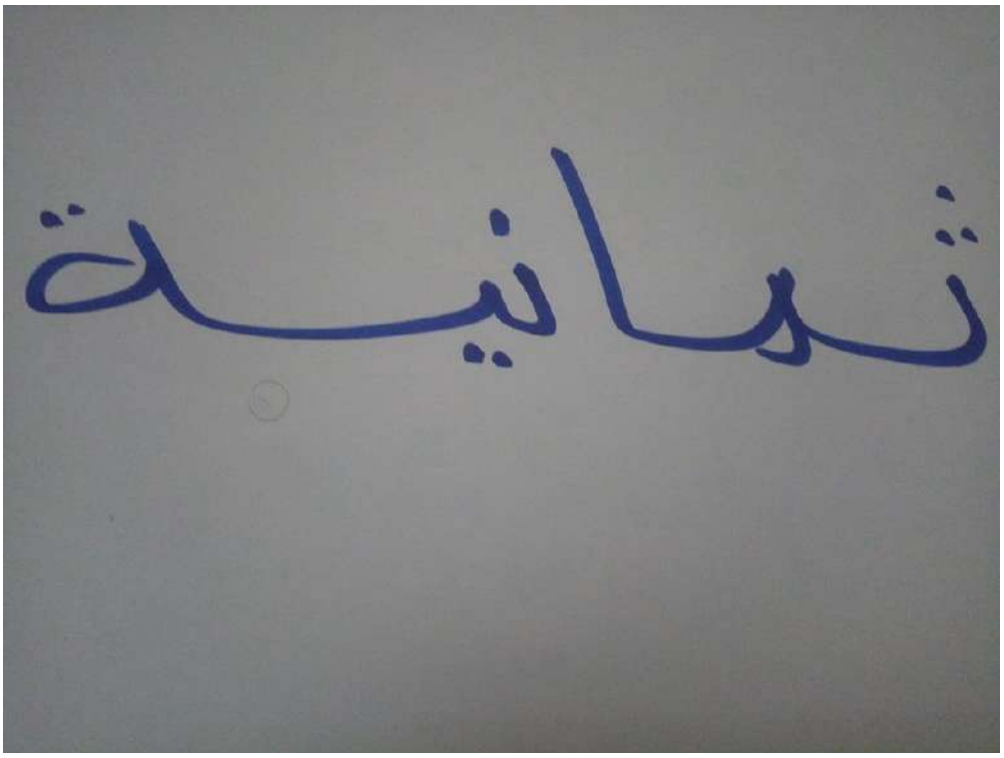


Figure III.14 La première image de test.

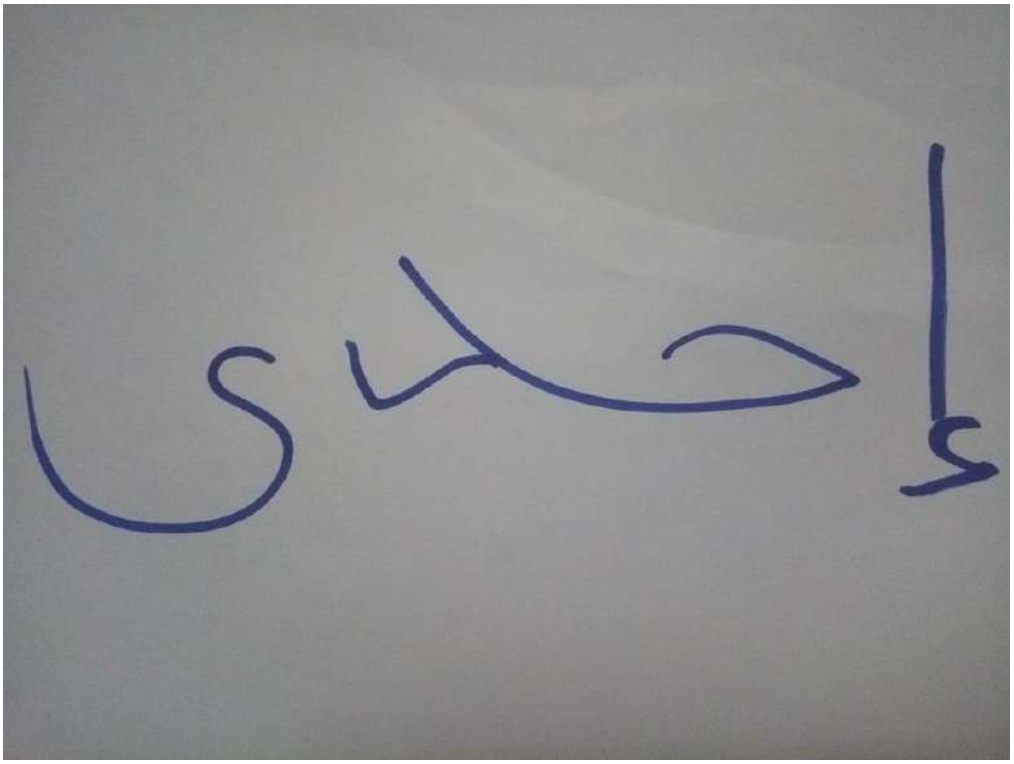


Figure III.15 La deuxième image de test.

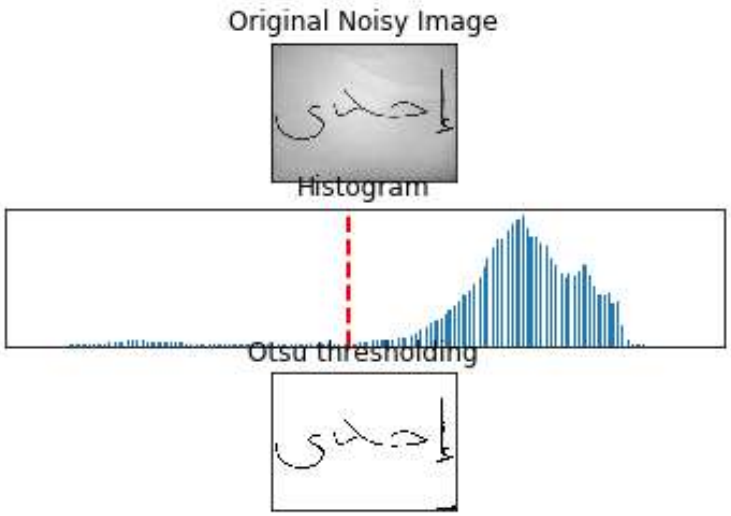


Figure III.16 résultat obtenus.

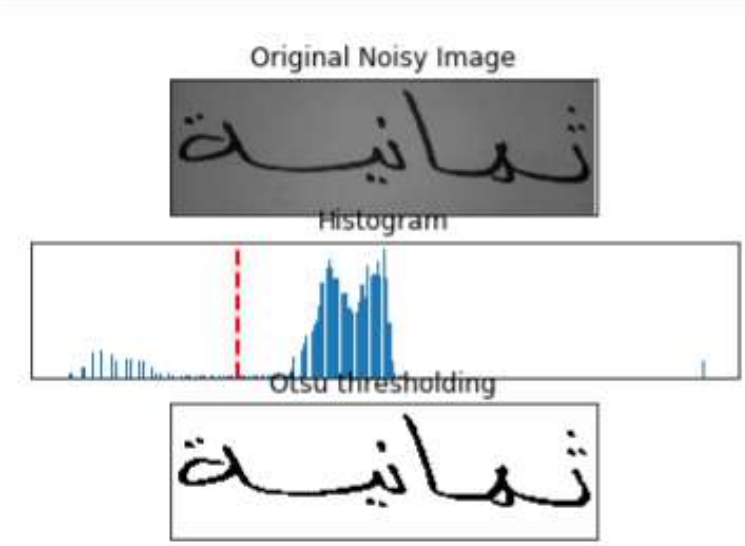


Figure III.17 résultat obtenus.

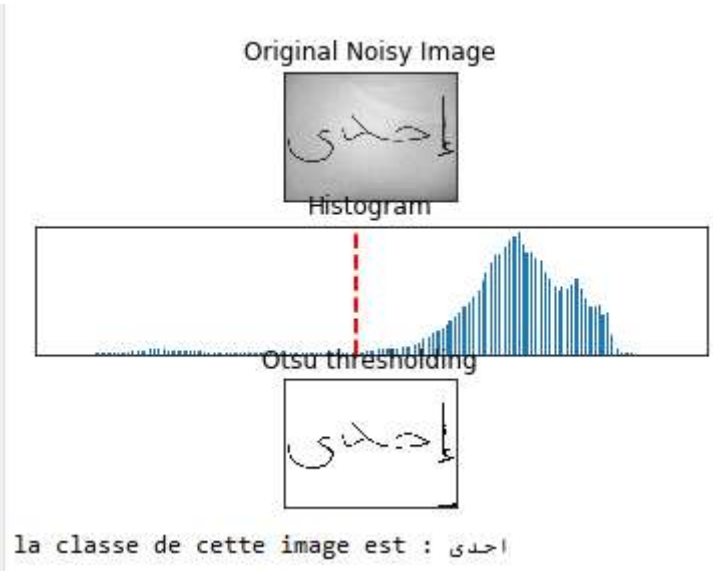


Figure III.18 résultat de classification de la première image.

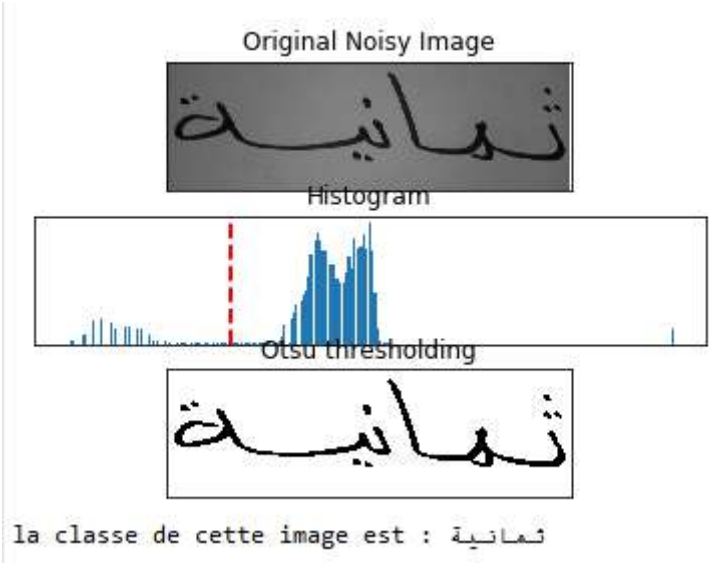


Figure III.19 résultat de classification de la deuxième image.

III.6 Conclusion :

Dans ce chapitre, nous avons présenté le fonctionnement détaillé de notre solution pour réaliser un système intelligent de reconnaissance de mots arabes. Ce système est basé sur le réseau de neurones convolutionnel. L'objectif que nous nous sommes fixé est d'obtenir les meilleures performances en termes de précision et de temps d'exécution. Pour ce faire nous avons implémenté et évalué plusieurs configurations. Nous avons commencé par augmenter le nombre de couches de convolution. Nous avons déduit que la configuration avec 3 couches était la plus adéquate pour la réalisation de notre tâche. Ensuite nous avons fait varier le nombre de couche de maxpooling est obtenu encore une meilleures précision. Nous avons également cherché la meilleure taille du filtre qui améliorerait les performances du système. Ainsi avec cette architecture et quelques autres modifications, nous avons obtenu système intelligent de reconnaissance de mots manuscrits arabes qui reconnaissent plus de 96 mots sur 100 mots. Ses performances ont été démontrées avec la présentation des exemples de reconnaissances de mot que nous avons écrit à la main.

Conclusion générale :

Au terme de ce travail, nous avons conçu et réalisé système de reconnaissance automatique intelligent des mots manuscrits en arabe. Son efficacité et ses performances sont le fruit de l'utilisation des réseaux de neurones convolutionnels.

Pour expliciter les concepts qui régissent le fonctionnement d'un réseau CNN, nous l'avons utilisé dans une expérience de reconnaissance automatique de nombres manuscrits. Cette application a permis de mettre en exergue l'influence de plusieurs paramètres sur les performances du réseau. De plus, cette première expérience nous a permis de maitriser son architecture et les différents pré-traitements nécessaires à l'obtention des meilleures performances. Ces expertises nous ont servis à réaliser notre projet.

Pour concevoir un système intelligent de reconnaissance de mots arabes performant, nous avons implémenté et évalué plusieurs configurations. Les différentes expérimentations réalisées au cours de ce travail ont permis de définir une architecture optimale du réseau CNN. Le système de reconnaissance intelligent que nous avons conçu autour de cette structure neuronale a permis d'obtenir les meilleures performances en termes de précision et de temps d'exécution. En effet ce système intelligent reconnaît automatiquement plus de 96 mots sur 100 mots en arabe.

Quant aux perspectives futures que nous envisageons pour compléter ce travail :

- adapter la base de données pour introduire tous les mots arabes qui permettent d'écrire des montants en dinars sur les chèques de banques. Ceci permettra à notre système d'être appliqué sur un cas concret.
- Concevoir une architecture matérielle embarquée puissante (FPGA par exemple) pour implémenter notre système.

Bibliographie :

[1] TOUZET, Claude. *les réseaux de neurones artificiels, introduction au connexionnisme*. 1992.

[2] Efiaut MUXIKA OLASAGASTI, " Application des Réseaux de Neurones a l'Identification d'un Axe de Machine-Outil", Thèse de doctorat, INPG, 2002

[3] TRECAT, J. APPLICATION DE RESEAUX DE NEURONES ARTIFICIELS A LA RECONNAISSANCE AUTOMATIQUE DE CARACTERES MANUSCRITS. Thèse de doctorat. Faculté Polytechnique de Mons.

[4] DREYFUS, Gérard. *Neural networks: methodology and applications*. Springer Science & Business Media, 2005.

[6] PETITJEAN, Gérald. Introduction aux réseaux de neurones. *gerald. petitjean@ eurodecision. com*, 2007.

[7] S.ZEGHLACHE, Commande Intelligente, cours, université Mohamed boudiaf-m'sila.2008

[8] BABA HAMED, Amel. *Vers Un Modèle De Classification Neuronale Des Données Médicales A Base De La Technologie FPGA*. Thèse de doctorat.

[9] BUYSSENS, Pierre. *Fusion de différents modes de capture pour la reconnaissance du visage appliquée aux e_transactions*. 2011. Thèse de doctorat.

[10] Cyrille FAUCHEUX, Alexandre LISSY, Rapport de projet, Réseaux de neurones sous Matlab, École Polytechnique de l'Université de Tours.

[11] Parizeau, Marc. "Réseaux de neurones." *GIF-21140 et GIF-64326* 124 (2004).

[12] A. Krizhevsky, I. Sutskever et G. E. Hinton, « ImageNet Classification with Deep Convolutional Neural Networks », Advances in neural Processing Systems de traitement, vol. 1, 2012, p. 1097–1105 lire en ligne archive du 16 février 2015

[14] Kadous Djamila, Utilisation des réseaux de neurones comme outil du datamining Génération de modèle comportemental d'un processus physique à partir de données, mémoire de master, université d'ABOU BAKR BELKAID de TELEMCEN, 2012

[15] M. Boughaba and B. Boukhris, "L`apprentissage profond (Deep Learning) pour la

classification et la recherche d'images par le conte, mémoire de master, université de KASDI MERBAH OUARGLA, 2017

[17] A. Krizhevsky, I. Sutskever et G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks », Advances in neural Processing Systems de traitement. 2012.

[18] Graham, Benjamin. Fractional Max-Pooling. 18 décembre 2014.

[19] Russakovsky, O., Deng, J., Su, H., et al. "ImageNet Large Scale Visual Recognition Challenge." International Journal of Computer Vision (IJCV). Vol 115, Issue 3, 2015, pp. 211–252.

[20] D. Learning, "Deep Learning pour Reconnaissance du Visage Deep Learning pour Reconnaissance du Visage," 2018.

[21] H. Fethallah, M. Mohammed, B. Akkacha, and S. M. Ismail, "Classification des images par les réseaux de neurones," 2017.

[22] LeCun, Yann. L'apprentissage prédictif est le grand défi scientifique de l'intelligence artificielle,2018.

Webographie :

[5]https://www.editionseni.fr/open/mediabook.aspx?idR=2dac0ec804e7e66899c47f486d228ff0

[13] https://optinum.co.za/whats-new-deep-learning

[16] http://neuralnetworksanddeeplearning.com/chap1.html#sigmoid_neurons

[23]https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun-cnn?fbclid=IwAR0epfLz_RE8iX20MJBXcZrdnj8hVTq1ET5zEas74DKIXRNzRpEIQElawqo

[25]https://jeanfistonmikwa.skyrock.com/3030412618-Python-est-un-langage-de-programmation.html

[26] https://www.spyder-ide.org/

[27] https://www.tensorflow.org/

[28] https://keras.io/

[29] https://scikit-learn.org/

[30]https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6?fbclid=IwAR0McIi_gOps6EjMJq4lX-s6PXG7myJq0I4hn72sHTXt9ouZoy0RrjV6JYE