REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté *des Sciences et de la technologie*

Département d'Electronique

Mémoire

Présenté pour obtenir

LE DIPLOME DE MASTER

FILIERE : **Electronique**

Spécialité : Industrie Electronique (MCIL)

Par

- **Meridji Ameur**
- **Ketfi Akram**

Intitulé

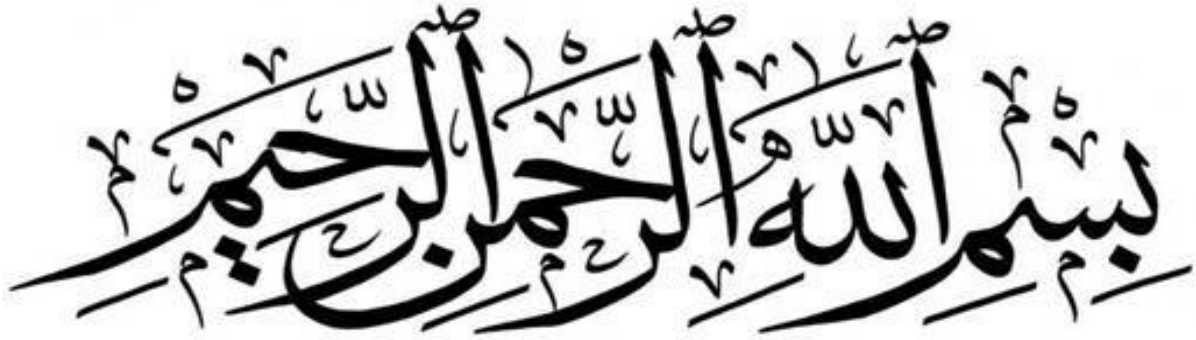
Classification des images utilisant les réseaux de neurones de convolution

Evalué le :

Par la commission d'évaluation composée de* :

Nom & Prénom	*Grade*	*Qualité*	*Etablissement*
M. Bekkouche Toufik	*MCB*	*Président*	*Univ-BBA*
M. Boudechiche Djamel Eddine	*MCB*	*Encadreur*	*Univ-BBA*
M. Sief Eddine Azoug	*MCB*	*Examinateur*	*Univ-BBA*

Année Universitaire 2020/2021


بسم الله الرحمن الرحيم

REMERCIEMENTS

Nous rendons nos profondes gratitudes tout d'abord à Dieu le tout puissant, qui nous a aidés et nous a donné la force et la patience pour accomplir ce modeste travail.

En préambule à ce mémoire, nous tenons à remercier vivement notre encadrant ***Dr. Boudechiche Djamel Eddine,*** *de nous avoir pris en charge et bien encadré, fait profiter de son expérience scientifique durant ce travail. On le remercie sincèrement pour son aide précieuse, son encouragement, ses judicieux conseils, sa patience et sa disponibilité tout au long de notre réalisation de ce mémoire.*

Nous désirons aussi remercier les personnes qui ont contribué à l'élaboration et l'avancement de notre projet fin d'étude ainsi que l'équipe pédagogique de l'université de Bordj-Bou-Arreridj plus précisément département d'Électronique.

Nous adressons tous nos remerciements les plus chers au président du jury et l'examinateur pour l'honneur qu'ils nous font en jugeant ce travail.

Enfin, nous adressons nos plus sincères gratitudes à tous nos proches et amis, pour leurs soutiens et leurs encouragements tout au long de nos études.

Tous les mots restent faibles pour exprimer notre profonde reconnaissance à nos parents pour le soutien et l'aide précieuse qu'ils nous ont apportés durant nos longues années d'études.

Résumé

Dans ces dernières années, l'apprentissage en profondeur connu une grande évolution et sa lui permet d'appliqué dans plusieurs domaines comme la classification d'images. La classification d'images a connu une avance majeure en termes de performance, grâce à l'essor des réseaux de neurones à convolution (CNN). Dans ce mémoire, nous avons proposées un modèle de CNN simple et efficace pour classifie des images de *rugby* et *soccer* en deux classes. Les résultats de classification obtenus avec le modèle proposé, sont comparables à ceux du modèle *ResNet-50*, LeNet-5 et Inception-v3 avec une justesse de validation plus de 80 %. Les résultats obtenus sont acceptables dans le terme de précision et dans terme de l'erreur de prédiction.

ملخص

في السنوات الأخيرة، شهد التعلم العميق تطورا كبيرا ومكن من تطبيقه في العديد من المجالات مثل تصنيف الصور. شهد تصنيف الصور تقدما كبيرا في الأداء، وذلك بفضل ظهور الشبكات العصبية التلافيفية (CNN). في هذا الموجز، اقترحنا نموذجا بسيطا وفعالا لشبكة CNN لتصنيف صور الرقبي وكرة القدم إلى فئتين. نتائج التصنيف التي تم الحصول عليها باستخدام النموذج المقترح قابلة للمقارنة مع تلك الخاصة بنموذج ResNet-50 وLeNet-5 وInception-v3 بدقة تحقق تزيد عن 80٪. النتائج التي تم الحصول عليها مقبولة من حيث الدقة ومن حيث خطأ التنبؤ.

.

Abstract

In recent years, deep learning has undergone a great evolution and has enabled it to be applied in several fields such as image classification. Image classification has seen a major advance in performance, thanks to the rise of convolutional neural networks (CNN). In this brief, we have proposed a simple and effective CNN model for classifying rugby and soccer images into two classes. The classification results obtained with the proposed model are comparable to those of the ResNet-50, LeNet-5 and Inception-v3 model with a validation accuracy of more than 80%. The results obtained are acceptable in terms of precision and in terms of the prediction error.

Table des matières

Liste des figures

Liste d’abréviation

• **ANN**: Artificiel Neural Network

• **ACP :** Analyse en Composantes Principales

• **CNN**: Convolution Neural Network (réseaux de neurones de convolution)

• **CONV :** couche de convolution

• **DRF**: Django REST framework

• **FC**: Full Connexion

• **GPU** : Graphical Processing Unit

• **LOSS :** couche de perte

• **MLP**: Multi-Layer Perceptron

• **MB-SGD** : *Mini Batch - Descente de gradient stochastique*

• **ONEIROS:** Open-ended Neuro-Electronic Intelligent Robot Operating System

• **Pixels: PIC**ture **El**ement

• **RNN**: Recurrent Neural Network

• **RMSProp*:*** Root Mean Squared Prop

• **ReLU :** Unité de rectification linéaire

• **ResNet :** Residual Neural Network

• **RVB**: Rouge Vert Bleu

• **SGD** : Stochastique Gradient Descente

• **Tanh** : Tangente Hyperbolique

Introduction générale :

Nous vivons dans un monde numérique, où les informations sont stockées, traitées, indexées et recherchées par des systèmes informatiques, ce qui rend leur récupération une tâche rapide et pas cher. Au cours des dernières années, des progrès considérables ont été réalisés dans le domaine de classification d'images. Ce progrès est dû aux nombreux travaux dans ce domaine et à la disponibilité des bases d'images internationales qui ont permis aux chercheurs de signaler de manière crédible l'exécution de leurs approches dans ce domaine, avec la possibilité de les comparer à d'autres approches qu'ils utilisent les mêmes bases.

Google et IBM avec l'aide du laboratoire de Geoff Hinton **[1]** ont montré que les réseaux profonds pouvaient diminuer de moitié les taux d'erreurs des systèmes de reconnaissance vocale et d'image. Et avec le développement de la technologie, le GPU (*Graphical Processing Unit*) qui devenir capables de réaliser plus de mille milliards d'opérations par seconde, ils peuvent, maintenant, de faire des calculs sur des réseaux neuronaux en temps réel.

Plusieurs records en reconnaissance d'image ont été battus par des CNN. L'événement le plus marquant a été la victoire éclatante de l'équipe de Toronto **[1]** dans la compétition de reconnaissance d'objets « ImageNet ».

Du jour au lendemain, la majorité des équipes de recherche en parole et en vision ont abandonné leurs méthodes préférées et sont passés aux CNN et autres réseaux neuronaux. L'industrie d'internet a immédiatement saisi l'opportunité et a commencé à investir massivement dans des équipes de recherche et développements en apprentissage profond (Deep Learning) **[1]**.

Dans notre mémoire, on va utiliser les CNN pour classifier des images. On va créer notre modèle de classification et le compare avec d'autres modèles de la classification qui existe déjà.

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

- Dans le premier chapitre, on va présenter les réseaux de neurones et l'apprentissage profond et leurs paramètres.
- Le deuxième chapitre est consacré à la description des CNN ainsi que leurs l'intérêt dans le domaine de la classification des images.
- Dans le troisième chapitre, on va montrer la partie expérimentale de notre travail et l'on discute les différents résultats obtenus.
- À la fin, on termine par une conclusion générale.

Chapitre I:

Les réseaux de neurones

I.1 Introduction :

Un réseau de neurones artificiels est une technique de calcul leur fonctionnement inspiré par des neurones biologiques et opèrent mécaniquement à partir de règles précises. Le réseau neuronal est construit avec des connexions de nœuds, qui sont des éléments qui correspondent aux neurones du cerveau.

Dans ce chapitre, on a introduit quelques notions fondamentales sur l'apprentissage profond qui devient dans notre jour l'un des domaines les plus vaste et les plus récents de la science et de l'ingénierie, comme la reconnaissance visuelle et sonore, d'espèces et aussi bien au traitement d'images.

I.2 Les réseaux de neurones :

I.2.1 Définition :

Les réseaux de neurones sont des constructions abstraites simulant l'activité d'un réseau de neurones biologique simplifiés. Ils sont utilisés dans apprentissage profond pour construire un modèle à partir de données existantes dans le but d'effectuer des prédictions sur de nouvelles données soit à l'aide de régression, dans le cas continu, ou de classification, dans le cas discret.

I.2.2 Les principales composantes du réseau de neurones :

Le réseau de neurones est composé des composants principaux suivants:

- **Le perceptron :**

Le perceptron peut être vu comme le type de réseau de neurones le plus simple (Figure I.1). C'est un classifier linéaire. Dans sa version simplifiée, le perceptron est monocouche et n'a qu'une seule sortie (booléenne) à laquelle toutes les entrées (booléennes) sont connectées. Plus généralement, les entrées peuvent être des nombres réels **[2]**.

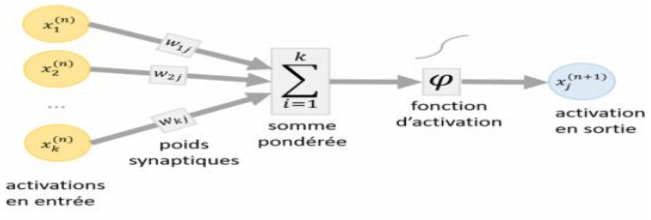


Figure I.1 : modelé général de perceptron.

- Un neurone possède des entrées.
- Chaque entrée possède un poids.
- La sortie est une fonction du poids et des entrées :

$$\mathbf{Y} = \mathbf{f}(\boldsymbol{W_1} * \boldsymbol{X_1} + \boldsymbol{W_2} * \boldsymbol{X_2}) \quad \text{(I.1)}$$

a) Fonctions d'activation (ou fonction de transfert) :

- **Unité de rectification linéaire (Relu) :**

$$f(x)= \begin{cases} 0 & si \quad x < 0 \\ x & si \quad x \geq 0 \end{cases} \quad \text{(I.2)}$$

- **Linéaire Sigmoïde:**

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{(I.3)}$$

- **tanh(x) :**

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad \text{(I.4)}$$

b) Calcul de l'erreur :

Dans les réseaux de neurones artificiels, l'erreur que l'on désire minimiser est donnée par :

$$E_k = (Y_k - S_k) \quad \text{(I.5)}$$

Cette erreur peut être positive ($\mathbf{Y_k} > \mathbf{S_k}$) ou négative ($\mathbf{Y_k} < \mathbf{S_k}$).

- $\mathbf{E_K}$ est l'erreur de sortie.
- $\mathbf{Y_K}$ est la sortie désirée.
- $\mathbf{S_K}$ est la sortie réelle du réseau.

c) Fonction de coût :

Pour réduire l'erreur, nous utilisons la fonction de coût (*cost fonction*). Autrement dit, nous allons améliorer $\boldsymbol{S_k}$, pour réaliser cette amélioration, nous modifions les valeurs des poids :

$$J(W_1,W_2) = \frac{1}{M}\sum_{i=1}^{M}(Y_K - S_K)^2 \quad \text{(I.6)}$$

Le but ici est de réduire cette fonction, et pour l'obtenir, nous utilisons un algorithme de descente du gradient.

d) Mise à jour des poids :

Descente de gradient signifie que l'on cherche à réduire l'erreur dans la direction de l'erreur, en descendant le long du gradient. Si l'on considère les entrées xi du réseau associé respectivement aux poids *Wi*, alors :

$$W_{i+1} \leftarrow W_i - \alpha\frac{\mathrm{dJ}}{\mathrm{dW}} \tag{I.7}$$

- **α** est le taux d'apprentissage.
- **W** : le poids.
- $\frac{\mathbf{dJ}}{\mathbf{dW}}$: Descente de gradient.

Les poids peuvent être mis à jour avec la formule suivante :

$$W_{i+1} = W_i - \alpha\frac{\mathrm{dJ}}{\mathrm{dW}} = W_i + \alpha(Y_K - S_K)X_i \tag{I.8}$$

$$W_{i+1} = W_i + \alpha(Y_K - S_K)X_i \tag{I.9}$$

- **Le perceptron multicouche :**

Dans le modèle du perceptron multicouche (Figure I.2), les perceptrons sont organisés en couches. Les perceptrons multicouches 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. Les MLP (*Multi-Layer Perceptron*) sont généralement organisés en trois couches, la couche d'entrée, la couche intermédiaire (dite couche cachée) et la couche de sortie. L'utilité de plusieurs couches cachées n'a pas été démontrée. Illustre la structure d'un MLP présentant quatre neurones en entrée, trois neurones sur la couche cachée et deux en sortie. Lorsque tous les neurones d'une couche sont connectés aux neurones de la couche suivante, on parle alors de couches complètement connectées **[3].**

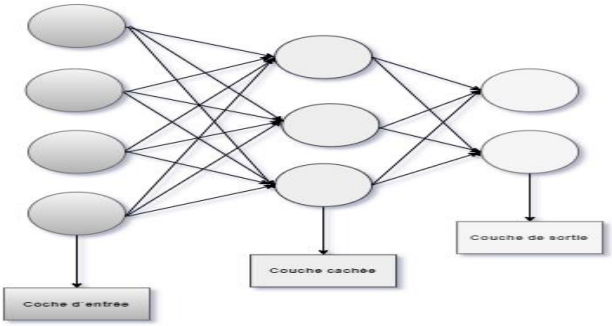


Figure I.2 : schéma d'un perceptron multicouche.

- **La couche : groupement des neurones**

Les couches (ou *layer*) contiennent des neurones et aident à faire circuler l'information. Il existe au moins deux couches dans un réseau de neurones : la couche d'entrée (*input layer*) et la couche de sortie (*output layer*). Les couches existent entre les couches d'entrée et de sortie, sont appelées *les couches cachées (ou hidden layer).*

- **Le poids et biais : valeurs numériques**

Les poids et biais sont des variables du modèle qui sont mises à jour pour améliorer la précision du réseau. Un poids est appliqué à l'entrée de chacun des neurones pour calculer une donnée de sortie.

Les réseaux de neurones mettent à jour ces poids de manière continue. Il existe donc une boucle de rétroaction mise en œuvre dans la plupart des réseaux de neurones.

Les biais sont également des valeurs numériques qui sont ajoutées une fois que les poids sont appliqués aux valeurs d'entrée. Les poids et les biais sont donc en quelque sorte des valeurs d'autoapprentissage de nos réseaux de neurones **[4].**

I.2.3 Fonctionnement des réseaux de neurones :

Le concept de réseau de neurones repose sur trois étapes principales :

1. Pour chaque neurone dans une couche, multiplier la valeur d'entrée par le poids.
2. Ensuite, pour chaque couche, additionner toutes les pondérations des neurones et ajouter un biais.
3. Enfin, appliquer la fonction d'activation sur cette valeur pour calculer une nouvelle sortie

I.2.4 les différents types de réseaux de neurones :

Il existe différents types de réseaux de neurones. Les deux réseaux de neurones les plus populaires sont :

a. Réseau de neurones récurrent (*Récurrent Neural Network* (RNN)) :

Ce sont des réseaux de neurones spécialisés qui utilisent le contexte des entrées lors du calcul de la sortie. La sortie dépend des entrées et des sorties calculées précédemment. Ainsi, les RNN conviennent aux applications où les informations historiques sont importantes. Ces réseaux nous aident à prévoir les séries chronologiques dans les applications commerciales et à prévoir les mots dans les applications de type chabot. Ils peuvent fonctionner avec différentes longueurs d'entrée et de sortie et nécessitent une grande quantité de données.

b. Réseau de neurones à convolution (*Convolution Neural Network* (CNN)) :

Ces réseaux reposent sur des filtres de convolution (matrices numériques). Les filtres sont appliqués aux entrées avant que celles-ci ne soient transmises aux neurones. Ces réseaux de neurones sont utilisés pour le traitement et la prévision d'images **[4].**

I.3 L`apprentissage en profondeur (*deep learning*) :

I.3.1 Définition :

C'est un nouveau domaine de recherche qui traite de la recherche de théories et d'algorithmes permettant à une machine d'apprendre par elle-même en simulant des neurones dans le corps humain. Et l'une des branches de la science qui traite des sciences de l'intelligence artificielle. Il s'agit d'une branche de la science de l'apprentissage automatique. La plupart des recherches approfondies sur l'apprentissage se concentrent sur la recherche de méthodes permettant de générer un degré élevé d'abstractions en analysant un vaste ensemble de données à l'aide de variables linéaires et non linéaires.

Les réseaux de neurones profonds partent du principe qu'il est plus efficace de multiplier le nombre de couches plutôt que le nombre de neurones. Ainsi, plusieurs réseaux de neurones profonds ont été développés récemment en empilant un nombre généralement important de couches de neurones aux propriétés spécifiques.

I.3.2 le principe de fonctionnement :

Comme à l'intérieur du cerveau humain, les signaux voyagent entre les neurones du cerveau artificiel. Le secret de cette prouesse repose en grande partie sur les algorithmes. Dans le cas de la reconnaissance visuelle, pour être performant, l'algorithme d'apprentissage profond doit être capable d'identifier toutes les formes existantes et dans tous les angles.

Ainsi, il sera capable de détecter une voiture sur la route au milieu du paysage. Ceci n'est possible que si la machine a suivi un entraînement poussé. Et ceci passe par la visualisation de milliers de photographies sur lesquelles apparaît une voiture, de toutes les formes et dans tous les angles possibles.

Lorsque l'image nouvelle apparaît, elle est envoyée au réseau de neurones qui se charge de les analyser et de déterminer si l'objet au milieu du cliché est bel et bien une voiture. La machine a gagné son pari ? Elle garde sa bonne réponse au chaud, car elle l'aidera à résoudre d'autres situations similaires le jour où elle devra reconnaître une autre voiture **[5].**

I.3.3 les défirent types d`apprentissage :

a) Apprentissage supervisé :

Nous voulons dire que nous sommes ceux qui supervisent les instructions de la machine en lui donnant toutes les entrées et les sorties (Figure I.3).

- Les données fournies à la machine sont appelées entraînement (*training*), et nous voulons dire que la machine s'entraîne sur ces données.
- Et quand nous disons superviser, nous devons être conscients des nouvelles données (*Tasting*), et nous voulons dire qu'après que la machine a appris ces données, nous lui donnons dans la phase *Tasting* de nouvelles données pour faire la prédiction.

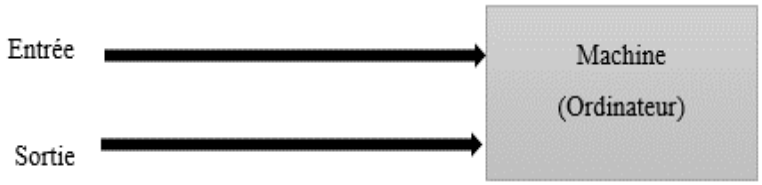


Figure I.3 : apprentissage supervisé.

Il y a deux types plus courants d'applications sont la classification et la régression.

La seule différence est le type de sorties correctes : la classification emploie des classes, tandis que la régression exige des valeurs. La régression peut devenir une classification lorsqu'elle a besoin d'un modèle pour juger de quel groupe les données d'entrée appartiennent et de la régression lorsque le modèle estime la tendance des données.

Dans la classification (Figure I.4**)**, le but est de trouver le lien entre les données d'apprentissage et les nouvelles entrées pour prévoir les classes auxquelles appartiennent les nouvelles entrées parmi un ensemble de classes prédéfinies qu'est prévu de l'apprentissage supervisé.

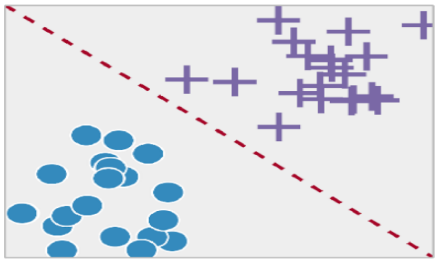


Figure I.4 : la méthode de classification.

Dans la régression (Figure I.5), le but est de faire la prédiction des résultats continus. Dans l'analyse de régression où les valeurs à prédire sont numériques, il faut donner un certain nombre de variables et de réponse prédictive (résultat), et nous essayons de trouver une relation entre ces variables qui nous aide à comprendre comment la valeur d'une variable dépendante change en fonction d'une variable indépendante. Donc, la tache de l'algorithme de régression est de trouver la fonction qui permet de mapper la variable d'entrée (X) à la variable de sortie continue (Y) **[6].** Par exemple :

• Prédire l'âge d'une personne.

• Prédire la nationalité d'une personne.

• Prédire si le cours des actions d'une entreprise augmentera demain.

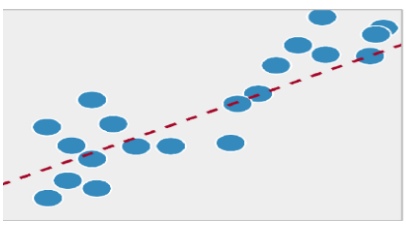


Figure I.5 : la méthode de régression.

b) Apprentissage non supervisé :

Dans cette méthode (Figure I.6) les données ne sont pas étiquetées, le réseau de neurones analyse l`ensemble de données, et une fonction-coût lui indique dans quelle mesure il est éloigné du résultat souhaité. Le réseau s`adapte alors pour augmenter la précision de l`algorithme.

c) Apprentissage renforcé :

Le réseau neurones est renforcé pour les résultats positifs et sanctionné pour les résultats négatifs (Figure I.7). C`est ce qui lui permet d`apprendre au fil du temps, de la même manière qu`un humain apprend progressivement de ses erreurs **[7].**

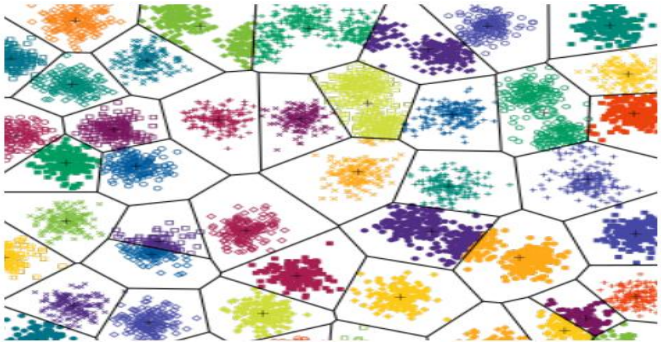


Figure I.6 : exemple sur l`apprentissage non supervisé.

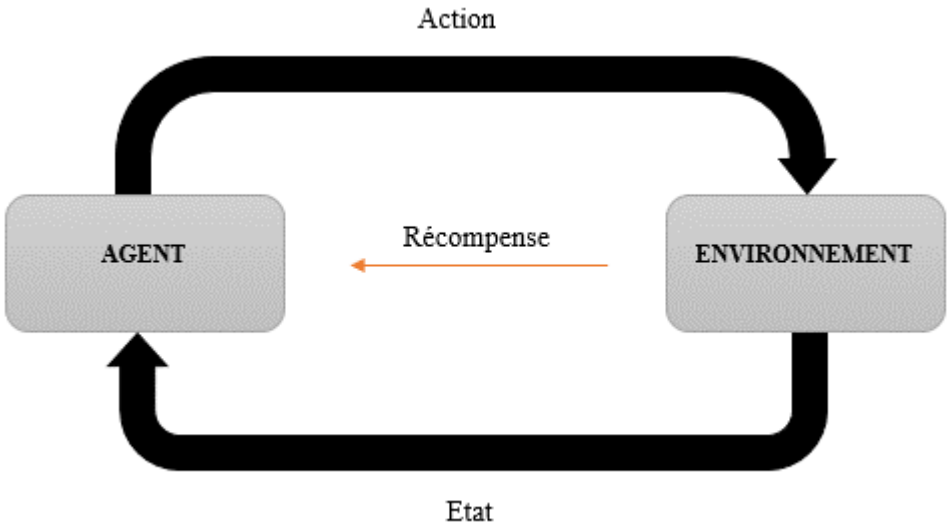


Figure I.7 : exemple sur l`apprentissage renforcé.

I.4 Les paramètres de l'apprentissage profond (*deep learning*) :

I.4.1 la régularisation :

La régularisation fait référence à un ensemble de techniques différentes qui réduisent la complexité d'un modèle de réseau de neurones pendant l'entraînement et empêchent ainsi le sur ajustement **[8].**

Il existe deux techniques de régularisation très populaires et efficaces appelées *L2* et *abandon* :

A. Régularisation L2:

La régularisation L2 est le type le plus courant de toutes les techniques de régularisation et est également connue sous le nom de perte de poids ou de régression de conduite.

La dérivation mathématique de cette régularisation, ainsi que l'explication mathématique des raisons pour lesquelles cette méthode fonctionne pour réduire le sur ajustement, est assez longue et complexe. Puisqu'il s'agit d'un article très pratique, je ne veux pas me concentrer sur les mathématiques plus que nécessaires. Au lieu de cela, je souhaite transmettre l'intuition derrière cette technique et surtout comment la mettre en œuvre afin que vous puissiez résoudre le problème de sur apprentissage.

Lors de la régularisation L2, la fonction de perte du réseau de neurones est étendue par un terme dit de régularisation, appelé ici Ω.

$$\Omega\,(\mathrm{W}) = \|W\|_2^2 = \sum_i \sum_j {W_{ij}}^2 \tag{1.10}$$

Le terme de régularisation **Ω** est défini comme la norme euclidienne (ou norme **L2**) des matrices de poids, qui est la somme de toutes les valeurs de poids au carré d'une matrice de poids. Le terme de régularisation est pondéré par l'alpha scalaire divisé par deux et ajouté à la fonction de perte régulière choisie pour la tâche en cours. Cela conduit à une nouvelle expression pour la fonction de perte:

$$\hat{\mathrm{L}}\,(\mathrm{W}) = \frac{\alpha}{2}\|\mathrm{W}\|_2^2 + \mathrm{L}(\mathrm{W}) = \frac{\alpha}{2}\sum_{\mathrm{i}}\sum_{\mathrm{j}} {W_{ij}}^2 \tag{I.11}$$

Alpha est appelé le taux de régularisation **[6]**.

B. Abandon (*dropout*) :

En plus de la régularisation L2, il y a une autre technique de régularisation célèbre et puissante est appelée la régularisation d'abandon. La procédure de régularisation des abandons est assez simple.

En un mot, l'abandon signifie que pendant l'entraînement on abandonne un certain nombre des neurones. Regardons un exemple visuel dans la figure I.8.

Supposons que sur le côté gauche, nous ayons un réseau de neurones à réaction sans interruption. Utiliser le décrochage avec une probabilité de $\boldsymbol{P = 0,5}$ qu'un neurone aléatoire s'éteigne pendant l'entraînement entraînerait un réseau de neurones sur le côté droit.

Dans ce cas, vous pouvez observer qu'environ la moitié des neurones ne sont pas actifs et ne sont pas considérés comme faisant partie du réseau neuronal. Et comme vous pouvez l'observer, le réseau neuronal devient plus simple.

Une version plus simple du réseau de neurones entraîne moins de complexité qui peut réduire le sur ajustement. La désactivation des neurones avec une certaine probabilité $\boldsymbol{P}$ est appliquée à chaque étape de propagation directe et de mise à jour de poids **[8].**

I.4.2 techniques d'optimisation :

A. *Mini Batch* - Descente de gradient stochastique :

MB-SGD est une extension de l'algorithme SGD (Stochastique Gradient Descente : utilisée pour la minimisation d'une fonction objectif qui est écrite comme une somme de fonctions différentiables) (Figure I.9). Il surmonte la complexité chronophage de SGD en prenant un lot de points sous-ensemble de points de l'ensemble de données pour calculer le dérivé. Ces derniers temps, les algorithmes d'optimisation adaptative gagnent en popularité en raison de leur capacité à converger rapidement. Ces algorithmes utilisent des statistiques d'itérations précédentes pour accélérer le processus de convergence **[9].**

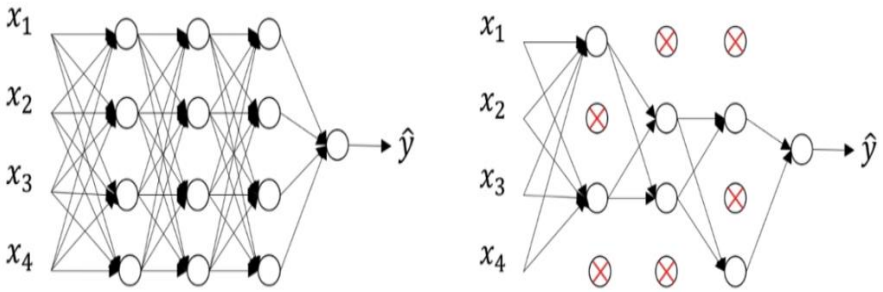


Figure I.8 : Réseau de neurones avec dropout (à droite) et sans dropout (à gauche).

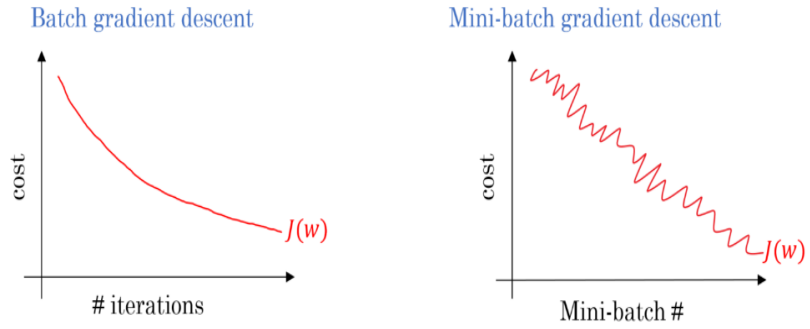


Figure I.9 : le gradient descente et le mini batch de gradient descente.

B. *Optimiseur basé sur Momentum:*

Il s'agit d'un algorithme d'optimisation adaptative qui utilise de manière exponentielle des gradients moyens pondérés sur les itérations précédentes pour stabiliser la convergence, ce qui se traduit par une optimisation plus rapide. Cela se fait en ajoutant une fraction (gamma) aux valeurs de l'itération précédente. Essentiellement, le terme d'élan augmente lorsque les points de gradient sont dans les mêmes directions et diminue lorsque les gradients fluctuent. En conséquence, la valeur de la fonction de perte converge plus rapidement que prévu.

C. *RMSProp :*

(*Root Mean Squared Prop****)*** est une autre méthode de taux d'apprentissage adaptatif qui tente d'améliorer AdaGrad. Au lieu de prendre la somme cumulative des gradients carrés comme dans AdaGrad, nous prenons la moyenne mobile exponentielle. La première étape dans AdaGrad et RMSProp est identique. RMSProp divise simplement le taux d'apprentissage par une moyenne en décroissance exponentielle.

D. *Estimation adaptative du moment (Adam) :*

C'est une combinaison de *RMSProp* et *Momentum*. Cette méthode calcule le taux d'apprentissage adaptatif pour chaque paramètre. En plus de stocker la moyenne décroissante précédente des gradients carrés, il contient également la moyenne du gradient passé similaire à *Momentum*. Ainsi, Adam se comporte comme une balle lourde avec frottement qui préfère les minima plats en surface d'erreur **[9].**

E. Les techniques d'augmentation :

Les techniques d'augmentation des données sont utilisées pour générer des échantillons de formation artificiels et étendre l'ensemble de données de formation. Elles ont été largement utilisées pour améliorer la performance du modèle et réduire *overfitting*, en particulier notamment pour l'apprentissage avec un petit ensemble de données ou un ensemble de données déséquilibré. Les techniques d'augmentation d'image couramment utilisées techniques d'augmentation d'image couramment utilisée incluent le recadrage, le retournement, la rotation de l'image **[10].**

I.4.3 Normalisation :

La normalisation d'un vecteur (par exemple, une colonne dans un jeu de données) consiste à diviser les données de la norme vectorielle**.** Typiquement, nous l'utilisons pour obtenir la distance euclidienne du vecteur égale à une certaine valeur prédéterminée **[11].**

I.5 conclusion :

Dans ce chapitre on va parlé sur les réseaux neurones et l`apprentissage profond (*deep learning*) et leurs paramètres on utilise dans beaucoup de domaines parmi ces domaines le traitement d`image.

Les réseaux de neurones reposent à présent sur des bases mathématiques solides qui permettent d`envisager des applications dans presque tous les domaines y compris industriel et à grande échelle, notamment dans le domaine de la classification.

Chapitre II :

Les réseaux de neurones de convolution

II.1 Introduction :

Dans les réseaux de neurones, le réseau de neurones à convolution (CNN) est l'une des principales catégories pour effectuer la reconnaissance d'images.

La détection d'objets, la reconnaissance de visages, les panneaux de signalisation sont quelques-uns des domaines où les CNN sont largement utilisées. Au cœur de la CNN se trouve la couche de convolution qui donne son nom au réseau.

Cette couche effectue une opération appelée convolution qui signifie mathématiquement l'intégrale mesurant combien deux fonctions se chevauchent lorsque l'une passe sur l'autre, la multiplication dans le réseau neuronal est effectuée entre un tableau de données d'entrée et un tableau bidimensionnel de poids, appelé filtre ou noyau.

Donc ces réseaux basent sur des filtres de convolution qu'est une petite matrice de taille variable appelée « *kernel* ». Les filtres sont adaptés aux entrées avant que celles-ci ne soient envoyées aux neurones. Les ordinateurs voient une image d'entrée comme un tableau de pixels et cela dépend de la résolution de l'image.

Avec la révolution scientifique d'Internet et l'expansion des applications multimédias, les exigences des nouvelles technologies pour la compression d'images se sont accrues.

Récemment, de nombreuses technologies différentes ont été d´enveloppées pour répondre à ces exigences.

II.2 les images numériques :

II.2.1 définition :

Les images numériques sont constituées d'un ensemble de pixels (*PICture éléments*), juxtaposés en lignes et en colonnes. Le pixel (qui correspond à un point ou petit carré) est le plus petit élément que l'on peut trouver dans une image. Chaque pixel possède des caractéristiques propres, couleurs, luminosité, brillance, qui permettent de les différencier et de composer les images **[12].**

II.2.2 les déférents types de format d`image numérique :

- **Image couleur RVB :** L'œil humain analyse la couleur à l'aide de trois types de cellules photo « les cônes ». Ces cellules sont sensibles aux basses, moyennes, ou hautes fréquences (rouge, vert, bleu). Pour représenter la couleur d'un pixel, il faut donc donner trois nombres, qui correspondent au dosage de trois couleurs de base : Rouge, vert, bleu. On peut ainsi représenter une image couleur par trois matrices chacune correspondant à une couleur de base.
- **Image d'intensités :** C'est une matrice dans laquelle chaque élément est un réel compris entre 0 (noir) et 1 (blanc). On parle aussi d'image en niveaux de gris, car les valeurs comprises entre 0 et 1 représentent les différents niveaux de gris.
- **Image binaire :** Une image binaire est une matrice rectangulaire dans l'élément valent 0 ou 1. Lorsque l'on visualise une telle image, les 0 sont représentés par du noir et les 1 par du blanc. **[13].**

II.2.3 les caractéristiques d`images numériques :

L'image est un ensemble structuré d'information caractérisé par les paramètres suivants :

a. Les pixels :

Une image numérique est constituée d'un ensemble de points appelés **pixels** (abréviation de ***PIC****ture* ***El****ement*) pour former une image (Figure II.1).

Le pixel représente ainsi le plus petit élément constitutif d'une image numérique. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant.

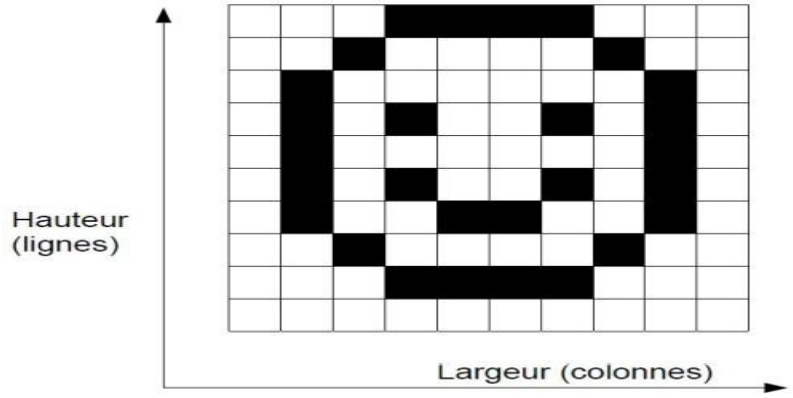


Figure II.1 : les dimensions d`images.

b. Dimension et Résolution :

La dimension est la taille de l'image. Elle se présente sous forme d'une matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image. Par contre, la résolution est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateur.

La résolution est exprimée en nombre de pixels par unité de mesure (pouce ou centimètre). On utilise aussi le mot résolution pour désigner le nombre total de pixels horizontaux et verticaux sur un moniteur. Plus ce nombre est grand, plus la résolution est meilleure.

c. Niveau de gris :

C'est la valeur d'intensité lumineuse d'un pixel. Cette valeur peut aller du noir (0) jusqu'au blanc (255) en passant par les nuances qui sont contenues dans l'intervalle [0, 255]. Elle correspond en fait à la quantité de la lumière réfléchie. Pour 8 bits, on dispose de 256 niveaux de gris dont 40 sont reconnus à l'œil nu. Plus le nombre de bits est grand plus les niveaux sont nombreux et plus la représentation est fidèle **[14].**

II.2.4 Méthodes de classification :

De nombreuses méthodes classiques ont été consacrées, elles peuvent être séparées en deux grandes catégories : les méthodes de classification supervisée et les méthodes de classification non supervisée.

a. Méthodes supervisées : L'objectif de la classification supervisée est principalement de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets. On dispose au départ d'un échantillon dit d'apprentissage dont le classement est connu. Cet échantillon est utilisé pour l'apprentissage des règles de classement. Il est nécessaire d'étudier la fiabilité de ces règles pour les comparer et les appliquer, évaluer les cas de sous apprentissage ou de sur apprentissage (complexité du modèle). On utilise souvent un deuxième échantillon indépendant, dit de validation ou de test.

b. Méthodes non supervisées : Procède de la façon contraire. C'est à dire ne nécessitent aucun apprentissage et aucune tâche préalable d'étiquetage manuel. Elle consiste à représenter un nuage des points d'un espace quelconque en un ensemble de groupes appelé Cluster. Il lié généralement au domaine de l'analyse des données comme l'ACP (Analyse en Composantes Principales). Un «Cluster» est une collection d'objets qui sont «similaires» entre eux et qui sont «dissemblables » par rapport aux objets appartenant à d'autres groupes **[14].**

II.2.5 Histogramme :

L'histogramme d'une image est une fonction discrète. Elle représente le nombre de pixels en fonction du niveau de gris (Figure II.2).

Lorsque cette fonction est normalisée entre 0 et 1 pour tous les niveaux de gris, on peut la voir comme une densité de probabilité qui fournit la probabilité de trouver un certain niveau de gris de l'image.

Ainsi le niveau de gris d'un pixel devient une variable aléatoire dont la valeur dépend du résultat d'une expérience aléatoire sous-jacente **[15].**

II.2.6 les filtre des images numériques :

a. Filtre moyen :

Les filtres moyens, comme leur nom l'indique, calculent la moyenne, éventuellement pondérée, des pixels situés dans le voisinage de chaque pixel. Cette famille de filtres permet de réduire le bruit dans l'image, ce qui rend les zones homogènes plus lisses. Par contre, les contours sont fortement dégradés, et les structures trop fines peuvent devenir moins visibles **[16].**

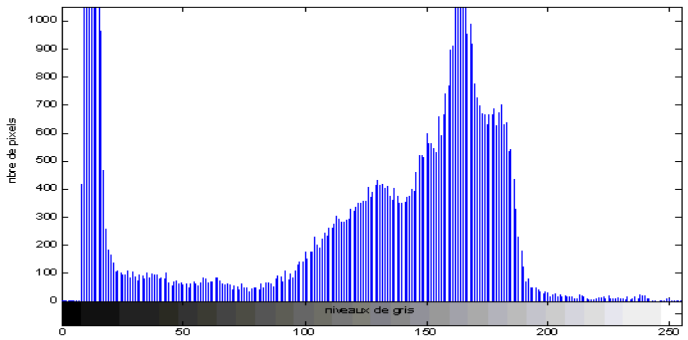


Figure II.2 : histogramme d`une image numérique.

Exemple :

$$H=\begin{bmatrix}\frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9}\end{bmatrix}=\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Pour des filtres de taille **n*n**, la matrice est de la forme :

$$\frac{\mathbf{1}}{\mathbf{n^2}}\times\begin{pmatrix}\mathbf{1} & \cdots & \mathbf{1}\\ \vdots & \ddots & \vdots\\ \mathbf{1} & \cdots & \mathbf{1}\end{pmatrix}$$

b. Filtre médian :

Le filtre médian est un filtre numérique non linéaire, souvent utilisé pour la réduction de bruit. La réduction de bruit est une étape de prétraitement classique visant à améliorer les résultats de traitements futurs (détection de bords par exemple). La technique de filtre médian est largement utilisée en traitement d'images numériques, car il permet sous certaines conditions de réduire le bruit tout en conservant les contours de l'image.

L'idée principale du filtre médian est de remplacer chaque entrée par la valeur médiane de son voisinage.

c. Filtre Sobel :

Le filtre de Sobel est un opérateur utilisé en traitement d'image pour la détection de contours. Il s'agit d'un des opérateurs les plus simples qui donne toutefois des résultats corrects.

L'opérateur utilise des matrices de convolution. La matrice de taille 3×3 subit une convolution avec l'image pour calculer des approximations des dérivées horizontale et verticale. Soit **A** l'image source, $\mathbf{G_x}$ et $\mathbf{G_y}$ deux images qui en chaque point contiennent des approximations respectivement de la dérivée horizontale et verticale de chaque point **[17].**

Les deux types de filtre *Sobel*:

Horizontale : $\mathbf{G_x}=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}*A$ **;** **Vectoriel :** $\mathbf{G_y}=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}*A$

II.3 le réseau de neurones de convolution :

Le réseau de neurones de convolution (CNN) est à ce jour les modèles les plus performants pour classer des images (Figure II.3). En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a 2 dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profond 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu]. La première partie d'un CNN est la partie convolutifs à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN.

Ce code CNN en sortie de la partie convolutifs est ensuite branché en entrée d'une deuxième partie, constituée de couches entièrement connectées (perceptron multicouche). Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer l'image. La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1, pour produire une distribution de probabilité sur les catégories **[14].**

II.3.1 les opérations des réseaux de neurones de convolution :

- **Remplissage *(padding)* :**

Le remplissage est un terme pertinent pour les réseaux de neurones convolutifs, car il fait référence à la quantité de pixels ajoutés à une image lorsqu'elle est traitée par le noyau d'un CNN (Figure II.4). Par exemple, si le remplissage dans un CNN est défini sur zéro, alors chaque

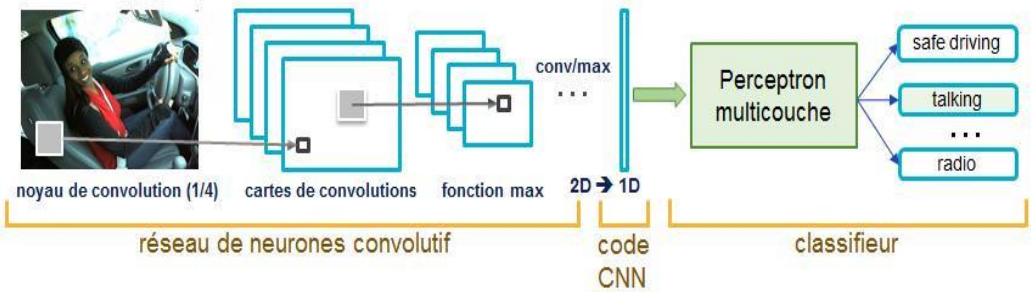


Figure II.3 : Architecture standard d'un CNN.

valeur de pixel ajoutée sera de valeur zéro. Si, cependant, le remplissage de zéro est défini sur un, il y aura une bordure d'un pixel ajoutée à l'image avec une valeur de pixel de zéro.

Le remplissage fonctionne en étendant la zone dont un réseau neuronal de convolution traite une image. Le noyau est le filtre des réseaux neuronaux qui se déplace à travers l'image, balayant chaque pixel et convertissant les données dans un format plus petit, ou parfois plus grand. Afin d'aider le noyau à traiter l'image, un remplissage est ajouté au cadre de l'image pour laisser plus d'espace au noyau pour couvrir l'image. L'ajout de remplissage à une image traitée par un CNN permet une analyse plus précise des images **[18].**

Il existe deux choix courants pour le remplissage :

a. Valide: cela ne signifie pas de remplissage. Si nous utilisons un remplissage valide, la sortie sera :

$$(n\text{-}f + 1) \times (n\text{-}f + 1) \qquad \text{(II.1)}$$

b. *Same*: Ici, nous appliquons un remplissage pour que la taille de sortie soit la même que la taille d'entrée, c'est-à-dire :

$$n + 2p\text{-}f + 1 = n \qquad \text{(II.2)}$$

Donc :

$$p = \frac{(f-1)}{2} \qquad \text{(II.3)}$$

- **Entrée:** n × n.
- **Taille du filtre:** f × f.
- **Sortie:** (n-f + 1) × (n-f + 1).

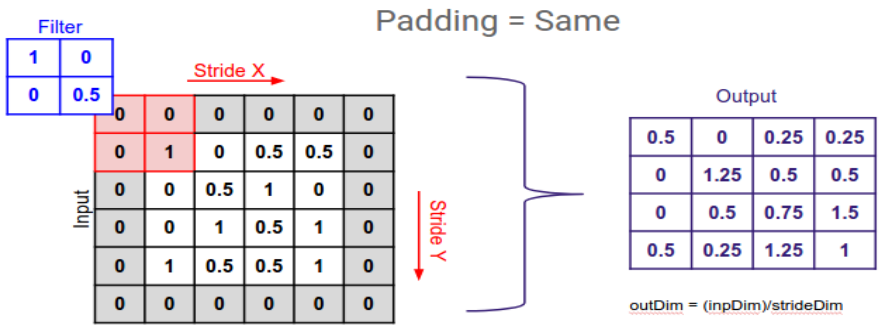


Figure II.4 : le fonctionnement de remplissage (*padding*).

- **Convolutions foulées (*stride*) :**

Stride est un composant de réseaux de neurones de convolution ou de réseaux de neurones réglés pour la compression d'images et de données vidéo (Figure II.5**)**. La foulée est un paramètre du filtre du réseau neuronal qui modifie la quantité de mouvement sur l'image ou la vidéo. Par exemple, si la foulée d'un réseau neuronal est définie sur 1, le filtre déplacera un pixel, ou une unité, à la fois. La taille du filtre affecte le volume de sortie codé, donc la foulée est souvent définie sur un entier, plutôt que sur une fraction ou une décimale **[18].**

II.3.2 les couches des réseaux de neurones de convolution :

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 sous échantillonnage (*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 », qui est une couche de type perceptron.
- La couche de perte (**LOSS**).

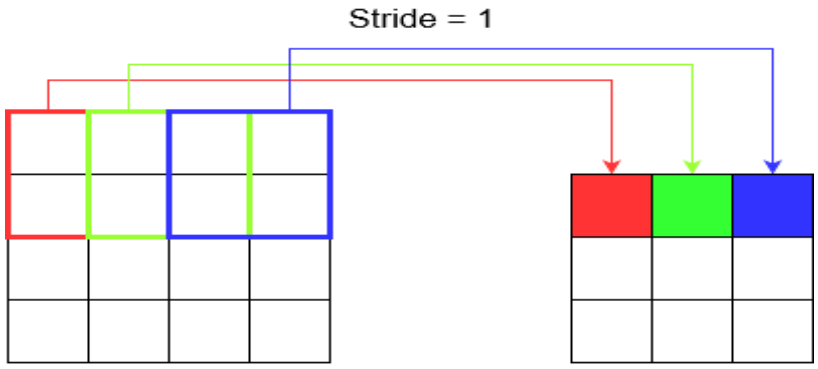


Figure II.5 : le fonctionnement de *Stride.*

- **Couche de convolution (CONV) :**

Une couche de convolution est un empilement de convolutions. En effet, l'image est parcourue par plusieurs noyaux de convolution qui donnent lieu à plusieurs cartes de caractéristiques de sorties. Chaque noyau de convolution possède des paramètres spécifiques à l'information qui est recherchée dans l'image (par exemple : un noyau de convolution de type filtre *Sobel* a des paramètres permettant de rechercher les contours dans l'image).

Le choix des paramètres du noyau de convolution dépend de la tâche à résoudre. Avec les méthodes *deep learning*, ces paramètres sont automatiquement appris par l'algorithme à partir des données d'entraînement.

Notamment, grâce à la technique de rétro propagation du gradient, qui permet l'ajustement des paramètres. En fonction de la valeur du gradient de la fonction de perte. La fonction de perte calcule l'erreur entre la valeur prédite et la valeur cible.

- **Couche de sous échantillonnage (POOL) :**

L'étape de *pooling* est une technique de sous-échantillonnage. Généralement, une couche de sous échantillonnage est insérée régulièrement entre les couches de correction et de convolution (Figure II.6). En réduisant la taille des cartes de caractéristiques, donc le nombre de paramètres du réseau, cela accélère le temps de calcul et diminue le risque de sur apprentissage.

L'opération de sous échantillonnage la plus courante est celle du maximum : *Max Pool* (2*2, 2). Elle est plus efficace que la moyenne, car elle maximise le poids des activations fortes. Elle est appliquée à la sortie de la couche précédente comme un filtre de convolution de taille (2*2) et se déplace avec un pas de 2. En sortie de la couche de *pooling* est obtenue une carte de caractéristique compressée par un facteur de 4 **[19].**

- **La couche de correction (ReLU) :**

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.

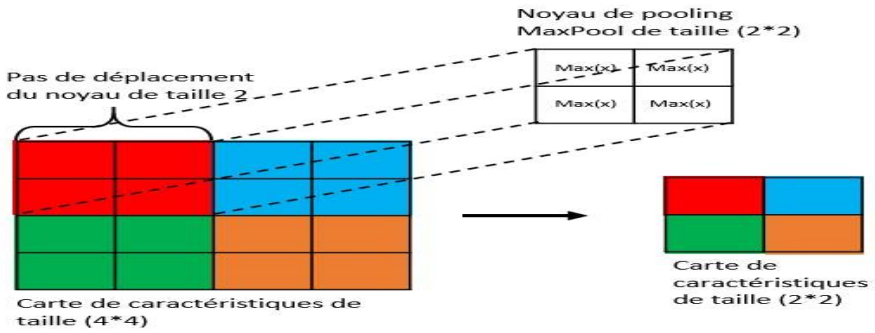


Figure II.6 : Schéma d'une opération de sous échantillonnage avec un noyau *Max Pool* de taille 2*2 et d'un pas de 2.

- **Couche entièrement connectée (FC) :**

Après plusieurs couches de convolution et de sous-échantillonnage, 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. Leurs fonctions d'activations donc être calculées avec une multiplication matricielle suivie d'un décalage de polarisation.

- **Couche de perte (LOSS) :**

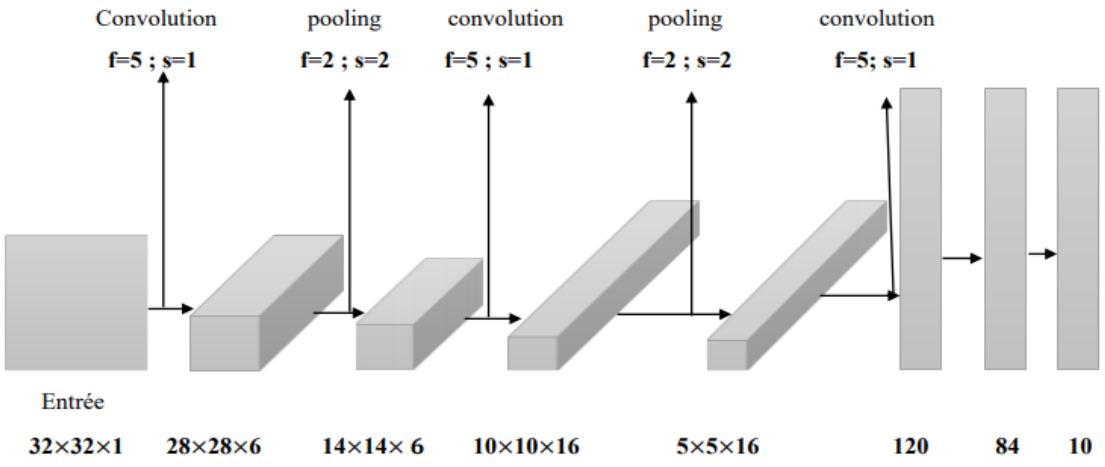
La couche de perte spécifie comment l'entraînement du réseau pénalise l'écart entre le signal prévu et réel. Elle est normalement la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La fonction « *Softmax* » permet de calculer la distribution de probabilités sur les classes de sortie **[14].**

II.3.3 les architectures des réseaux de neurones de convolution :

- **Lenet-5 :**

Comprenons l'architecture de **Lenet-5**. Le réseau a cinq couches avec des paramètres apprenable et donc nommé **Lenet-5** (FigureII.7). Il comporte trois ensembles de couches de convolution avec une combinaison de regroupement moyen. Après la convolution et la mise en commun des couches moyennes, nous avons deux couches entièrement connectées. Enfin, un classificateur *Softmax* qui classe les images dans la classe respective.

- L'entrée de ce modèle est une image **32 × 32** en niveaux de gris donc le nombre de canaux est un.
- Nous appliquons ensuite la première opération de convolution avec la taille de filtre **5×5** et nous avons 6 de ces filtres. En conséquence, nous obtenons une carte des caractéristiques de taille **28×28×6**. Ici, le nombre de canaux est égal au nombre de filtres appliqués.
- Après la première opération de regroupement, nous appliquons le regroupement moyen et la taille de la carte des caractéristiques est réduite de moitié. Notez que le nombre de canaux est intact.
- Ensuite, nous avons une couche de convolution avec seize filtres de taille **5×5**. Encore une fois, la carte des fonctionnalités a changé, elle est **10×10×16**. La taille de sortie est calculée de la même manière. Après cela, nous avons à nouveau appliqué une couche moyenne de regroupement ou de sous-échantillonnage, qui réduit à nouveau la taille de la carte d'entités de moitié, c'est-à-dire **5×5×16**.
- Ensuite, nous avons une couche de convolution finale de taille **5×5** avec **120** filtres. Comme le montre l'image ci-dessus. Laissant la taille de la carte des caractéristiques **1×1×120**. Après quoi, le résultat d'aplatissement est de **120** valeurs.
- Après ces couches de convolution, nous avons une couche entièrement connectée avec quatre-vingt-quatre neurones. Enfin, nous avons une couche de sortie avec dix neurones puisque les données ont dix classes **[20].**



FigureII.7: l'architecture finale du modèle **Lenet-5.**

- **Resnet-50 :**

ResNet (*Residual Neural Network*) est une architecture de réseau profond avec 152 couches (Figure II.8). Ceci a été développé par Microsoft en 2015 **[21].** Il a remporté l'ILSVRC2015 (*ImageNet Large Scale Visual Recognition* Challenge 2015) avec un taux d'erreur de 3,6 %, ce qui est considéré comme meilleur que la précision au niveau humain. Les couches résiduelles dans *ResNet* calculent les changements dans l'entrée. Ceci est ensuite ajouté à l'entrée pour produire la sortie. ResNet-50 est l'un des premiers à adopter la normalisation des batchs. Le *ResNet* est constitué de deux blocs CONV et Identité.

Architecture ResNet-50 représentée avec les unités résiduelles, la taille des filtres et les sorties de chaque couche de convolution. La DRF (*Django REST framework*) extraite de la dernière couche de convolution de ce réseau est également représentée. FC1000 désigne la couche entièrement connectée avec **1000** neurones. Le nombre en haut du bloc de couches de convolution représente la répétition de chaque unité. n Classes représente le nombre de classes de sortie **[22].**

- **Inception-V3 :**

L'objectif du module *Inception* est d'agir comme un « extracteur de caractéristiques à plusieurs niveaux » en calculant des convolutions **1×1, 3×3** et **5×5** au sein du même module du réseau - la sortie de ces filtres est ensuite empilée le long de la dimension du canal et avant d'être introduite dans la couche suivante du réseau.

L'incarnation originale de cette architecture a été appelée *GoogLeNet*, mais les manifestations ultérieures ont simplement été appelées *Inception-VN*, où N fait référence au numéro de la version publiée par Google.

Les poids d'Inception-V3 sont plus petits que ceux de VGG et *ResNet*, avec 96 Mo **[23].**

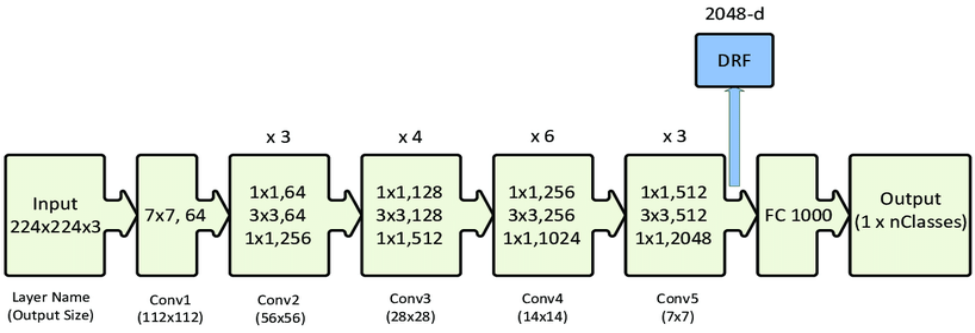


Figure II.8 : l`architecture de **Resnet-50.**

II.4 Conclusion :

Nous avons consacré ce chapitre à la présentation des notions de la classification ainsi que leurs intérêts dans le domaine d'imagerie et on a parlé aussi sur l'utilisation des réseaux de neurones de convolution dans ce domaine.

Dans ce chapitre, nous avons présenté les réseaux de neurones de convolution. Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée et de classifier ces caractéristiques, ils sont fondés sur la notion de « champs récepteurs » (*receptive fields*), ils implémentent aussi l'idée de partage des poids qui permettant de réduire beaucoup de nombres de paramètres libres de l'architecture. Ce partage des poids permet en outre de réduire les temps de calcul, l'espace mémoire nécessaire, et également d'améliorer les capacités de généralisation du réseau.

Chapitre III :

Résultats et discussion

III-1 Introduction :

Ce chapitre est consacré à l'implémentation d'algorithmes d'apprentissage profond basés sur les réseaux de neurones pour la classification des images.

Dans ce chapitre, on va définir l'architecture des trois modèles (*lenet*-5, *resnst-50*, *inception*-V3) qu'on a créés et par la suite, et on va appliquer ces modèles sur la base d'images qui on a proposé. Pour cela, on va travailler avec les bibliothèques d'intelligence artificielle *Tensorflow* et *Keras* existe dans l'environnement Python pour l'apprentissage et la classification d'images. Pour améliorer les performances des modèles, on va utiliser quelques techniques comme *data augmentation* et *dropout* pour évitera les problèmes du sur apprentissage.

Notre modèle CNN et les trois autres algorithmes préentraînés (*lenet*-5, *resnst-50*, *Inception* V3) sont utilisés pour classer les images en deux classes (*rugby* et *soccer*). Dans notre travail, nous avons utilisé deux bases de données. Pour chaque base d'images, il y a une base d'images pour l'entraînement et d'autres pour le test.

III.2 Logiciels, librairies et matériels utilisés dans l'implémentation :

III.2.1 Python :

Python est un langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels. En effet, parmi ses qualités, Python permet notamment aux développeurs de se concentrer sur ce qu'ils font plutôt que sur la manière dont ils le font. Il a libéré les développeurs des contraintes de formes qui occupaient leur temps avec les langages plus anciens. Ainsi, développer du code avec Python est plus rapide qu'avec d'autres langages **[24].**

III.2.2 Tensorflow :

Tensorflow est un *framework* de programmation pour le calcul numérique qui a été rendu Open Source par Google en novembre 2015. Depuis sa création, le *Tensorflow* n'a cessé de gagner en popularité, pour devenir très rapidement l'un des *framework* 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 multidimensionnelles, 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 **[25].**

III.2.3 Keras :

Keras est un réseau 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. 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* **[26].**

III.2.4 Colab Pro :

Google *Colab Pro* ou *Colaboratory* est un service cloud, offert par Google (gratuit), basé sur *Jupyter* Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur **[27].**

III.2.5 les ressources matérielles :

Dans ce projet en va utiliser Colab Pro qui contient des GPU et TPU plus rapide vous permet d'attendre moins longtemps lors de l'exécution du code, bénéficiez de plus de RAM et d'une capacité de disque plus grande, et donc de plus d'espace pour vos données, avec un notebook qui dispose d'une autonomie plus longue et de délais d'inactivité réduits, vous êtes moins souvent déconnecté et un Processeur Intel(R) Xeon(R) à 2,30 GHz et avec un GPU Tesla P100-PCIE de16GB du Ram **[28].**

III.3 Les bases de données d'images :

Dans notre base de données (*Our_DATA*) qui contient de 3058 images et qui ensuite divisé sur deux fichiers (*train* et *test*).

Les fichiers d'entraînement et test contiennent deux classes (*rugby, soccer*). Pour le fichier *test*, on trouve 305 images pour la classe *rugby*, et 305 images pour la classe *soccer*. Pour le fichier *train*, on trouve 1623 images pour la classe *rugby*, et 1623 images pour la classe *soccer*.

III.4 Architecture de notre réseau :

Au cours de nos expérimentations, nous avons utilisé les réseaux préentraînés :

Les trois réseaux préentraînés utilisés sont Lenet-5, ResNet-50 et Inception-v3. Ces réseaux ont été entraînés sur plus d'un million d'images dans la base de données *ImageNet*, et peuvent classer les images en 1000 classes.

Les principales caractéristiques de ces réseaux sont données dans lesquelles nous remplaçons la dernière couche apprenable par trois nouvelles couches de *MaxPooling2D* et deux couches entièrement connectées (*fully connected*) avec un nombre de sorties égal à 1024 et 2, respectivement.

III.4.1 architecture lenet-5 :

Le premier modèle est composé de deux couches de convolution et deux couches de *sous échantillonnage* et trois couches d'entièrement connectées. L'image en entrée est de taille 224x224 pixels, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 5x5, chacune de nos couches de convolution est suivie d'une fonction d'activation *ReLU,* cette fonction force les neurones à retourner des valeurs positives, après cette, on applique *sous échantillonnage* pour réduire la taille de l'image ainsi la quantité de paramètres et de calcul. À la sortie de cette couche, nous aurons 32 *feature maps* de taille 112x112.

On répète la couche de convolution qui composées de 48 filtres de taille 5x5, la fonction d'activation *ReLU* est appliquée toujours sur chaque convolution. Une couche de sous

échantillonnage est appliquée après la couche de convolution. À la sortie de la dernière couche sous échantillonnage, nous aurons 48 *feature maps* de taille 56x56.

Après ces couches de convolution et sous échantillonnage, nous utilisons un réseau de neurones composé de trois couches entièrement connectées. Le premier couche contient de 256 neurones et la deuxième couche contient de 84 neurones où la fonction d'activation utilisée est le *ReLU*, et la troisième couche est un *sigmoïde* qui permet de calculer la distribution de probabilité de deux classes (*rugby, soccer*).

Dans la figure III.1, on peut voir les résultats de la classification. D'après la Figure III.1, la précision (*accuracy*) de l'apprentissage de l'entraînement (*train*) et de teste (*test)* sont augmenté avec le nombre d'itérations, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. Si la précision est diminuée, alors on aura besoin de plus d'information pour faire apprendre notre modèle, et par conséquent, on doit augmenter le nombre d'itérations et vice versa. De même, l'erreur d'apprentissage diminue avec le nombre d'itérations et l'erreur de test augmente avec le nombre d'époque. Nous remarquons aussi qu'on a fixé le nombre d'époques (300 itérations) et le nombre de *batch size* (32), le modèle prend un temps d'exécution de 8 heures. Pour l'entraînement, l'erreur de la classification est 0.11 et la précision de la classification est 95.34 %, et pour le test, l'erreur est 0.64 et la précision est 83.22 %.

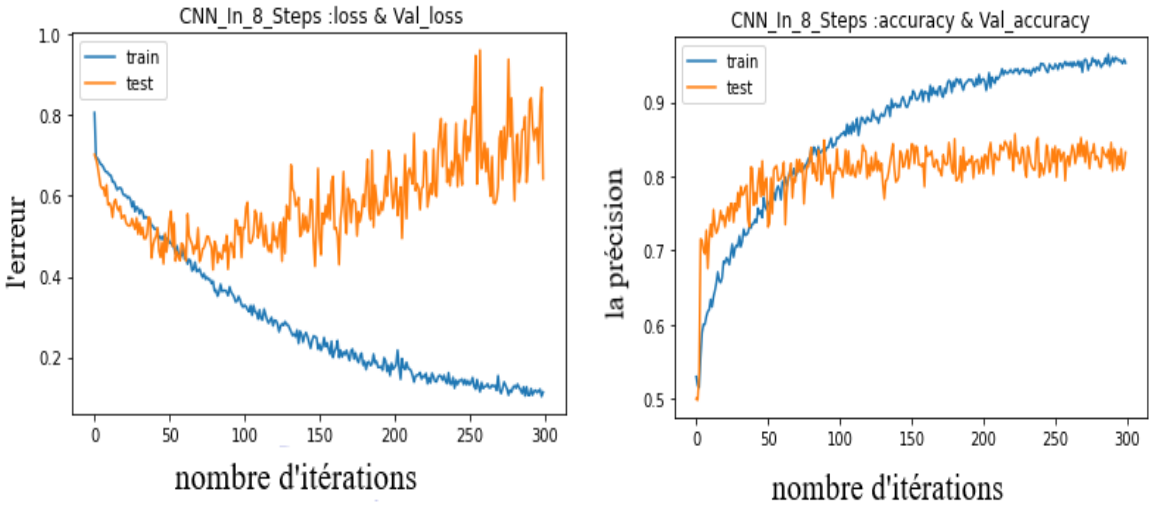


Figure III.1 : La précision et l'erreur du modèle Lenet-5.

III.4.2 architecture inception-V3 :

Dans le deuxième modèle, on implémente le modèle inception-V3 ou on utilise notre base d'images. Les résultats d'entraînement et de teste on les voir dans la figure III.2.

La précision (*accuracy*) de l'apprentissage (*train*) et de test sont augmenté plus rapide par rapport à *lenet-5* avec le nombre d'itérations et ça nous a donné un bon résultat pour la précision et ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. Et pour l'erreur d'apprentissage, diminue rapidement par rapport à *lenet-5*.

Nous remarquons aussi qu'on a fixé le nombre itérations (300 itérations) et le nombre de *batch size* =32 et *dropout* = 0.3. Le temps d'entraînement est 8 heures.

Pour l'entraînement, l'erreur de la classification est 0.0753 et la précision de la classification est 97.72 %. Pour le test, l'erreur est 0.9541 et la précision est 87.17 %.

III.4.3 architecture Resnet-50 :

Dans le troisième modèle, on implémente le modèle Resnet-50 ou on utilise notre base d'images. Les résultats d'entraînement et de teste on les voir dans la figure III.3.

La précision (*accuracy*) de l'apprentissage (*train*) et de test sont augmenté plus rapide par rapport à *lenet-5* et *inception v3* avec le nombre itérations et ça nous a donné un très bon résultat pour la précision Et les résultats sont meilleurs que le modèle précédent (*inception-v3*) et pour l'erreur d'apprentissage diminue rapidement par rapport à *lenet 5* et *inception v3*.

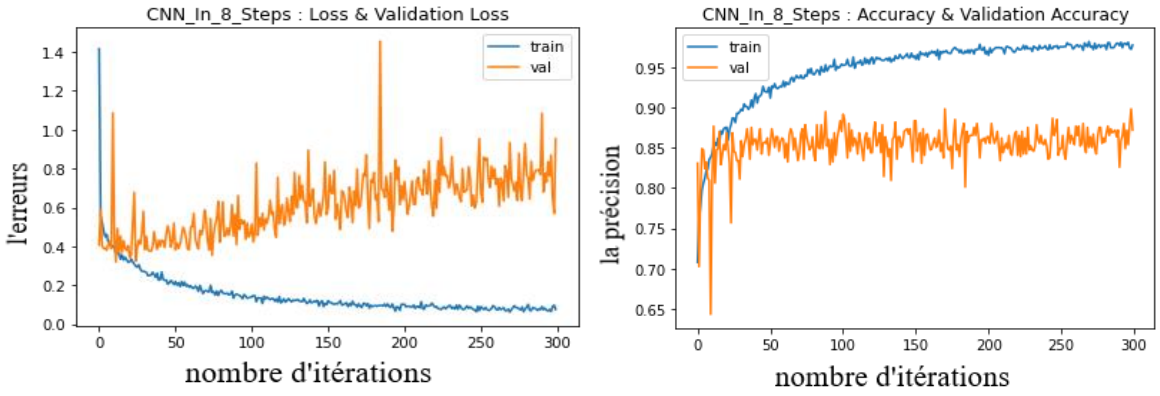


Figure III.2 : La précision et l'erreur pour le modèle *inception-v3*.

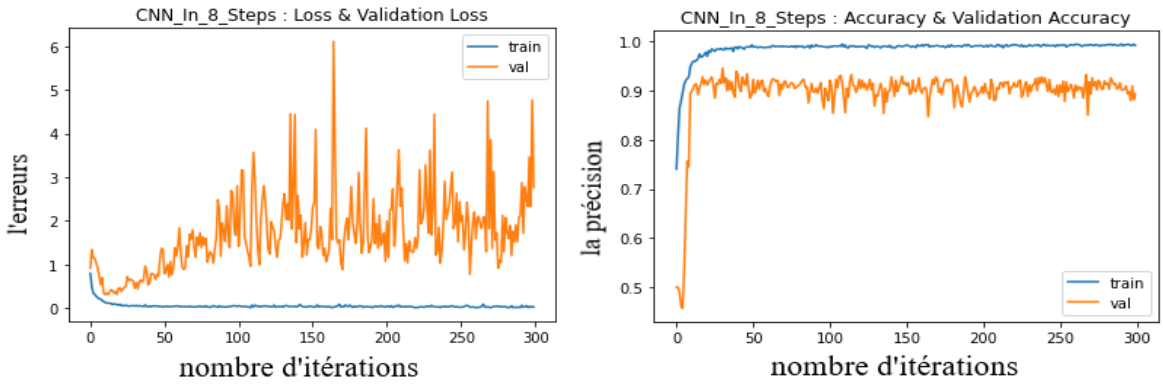


Figure III.3 : La précision et l'erreur pour le modèle resnet-50.

Nous remarquons aussi qu'on a fixé le nombre itérations (300 itérations) et le nombre de *batch size* =32 et *dropout* = 0.3. Le temps d'entraînement est 8 heures.

Pour l'entraînement, l'erreur de la classification 0.0350 et la précision de la classification sont 98.90 %. Pour le test, l'erreur est 0.962 et la précision est 89.75 %.

III.4.4 architecture de modèle proposé :

On a proposé un modèle très simple et défirent avec les modèles précédents et nous avons obtenu des résultats bons et acceptables.

Notre modèle est composé de quatre couches de convolution et quatre couches de sous échantillonnage et dropout de (0.4) et deux couches d'entièrement connectées.

L'image en entrée est de taille 224x224, l'image passe d'abord dans la première couche de convolution. Cette couche est composée de 32 filtres de taille 3x3, chacune de nos couches de convolution est suivie d'une fonction d'activation *ReLU,* cette fonction force les neurones à retourner des valeurs positives, après cette convolution on applique sous échantillonnage pour réduire la taille de l'image ainsi la quantité de paramètres et de calcul. À la sortie de cette couche, nous aurons 32 *feature maps* de taille 112x112.

On répète la couche de convolution qui composées de 48 filtres de taille 3x3, la fonction d'activation *ReLU* est appliquée toujours sur chaque convolution. Une couche de sous échantillonnage est appliquée après la couche de convolution. À la sortie de la dernière couche sous échantillonnage, nous aurons 48 *feature maps* de taille 56x56.

On répète la couche de convolution qui composées de 64 filtres de taille 3x3, la fonction d'activation *ReLU* est appliquée toujours sur chaque convolution. Une couche de *Sous échantillonnage* est appliquée après la couche de convolution. À la sortie de la dernière couche sous échantillonnage, nous aurons 64 *feature maps* de taille 28x28.

On répète la couche de convolution qui composées de 128 filtres de taille 3x3, la fonction d'activation *ReLU* est appliquée toujours sur chaque convolution. Une couche de *sous échantillonnage* est appliquée après la couche de convolution. À la sortie de la dernière couche sous échantillonnage, nous aurons 128 *feature maps* de taille 14x14.

Après ces couches de convolution et sous échantillonnage, nous utilisons un réseau de neurones composé de trois couches entièrement connectées.

Nous avons entraîné notre modèle 3 fois avec des nombres d'itérations différentes.

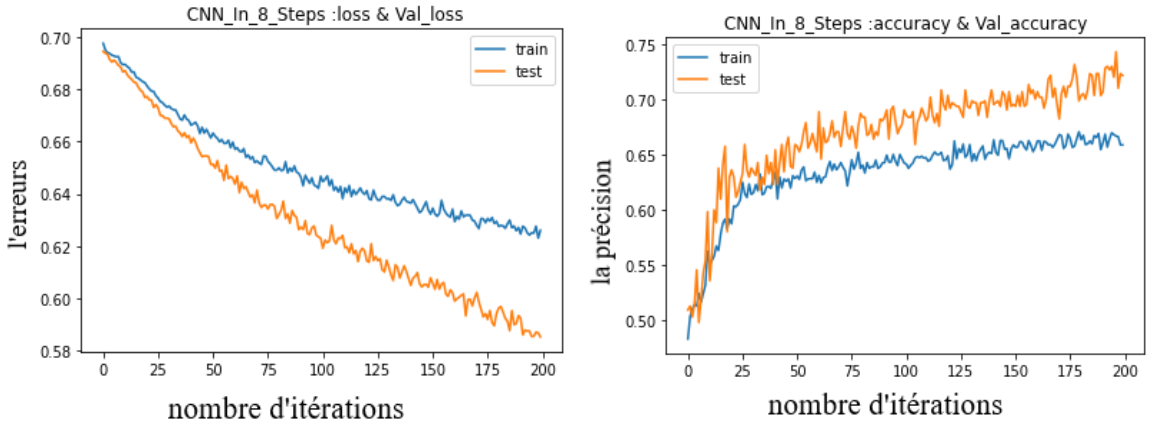


Figure III.4 : La précision et l'erreur pour le modèle proposer à 200 itérations.

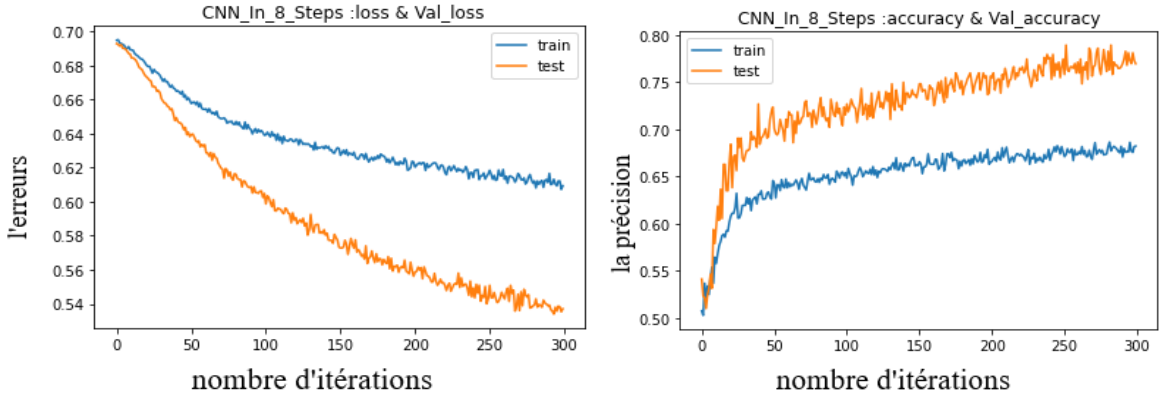


Figure III.5 : La précision et l'erreur pour le Modèle proposer à 300 itérations.

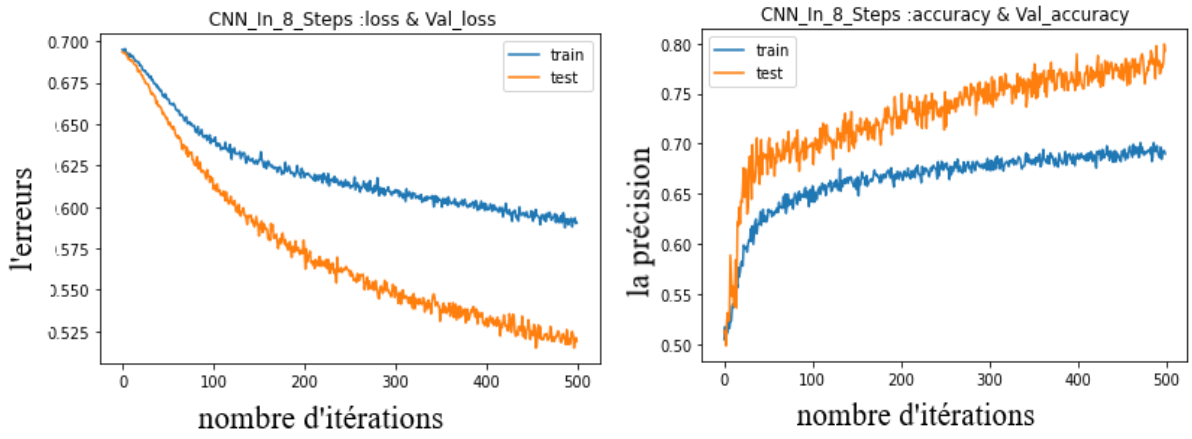


Figure III.6 : La précision et l'erreur pour le modèle proposer à 500 itérations.

Dans la figure III.4, on peut voir la précision et l'erreur de la classification de notre modèle par les paramètres suivantes : batch size = 32, nombre itération =200.

Dans la figure III.5, on peut voir la précision et l'erreur de la classification de notre modèle par les paramètres suivantes : batch size = 32, nombre itération =300

Dans la figure III.6, on peut voir la précision et l'erreur de la classification de notre modèle par les paramètres suivantes : batch size = 32, nombre itération =500.

D'après les trois figures, la précision de l'apprentissage augmente avec le nombre d'itérations, ceci reflète qu'à chaque itération le modèle apprenne plus d'informations. Si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'itérations et vice versa. De même, l'erreur d'apprentissage et de test diminue avec le nombre itérations.

Nous remarquons aussi qu'on a fixé le nombre itérations à 200 itérations et le nombre de *batch size* = 32, et avec *dropout* = 0.4. Le modèle prend un temps d'exécution de 6 heures. Pour l'entraînement, il y a une erreur de 0.63 et une précision de 68.50 %. Pour le test, il y a une erreur de 0.5220 et une précision de 77.30 %.

Lorsqu'on fixe le nombre d'itérations à 300 itérations, le temps d'exécution est 8 heures. Pour l'entraînement, il y a une erreur de 0.62 et une précision de 69.50 %. Pour le test, il y a une erreur de 0.52 et une précision de 78.50 %.

Lorsqu'on fixe le nombre d'itérations à 500 itérations, le temps d'exécution est 13 heures 30 min. Pour l'entraînement, il y a une erreur de 0.5904 % et une précision de 69 %. Pour le test, il y a une erreur de 0.5189 et une précision de 79.28 %.

Plus on augmente en nombre d'époques on a obtenus des meilleurs résultats pour la précision (*accuracy*) de l'apprentissage (*train*) et de test.

- **Tableau de comparaissant des résultats :**

Le tableau suivant montré les résultats obtenus sur les quatre modèles :

Les modèles	Architecture utilisée			Nombre d'itération	La Précision d'entraînement	La précision de test	Nombre de Paramètres	Temps d'entraînement
	Couche de Convolution	couche de Pooling	Fully connected					
Lenet-5	02	02	03	300	95.34%	83.28%	38, 598, 062	28800 seconds (8 heures)
Resnet-50	46	02	02	300	98.90%	89.75%	21, 636, 712	28800 seconds (8 heures)
Inception-v3	/			300	97.72%	87.17%	23, 851, 784	28800 seconds (8 heures)
Modèle proposé	04	04	02	200	68.50%	77.30%	3, 127, 258	21600 seconds (6 heures)
				300	69.50%	78.50%	3, 127, 258	28800 seconds (8 heures)
				500	69%	79.28%	3, 127, 258	48600 Seconds (13 heures 30 min)

Tableau III.1 : Tableau de comparaissant des résultats entre les modèles.

Pour tester notre modèle, nous avons utilisé 49 photos pour faire la prédiction et nous avons obtenu les résultats indiqués dans la Figure (III.7).

- Le nombre 7 représente le nombre d'images de prédiction correctes, cela signifie que lorsque nous lui avons donné des photos de rugby, il a prédit qu'il s'agissait de photos de rugby.
- Le nombre 18 représente le nombre d'images de fausses prédictions, cela signifie que lorsque nous lui avons donné un échantillon de photos de soccer, il a prédit qu'il s'agissait de photos de rugby.

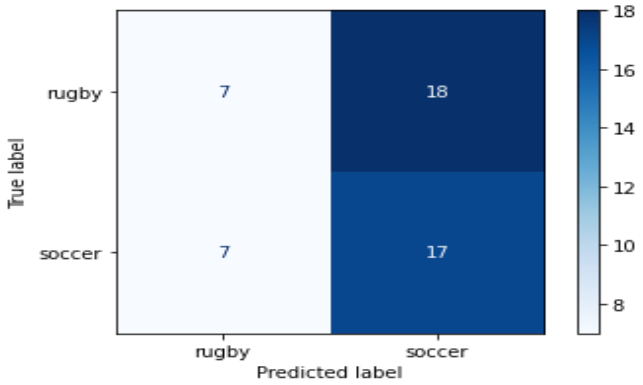


Figure III.7 : Matrice de confusion pour le modèle proposé.

- Le nombre 7 représente le nombre d'images de fausses prédictions, cela signifie que lorsque nous lui avons donné un échantillon de photos de rugby, il a prédit qu'il s'agissait de photos de soccer.
- Le nombre 17 représente le nombre d'images de prédiction correctes, cela signifie que lorsque nous lui avons donné des photos de soccer, il a prédit qu'il s'agissait de photos de soccer.

III.5 Conclusion :

Nous avons présenté dans ce chapitre, une approche de classification basée sur la CNN, pour cela, on a utilisé trois modèles avec différentes architectures (*lenet-5*, *resnet-50* et *inception-v3*) et avec un notre modèle proposé. La comparaison des résultats trouvés a montré que le nombre d'itérations, batch size et la profondeur de réseaux, sont des facteurs importants pour l'obtention de meilleurs résultats.

Conclusion générale :

Dans cette mémoire, nous avons discuté des notions fondamentales des réseaux de neurones en général et des CNN en particulier. Nous avons introduit ces CNN en présentant les différents types de couches utilisées dans la classification : la couche de convolution, la couche de rectification, la couche de sous échantillonnage et la couche entièrement connectée. Nous avons parlé aussi sur les méthodes de régularisation (*dropout* et *data augmentation*) utilisées pour éviter le problème de sur apprentissage.

Les paramètres du réseau sont difficiles à définir a priori. C'est pour cette raison que nous avons définie différents modèles avec des architectures différentes afin d'obtenir des meilleurs résultats en termes de précision et d'erreur.

Nous avons rencontré quelques problèmes dans la phase d'implémentation, l'utilisation d'un CPU a fait que le temps d'exécution était trop coûteux. Afin de régler ce problème, on doit utiliser des CNN plus profondes déployées sur un GPU au lieu d'un CPU sur des bases plus importantes.

Bibliographie :

[1] lecun/UPL4485925235409209505_Intelligence_Artificielle_____Y._LeCun.pdf. https://www.college-de-france.fr/media/yann-

[2] Perceptron-Wikipedia
http://fr Wikipedia.org.

[3] Pierre Buyssens, Fusion de différents modes de capture pour la reconnaissance du visage appliquée aux e_ transactions DOCTORAT de l'UNIVERSIT´E de CAEN Le 4 janvier 2011.

[4] Comprendre les réseaux de neurones moncoachdata
http://moncoach data.com.

[5] Definition deep learning-apprentissage profound
https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262.

[6] MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLOME MASTER EN. Etude d'une méthode de compression qui utilise l'intelligence artificielle.

[7] Lebigdata.fr/ réseau de neurones artificiel définition. http://www.lebigdata.fr.

[8] Regulization in deep learning L1, L2 and dropout.
https://towardsdatascience.com/regularization-in-deep-learning-l1-l2-and-dropout 377e75acc036.

[9] Optimization technique popularly used in deep learning. http:// medium.com.

[10] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural net- work (ann) modeling and its application in pharmaceutical research," Journal of pharmaceutical and biomedical analysis, vol. 22, no. 5, pp. 717–727, 2000.

[11] Normalizing Inputs of Neural Networks| Baeldung.
https://www.baeldung.com/cs/normalizing-inputs-artificial-neural-network.

[12] Qu`est-ce qu`une image numérique. http://tecfaetu.unige.ch/staf/staf-f/anzani/staf14/ex2/welcome.html.

[13] Naciri H., Chaoui N, Conception et Réalisation d'un système automatique d'identification des empreintes digitales, Mémoire de PFE, Université de Tlemcen, 2003.

[14] classification des images avec les réseaux de neurones. http://dspace.univ-tlemcen.dz/bitstream/112/12235/1/Classification-des-images-avec-les-reseaux-de-neurones.pdf.

[15] Traitement des images numériques.
http://www.tsi.enst.fr/pages/enseignement/ressources/mti/egal-histo/rapport.htm/.

[16] filtres moyenneurs-platforme logicieles copia
https://www.pfl-cepia.inra.fr/index.php?page=tutoImg-filtres-moyenneurs.

[17] filtre médian
http://fr.wikipédia.org.

[18] Padding (machine learning) definition | deepai
https://deepai.org/machine-learning-glossary-and-terms/padding.

[19] Deep learning : les réseaux de neurones convolutifs
https://www.imaios.com/fr/Societe/blog/Classification-des-images-medicales-comprendre-le-reseau-de-neurones-convolutifs-CNN.

[20] Lenet-5| Lenet-5 Architecture| introduction to Lenet-5
https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5.

[21] Mémoire de Fin d'étude Master (Détection d'objets basé Faster R-CNN)
https://dspace.univguelma.dz/jspui/bitstream/123456789/10135/1/CHOUINI_MOHAMMED%20EL%20MOUNSIF1603395873%20%281%29.pdf.

[22] Resnet-50
https://www.researchgate.net/figure/ResNet-50-architecture-26-shown-with-the-residual-units-the-size-of-the-filters-and_fig1_338603223.

[23] ImageNet : VGGNet, ResNet, Inception.
https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/.

[24] Python : définition et utilisation de ce langage informatique
https://www.journaldunet.fr

[25] http://www.tensorflow.org/

[26] http://www.keras.io/

[27] Google Colab : Le guide Ultime | Le Data Scientist
https://ledatascientist.com ›

[28] Colab pro.
https://www.google.com/search?q=carte+graphique+de+colab+pro&oq=carte+graphique+de+colab&aqs=chrome.1.69i57j33i160.21048j1j15&sourceid=chrome&ie=UTF-8#.