République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Larbi Tébessi - Tébessa

Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie

Département : Mathématiques et Informatique

Mémoire de fin d'études
Pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Option : Réseau et sécurité

Classification d'image numériques par de procédés de Deep Learning : Application aux panneaux de signalisation

Thème

Présenté Par :

Fares Imtiez

Devant le jury :

Dr.Guettal Abdeldjalil	MCA	Univérsité Larbi Tébessi	Président
Dr.Laiemeche lakhdar	MCA	Univérsité Larbi Tébessi	Examinateur
Dr.Mekhaznia Tahar	MCA	Univérsité Larbi Tébessi	Rapporteur

Session Juin 2021

Résumé

La détection et la reconnaissance automatiques des panneaux de signalisation sont très importantes et pourraient potentiellement être utilisées par l'assistant de conduite pour réduire les accidents. Dans ce mémoire de fin d'études, un réseau de neurones convolutifs profonds (CNN) a été utilisé pour développer un système de détection et de reconnaissance autonome de la signalisation routière (ATSR). Dans le but d'améliorer la précision de la classification, l'architecture CNN a été utilisée avec différents paramètres dans divers scénarios afin d'obtenir le meilleur taux de reconnaissance, Les résultats expérimentaux montrent qu'architecture proposé a atteint une précision de 99,67%, sur l'ensemble de données de référence de reconnaissance des panneaux de signalisation allemands (GTSRB). Sans aucune adition des modules ou bien des modèles pré-entrainés

Les mots clés : Détection et reconnaissance des panneaux de signalisation, réseaux de neurones convolutifs (CNN), Deep Learning

Abstract

Automatic detection and recognition of traffic signs is very important and could potentially be used by the driving assistant to reduce accidents. In this final thesis, a deep convolutional neural network (CNN) is used to develop an autonomous road sign detection and recognition system (ATSR). In order to improve the accuracy of the classification, the CNN architecture is used with different parameters through many scenarios in order to obtain the best recognition rates, The experimental results show that the proposed architecture achieved an accuracy of 99,67%, on German Traffic Sign Recognition Reference Data Set (GTSRB). Without any addition of modules or pre-trained models

Keywords: Detection and recognition of traffic signs, convolutional neural networks (CNN), Deep Learning

ملخص

يعد الإكتشاف و التعرف التلقائي على إشارات المرور أمرا مهما للغاية و يمكن إستخدامه بواسطة مساعد آلي خاص بالقيادة لتقليل الحوادث، في هده الأطروحة، يتم يتم إستخدام شبكية عصبية تلافيفية عميقة من إجل تطوير نظام مستقل للكشف عن إشارات المرور و التعرف عليها، تم إستخدام هده البنية مع العديد من البرميترات في العديد من السيناريهات من إجل تحسين دقة التصنيف، و أظهرت النتائج التجريبية أن البنية المقترحة حققت دقة 99،67 بالمئة على مجموعة البيانات المرجعية للتعرف على إشارات المرور الألمانية بدون إية إضافة أو إستخدام نمادج مدربة مسبقا

الكلمات المفتاحية الكشف و التعرف على إشارات المرور ، التعلم العميق، الشبكات العصبية التلافيفية

Remerciement

On remercie Allah le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce projet.

Tout d'abord ce travail ne serait pas aussi riche et n'aurait pas pu avoir le jour sans l'aide et l'encadrement de Dr.MEKHAZNIA Taher, on le remercier pour la qualité de son encadrement exceptionnel, pour sa patience, sa rigueur et sa disponibilité durant notre préparation de ce travail.

Je remercier également toutes les personnes qui nous ont aidés et soutenue de près ou de loin.

Dédicaces

Je dédier le fruit de ce travail :

A l'homme de ma vie, mon exemple éternel, mon soutien moral et source de joie et de bonheur, celui qui s'est toujours sacrifie pour me voir réussir, que dieu te garde dans son vaste paradis à toi mon père.

A la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur : maman, Sans oublier ma tante Zoubida qui m'a soutenue tout au long de ce projet

Aux Dr.FARES Tarek et Dr.FARES Abdelhafid pour ses efforts qui me vraiment e fait éclaire le chemin pour une vie académique parfaite

Aux personnes qui m'ouvre le cœur, le remplis de bonheur, reste dans ma vie, remplis la d'envie

Aux personnes qui m'ont toujours aidé et encouragé, qui étaient à mes cotes, et qui m'ont accompagné durant mon chemin d'études, mes aimables amis, collègues d'étude et mon frère et ma sœur

FARES Imtiez

Tableau des matières

Liste des figures

Liste des tableaus

Introduction générale

Avec l'expansion démographique et la diversité des activités, l'usage des moyens de transports est devenu une nécessite dans notre vie quotidienne, la gestion du trafic doit être gérée par un règlement misant sur la fluidité et la sécurité : Les panneaux de signalisation, qui doit être assimilée et respectée par les conducteurs. Cependant dans certains cas, cette tache devient un peut compliquer et difficile à cause de la fatigue des conducteurs ou bien a cause des males condition talque la pluie, le bouillard, état des panneaux. Pour réduire le taux de la mortalité causé par les accidents des voitures, les systèmes informatiques qui déchiffrent les signaux et servir en guise de guide pour les conducteurs

Les systèmes de reconnaissance des panneaux de signalisation (TSR) représentent une des parties intégrantes les plus importantes des véhicules autonomes et des systèmes avancés d'assistance à la conduite (ADAS). Ils permettent d'aider le conducteur sur la route et réduire les accidents de route qui sont essentiellement causés par le fait de ne pas remarquer un changement de limitation de vitesse ou l'avertissement d'un danger potentiel ou une mauvaise lecture des panneaux.

Les systèmes de reconnaissance des panneaux de signalisation sont généralement constitués de deux parties : détection du panneau puis sa classification. Dans le cadre de ce travail, nous nous intéressons à la classification des panneaux de signalisation. Les panneaux de signalisation sont classés en des catégories selon le rôle ou la fonction associée au panneau et chaque catégorie contient des panneaux ayant la même forme mais avec des détails différents d'où la difficulté de classification.

La classification des panneaux de signalisation nécessite d'abord l'extraction des caractéristiques présentes dans les images des panneaux puis la classification se fait sur la base de ces caractéristiques. Plusieurs classifieurs peuvent après être utilisés tel que, le k-means, KNN, SVM etc.

Le but du travail proposé consiste à identifier le sens de l'image. Pour ce fait, une classification du contenue de la base de données s'avère nécessaire. L'usage des classifieures apparait la solution de salut. Il est donc utile de faire recours aux réseaux de neurones qui paraissent d'une utilité majeure, leur rôle consiste a extraire les caractéristiques des images afin de construire certains class dédiées à l'entrainement et d'autre aux tests

La particularité de notre travail consiste à réaliser les dites taches sans recours aux modules alternatifs qui apparaissent dans la plupart des travaux similaires

Le manuscrit est structuré comme suit :

Le premier chapitre on expose les notions de base nécessaires au traitement d'images pour permettre de comprendre la nature des traitements décrits après, ainsi qu'une description des principaux algorithmes de classification d'images.

Dans le deuxième chapitre, au regard de l'approche développée basée sur les réseaux de neurones, après une description sur les algorithmes de Machine Learning et les réseaux de neurones profonds nous focalisons notre étude sur l'étude de spécificités et l'usage des réseaux de neurones convolutionnels (CNN).

Le troisième chapitre est consacré à la classification supervisée des panneaux de signalisation. Ce chapitre met également l'accent sur l'architecture proposée pour chacun des réseaux profonds ainsi que ses hyperparamètres, le chapitre englobe également une l'implémentation et à la présentation des résultats expérimentaux obtenus par chacun des réseaux profonds CNN

Enfin, l'analyse et la discussion des différents résultats obtenus et les suites à mener à ce travail pour améliorer la qualité des résultats feront l'objet d'une conclusion.

Chapitre 1

Reconnaissance d'images

1. Introduction

A l'heure actuelle, L'utilisation de l'intelligence artificielle permet d'ajouter plus de valeurs aux plusieurs domaines : administratif et automobile. L'une d'applications de l'intelligence artificielle qu'on peut citez, la classification d'image qui a évolué très rapidement. Cette sous discipline permet à la machine de voir et comprendre les images, et de prendre une décision en fonction de la situation, ou l'association des objets présents dans la scène

Dans ce chapitre ; nous allons présenter la classification d'image. Dans un premier temps, nous décrivons les notions de base nécessaires aux traitements d'image. Enfin nous détaillerons l'opération de classification.

2. Définition d'image

L'image rassemble les icones qui entretiennent une relation d'analogie qualitative entre signifiant et référent : formes, couleurs, proportions qui permettent de les reconnaître[1]

L'image numérique est un code qui est acquis, stocké et interprété.

Cette abstraction de l'image en un code apporte un grand nombre d'avantages. Tout d'abord, cela permet sa reproductibilité à l'infini sans dégradation, puisqu'un fichier numérique est une série d'octets que l'on peut dupliquer exactement, quelle que soit la nature de ce fichier.

Le codage permet aussi de faciliter la transmission de l'image, par exemple en la scindant en plus petites parties dont on conserve simplement les coordonnées dans le plan

Comme elle permet d'abstraire l'image de son support, il est également possible de la visualiser simultanément de manière rigoureusement identique. On peut par ailleurs l'afficher en fonction d'un environnement déterminé, ce qui permet l'interactivité (c'est le cas par exemple lorsqu'on navigue dans les grossissements d'une lame virtuelle ou que l'on insère ou affiche des annotations).

En revanche, une image numérique est contrainte par l'espace de stockage que l'on souhaite lui assigner. Elle dépend également du système de codage de l'image et de son décodage (algorithme de compression et de décompression qui constitue le format de l'image), et enfin du matériel d'affichage (écran, projecteur, etc.).[2]

3. Type d'images

3.1. Matricielle

Celui qui considère l'image comme un rectangle constitué de points élémentaires de couleur uniforme, les pixels. Une image plein écran en résolution standard 800 x 600 est constituée de 480 000 pixels. Décrire l'image revient alors à préciser la couleur de chaque point. Le fichier graphique sera une liste de nombres binaires, correspondants à ces couleurs, précédée par un en-tête (header) décrivant la méthode utilisée. On peut en effet imaginer plusieurs façons de décompter les points. Ce principe est celui des images dites numériques ou bitmap.[3]

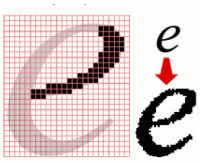


Figure 1. Image matrecielle en détaille

Images matrecielles	
Points forts	**Points faibles**
Elles autorisent la qualité photographique.	Les fichiers peuvent être encombrants.
	Leur agrandissement provoque un effet de mosaïque (les pixels agrandis deviennent des carrés visibles).
	La création d'une image « à la souris » est difficile. Usage conseillé d'un périphérique de numérisation : scanner, appareil photo numérique...
	Les retouches sont délicates : effacer un élément de l'image crée un « trou ».

Tableau 1. Points faibles et forts d'une image matrecielle

3.2. Vectorielle

L'autre qui considère l'image comme un ensemble de figures élémentaires pouvant être décrites par des données mathématiques (coordonnées de points, tangentes en un point d'une courbe, etc.). Le fichier décrit ces différentes figures, véritableaus "objets" graphiques indépendants les uns des autres. Ce principe est celui des images dites vectorielles (WMF, CGM, etc.) [3]

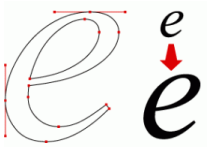


Figure 2. Image Vectorielle en détaille

Images Vectoriels	
Points forts	**Points faibles**
Les fichiers sont petits.	Inutilisables pour des images complexes, des photographies.
Les images sont redimensionnables sans perte de qualité, les courbes sont lissées quel que soit l'échelle d'affichage.	Non reconnues par les navigateurs Internet et par certains logiciels multimédia.
Les retouches sont aisées puisque les différents éléments de l'image sont indépendants.	

Tableau 2. Points faibles et forts d'une image vectoriel

La méthode utiliser passer d'une image matricielle vers une image vectoriel s'appelle : La vectorisation. L'algorithme doit analyser l'image pour en déduire un ensemble de primitives géométriques. L'opération s'accompagne d'une perte de détails et de nuances de couleurs

Figure 3. La vectorisation

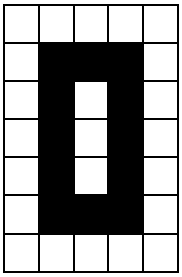
4. Carécteristique d'une image

4.1. Pixel

Tant qu'une image est définie par sa dimension (la surface), implique que parmi les caractéristiques qu'on peut la remarque et nous s'intéresse dans notre étude que

l’image numérique est constituée de ‘PICTure ELementS’ ou bien le terme le plus connus ‘pixels’

Un pixel est un point dans l’espace colorimétrique de l’image numérique



0	0	0	0	0
0	1	1	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	1	1	0
0	0	0	0	0

Figure 4. Représentation du 0 ‘Zéro’ dans une image numérique

Comme il est illustré en noire et blanc dans la Figure 4 un pixel est présenté par 1 pour le noire et 0 pour le blanc et dans le cas d’une image en couleur RGB un pixel est représenté par 3 octets

4.2. Luminosité

Le mot luminance est substitué au mot brillance, qui correspond à l’éclat d’un objet.

La luminance est le degré de luminosité des pixels de l’image.

				
- 100%	- 50%	Original	+50%	+100%

Tableau 3. Diffèrent degrés du Luminance

4.3. Contraste

C’est l’opposition marquée entre deux régions d’une image, plus précisément entre les régions sombres et les régions claires de cette image.

Le contraste est défini en fonction des luminances de deux régions d’une image.

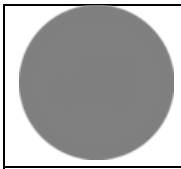
				
- 100%	- 50%	Original	+50%	+100%

Tableau 4. Diffèrent degrés du Contraste

4.4. Résolution

La résolution d'une image est définie par le nombre de pixels par unité de surface.

Une image ayant des pixels plus petits dispose d'une résolution supérieure, affiche plus de détails et pèse plus lourd au niveau de la taille du fichier.

Figure 5. Différence entre différentes résolutions sur un seul objet

4.5. Histogramme

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

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. D'où un traitement statistique des images.[4]

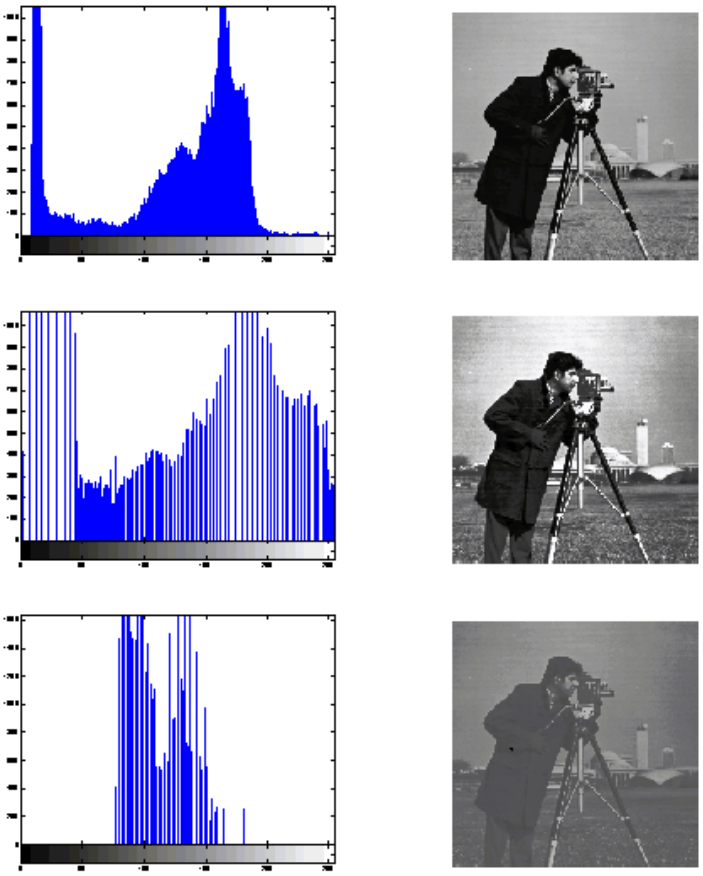


Figure 6. Exemple d'histogramme au niveau de gris

5. Couleurs et Encodage

5.1. Binaire

Comme il est illustrer dans la Figura 4, dans une image en codage de couleur binaire un pixel prend 2 'deux' valeurs soit 0 pour le blanc et un pour le noire, donc un pixel sera encodé sur 1 bit

5.2. Gris

Dans une image gris un pixel prend une valeur entre le 0 et 255, donc un pixel sera encodé sur 8 bits

5.3. RGB

Dans une image couleur un pixel contient les valeurs des 3 'trois' composants RGB (Red, Green, Blue)

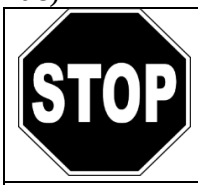
		
Binaire	Gris	RGB

Tableau 5. Couleurs et Encodage

6. Représentation d'une image

La figure 7 représente les différents éléments d'une image matricielle

- Une image I est une Matrice de dimension L x H
- Chaque élément à une valeur entière dans l'intervalle [Nmin , Nmax]
- Le nombre de « bits » requis pour représenter les niveaux de gris dans l'intervalle « N » est « K »
- La relation entre « K » et « N » est : $N = 2^K$
- Le nombre de bit pour représenter une image est donc : b = L . H . K

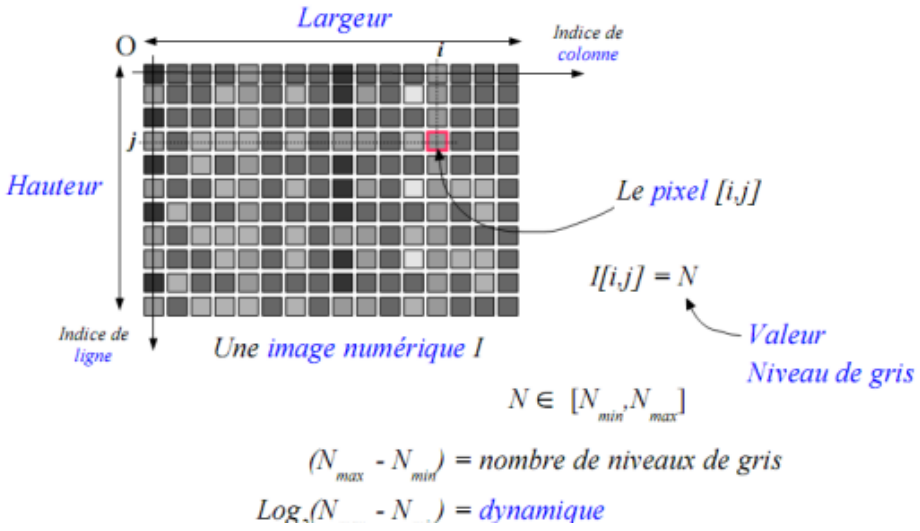


Figure 7. Représentation d'une image matricielle

7. Transformations Possibles sur l'Image Numérique :

Parmi les transformations possibles sur une image numérique en trouve :

- Le Traitement des Images : image -> image («image processing»).
- L'Analyse des Images : image -> mesures («image analysis») analyse sémantique
- Image description de haut niveau : images -> Classes d'objet («image understanding or recognition»).
- Reconstruction d'Images : Opérateurs ensemble d'informations->image

8. Formats d'images

La sauvegarde des images sous forme de fichier nécessite de stocker non seulement la liste des valeurs des pixels, mais aussi la taille de l'image et la manière dont sont codés les pixels.

8.1. Structure des fichiers

Un fichier image est en général composé de deux parties : un en-tête, et des données. L'exception concerne les fichier raw, qui ne contiennent que les données brutes, et qui ne peuvent pas être lues sans connaître les conditions dans lesquelles a été acquise l'image.

Ne peuvent pas être lues sans connaître les conditions dans lesquelles a été acquise l'image. L'en-tête contient la taille de l'image, le type de donnée (binaire, couleur, niveau de gris...), le nombre de bits utilisés pour le codage, l'ordre dans lequel sont stockés les valeurs, le type de compression utilisé... On peut parfois y trouver des informations utilisateurs (métadonnées), telles que les données EXIF. Suivent ensuite les données, éventuellement compressées, qui ne peuvent être lues qu'une fois l'en-tête décodé.[5]

8.2. Compression des images

La compression permet de transformer un ensemble de données de départ, correspondant aux valeurs des pixels, en un autre ensemble de données, de taille plus réduite, mais qui contient l'information de l'ensemble de départ. La décompression est l'opération de reconstruction des valeurs de départ à partir des données compressées. On distingue deux modes de compression : avec perte (ou destructif) et sans perte (ou non destructif).

Dans un mode de compression sans perte, l'image obtenue après décompression correspond exactement à l'image initiale. Plusieurs algorithmes existent, tels que le RLE (Run-Length-Encoding), qui détecte les plages de pixels de même valeur, ou le LZW, utilisé par les images GIF.

En pratique, ces types de compression sont plutôt e-caces pour des images avec beaucoup de redondances (beaucoup de zones avec des couleurs uniforme). Les images scientifiques, au même titre que les photographies, contiennent très peu de redondances, et les algorithmes de compression sans perte sont donc assez peu e-caces.

Dans un mode de compression avec perte, on accepte que l'image obtenue après décompression présente quelques différences avec l'image initiale. Les algorithmes utilisent pour cela une décomposition de l'image selon les composantes fréquentielles (compression JPEG) ou basée sur des ondelettes (JPEG2000), et ne conservent que les composantes principales de la décomposition. Il en résulte une détérioration de l'image reconstruite

Il va de soi que si l'on désire exploiter l'information stockée dans les images, il faut privilégier les modes de compression sans perte [5]

8.3. Stockage

Pour obtenir l'écriture binaire d'un pixel ayant comme valeur 179, il faut décomposer cette valeur comme somme de puissances de deux. On obtient ainsi $179=2^7+2^5+2^4+2+1$

Où l'on a pris soin d'ordonner les puissances de deux par ordre décroissant. Afin de faire mieux apparaître l'écriture binaire, on ajoute « 1× » devant chaque puissance qui apparaît dans l'écriture, et « 0× » devant les puissances qui n'apparaissent pas $179=1\times2^7+0\times2^6+1\times2^5+1\times2^4+0\times2^3+0\times2^2+1\times2+1\times2^0$.

L'écriture binaire de la valeur 179 du pixel est ainsi (1,0,1,1,0,0,1,1), où chaque 1 et chaque 0 correspond au facteur multiplicatif qui apparaît devant chaque puissance.

On peut écrire toute valeur entre 0 et 255 de cette manière, ce qui nécessite d'utilisation de 8 bits. Il y a en effet 256 valeurs possibles, et $256=2^8$. Pour stocker l'image complète, on a donc besoin de $n\times p\times 8$ bits.

Pour l'image montrée à la première figure, on a ainsi besoin de $240\times240\times8=460800$ bits.

On utilise le plus souvent l'octet (8 bits) comme unité, de sorte que cette image nécessite 57,6 ko (kilo octets).[7]

8.4. Quelques formats

Un format d'image c'est la représentation des informations codées sur l'image. Selon le support de la communication on trouve les formats suivants :

- Sans compression
  - TIF : pour l'archivage
  - PSD : format de travail Photoshop
  - XCF : format de travail Gimp
- Compression non-destructif
  - GIF : 256 tons seulement, peut gérer la transparence, les animations
  - PNG : millions de couleur, peut gérer la transparence
- Compression destructive
  - JPG : gère mieux les nuances de couleur, plus léger, perte de qualité

Dans le cadre de notre étude en s'intéresse avec les fichiers d'images de format PNG :

- Profondeur de 16 bits, mais utilise 4 canaux pour afficher des millions de couleurs
- Canal alpha permet d'avoir une transparence graduelle (un pixel GIF est transparent ou non)
- Permet l'affichage progressif en 2 dimensions (GIF est vertical seulement)
- Prédit les valeurs d'un pixel selon les valeurs des pixels environnants

9. Analyse d'image

L'analyse d'image a pour but d'extraire des paramètres quantitatifs des images. Ces paramètres peuvent ensuite être soumis à une analyse statistique plus poussée, et/ou être confrontés à d'autres paramètres issus de méthodes d'acquisition complémentaires : tests mécaniques, analyses sensorielles, analyses physico-chimiques...

On distingue différentes familles de méthodes en fonction du type d'image à analyser :

- L'analyse de la morphologie de particules est la démarche qui vient souvent en premier lieu à l'esprit. Elle consiste à déterminer, pour chaque particule identifiée dans l'image, des paramètres décrivant le plus souvent la taille, la forme, ou les statistiques des niveaux de gris de la particule
- Pour des images 3D, l'analyse des particules est un peu plus difficile, notamment du fait qu'il est plus compliqué d'observer entièrement un nombre significatif de particules sur une même image.
- Dans certains cas, la structure d'intérêt se prête mal à une description par un ensemble de particules : les solides alvéolaires comme la mie de pain, les structures ramifiées... Des paramètres géométriques globaux permettent néanmoins une description qui peut s'avérer pertinente
- Un problème particulièrement ardu est la détermination de paramètres 3D quand on ne dispose que d'images de coupes planes 2D. On entre ici dans le champ de la stéréologie, une discipline qui mêle géométrie stochastique et statistiques.

- Enfin, de nombreuses applications fournissent des images dans lesquelles les structures d'intérêt ne peuvent être clairement identifiées. Dans ce cas, les méthodes d'analyse de texture d'image peuvent se révéler fructueuses.

Il peut être utile de rappeler que dans tous les cas, il est nécessaire de disposer de la résolution de l'image. Cette résolution peut être fournie par le dispositif d'acquisition. Dans le cas inverse, l'utilisation de mires de calibration permet de connaître a posteriori la résolution des images.[5]

10. Analyse pour la reconnaissance des formes

Dans un monde de plus en plus riche en informations électroniques et où le développement de la technologie conduit à une explosion de l'information, les solutions automatiques de reconnaissance des formes, permettront d'aboutir à des systèmes d'indexation plus efficaces pour les domaines d'application comme : le contrôle qualité de pièces usinées, la détection et la reconnaissance de visages, l'authentification, la détection, la reconnaissance et le suivi d'objets dans une séquence vidéo, la recherche de formes par le contenu, la lecture automatique des plaques d'immatriculation, la vidéosurveillance…

10.1. Analyse multi échelle de contours de formes planaires

Une méthode originale d'analyse multi échelle basée sur un lissage progressif du contour par réduction de la largeur de bande du filtre. La détection des points d'intersection entre le contour original et les contours filtrés (après une remise à l'échelle de ces derniers) permet de générer la carte des points d'intersection (CPI) qui est caractéristique de la forme.

Dénommée méthode MSGPR (pour Multi-Scale curve smoothing for Generalised Pattern Recognition), cette méthode qui est de la famille des méthodes dites d'espaceéchelle (ou scale-space) permet d'appréhender, à la fois, les caractéristiques spatiales et les caractéristiques d'échelle. Comme dans les autres méthodes de cette famille, le lissage est réalisé à l'aide d'un filtrage passe-bas progressif du contour fermé.[6]

10.2. Description de formes basée sur l'analyse de la silhouette d'objets

Dans ce même contexte, Mingqiang YANG a réalisé une étude d'analyse et de description de formes basée tant sur les contours que sur la silhouette des objets. Dans le cadre de cette étude, il a proposé une méthode originale d'échantillonnage d'un contour en se basant non pas sur la distance entre les points consécutifs mais sur l'aire entre deux points consécutifs et le centre de gravité de la forme. En s'appuyant sur ce nouveau type d'échantillonnage, il a développé un descripteur de formes dénommé Chord Context et basé sur les statistiques des longueurs de segments (ou cordes) dans la forme dans une direction donnée. [6]

11. Conclusion

Dans ce chapitre, nous avons abordé la classification des images. Dans un premier temps, nous avons présenté de manière générale la notion d'image, ses différents types, ses caractéristiques et les traitements qu'on peut effectuer sur elle.

Dans le prochain chapitre, nous décrirons en détail les algorithmes du Machine Learning et Deep Learning. Nous aborderons également les concepts théoriques des réseaux de neurones profonds qui seront employés dans notre système de classification des panneaux de signalisation.

Chapitre 2

Classification de signaux routiers

1. Introduction

La puissance de calcul s'est accrue de façon exponentielle et l'intelligence artificielle a assisté à une "explosion" de techniques de calcul qui n'était pas possible auparavant en plus de la disponibilité des données, tout cela a permis la renaissance du Deep Learning

Dans ce chapitre, nous aborderons les réseaux de neurones profonds. Dans un premier temps, nous présentons la notion de segmentation après les Machine Learning et le Deep Learning. Ensuite, nous expliquons les réseaux de neurones convolutifs (CNN)

2. La segmentation

La segmentation joue un rôle prépondérant dans le traitement d'image, elle est réalisée avant les étapes d'analyse et de prise de décision dans plusieurs processus d'analyse d'image, telle que la détection des objets. Elle facilite la localisation et la délimitation des entités présentes dans l'image

2.1. Définition

La segmentation est un traitement de bas niveau qui consiste à créer une partition de l'image A en sous-ensembles Ri, appelés régions tels qu'aucune région ne soit vide, l'intersection entre deux régions soit vide et l'ensemble des régions recouvre toute l'image. Une région est un ensemble de pixels connexes ayant des propriétés communes qui les différencient des pixels des régions voisines [8]

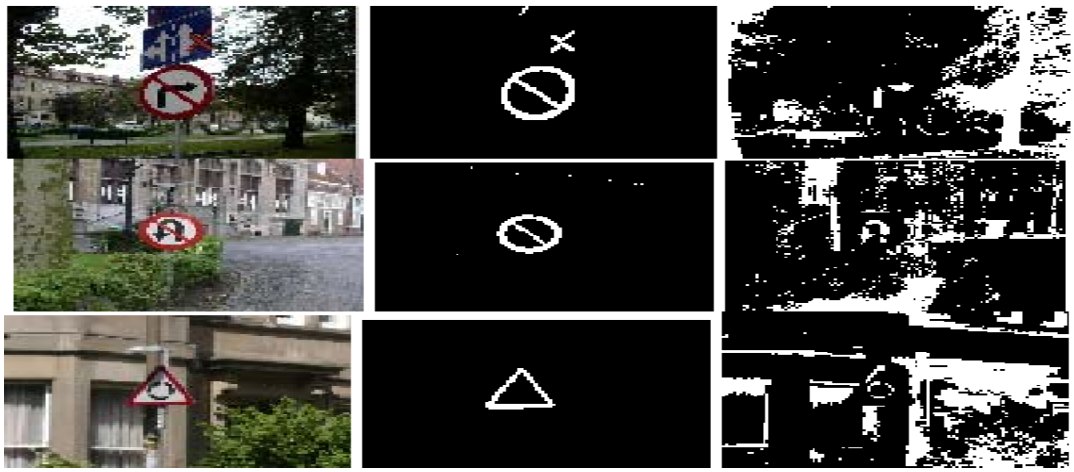


Figure 8. Segmentation d'une image

La figure 8 représente une image en RVB avec sa représentation en NB qui représente une partie de segmentation. Alors la segmentation est la partition d'une image en un ensemble de régions qui ne se chevauchent pas et dont l'union est l'image entière. Quelques règles à suivre pour obtenir une segmentation sont [9] :

- Les régions doivent être uniformes et homogènes par rapport à certaines caractéristiques (niveau de gris, écart type, gradient).
- Leurs intérieurs doivent être simple et sans beaucoup de petits trous (des parties de région non segmentés).
- Les régions adjacentes doivent avoir des valeurs très différentes par rapport à la caractéristique prise en compte dans la segmentation.
- Les limites de chaque région doivent être simples et spatialement précises.

Une technique de segmentation c'est chercher à identifier des régions de pixels homogènes au sein de l'image. Le critère d'homogénéité peut être l'intensité, la couleur, ou même la texture locale. Le résultat se présente soit sous la forme d'une image binaire, soit d'une image étiquetée, chaque étiquette ou label correspondant à une région.[21]

Donc on peut dire que le choix d'une technique de segmentation sera basé sur les critères suivant

- Type d'image
- Contenus de l'image
- But ou objectif à atteindre
- Outils utilisés

2.2. Techniques de segmentation

Il existe plusieurs techniques, et dans le cadre de notre étude en peut citer les techniques suivantes :

1- Technique du seuillage (threshold based methods)

Elle permet d'extraire une partition par la méthode du seuillage basée sur l'histogramme de niveaux de gris dans l'image qui nous permettent de segmenter en *n* classes d'intensité. [11]

Un exemple illustratif : [10]

11	12	11	10
12	14	15	11
10	15	14	12
10	11	10	11

Figure 9. Image original

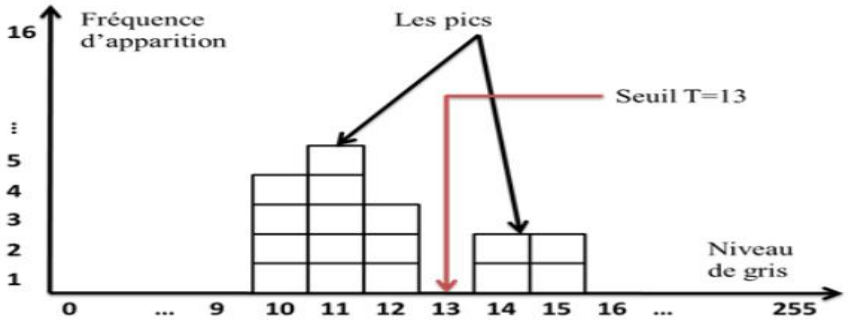


Figure 10. Histogramme de l'image

L'image résultat de la segmentation par seuillage (Seuil T = 13) est une image binaire contenant deux classes. En effet, les pixels qui ont une valeur inférieure à T=13 prennent la valeur 0 et constituent la première classe et les pixels qui ont une valeur supérieure à T=13 sont codés à 1 et constituent la deuxième classe.

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0

Figure 11. Image segmentée par seuillage

2- Technique par croissance de régions (Région growing methods)

C'est une technique locale. L'idée est de choisir un pixel de départ qu'on appelle point d'amorce puis on agrandit les régions autour de ce point, par similarité tant que le regroupement est possible on continue. [12]. Un exemple illustratif : [10]

Figure 12. Exemple segmentation d'image par croissance de régions

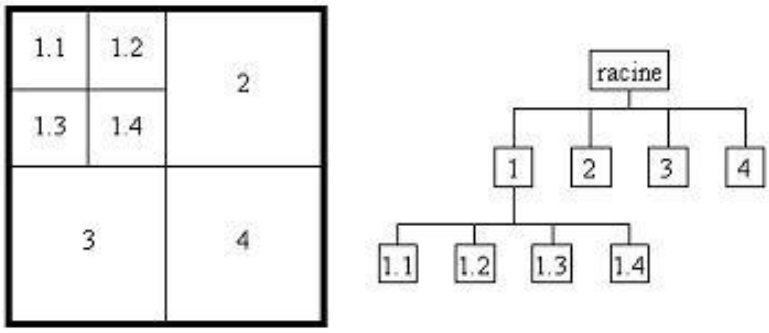


Figure 13. Exemple de segmentation par division-fusion de régions

3- Segmentation par Quadtree

C'est la segmentation par Quadtree qui ne travaille pas sur les pixels mais sur les zones. Un quadtree est une structure de données de type arbre dans laquelle chaque nœud interne a exactement quatre sous nœuds. Cette méthode permet de diviser l'image en quatre zones selon le critère d'homogénéité, à chaque itération la segmentation crée des objets carrés réguliers, où la taille est définie par la variation des objets.[11]

4- Technique basée sur les contours (edge detection based methods)

La segmentation par contours permet de détecter les limites entre les régions de l'image. Plusieurs zones de niveaux de gris différents correspondants aux différents objets de la scène, pour cela il faut trouver une manière de localiser ces objets dans l'image. [11]

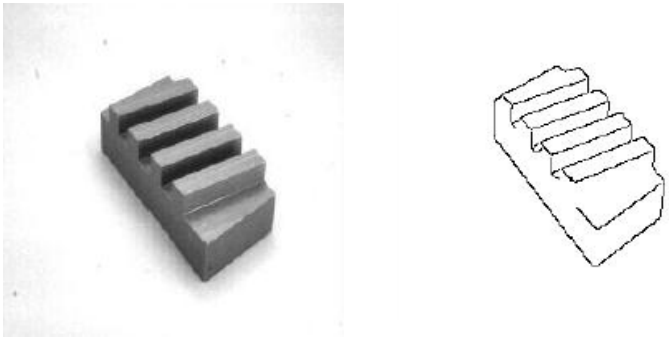


Figure 14. Image originale et image résultat segmentée par contour

Deux grandes approches peuvent être envisagées pour extraire les zones pertinentes des images la technique de détection et fermeture de contours ou la technique de contours déformables ou actifs (snakes).

3. Classification

La classification permet d'affecter une forme, à une ou plusieurs classes prédéfinies ou non, sur la base d'une ressemblance de caractéristiques. Cette affectation est établie via une règle de décision. Les approches de classification se subdivisent en deux grandes catégories [22]

3.1. L'apprentissage automatique (Machine Learning)

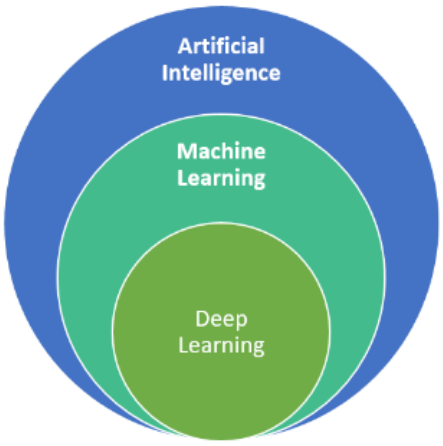


Figure 15. Intelligence artificielle & apprentissage automatique & apprentissage profond

Machine Learning (apprentissage automatique en français) (ML), fait partie de l'intelligence artificielle (IA), c'est la couche intelligente de l'IA qui est de plus en plus émergente et utilisée dans plusieurs domaines.

Si en fait sur les algorithmes d'apprentissage automatique, ils ont exactement comme notre cerveau capable capable d'extraire des informations à partir de données tout en les représentant dans un certain type de modèle permettant ainsi aux applications logicielles de devenir de plus en plus précises dans la prévision des résultats sans être explicitement programmées. Le principe de base est que ces algorithmes puissent recevoir des données d'entrées et utiliser une analyse statistique pour prédire une sortie tout en les mettant à jour à mesure que de nouvelles données deviennent disponibles.

Plusieurs modèles peuvent représenter la structure des données permettant par la suite de faire des prédictions, comme les SVM, Ramdom Forest, k-nearest neighbors...etc

3.2. Apprentissage

L'apprentissage est le noyau de toute la méthode, Il est basé sur des techniques algorithmiques visant à minimiser l'erreur entre les données entré et les résultats obtenus. L'apprentissage a pour but de changer les valeurs des poids jusqu'à ce que l'on ait des sorties ayant des valeurs les plus proches des valeurs entrées

Nous pouvons décrire 2 (deux) grandes catégories d'algorithmes d'apprentissage automatique

3.2.1. Apprentissage Supervisé :

Cette approche suppose que toutes les classes sont connues ou prédéfinies. La problématique consiste à associer la forme testée à la classe la plus adaptée. Pour cela, le classifieur doit effectuer un processus appelé "apprentissage " qui consiste à déterminer les frontières de séparation entre les différentes classes, selon deux approches : classification par séparation et classification par modélisation.

Les différentes approches de classification supervisée [13] :

- Approche par séparation de données **:** Elle consiste plutôt à partager l'espace de décision en plusieurs zones dont les frontières sont définies par des hyperplans. Parmi les approches par séparation, nous citons : les machines à vecteurs de supports (SVM) et les réseaux de neurones artificiels (ANN), ...etc.
- Approche par modélisation de données : Elle est basée sur la construction d'un modèle pour chaque classe. Parmi les approches par modélisation, nous citons : la mixture de gaussiennes (GMM) et le modèle de Markov caché (HMM),...etc.

3.2.2. Apprentissage Non Supervisé :

Dans les méthodes non supervisées, on ne dispose pas de l'ensemble des échantillons d'apprentissage. Le nombre de classes et les règles d'affectation à ces classes doivent être établis seulement à partir d'observations, sans faire référence à un ensemble d'entraînement. Le groupement des individus est réalisé sur la base de similarités et il est généralement conditionné par le choix du nombre de classes. L'utilisateur n'intervient qu'une fois la classification effectuée pour interpréter le contenu des classes, sans faire appel à d'autres hypothèses sur les images ou sur les classes.

On peut distinguer deux types de méthodes de classification non supervisée : les méthodes hiérarchiques et les méthodes de partitionnement.[13]

3.3. Classification des images et l'apprentissage machine

Avec l'évolution de matériel informatique, les méthodes classiques sont devenues très compliqué a implémenté dans le cadre de la classification des images, la reconnaissance des objets et du vocale. Les données se passe en temps réel la qualité du son ou nombres des pixels sont très complexe

Comme il est expliqué dans le premier chapitre, pour une machine une image est une matrice des valeurs indiquant la luminosité de chaque pixel

Donc la question qui se pose est : comment une machine peut-elle identifier une telle classe dans une variétés des objets d'une image quand l'apparence d'un tel panneau de signalisation et des objets qui les entourent peut varier infiniment ?

Avec l'évolution de la technologie, la réaliser un algorithme statique qui fonctionnera de manière robuste dans toutes les situations devient impossible pratiquement, c'est pour quoi ils ont pensé à l'apprentissage machine (auto apprentissage), il est utilisé depuis longtemps pour filtrer les contenus indésirables, ordonner des réponses à une recherche, faire des recommandations, ou sélectionner les informations intéressantes pour chaque utilisateur

Figure 16. System de reconnaissance de signaux routiers

Comme il est illustré dans la figure 16 un système de reconnaissance entraînable peut être vu comme une boite avec une entrée(image), et une sortie qui peut représenter la catégorie de l'image On parle alors de systèmes de classification ou de reconnaissance des images.

Le type le plus de l'apprentissage machine dans le domaine de l'imagerie est supervisé : on fixe en entrée de la machine une photo d'un objet, par exemple une plaque de lime vitesse, et on lui donne la sortie désirée pour une classe. Après chaque itération, la machine ajuste ses paramètres internes de manière à rapprocher sa sortie de la sortie désirée. Après avoir montré à la machine une data set avec des milliers ou des millions d'exemples d'image avec leur catégorie, la machine devient capable de classifier correctement la plupart d'entre eux. Mais l'extraordinaire que l'algorithme peut aussi classifier correctement des images de panneaux de signalisations qu'elle n'a jamais vues durant la phase l'apprentissage. C'est ce qu'on appelle la capacité de généralisation

A l'heure actuelle, les systèmes de reconnaissance des images classiques étaient composés de deux blocs: un extracteur de caractéristiques (feature extractor en anglais), suivi d'un classifieur entraînable simple. L'extracteur de caractéristiques transforme la matrice des valeurs (comme on a expliqué haut de sous) en vecteur de caractéristiques dont chacun indique la présence ou l'absence d'un motif simple dans l'image. Ce vecteur est envoyé au classifieur, dont un type commun est le classifieur linéaire. Ce dernier calcule une somme pondérée des caractéristiques : chaque nombre est multiplié par un poids (positif ou négatif) avant d'être sommé. Si la somme est supérieure à un seuil, la classe est reconnue. Les poids forment une sorte de

«prototype» pour la classe à laquelle le vecteur de caractéristiques est comparé. Les poids sont différents pour les classifieurs de chaque catégorie, et ce sont eux qui sont modifiés lors de l'apprentissage. Les premières méthodes de classification linéaire entraînable datent de la fin des années cinquante et sont toujours largement utilisées aujourd'hui. Elles prennent les doux noms de perceptron ou régression logistique. [14]

3.4. Extraction des caractéristiques basée sur l'apprentissage profond (Deep Learning)

Comme il est illustrés dans la Figure 14 que l'apprentissage automatique ou bien Deep Learning (DL) en anglais, est un sous-domaine de l'apprentissage automatique qu'est basé sur les réseaux neurones multicouches pour aboutir à meilleurs résultats de précision dans une variance des domaines déférents, sa force est de sa apprentissage, extraction et traduction des caractéristiques des données sans l'utilisation d'un codage manuellement ou bien des algorithmes statiques. Dans le cadre de notre étude le DL se devient très efficace pour résoudre des problèmes difficiles tels que l'extraction de caractéristiques à partir d'images de très grande taille contenant un nombre important de pixels facilitant ainsi le processus de classification

3.5. Apprentissage automatique VS Apprentissage profond (justification du choix)

Dans cette partie on va justifier le choix d'attaquer le Deep Learning au lieu du Machine Learning toujours dans le contexte de notre étude

Machine Learning	Deep Learning
L'apprentissage automatique utilise des algorithmes pour analyser les données, apprendre de ces données et prendre des décisions éclairées en fonction de ce qu'il a appris	L'apprentissage profond structure les algorithmes en couches pour créer un réseau neurone capable d'apprendre et de prendre des décisions intelligentes par lui-même
Peut s'entraîner avec moins de données d'entraînement	Peut Nécessite de grands ensembles de données pour la formation
Prend moins de temps pour s'entraîner	Prend plus de temps pour s'entraîner
Trains sur CPU	S'entraîne sur GPU pour une formation appropriée
La sortie est sous forme numérique pour les applications de classification et de notation	La sortie peut être sous n'importe quelle forme, y compris des éléments de forme libre tels que du texte et du son libres
Capacité de réglage limitée pour le réglage des hyperparamètres	Peut être réglé de différentes manières
Besoin de comprendre les fonctionnalités qui représentent les données	Pas besoin de comprendre la meilleure caractéristique qui représente les données

Tableau 6. Comparaison générale entre DL & ML

Processus

3.5.1. Machine Learning

Dans le cadre de notre étude on vas créer un programme qui reconnaît les panneaux de signalisation . Pour entraîner le modèle, nous utilisons un classificateur. Un classificateur utilise les caractéristiques d'une plaque routière pour essayer d'identifier la classe à laquelle il appartient.

Dans l'exemple suivant, le classificateur sera formé pour détecter si l'image est:

- Limite de vitesse (20 km / h)
- Limite de vitesse (60 km / h)
- Dépassement Interdit
- Courbe dangereuse à gauche

Les quatre classes ci-dessus sont la classe que le classificateur doit reconnaître. Pour construire un classificateur, il faut avoir des données en entrée et lui attribuer une étiquette. L'algorithme prendra ces données, trouvera un modèle et le classera ensuite dans la classe correspondante.

Cette tâche est appelée apprentissage supervisé. Dans l'apprentissage supervisé, les données d'entraînement que vous transmettez à l'algorithme comprennent une étiquette.

La construction d'un algorithme nécessite de suivre quelques étapes standard:

- Collectez les données
- Construire le classificateur
- Faire des prédictions

La première étape est nécessaire, choisir les bonnes données fera réussir ou échouer l'algorithme. Les données que nous avons les choisir pour entraîner notre modèle sont appelées une fonction.

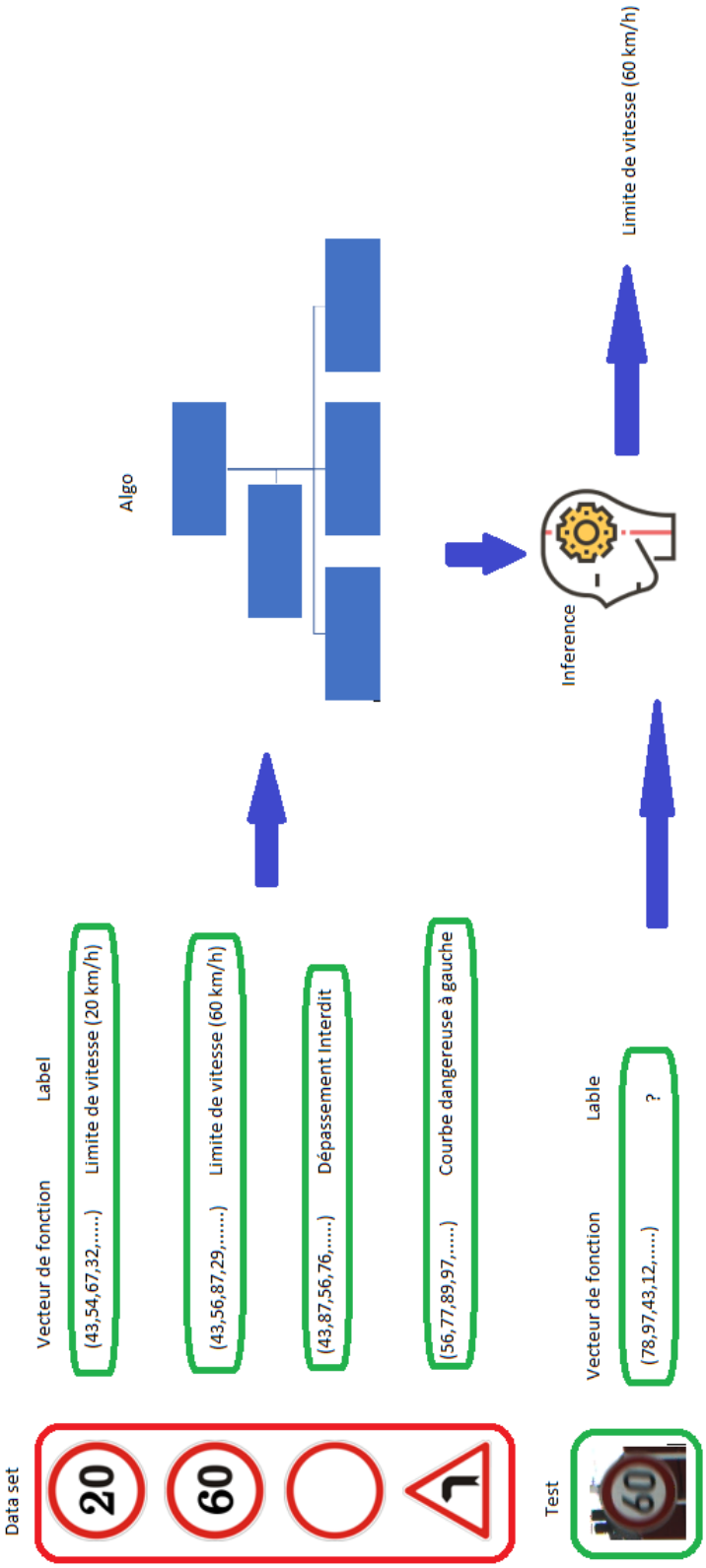


Figure 17. Processus Machine Learning

L'objectif est d'utiliser ces données d'apprentissage pour classer le type d'objet. La première étape consiste à créer les colonnes de caractéristiques. Ensuite, la deuxième étape consiste à choisir un algorithme pour entraîner le modèle. Une fois l'apprentissage terminé, le modèle prédira quelle image correspond à quel objet.

Après cela, il est facile d'utiliser le modèle pour prédire de nouvelles images. Pour chaque nouvelle image alimentée dans le modèle, la machine prédira la classe à laquelle elle appartient. Par exemple, une image entièrement nouvelle sans étiquette traverse le modèle. Pour un être humain, il est trivial de visualiser l'image comme une voiture. La machine utilise ses connaissances antérieures pour prédire également que l'image est une voiture.

3.5.2. Dee Learning

En apprentissage profond, la phase d'apprentissage se fait via un réseau de neurones. Un réseau de neurones est une architecture où les couches sont empilées les unes sur les autres.

Considérez le même exemple d'image ci-dessus. L'ensemble de formation serait alimenté à un réseau de neurones

Chaque entrée va dans un neurone et est multipliée par un poids. Le résultat de la multiplication passe à la couche suivante et devient l'entrée. Ce processus est répété pour chaque couche du réseau. La couche finale est appelée la couche de sortie; il fournit une valeur réelle pour la tâche de régression et une probabilité de chaque classe pour la tâche de classification. Le réseau de neurones utilise un algorithme mathématique pour mettre à jour les poids de tous les neurones. Le réseau de neurones est entièrement entraîné lorsque la valeur des poids donne une sortie proche de la réalité. Par exemple, un réseau neuronal bien formé peut reconnaître l'objet sur une image avec une plus grande précision que le réseau neuronal traditionnel.

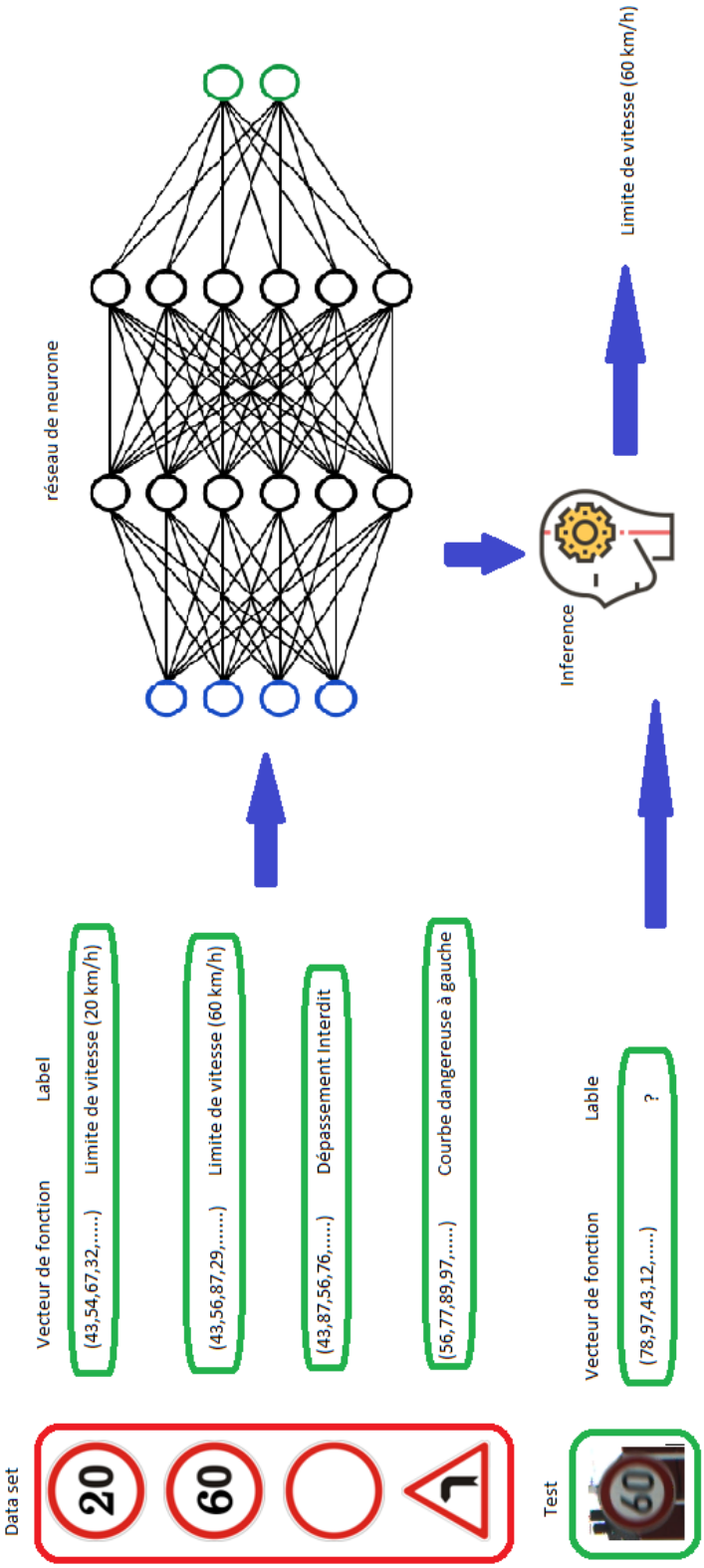


Figure 18. Processus Deep Learning

4. Les réseaux neurones artificiels

4.1. Définition

Les travaux exercés sur les réseaux de neurones artificiels ont été dérivé du fonctionnement de notre cerveau humain grâce à sa capacité de capacité de calculs et d'organisation des données et des instructions en utilisant les neurones.

On s'inspire de notre cerveau humain

De manière formelle, un réseau de neurones est un système composé de plusieurs éléments de calcul simples fonctionnant en parallèle, reliés entre eux par des opérations de somme. Pour distinguer cette approche par rapport à un réseau biologique, nous utilisons le terme de réseau de neurones artificiel.[8]

Un réseau de neurone ne se programme pas, il est entraîné grâce à un mécanisme d'apprentissage. Le premier modèle de réseau de neurone proposé est le perceptron monocouche. [15]

Les réseaux de neurones ont de nombreuses applications dans des domaines très variés :

- Traitement des images.
- Identification des signatures.
- Reconnaissance des caractères (dactylos ou manuscrits)
- Reconnaissance de la parole.
- Reconnaissance de signaux acoustiques (bruits sous- marins, ...).
- Extraction d'un signal du bruit.
- Contrôle de systèmes asservis non-linéaires (non modélisables).
- Robotique (apprentissage de tâches).
- Aide à la décision (domaine médical, bancaire, management, ...)

4.2. Model d'un neurone

Un neurone est une unité de traitement de l'information fondamentale pour le fonctionnement d'un réseau neuronal. Le schéma de principe de la figure 17 montre le modèle d'un neurone, qui forme la base pour la conception d'une grande famille de réseaux de neurones étudiés dans les chapitres suivants. Ici, nous identifions trois éléments de base du modèle neuronal [15] :

1- Un ensemble de synapses, ou liens de connexion, dont chacun est caractérisé par un poids ou une force qui lui est propre. Plus précisément, un signal xj à l'entrée de la synapse j connecté au neurone k est multiplié par le poids synaptique wkj. Il est important de noter la manière dont les indices du poids synaptique wkj sont écrits. Le premier indice de wkj se réfère au neurone en question, et le second indice se réfère à l'extrémité d'entrée de la synapse à laquelle le poids se réfère. Contrairement au poids d'une synapse dans le cerveau, le poids synaptique d'un neurone artificiel peut se situer dans une plage qui comprend des valeurs aussi bien négatives que positives.

2- Un additionneur pour additionner les signaux d'entrée, pondérés par les forces synaptiques respectives du neurone, les opérations décrites ici constituent un combineur linéaire.

3- Une fonction d'activation pour limiter l'amplitude de la sortie d'un neurone. La fonction d'activation est également appelée fonction d'écrasement, en ce qu'elle écrase (limite) la plage d'amplitudes admissible du signal de sortie à une valeur finie.

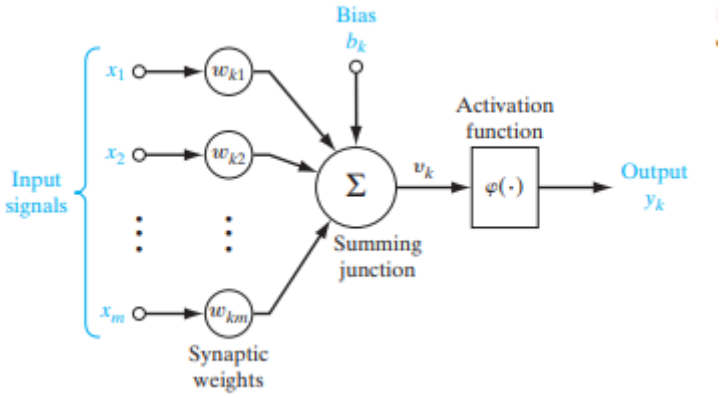


Figure 19. Un modèle d'un neurone

5. Réseaux de neurones convolutifs (CNN)

Le réseau de neurones convolutif de l'anglais convolutional neural networks (CNN) est une évolution, et qu'est une classe de réseaux neuronaux d'apprentissage profond. Ils sont directement inspirés par des processus biologiques par les travaux de Hubel et Wiesel en 1968 sur le cortex visuel des mammifères. [16] Ils sont conçus pour la classification supervisée des images et de vidéos, et de systèmes de recommandation au traitement du langage naturel.

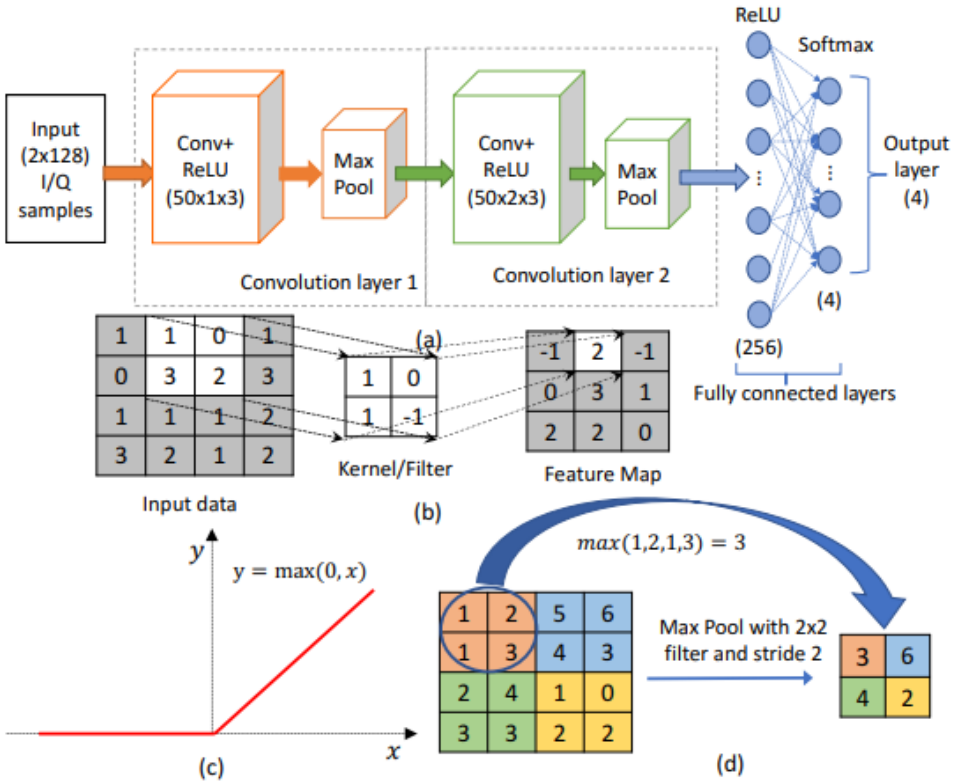


Figure 20. Réseau de neurones convolutif [18]

5.1. Composant d'un CNN

Yann Le Cun et Yoshua Bengio, ont introduit une architecture des réseaux de neurones convolutionnels de base [17] qui permet de résoudre certaines difficultés en limitant le nombre de paramètres à apprendre.

Une architecture CNN est formée par une couche d'entrée, une couche de sortie et des couches cachées. Les couches cachées sont généralement un ensemble de couches de traitement indépendantes :

- La couche de convolution (CONV) qui traite les données d'un champ récepteur ;
- La couche de pooling (POOL) aussi appelée couche de regroupement ;
- La couche de correction (ReLU), souvent appelée par "ReLU" en référence à la fonction d'activation (Unité de rectification linéaire) ;
- La couche "entièrement connectée (FC)", qui est une couche de type perceptron.

5.1.1. Couche de convolution

La première couche dans les CNN, considérée comme son cœur, est l'étape la plus importante.

Un ensemble de filtres de convolution peut être combiné pour former une couche de convolution (convolutional layer) d'un réseau de neurone. Les

neurones de la couche de convolution ne sont connectés qu'à une région d'entrée et ont généralement les mêmes paramètres. La sortie de la couche est généralement décrite comme un volume dont la taille et le poids dépendent des dimensions de la carte d'activation ou la carte de convolution (feature map).

Les cartes d'activation sont des résultats de convolution de chaque filtre avec les images d'entrées. Pour chaque paire (image/filtre) on en obtient une qui nous indique où se situent les caractéristiques dans l'image. Chaque caractéristique est une mini-image (tableauau de 2D) qui rassemble les aspects les plus communs des images.

Les entrées des couches de convolution sont soit des images, dans le cas du premier réseau, soit des cartes d'activations résultantes des couches précédentes. Les filtres sont généralement carrés et leur largeur peut s'étendre de 3 à N pixels ce qui engendre des cartes d'activation de taille différente de celle de l'entrée. [18] Dans la pratique, les filtres d'un réseau de neurones convolutif ont un noyau (kernel en anglais), un pas (stride) et un padding définis à l'avance. Leurs valeurs sont générées aléatoirement à l'initialisation. Ensuite, lorsque le réseau est en phase d'apprentissage, les valeurs des filtres seront mises à jour pour améliorer les résultats du CNN : les valeurs des filtres font donc parties des variables (poids, biais..) que le réseau change en apprenant.

Trois paramètres permettent de dimensionner le volume de la couche de convolution la profondeur, le pas et la marge.

- La profondeur de la couche est le nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur).
- Le pas (stride) 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 zéro ou zero-padding est la taille du zero-padding.

Au final, les cartes de convolutions sont mises à plat et concaténées en un long vecteur qui comprend les caractéristiques les plus pertinentes de l'image, appelé code CNN.

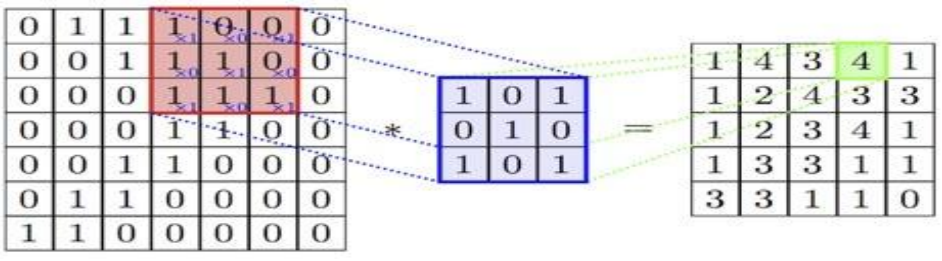


Figure 21. Couche de convolution

5.1.2. Couche de regroupement

Les couches de regroupement sont généralement placées après les couches de convolution. Elles ne jouent pas de rôle dans le processus d'apprentissage, par contre, elles servent à réduire le nombre de paramètres à apprendre. En effet, les couches profondes du réseau ont besoin de moins d'informations à propos de la localisation exacte des caractéristiques spatiales mais requièrent plus de matrices de filtres pour reconnaître d'avantage de modèles de haut niveau. L'entrée de ces couches est divisée en plusieurs éléments rectangulaires qui ne se chevauchent pas et les unités des éléments sont utilisées pour créer une seule unité de sortie. C'est une opération de compression qui garde les informations les plus importantes. Le type d'opération effectuée sur chaque élément détermine le type de couche de regroupement. Dans la plupart des cas, cette opération consiste à faire glisser un filtre pas à pas sur les données en entrée et récupérer certaines valeurs de l'image en prenant la valeur maximale. C'est l'opération de regroupement la plus courante. Dans ce cas, la couche est appelée Max-Pooling. La deuxième opération est l'Average-Polling, qui consiste à récupérer les valeurs en prenant une moyenne. Après avoir procédé au Pooling[19], l'image n'a plus qu'un quart du nombre de ses pixels de départ. La sortie aura le même nombre d'images mais un nombre inférieur de pixels. Ainsi, le CNN peut trouver si une caractéristique est dans une image sans se soucier de l'endroit où elle se trouve.

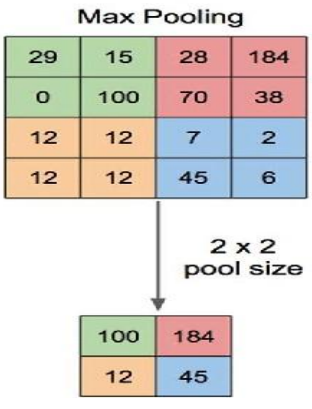
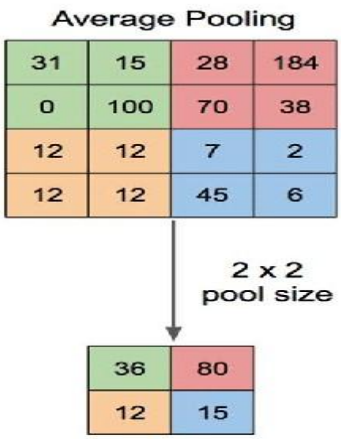
Type	**Max pooling**	**Average pooling**
But	Chaque opération de pooling sélectionne la valeur maximale de la surface.	Chaque opération de pooling sélectionne la valeur moyenne de la surface.
Illustration		

Tableau 7. Comparaison entre le max pooling et l'average pooling

5.1.3. Couche de correction

L'application des filtres de convolution génère souvent des intensités négatives. Pour les supprimer, une fonction d'activation (fonction mathématique appliquée à un signal en sortie d'un neurone artificiel). Est utilisée qui permet de régulariser la sortie de la convolution, de manière à pouvoir éliminer les valeurs négatives selon une règle de seuillage.

Cette fonction est basée sur l'unité linéaire de rectification (appelée RELU) qui permet de rectifier les caractéristiques et ne laisse passer que les valeurs non-négatives. Cette fonction permet ainsi de rectifier la linéarité des données non-linéaires. [18]

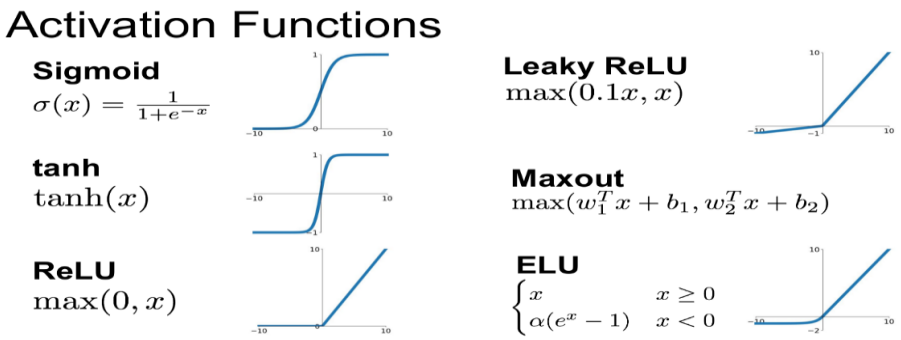


Figure 22. Fonction d'activation [19]

5.1.4. Couche entièrement connectée

Une couche entièrement connectée, fully-connected layer(FCL) est un perceptron multicouche traditionnel. Il utilise un classificateur dans la couche de sortie pour classer l'image d'entrée en fonction des données d'apprentissage. La FCL est la dernière couche et son rôle est de transformer les cartes d'activations en probabilités. Si l'on cherche à prédire 6 catégories, la FCL contiendra 6 neurones.

Un réseau de convolution qui n'inclut aucune couche entièrement connectée est appelé réseau entièrement convolutionnel (Fully Convolutional Network FCN). [20]

6. Conclusion

Dans ce chapitre, nous avons présenté les concepts de Machine Learning et le Deep learning et nous avons également parlé sur les réseaux de neurones et les types les plus importants utilisés dans la classification des images. L'un de ces réseaux, le CNN qui implémente l'idée de partage des poids qui permettant de réduire beaucoup de nombre de paramètres, aussi le temps de calcul, l'espace mémoire nécessaire, et également d'améliorer les capacités de généralisation du réseau.

Dans le prochain chapitre, nous allons faire la conception de notre système de reconnaissance des plaques de signalisation basé sur le CNN.

Chapitre 3

Implémentation

1. Introduction

Dans ce chapitre on a proposé un système de reconnaissance des images de panneaux de signalisation entrainée sur la base de données 'GTSRB' en utilisant les CNN pour une classification non supervisée. Ainsi, la première partie ce chapitre est consacré à la présentation de la méthode de classification des panneaux de signalisation par les réseaux de neurones profonds convolutionnel (CNN). Nous exposons d'abord l'architecture de notre système. Et le reste du chapitre, il est consacré pour tester les différentes implémentations de cette conception puis nous allons tester les performances du système de reconnaissance développé et les exprimer sous forme des graphes

On parle aussi des bases de données utilisées, ainsi que l'environnement du développement employés. Enfin, nous commenterons les résultats et établirons une conclusion.

2. Environnement

Dans cette section en vas aborder l'approche de notre system ainsi que l'architecture du réseau de neurones et l'algorithme utilisé dans cette approche sur la base de données 'GTSRB'

2.1. Classification des images par un réseau de neurones profond convolutionnel

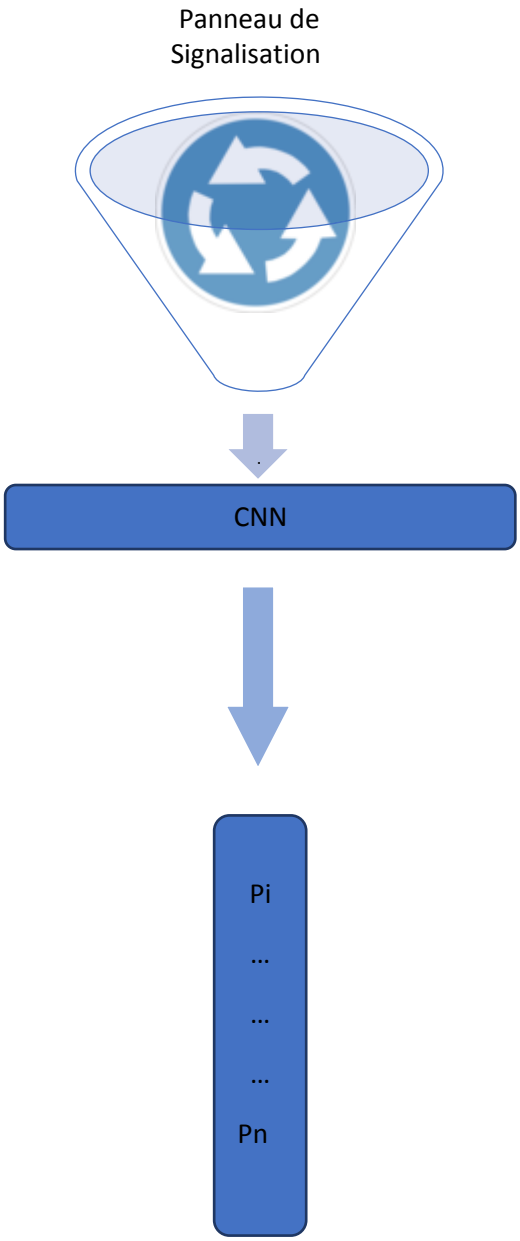
CNN reçoit des images en entrée, puis il décide quelles caractéristiques, d'un ensemble de données, peuvent être utilisées comme indicateurs pour étiqueter ces images de manière fiable. Ensuite il fait correspondre ces caractéristiques extraites à des classes établies au préalable.

Un CNN fonctionne en deux étapes :

La première étape consiste en la génération de caractéristiques qui visent à extraire les informations pertinentes de l'image représentées sous forme d'un lot caractéristiques.

La deuxième étape est la classification qui consiste à attribuer à chaque image d'entrée une classe en sortie suivant une règle de décision.

La figure 23 représente la première approche de notre proposé qui sera détaillée dans la section Algorithme dans la partie expérimentation



P_i est le panneau de Signalisation d'indice i avec i=1,…, n désigne le nombre de type de panneaux de signalisation

Figure 23. Première approche par CNN

2.2. Base de données

La GTSRB « German Trafic Sign Recognition Base » est une base de données qui fut créé par l'université de Frankfurt, en partenariat avec le ministère de la santé allemande, dans le cadre d'une compétition internationale organisée en 2011[24].

La base d'apprentissage est constituée de 39 209 images en RVB, réparties sur 43 classes comme il est illustré dans la Figure 24 représentant un panneau de signalisation donné. Chaque classe possède plusieurs collections d'images prisent dans de différentes vues. Une collection comprend 30 images du même, dont chaque collection contient 30 images prisent dans une seule seconde. Leurs tailles varient entre 15x15px et 250x250px et elles ne sont pas de forme carrée, chaque image dispose de son étiquette.

Le nombre d'images par classes est variable entre 220 pour la classe la moins représentée jusqu'à 2260 images pour la classe la plus représentée, il faut donc prendre en considération ce déséquilibre des données. Les images sont capturées dans des conditions de luminosité différentes à l'aide d'une caméra de faible résolution située à bord d'un véhicule en mouvement.

Lors de la compétition organisée en 2011[24], les participant ne disposaient que la base d'apprentissage, puis la base de test fut mise à disposition de la communauté scientifique.

Le choix de cette base de données est basé sur le code de la route de l'Allemagne qu'est similaire de l'Algérie

Figure 24. Les 43 classes en total du GTSRB

2.3. Outils de développement

2.3.1. Langage de programmation Python

Python est un langage de programmation qui fut lancé le 20 février 1991 par Guido van Rossum, il s'agit d'un langage interprété, c'est-à-dire que l'exécution du programme se fait ligne par ligne, et il peut être utilisé en programmation orientée objet et en programmation procédurale. [25]

Le langage est très populaire dans la réalisation de solutions qui impliquent l'apprentissage profond, avec un nombre très importants de modules et d'API à disposition.

Le langage est disponible en version 2.x et 3.x, pour les tests réalisés dans ce travail, nous avons employés la version 3.8.5 de l'interpréteur python.

2.3.2. Environnement de développement 'Spyder'

Spyder est un environnement de développement informatique qui contient un éditeur de texte, un interpréteur python et un débogueur. Il s'agit d'une solution tout en un pour tout travail réalisé sous python. [26]

2.3.3. Module TensorFlow

TensorFlow est une bibliothèque en libre accès (open source) pour le calcul numérique à base de flots de données. A l'origine, elle a été développée par des chercheurs et ingénieurs travaillant au sein de « Google Brain Team » pour la recherche sur l'apprentissage machine et l'apprentissage profond.

TensorFlow peut utiliser en parallèle la puissance du processeur CPU ainsi que celle du processeur de la carte graphique (GPU) afin d'accélérer les calculs intensifs. [27]

2.3.4. API Keras

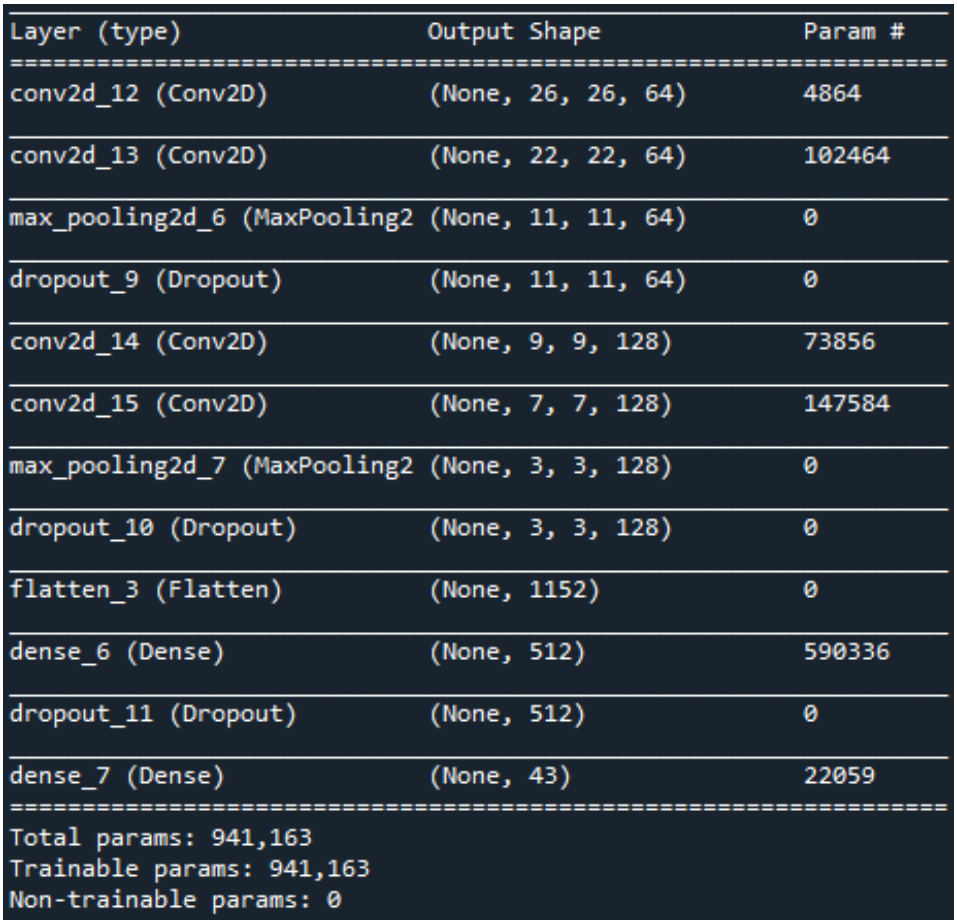
Keras est une API de haut niveau, programmée en python et qui peut utiliser de différents modules de réseau de neurones, tel que Tensorflow et Theano.Elle a été développée principalement pour les chercheurs suivant la philosophie de « Being able to go from idea to result with the least possible delay is key to doing good research ». Keras simplifie l'utilisation de Tensorflow pour l'utilisateur, en offrant une API simple et intuitive d'utilisation. [28]

3. Expérimentations

3.1. Architecture

Les CNN possèdent deux phases :

- La première phase est la phase convolutive qui fonctionne comme extracteur de caractéristiques pertinentes des images d'entrées. Les données extraites seront sous forme d'une matrice qu'on appelle "**features map**".
- La deuxième phase est la partie de couche de classification. Dans cette phase les caractéristiques qui sont extraites seront aplaties et concaténées dans un vecteur, qui passera par les couches suivantes. Ces dernières prennent la décision dans quelle classe les **features map** appartiennent en donnant un ensemble de **score de classe** et le **Loss** montre la différence entre le signal prévu et réel. [23]



Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 26, 26, 64)	4864
conv2d_13 (Conv2D)	(None, 22, 22, 64)	102464
max_pooling2d_6 (MaxPooling2	(None, 11, 11, 64)	0
dropout_9 (Dropout)	(None, 11, 11, 64)	0
conv2d_14 (Conv2D)	(None, 9, 9, 128)	73856
conv2d_15 (Conv2D)	(None, 7, 7, 128)	147584
max_pooling2d_7 (MaxPooling2	(None, 3, 3, 128)	0
dropout_10 (Dropout)	(None, 3, 3, 128)	0
flatten_3 (Flatten)	(None, 1152)	0
dense_6 (Dense)	(None, 512)	590336
dropout_11 (Dropout)	(None, 512)	0
dense_7 (Dense)	(None, 43)	22059

Total params: 941,163
Trainable params: 941,163
Non-trainable params: 0

Figure 25. Architecture de notre CNN

La figure 25 représente une capture (consol) de l'architecture de notre CNN, nous allons construire un réseau de neurones profond (son utilisé des modules alternatives) qui est entrainé sur notre base d'images de panneaux de signalisation. Ce réseau est constitué des couches suivantes :

quatre couches de convolution,

deux couches de max pooling

deux couches fully connected.

Chaque image en entrée de ce réseau est redimensionnée en taille 30*30 px. D'abord cette image passe par les deux couches de convolution. Pour chacune des couches, dans chaque scenario on utilise défirent nombre de filtres de taille 5*5 avec un zéro padding et un stride égale à 1. Ces couches sont suivies par une fonction d'activation de type Relu qui force les neurones à retourner des valeurs positives. On obtiendra à la sortie de la deuxième couche le même nombre utilisé déjà de features maps (nombre de filtres).

Ensuite, on utilisons une couche de max pooling de taille 2*2, afin de réduire la taille des features maps et ses paramètres. Nous aurons en sortie la valeur présidente features map de taille 12*12 à la sortie. Les deux dernières couches de convolution ont des filtres de taille 30*30. On obtiendra à la sortie de la dernière couche de max pooling un nombre features maps de tailles 4*4 selon le scénario implémenté . Ces features maps sont aplaties à un vecteur 1D de taille 480, qui sera introduit par la suite aux 2 couches fully connected où la première couche est constituée de 500 neurones, elle est suivie d'une fonction d'activation de type Relu car elle supprime les pixels de valeurs négative, et grâce a cette suppression cette fonction permet d'accélérer le processus. Ensuite nous appliquons dropout avec une probabilité 50% pour éteindre les neurones aléatoirement pour éviter le problème de overfitting. Ainsi, avec moins de neurones, le réseau est plus réactif et peut donc apprendre plus rapidement. Tandis que la dernière couche fully connected est constituée de 43 neurones (nombre de classes dans la base d'images) avec une fonction d'activation softmax qui permet de calculer les probabilités d'appartenances à chacune des classes.

3.2. Optimisation

On a utilisé l'algorithme d'optimisation Adam , cette dernière on peut la considéré comme combinaison de la descente de gradient stochastique et de la propagation quadratique moyenne. Les taux d'apprentissage individuels sont calculé pour chaque paramètre différent car il adopte une méthode adaptative d'apprentissage. Adam a utilisé des estimations de premier et deuxième moments de gradient pour adapter le taux d'apprentissage à chaque poids du réseau de neurones. Le nième moment d'une distribution pour un nombre est donné comme la valeur attendue de cette variable élevée à la puissance n. [29]

3.3. Algorithme

Dans le cadre de notre étude nous avons utilisé deux algorithmes
1- ALGORITHEM 1 CNN : Entrainement et construction du model
2- ALGORITHEM 2 TEST : test des modèles a la place d'un capteur réel

La figure 26 représente un diagramme de block qui résume les étapes d'algorithme CNN (consulter le pseudocode dans la partie Annex) , notre algorithme prend les images de la base de données comme entrée pour passer vers le prétraitement ou il les redimensionner, après cette étape on commence l'extraction des caractéristiques (les éléments qu'ils vas être prédicté) en passant par un nombre définit des filtres, et avant de générer le model il faut divisé les résultats de l'extraction des caractéristiques en deux listes une pour l'entrainement et l'autre pour les tests, en fin une effectuer des tests de classification sur notre model

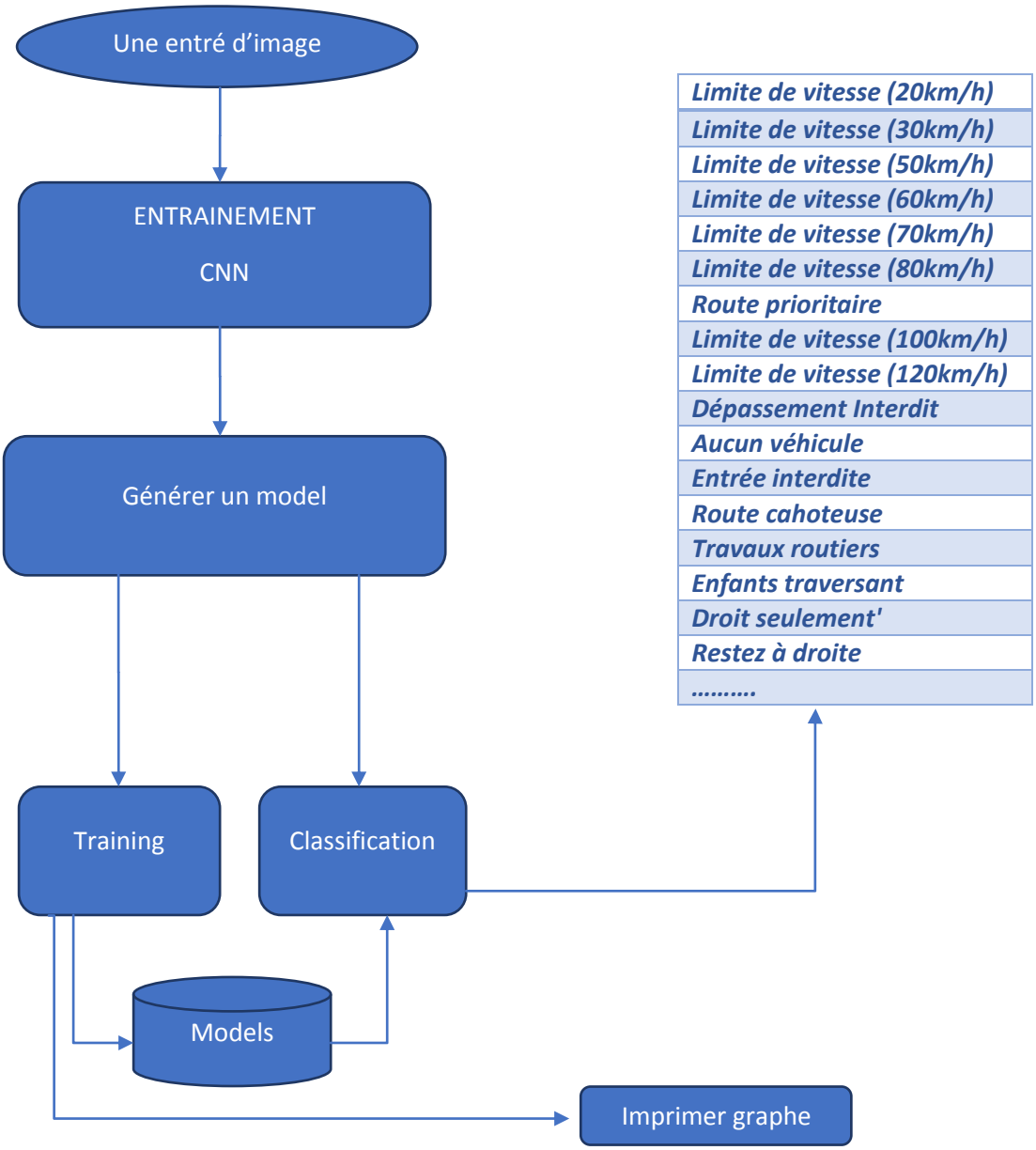


Figure 26. Diagramme de block de notre algorithme CNN

3.4. Configuration

Pour exécuter l'algorithme de la section présidente on a utilisé une machine avec les caractéristiques résumées dans le tableau 8

Module	Caractéristique
Machine	DELL
CPU	Intel(R) core(TM) i5-5200 2,20 GHz
GPU	NVIDIA GeForce 920M 4Go
RAM	8 Go DD3

Tableau 8. Caractéristiques de la machine

3.5. Scenarios

Durant la réalisation de notre system, nous avons exécuté 14 scenarios résumé dans le tableau 9

	Epoch	Filtre 1	Karnel size 1	Pool size	Filtre 2	Karnel size 2	Batch size	Dense 1	Dense 2	loss
01	15	32	(5,5)	(2,2)	64	(3, 3)	32	256	43	crossentropy
02	20	32	(5,5)	(2,2)	64	(3, 3)	32	256	43	crossentropy
03	30	32	(5,5)	(2,2)	64	(3, 3)	32	256	43	crossentropy
04	30	32	(5,5)	(2,2)	64	(3, 3)	64	256	43	crossentropy
05	30	32	(5,5)	(2,2)	64	(3, 3)	08	256	43	crossentropy
06	30	32	(5,5)	(2,2)	64	(3, 3)	128	256	43	crossentropy
07	30	32	(5,5)	(2,2)	64	(3, 3)	256	256	43	crossentropy
08	30	32	(5,5)	(2,2)	64	(3, 3)	512	256	43	crossentropy
09	30	32	(5,5)	(2,2)	64	(3, 3)	1024	256	43	crossentropy
10	40	32	(5,5)	(2,2)	64	(3, 3)	512	256	43	crossentropy
11	40	32	(5,5)	(2,2)	64	(3, 3)	256	256	43	mse
12	40	64	(5,5)	(2,2)	128	(3, 3)	256	256	43	crossentropy
13	40	64	(5,5)	(2,2)	128	(3, 3)	512	512	43	crossentropy
14	49	64	(5,5)	(2,2)	128	(3, 3)	512	512	43	crossentropy

Tableau 9. Ensemble des scénarios exécutés

D'après les définitions et les explications mentionné dans le deuxième chapitre on concentrer durant l'exécution des scenarios sur les paramètres suivants

- Epoch : nombre d'itération du notre algorithme (entrainement sur la base de données), avec une epoch chaque sample (ligne de caractéristique sur la base de données) a la probabilité de faire une mise a jour pour les paramètres interne de notre modèle
- Filtre : on peut considérer un filtre comme une fenêtre de scan d'une seule image, il est utilisé pour détecter les caractéristiques d'un seul pixel par ex est ce qu'il appartient à un arc ou bien une ligne ? donc en peut considérer le filtre comme un détecteur des caractéristiques
- Karnel : c'est scanner l'image région par région, donc le Karnel représente la taille d'une région
- Pool : c'est de la couche 'Pooling layer' , il est utilisé pour réduire la taille de l'image
- Batch : la taille du Batch est un hyperparamètre qui définis le nombre des des lignes des caractéristiques (samples) que sont déjà dans la procédure d'entrainement avant de faire la mise a jours des paramètres (comme il est expliqué dans le paramètre 'epoch'),on peut considéré ce lot (batch) comme une itération d'une boucle 'For' pour chaque ensembles des caractéristiques (samples) et de produire des prédictions, ou dans chaque fin d'itération les prédictions sont comparées par la variable de la variable de sortie et il produit aussi une calcule d'erreur (consulter les résultats de débogage dans la Annex) , cette dernière nous aide d'améliorer notre model a chaque fois, pour le choix des valeurs pour ce hyperparamètre et le paramètre d'epoch il n'existe pas une régle a suivre il faut

test des différents valeurs, car le contexte se change selon le domaine de l'application et la problématique a résoudre. Par exemple dans le Scénario n°01 la taille du lot est de 32 et le nombre d'epoch et de 15 et notre base de données contient 39209 samples cela implique que notre base sera divisée on 1226 lots avec 32 sample dans chaque lot, et le pois sera maitre à jour apres chaque lot d'une taille de 32 donc en total 1226 mise a jours et avec 15 epoch sera 18390 mises à jour

- Dense : La couche Dense consiste à la réception les sorties (résultats) des couches présidentes, et la sortie de cette couche est de la forme d'un lot d'une taille représenté par le paramètre 'Unists'
- Loss : Généralement avec les réseaux de neurones on veut toujours de minimiser l'erreur, et ce paramètre représente la fonction du cout qui calcul cette valeur de cout d'un modèle, le choix de cette fonction sera selon le type de la problématique, dans le contexte de notre problématique de la classification on va utiliser une fonction très connue dans ce domaine la fonction 'categorical_crossentropy' et une autre fonction 'mse' pour voire la différence

Dans le module TensorFlow les fonctions :

categorical_crossentropy : Calcule la perte d'entropie croisée entre les étiquettes (labels, consulter le pseudo code dans l'Annex de la mémoire) et les prédictions

mse : calcule l'erreur moyenne entre les étiquettes et les prédictions

4. Résultats

Comme il est illustré dans la figure 14 qu'après l'exécution de notre scripte de chaque scénario, un graphe sera généré avec un x-axis pour le temps et les valeurs du couts(loss) et précision (accuracy) du model entrainé sont sur y-axis. Ces graphes sont appelés 'courbes d'apprentissage'

Ces courbes nous aident de diagnostiquer nos modèles

Dans cette section on va présenter des résultats différents des scenarios 1,4,5,11, et 14, et Le reste sera représenter dans l'Annex de cette mémoire

Le tableau 10 représente les résultats des scénarios 1,4,5,11, et 14

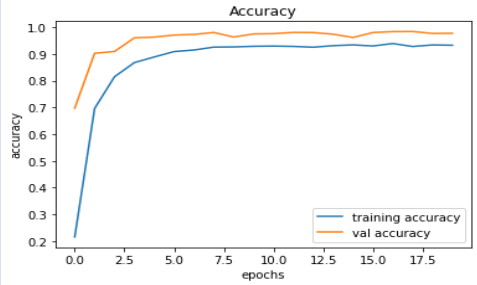
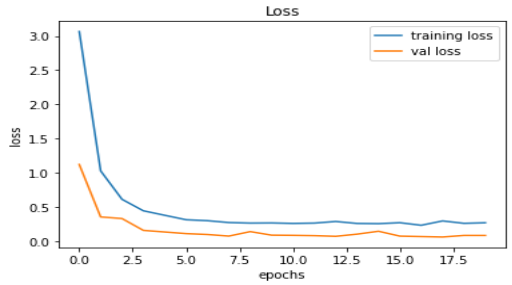
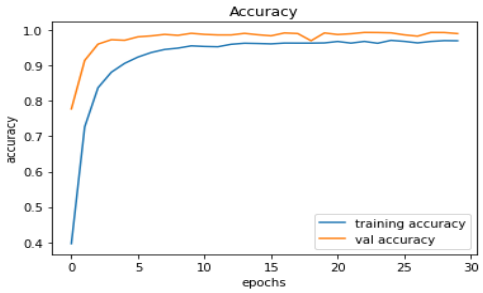
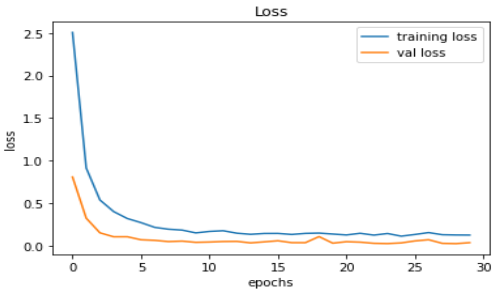
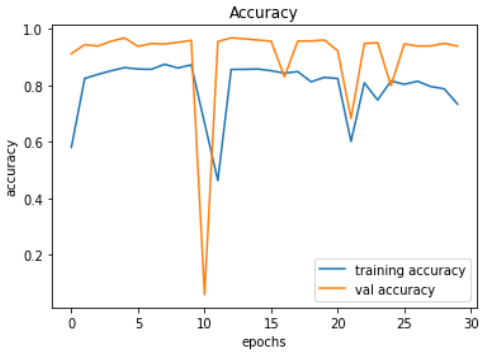
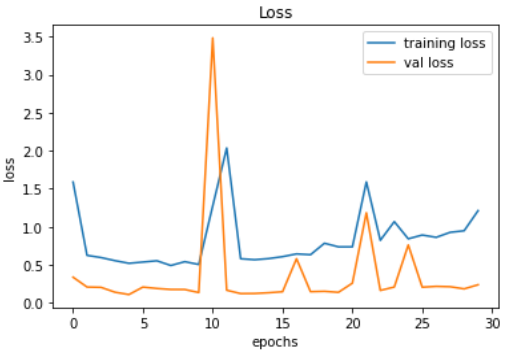
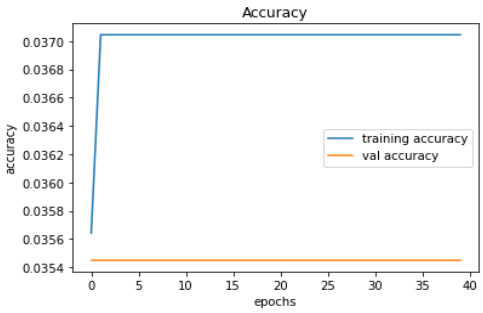
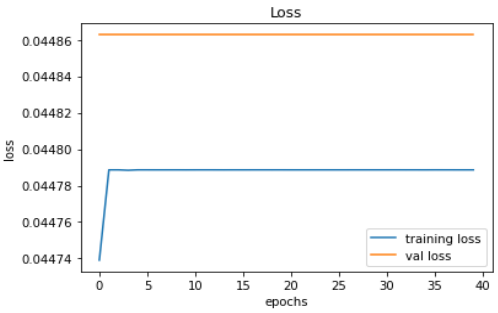
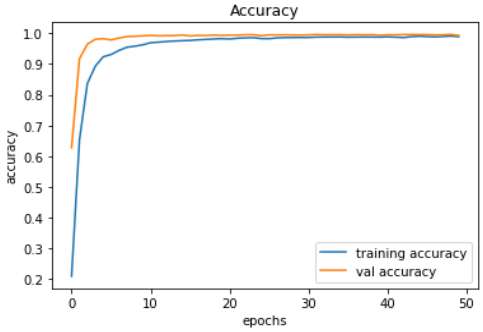
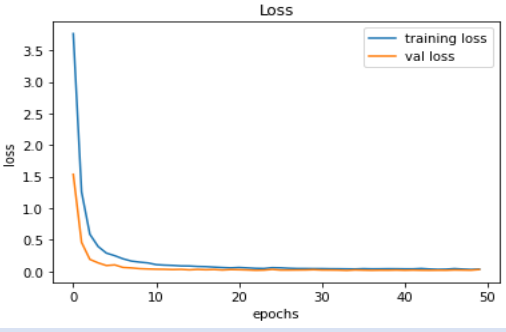
	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
1	0.9329 - 0.9776	0.2672 - 0.0815
		
4	0.9698 - 0.9902	0.1265 - 0.0373
		
5	0.7343 - 0.9393	1.2127 - 0.2379
		
11	0.0370 - 0.0355	0.0448 - 0.0449
		
14	0.9914 - 0.9967	0.0295 - 0.0182
		

Tableau 10. Résultat des scenarios 1,4,5,11 et 14

A cause la non disponibilité des capteurs performent on a implémentée une interface graphique pour tests le model de chaque scenario

La figure 27 représente une capture d'écran de notre application du test avec un exemple d'une photo différente totalement du notre base de données, le système a identifié cette signalisation comme une limite de vitesse 30km\h avec un vecteur de caractéristiques :

(1, 30, 30, 3)

Figure 27. Capture d'écrans application test

5. Analyse et discussion

L'objective de ce travail est de permettre un taux de précision acceptableau en regard des travaux similaires avec un CNN pure sans l'ajout d'aucun module complémentaire, le tableau 11 résume les résultats d'état de littératures

	Auteurs	Article	Accourcie	Epochs	Addition
01	Lihua Wen & Kang-Hyun Jo	Traffic Sign Recognition and Classification with Modified Residual Networks	99.66%	200	mResNets
02	Tejas Chaudhari , Ashish Wale, Amit Joshi, Suraj Sawant	Traffic Sign Recognition Using Small-Scale Convolutional Neural Network	97.71	50	ADAS
03	Pierre Sermanet & Yann LeCun	Traffic Sign Recognition with Multi-Scale Convolutional Networks	98.97%	-	EBLearn
04	Mrinal Haloi IIT Guwahati	Traffic Sign Classification Using Deep Inception Based Convolutional Networks	99.81%	-	GoogLeNet
05	Fares Imtiez & Mekhaznia taher	Classification d'images numériques par de procédés de deep learning : Application aux panneaux de signalisation	99.67%	49	Aucune

Tableau 11. Etat de littératures

5.1. Analyse

Si on veut comparer les résultats mentionner dans le tableau 11, on remarque que le scénario 14 est le plus parfait, mais pour quoi pas les autres ?

- Scenario 1 : L'accuracy est de 97,76% avec un loss de 0,09 mais avec un comportement bon sauf que dans la 2eme et la 13eme epoch , implique que le model n'est pas bien configuré , nombre bas d'epoch (15) batch size un peut minimiser (32)
- Scenario 4 : L'accuracy est de 99,0 % avec un loss de 0,03 , on remarque que l'accuracy a été augmenté car on augmenter le nombre d'epoch deux fois que le premier scenario la meme chose avec la batch size mais cette modification a touché l'accuracy plus que loss
- Scenario 5 : L'accuracy est de 93,93 % avec un loss de 0,23 , on remarque que les courbes sont du nature instableau avec des modification agressives a cause de la tailler très petite de la taille du lot (batch size) par contre le nombre d'epoch est toujours du 30
- Scenario 11 : L'accuracy est de 3,55% avec un loss de 0,04 , les courbes sont stableau avec une mauvaise performance a cause de le mauvais choix de la fonction loss 'mse' tanque il est aussi dédié au contexte du multi-classe aussi, elle

travail juste sur la difference et dans notre problématique : samples sont très similaire

- Scenario 14 : L'accuracy est de 99,67% avec un loss de 0,01 C'est le plus parfait scenario avec la meilleure configuration , est tous les courbes sont stableau parfaitement avec une très bonne performance

5.2. Discussion

En analysant tous les résultats de tous les scénarios on peut dire que :

1- Justification d'utilisation de plusieurs couches (figure 23) : CNN c'est d'une architecture hiérarchie, chaque couche utilise la sortie de la dernière couche comme une entré
2- Plus que nombre de filtre est élevé plus que nombre d'abstraction sera élevé aussi par ce que la couche d'entré dans le réseau neurone recevoir sample ; cette ligne est toujours bruitée, et c'est pour cela on laisse notre algorithme d'entrainement extracter des lignes bruitées, et aux moments il catch une bonne ligne on passe vers la couche suivante pour créer plus de complexité. Avec cette méthode on a assuré une des conditions de l'apprentissage profond : l'ajout de la complexité croisentte (figure 23 + tableau 10)
3- Il faut bien choisi les bonnes fonctions selon la problématique (scenario 11)
4- Des filtres petits peuvent capturer plus de détails, et le contraire avec les grands filtres (scenario 1 et 14)
5- Karnels de grande taille peuvent être utiles pour capter des informations avec un grand champs de la prochaine couche (deux couches successives peut augmenter le champ de la réception), donc en peut utiliser cette remarque comme une justification de placer les grands karnels avant les petits karnels dans tous les scénarios
6- Un batch d'une taille plus grande peut introduire accuracy inférieure (scenario 9)
7- Une grande taille du batch implique que le model fait des grand mis a jours, et le contraire pour les batch de petites tailles (la distance eiclidienne) (scenario 5 et 14)
8- En peut compenser la taille du batch par le nombre d'épochè (scenario 1 et 3)

On conclure que la bon choix des paramètres dans un CNN joue un grand rôle dans le processus d'entrainement. D'après le scenario 14 24 l'accourcie de train et de validation augmente avec le nombre d'époque, ceci signifie que le modèle apprenne plus d'informations à chaque époque. Si l'accuracy est diminuée, alors on aura besoin de plus du filtre et un batch plus grand et aussi on doit augmenter le nombre d'époque pour faire apprendre notre modèle. Ainsi, la même chose pour l'erreur de train et de la validation qui diminue avec le nombre d'époque. Le coût d'apprentissage est de 0.0295 pour un nombre d'itération de 49 et un coût de validation de 0.0182 Précision d'apprentissage : 99,14% & Précision de validation : 99,67%

5.3. Limites

Durant la partie expérimentale en a trouvé des difficultés, en peux les résumer dans les points suivants

- Performance : Manque du performances machine mentionné dans le tableau 9
- Débit de connexion internet : manque de la performance durant la phase d'installation
- Compatibilité des modules mathématiques avec la version du Tensorflow
- Ordre des différentes couches du l'architecture CNN

6. Conclusion

Ce chapitre était dédié à l'évolution de notre système, nous avons pu l'évaluer en montrons les différents résultats obtenus en termes de précision et d'erreur.

Conclusion générale

Le domaine de classification d'images qui comme tout domaine faisant partie de l'intelligence artificielle a connu une évolution majeure depuis l'avènement du Deep Learning.

Les travaux réalisés dans ce domaine sont multiples, aucune méthode proposée jusqu'à présent n'a réussi à atteindre une précision de 100% ou faire une reconstruction parfaite des images, mais elles tentent sans cesse de s'en approcher.

Notre travail a mis en œuvre un système capable de faire la classification et reconstruction des images de panneaux de signalisation par un CNN pure (sans ajouter aucun module alternative). Nous avons pu obtenir des résultats assez satisfaisants (99,67%), et prouvent l'intérêt de notre architecture des modèles utilisée.

Comme perspective, le travail réalisé peut être amélioré par d'avantage de tests sur de BDD plus importantes et utilisant d'autres outils de reconnaissance et classification plus efficaces

Reférences

1- What Is an Image? NEW LITERARY HISTORY W. J. T. Mitchell
2- D.Ameisen.Qu ’ est-ce qu ’ une image numérique ?.*Conference Paper*, vol.57, pp.169-172, 2013
3- 23 mars 2012 – Lycée du Mené – CRIPT Bretagne
4- http://www.tsi.enst.fr/pages/enseignement/ressources/mti/egal-histo/rapport.htm
5- Le traitement d'images D. Legland 12 juin 2019
6- Analyse et Interprétation d’Images : Analyse Multiéchelle pour la Description des Formes Planaires Kidiyo Kpalma
7- Le traitement numérique des images le 28 novembre 2011 à 11:28, par Jean-Paul Allouche
8- J.-P. Cocquerez And S. Philipp,1995 , Analyse d’Images : filtrage et segmentation, Masson
9- M.Meliani 2012’’Segmentation d’Image par Coopération Régions-Contours’’. Thèse de magister,Ecole nationale Supérieure en Informatique, Oued-SmarAlger, Ecole Doctorale STIC.
10- M. GUERROUT EL-HACHEMI. Performances dans la Segmentation d'images médicales. Thèse de doctorat .p16,2018
11- Abdelkrim MAARIR, Ilhame AGNAOU, Belaid BOUIKHALENE, « Evaluation de Quelques Méthodes de Segmentation ».
12- Antoine Olivier & Shen Danfei, « Segmentation par croissance de régions », http://www.tsi.enst.fr/pages/enseignement/ressources/mti/croissance/index.html, 2000.
13- A Meziane, A Merabet, « Détection de zones d’intérêt dans des images par les réseaux de neurones convolutifs », Mémoire de master en Systéme de télécommunication, sous la direction de Serir Amina, Alger, Université des sciences et de la technologie Houari Boumediene
14- Yann LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE
15- Simon Haykin. Neural Networks and Learning Machines. 3rd Edition
16- D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex.the journal of physiology
17- OLAVI S, Object detection from images using convolutional neural networks, Master, Aalto Universiy, 2017
18- Deep Learning Convolutional Neural Networks for Radio Identification Shamnaz Riyaz, Kunal Sankhe , Stratis Ioannidis, and Kaushik Chowdhury IEEE Communications Magazine, vol. 56, Issue no. 9, September, 2018
19- Sumit Saha. “ A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way“.(2018).sur le site www.towardsdatascience.com
20- Josh patterson adam gibson “deep learning a practitioner's approach“,O'Reilly Media,aout 2017
21- Le traitement d'images pour les nuls D. Legland 12 juin 2019
22- A Meziane, A Merabet, « Détection de zones d’intérêt dans des images par les réseaux de neurones convolutifs », Mémoire de master en Systéme de télécommunication, sous la direction de Serir Amina, Alger, Université des sciences et de la technologie Houari Boumediene

23- Ouhda M, El Asnaoui K, Ouanan M, and Aksasse B. Content-Based Image Retrieval Using Convolutional Neural Networks.
24- OFFE,Sergey et SZEGEDY,Christian.Batch normalization:accelerating deep network training
25- https://fr.wikipedia.org/wiki/Python_(langage)
26- https://www.supinfo.com/articles/single/7899-top-10-ide-developpeurs-python
27- https://fr.wikipedia.org/wiki/TensorFlow
28- https://deepai.org/machine-learning-glossary-and-terms/keras
29- https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=Adam%20is%20an%20optimization%20algorithm,iterative%20based%20in%20training%20data.&text=The%20algorithm%20is%20called%20Adam,derived%20from%20adaptive%20moment%20estimation.

Annexes

Scenario 1				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	1.9654	0.4959	0.6770	0.8498
02	0.8227	0.7570	0.2757	0.9211
03	0.5431	0.8394	0.1595	0.9551
04	0.4380	0.8683	0.1383	0.9624
05	0.3814	0.8853	0.1231	0.9652
06	0.3593	0.8934	0.1054	0.9670
07	0.3375	0.9007	0.0775	0.9769
08	0.3112	0.9089	0.0690	0.9809
09	0.2872	0.9165	0.1200	0.9649
10	0.2772	0.9195	0.2042	0.9388
11	0.2739	0.9228	0.0749	0.9779
12	0.2786	0.9213	0.0718	0.9779
13	0.2583	0.9259	0.0948	0.9700
14	0.2542	0.9266	0.0646	0.9793
15	0.2640	0.9261	0.0949	0.9722

Tableau 12. Débogage scenario 1

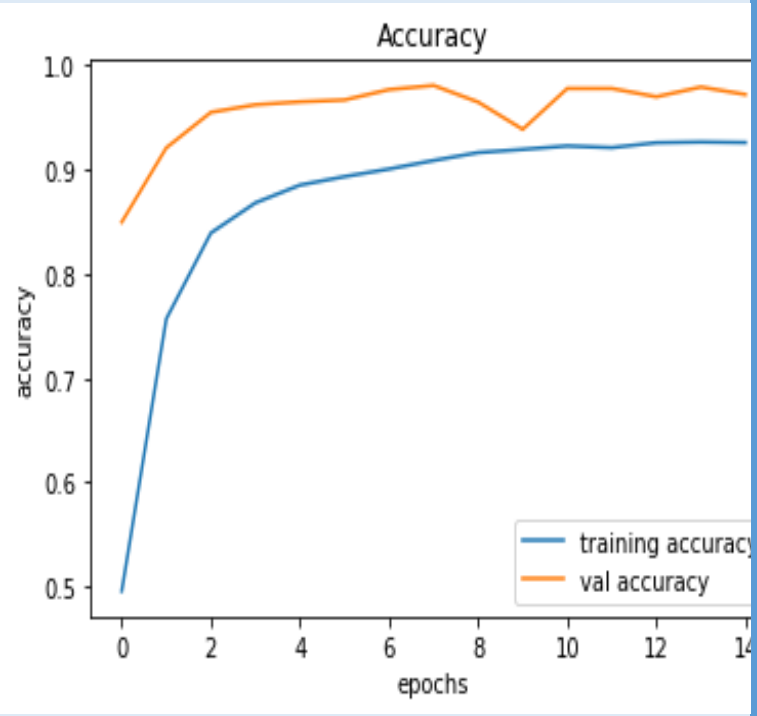
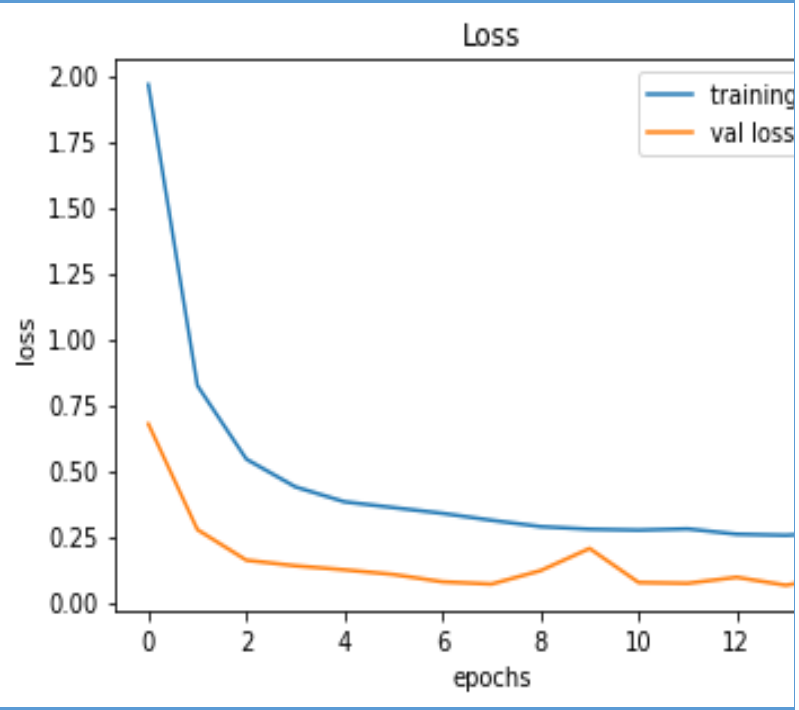
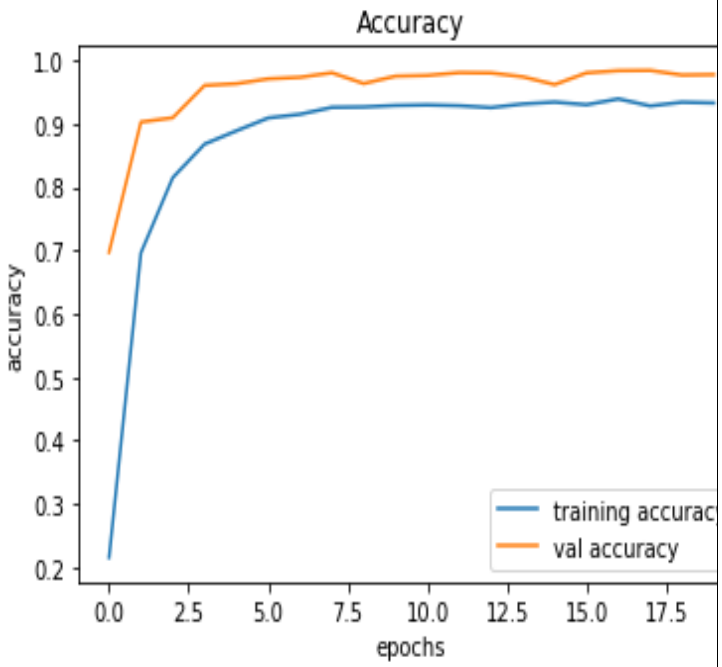
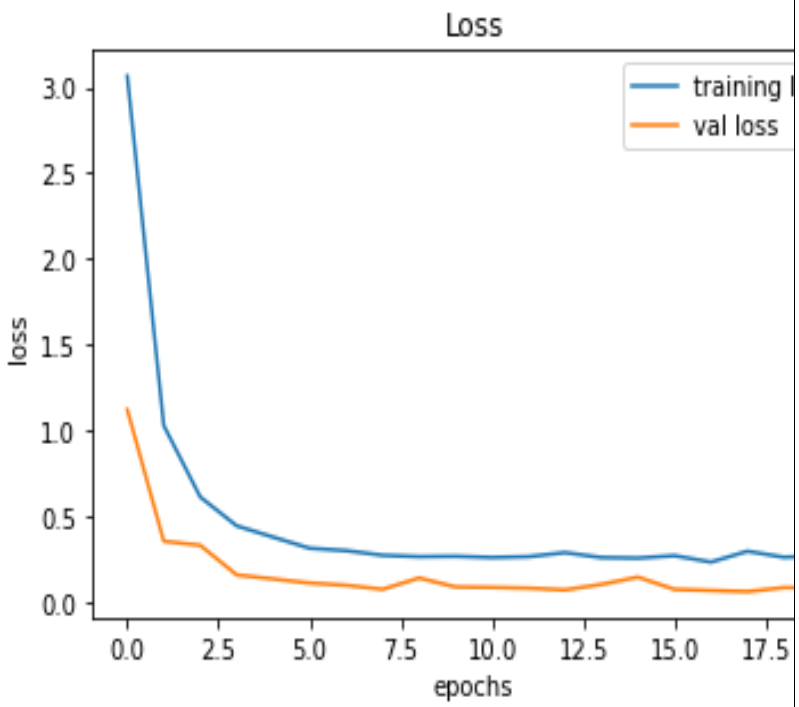
Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
01		
	0.9261-0.9722	**0.2640-0.0949**

Tableau 13. Résultat Scénario 01

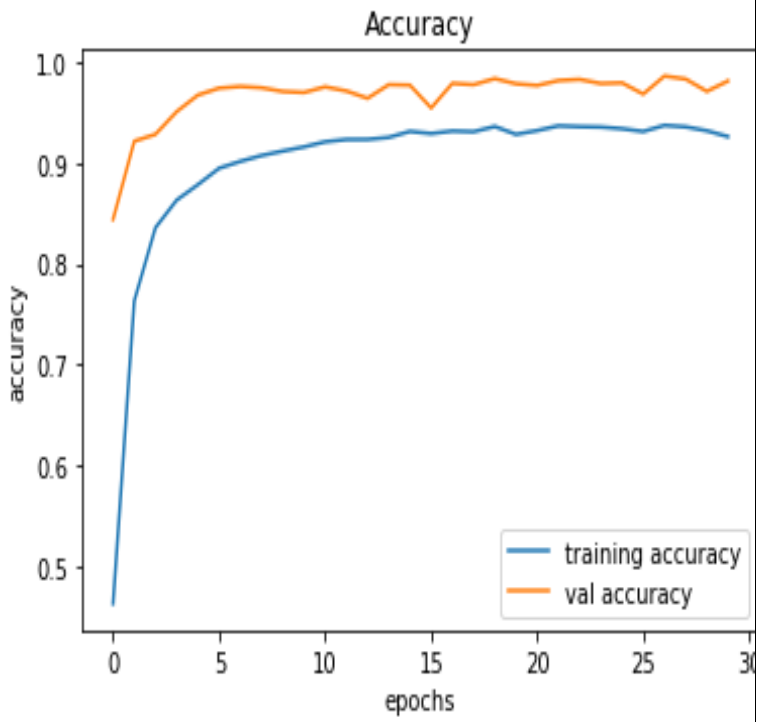
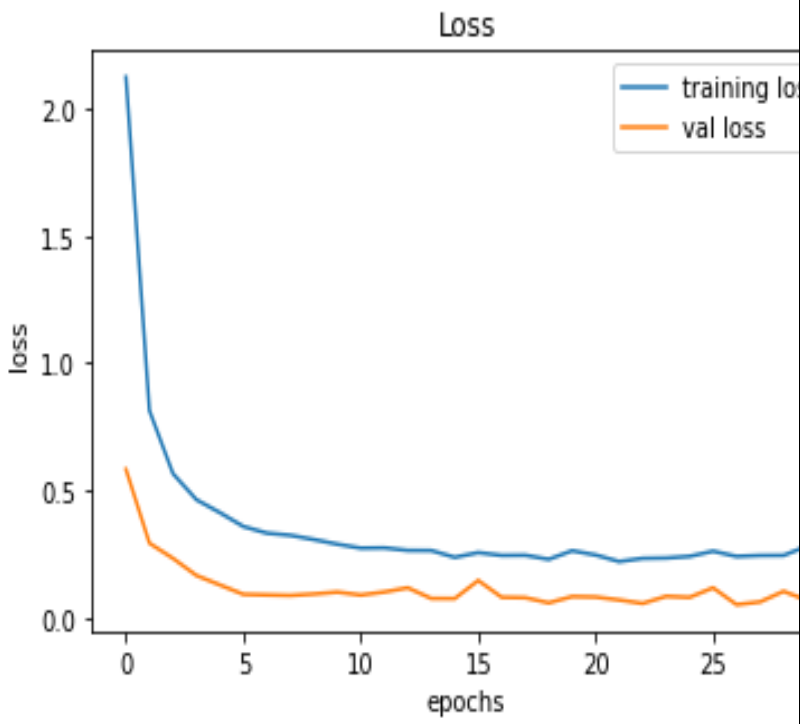
Scenario 2				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	3.0660	0.2150	1.1201	0.6971
02	1.0272	0.6960	0.3524	0.9027
03	0.6106	0.8148	0.3287	0.9095
04	0.4421	0.8679	0.1559	0.9605
05	0.3763	0.8888	0.1323	0.9633
06	0.3113	0.9091	0.1085	0.9709
07	0.2975	0.9150	0.0953	0.9733
08	0.2705	0.9259	0.0722	0.9805
09	0.2625	0.9265	0.1379	0.9635
10	0.2645	0.9289	0.0858	0.9751
11	0.2576	0.9297	0.0831	0.9763
12	0.2625	0.9285	0.0786	0.9807
13	0.2860	0.9255	0.0687	0.9802
14	0.2573	0.9313	0.1007	0.9741
15	0.2542	0.9342	0.1430	0.9619
16	0.2681	0.9301	0.0716	0.9804
17	0.2307	0.9393	0.0652	0.9838
18	0.2941	0.9281	0.0586	0.9842
19	0.2590	0.9340	0.0818	0.9769
20	0.2672	0.9329	0.0815	0.9371

Tableau 14. Débogage scenario 2

Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
02		
	0.9329-0.9776	0.2672-0.0815
	Tableau 15. Résultat Scénario 02	

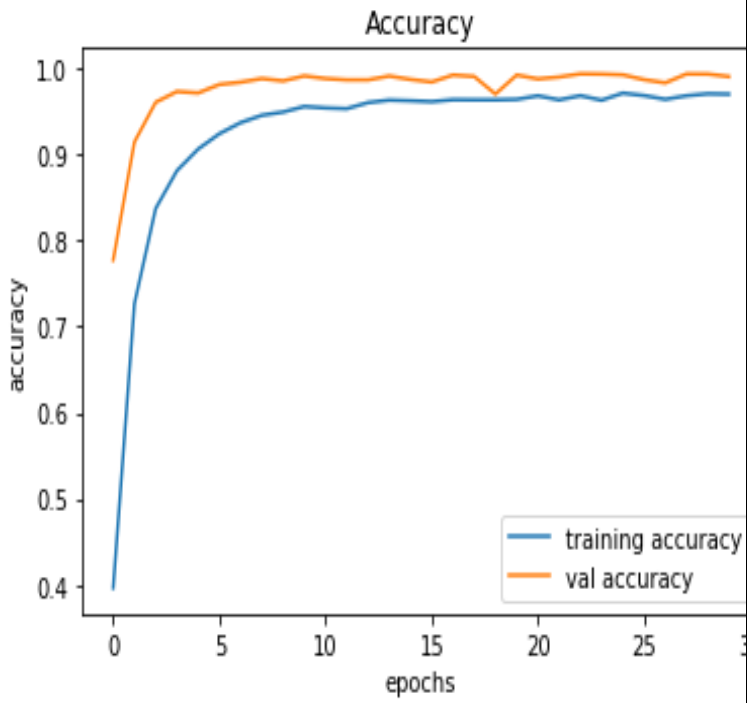
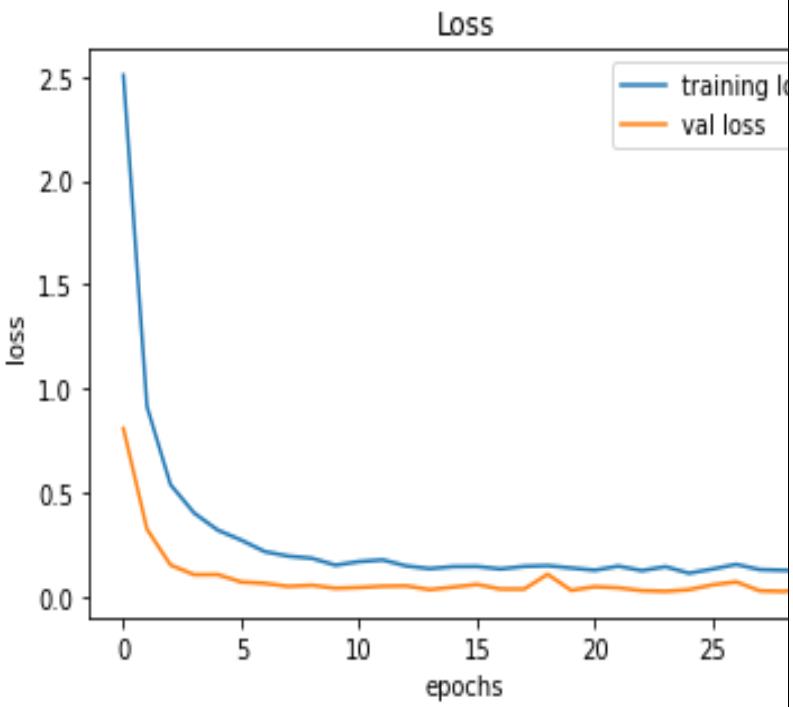
Scenario 3				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	2.1229	0.4627	0.5830	0.8439
02	0.8138	0.7631	0.2931	0.9214
03	0.5667	0.8357	0.2338	0.9283
04	0.4638	0.8630	0.1664	0.9508
05	0.4131	0.8784	0.1292	0.9672
06	0.3586	0.8946	0.0923	0.9741
07	0.3326	0.9015	0.0900	0.9758
08	0.3240	0.9073	0.0880	0.9745
09	0.3074	0.9117	0.0935	0.9708
10	0.2892	0.9158	0.1011	0.9698
11	0.2736	0.9209	0.0896	0.9754
12	0.2747	0.9232	0.1015	0.9713
13	0.2648	0.9233	0.1180	0.9640
14	0.2643	0.9252	0.0761	0.9776
15	0.2376	0.9312	0.0757	0.9770
16	0.2560	0.9291	0.1467	0.9545
17	0.2451	0.9316	0.0807	0.9788
18	0.2457	0.9309	0.0792	0.9776
19	0.2298	0.9360	0.0589	0.9836
20	0.2635	0.9283	0.0836	0.9784
21	0.2467	0.9320	0.0818	0.9768
22	0.2210	0.9367	0.0702	0.9816
23	0.2335	0.9359	0.0566	0.9828
24	0.2350	0.9355	0.0845	0.9787
25	0.2413	0.9338	0.0806	0.9793
26	0.2617	0.9311	0.1179	0.9682
27	0.2412	0.9372	0.0518	0.9861
28	0.2448	0.9357	0.0627	0.9832
29	0.2451	0.9318	0.1041	0.9709
30	0.2813	0.9260	0.0689	0.9813

Tableau 16. Débogage scenario 3

Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
03		
	0.9260 -0.9813	0.2813 - 0.0689
	Tableau 17. Résultat Scénario 03	

Scenario 4				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	2.5072	0.3965	0.8089	0.7772
02	0.9146	0.7265	0.3250	0.9143
03	0.5387	0.8371	0.1529	0.9603
04	0.4028	0.8809	0.1063	0.9728
05	0.3209	0.9060	0.1067	0.9712
06	0.2725	0.9238	0.0713	0.9810
07	0.2172	0.9367	0.0646	0.9837
08	0.1955	0.9453	0.0501	0.9879
09	0.1851	0.9491	0.0556	0.9852
10	0.1520	0.9555	0.0411	0.9909
11	0.1693	0.9538	0.0447	0.9879
12	0.1774	0.9529	0.0508	0.9864
13	0.1485	0.9600	0.0524	0.9864
14	0.1354	0.9629	0.0346	0.9908
15	0.1453	0.9621	0.0468	0.9869
16	0.1459	0.9611	0.0594	0.9839
17	0.1338	0.9634	0.0374	0.9920
18	0.1462	0.9632	0.0370	0.9903
19	0.1497	0.9632	0.1083	0.9699
20	0.1386	0.9638	0.0314	0.9920
21	0.1278	0.9679	0.0486	0.9875
22	0.1474	0.9632	0.0432	0.9897
23	0.1259	0.9680	0.0297	0.9934
24	0.1448	0.9628	0.0264	0.9931
25	0.1135	0.9710	0.0344	0.9922
26	0.1335	0.9682	0.0576	0.9866
27	0.1557	0.9638	0.0717	0.9829
28	0.1306	0.9680	0.0292	0.9932
29	0.1275	0.9703	0.0259	0.9931
30	0.1265	0.9698	0.0373	0.9902

Tableau 18. Débogage scenario 4

Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
04		
	0.9698 0.9902	0.1265 - 0.0373
	Tableau 19. Résultat Scénario 04	

Scenario 5				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	1.5887	0.5809	0.3373	0.9130
02	0.6244	0.8248	0.2078	0.9447
03	0.5956	0.8393	0.2039	0.9397
04	0.5547	0.8524	0.1399	0.9570
05	0.5200	0.8634	0.1088	0.9684
06	0.5364	0.8583	0.2073	0.9385
07	0.5531	0.8573	0.1890	0.9485
08	0.4910	0.8751	0.1760	0.9468
09	0.5414	0.8619	0.1753	0.9528
10	0.5045	0.8730	0.1353	0.9600
11	1.2747	0.6670	3.4853	0.05870
12	2.0367	0.4631	0.1659	0.9555
13	0.5800	0.8569	0.1217	0.9688
14	0.5671	0.8573	0.1229	0.9654
15	0.5818	0.8583	0.1329	0.9607
16	0.6074	0.8524	0.1459	0.9570
17	0.6435	0.8437	0.5804	0.8308
18	0.6329	0.8492	0.1470	0.9573
19	0.7838	0.8130	0.1514	0.9578
20	0.7364	0.8287	0.1383	0.9610
21	0.7366	0.8246	0.2589	0.9232
22	1.5905	0.6013	1.1845	0.6832
23	0.8221	0.8094	0.1631	0.9489
24	1.0681	0.7485	0.2080	0.9515
25	0.8420	0.8160	0.7608	0.7994
26	0.8920	0.8042	0.2059	0.9473
27	0.8614	0.8150	0.2166	0.9396
28	0.9275	0.7959	0.2121	0.9401
29	0.9477	0.7879	0.1837	0.9494
30	1.2127	0.7343	0.2379	0.9393

Tableau 20. Débogage scenario 5

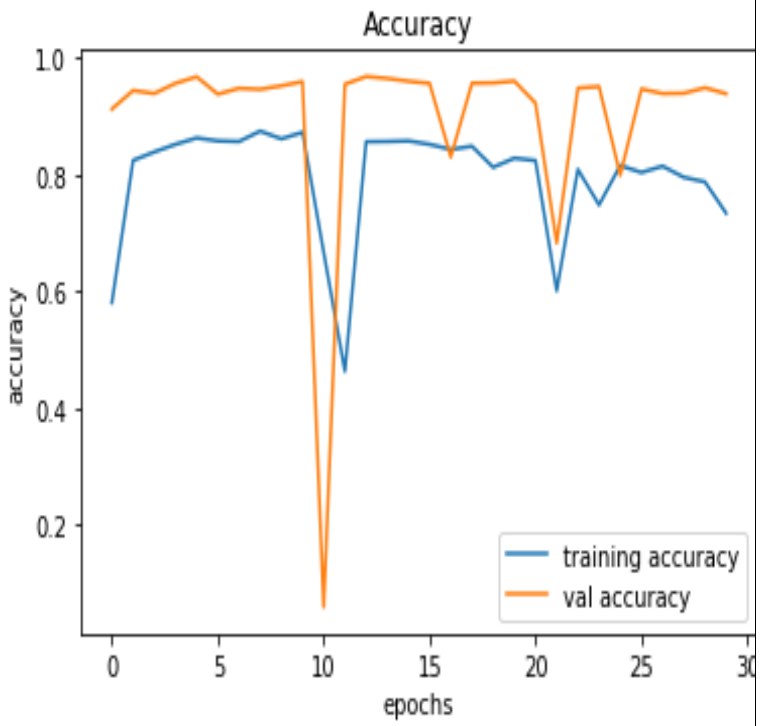
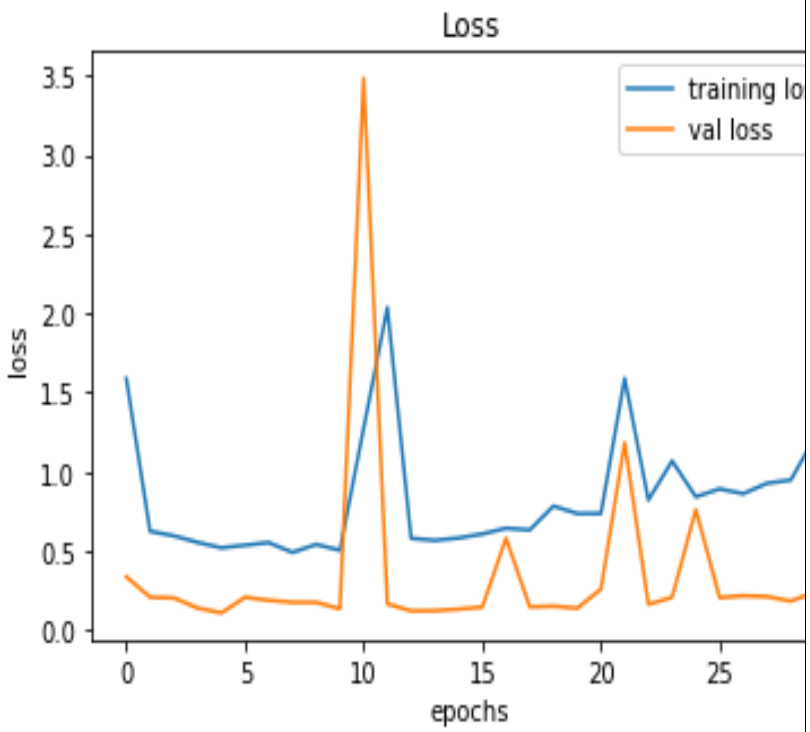
Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
05		
	0.7343 0.9393	1.2127 - 0.2379

Tableau 21. Résultat Scénario 05

Scenario 6				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	3.3320	0.2853	1.1536	0.7489
02	1.1576	0.6845	0.3924	0.9209
03	0.6050	0.8335	0.1554	0.9658
04	0.3712	0.8978	0.1166	0.9755
05	0.2901	0.9210	0.0724	0.9836
06	0.2167	0.9394	0.0735	0.9800
07	0.1946	0.1946	0.0520	0.9875
08	0.1785	0.9497	0.0405	0.9895
09	0.1458	0.9600	0.0755	0.9795
10	0.1661	0.9541	0.0309	0.9930
11	0.1245	0.9651	0.0408	0.9893
12	0.1226	0.9664	0.0320	0.9913
13	0.1174	0.9685	0.0292	0.9917
14	0.1064	0.9712	0.0290	0.9934
15	0.1018	0.9711	0.0252	0.9945
16	0.0951	0.9741	0.0343	0.9912
17	0.0995	0.9724	0.0340	0.9917
18	0.0989	0.9726	0.0346	0.9916
19	0.1096	0.9710	0.0354	0.9902
20	0.0956	0.9749	0.0245	0.9940
21	0.0939	0.9759	0.0253	0.9936
22	0.0754	0.9796	0.0183	0.9958
23	0.0990	0.9738	0.0247	0.9948
24	0.0953	0.9759	0.0241	0.9938
25	0.0698	0.9817	0.0227	0.9957
26	0.1051	0.9744	0.0267	0.9923
27	0.0791	0.9791	0.0214	0.9953
28	0.0792	0.9793	0.0201	0.9957
29	0.0862	0.9792	0.0182	0.9959
30	0.0721	0.9816	0.0433	0.9908

Tableau 22. Débogage scenario 6

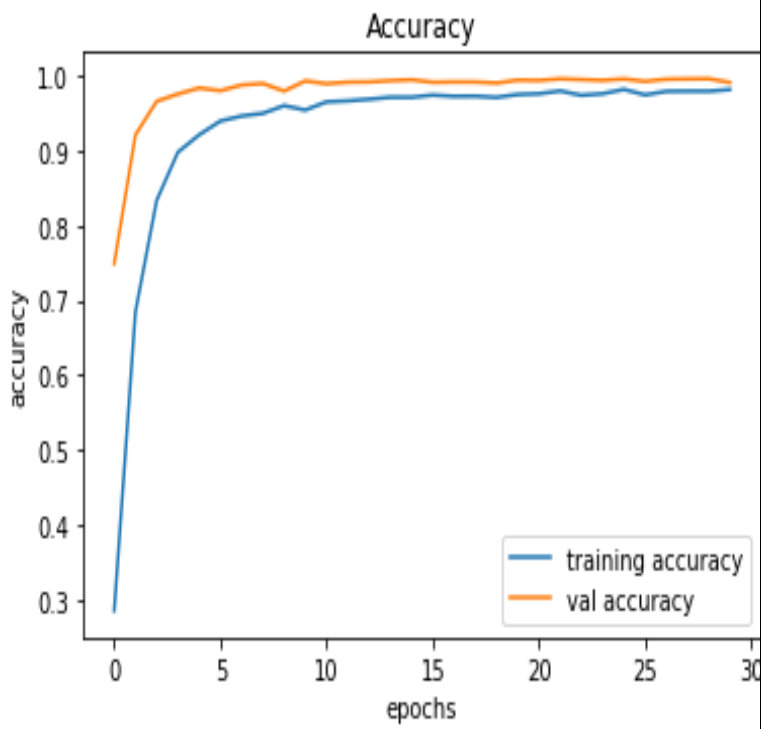
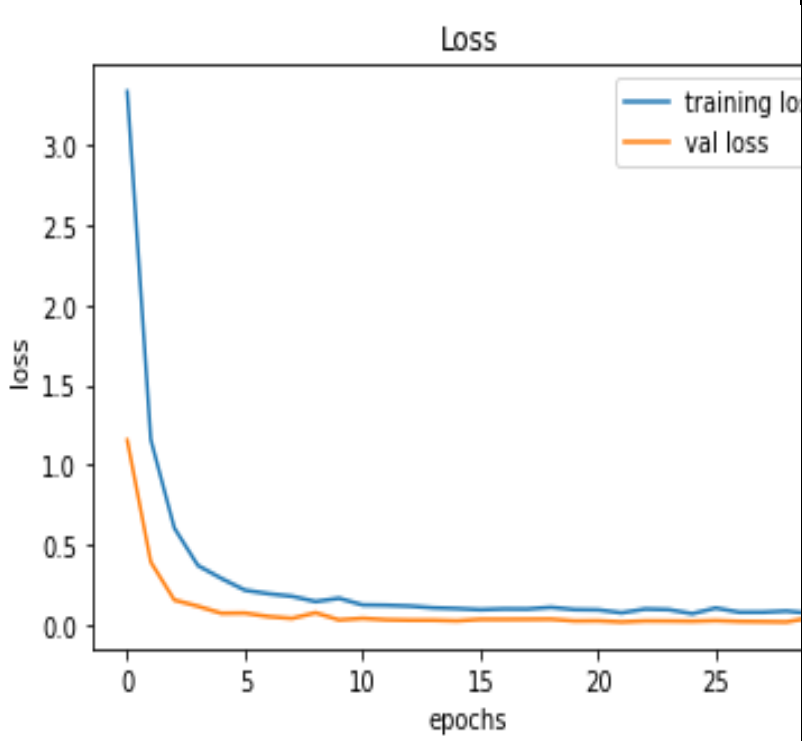
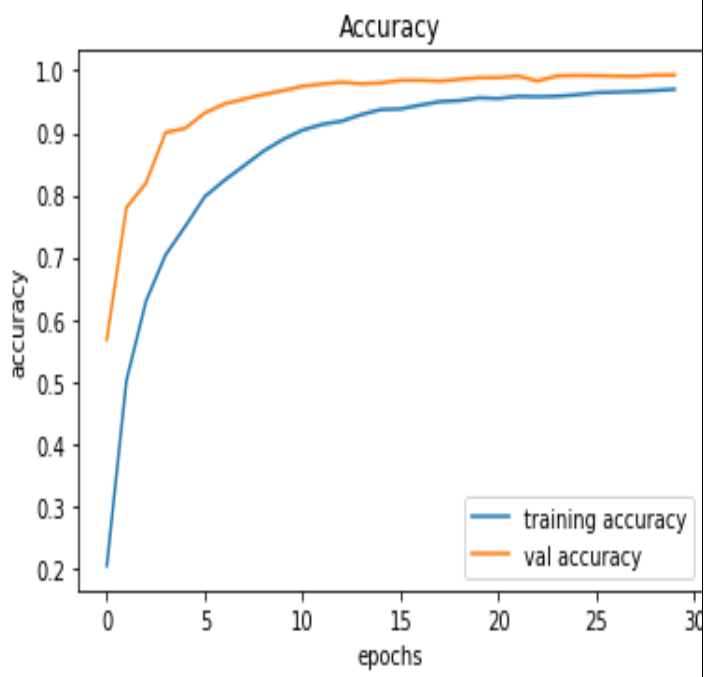
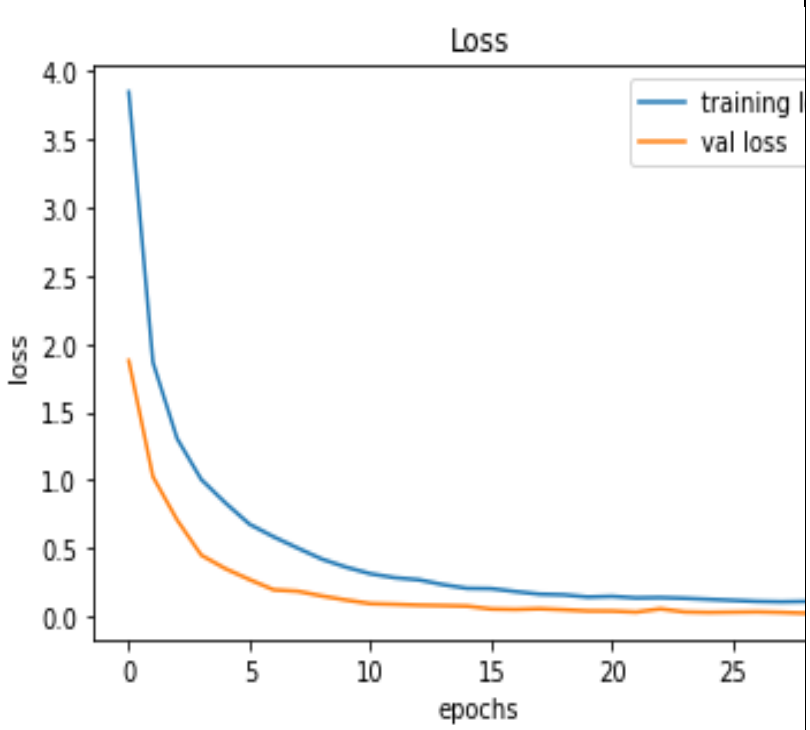
Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
06		
	0.9816 0.9908	0.0721 - 0.0433

Tableau 23. Résultat Scénario 06

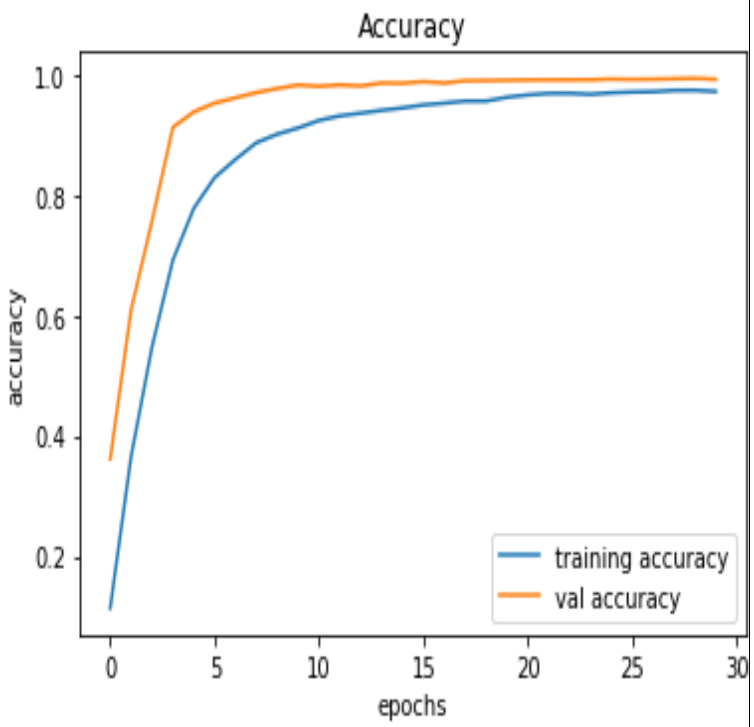
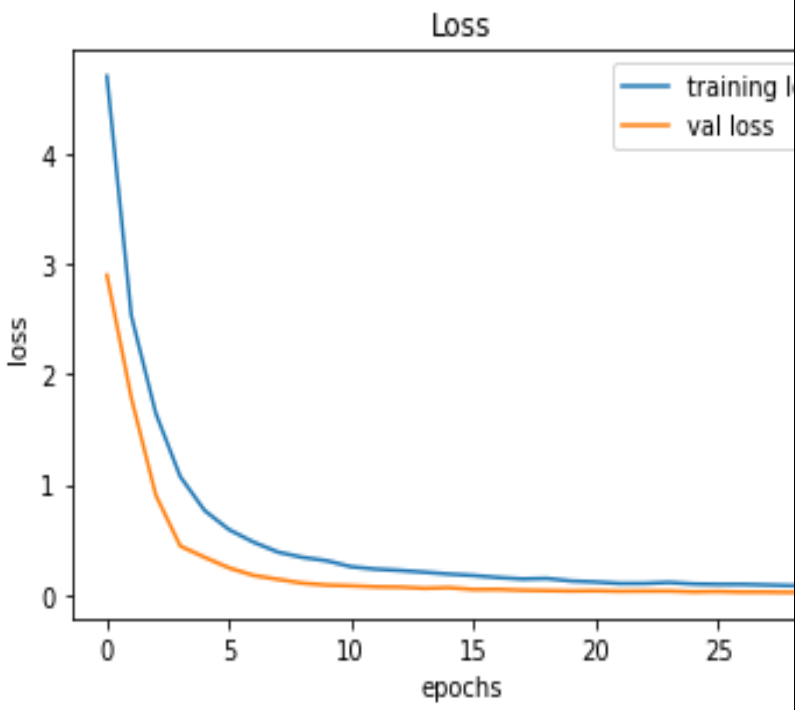
Scenario 7				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	3.8426	0.2039	1.8747	0.5677
02	1.8632	0.5013	1.0250	0.7795
03	1.3044	0.6292	0.7084	0.8193
04	1.0048	0.7037	0.4508	0.9004
05	0.8334	0.7492	0.3518	0.9072
06	0.6769	0.7973	0.2717	0.9318
07	0.5847	0.8234	0.1967	0.9464
08	0.5017	0.8468	0.1861	0.9542
09	0.4212	0.8705	0.1515	0.9617
10	0.3623	0.8894	0.1217	0.9677
11	0.3145	0.9043	0.0954	0.9749
12	0.2858	0.9135	0.0906	0.9784
13	0.2704	0.9187	0.0845	0.9813
14	0.2348	0.9293	0.0813	0.9788
15	0.2077	0.9374	0.0774	0.9802
16	0.2055	0.9384	0.0564	0.9843
17	0.1842	0.9445	0.0533	0.9844
18	0.1650	0.9500	0.0584	0.9830
19	0.1602	0.9517	0.0509	0.9860
20	0.1440	0.9561	0.0422	0.9885
21	0.1488	0.9551	0.0413	0.9887
22	0.1375	0.9586	0.0338	0.9912
23	0.1399	0.9579	0.0574	0.9834
24	0.1347	0.9587	0.0350	0.9913
25	0.1271	0.9611	0.0316	0.9921
26	0.1197	0.9644	0.0334	0.9917
27	0.1121	0.9653	0.0354	0.9911
28	0.1090	0.9663	0.0315	0.9909
29	0.1116	0.9680	0.0255	0.9927
30	0.1022	0.9701	0.0267	0.9929

Tableau 24. Débogage scenario 7

Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
07		
	0.9701 0.0267	0.1022 - 0.0267
	Tableau 25. Résultat Scénario 07	

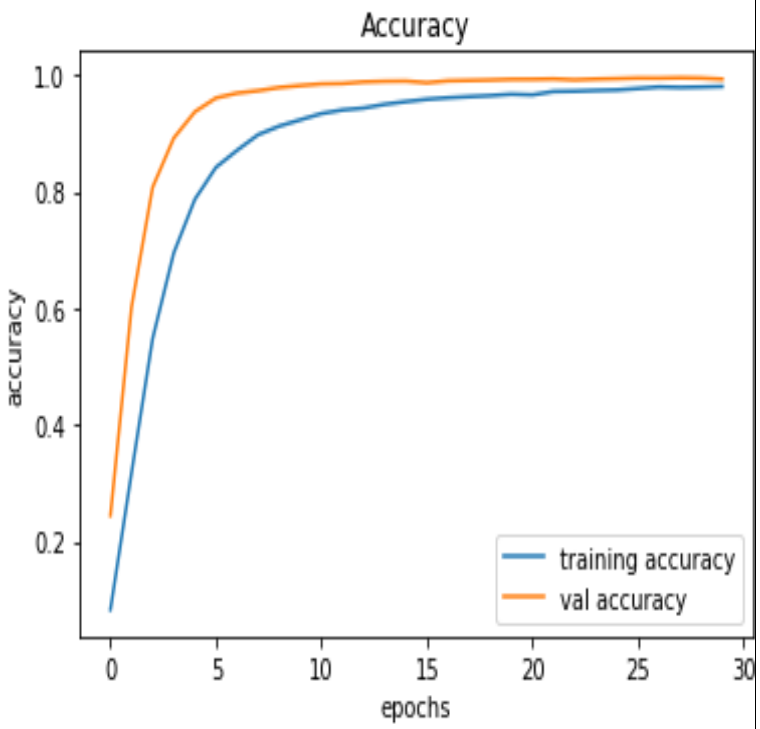
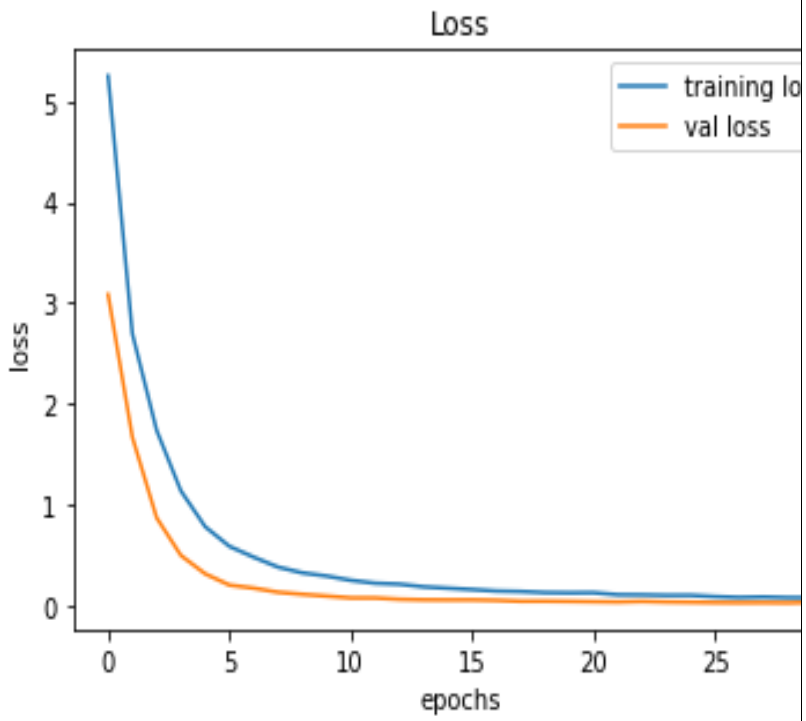
Scenario 8				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	4.7011	0.1136	2.8970	0.3618
02	2.5291	0.3658	1.7842	0.6100
03	1.6458	0.5477	0.9043	0.7573
04	1.0720	0.6922	0.4433	0.9132
05	0.7641	0.7784	0.3403	0.9388
06	0.5910	0.8296	0.2440	0.9535
07	0.4776	0.8598	0.1764	0.9623
08	0.3874	0.8876	0.1406	0.9709
09	0.3399	0.9017	0.1060	0.9777
10	0.3090	0.9121	0.0883	0.9834
11	0.2542	0.9246	0.0821	0.9813
12	0.2334	0.9323	0.0727	0.9837
13	0.2209	0.9367	0.0699	0.9819
14	0.2052	0.9414	0.0592	0.9869
15	0.1882	0.9454	0.0649	0.9865
16	0.1750	0.9504	0.0483	0.9890
17	0.1571	0.9534	0.0494	0.9869
18	0.1437	0.9568	0.0417	0.9907
19	0.1485	0.9568	0.0386	0.9909
20	0.1253	0.9632	0.0350	0.9913
21	0.1165	0.9671	0.0372	0.9918
22	0.1031	0.9695	0.0329	0.9920
23	0.1036	0.9697	0.0343	0.9921
24	0.1124	0.9681	0.0347	0.9922
25	0.0985	0.9705	0.0268	0.9934
26	0.0948	0.9719	0.0300	0.9929
27	0.0947	0.9725	0.0249	0.9931
28	0.0897	0.9746	0.0250	0.9939
29	0.0837	0.9748	0.0219	0.9945
30	0.0902	0.9728	0.0253	0.9930

Tableau 26. Débogage scenario 8

Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
08		
	0.9728 0.9930	0.0902 - 0.0253
	Tableau 27. Résultat Scénario 08	

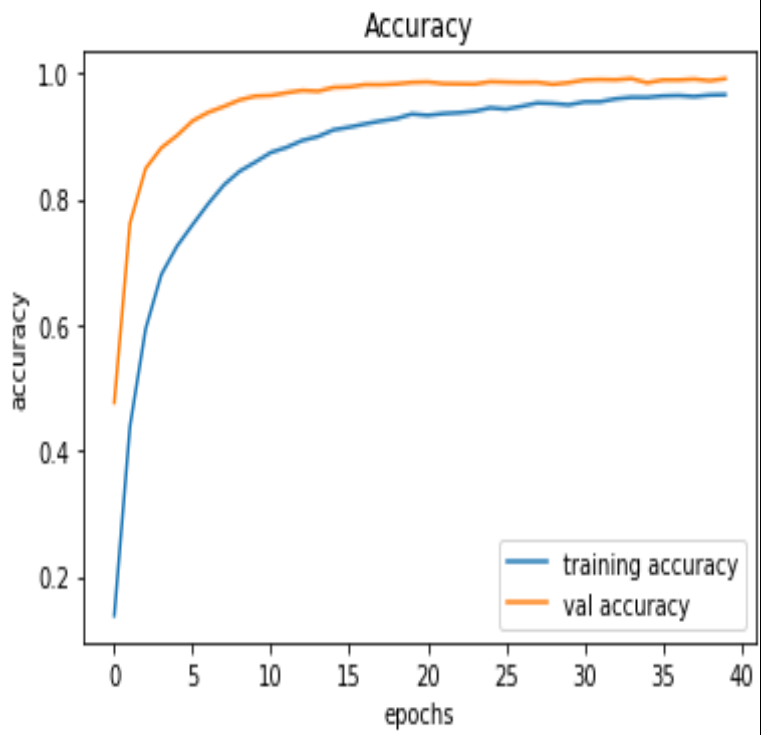
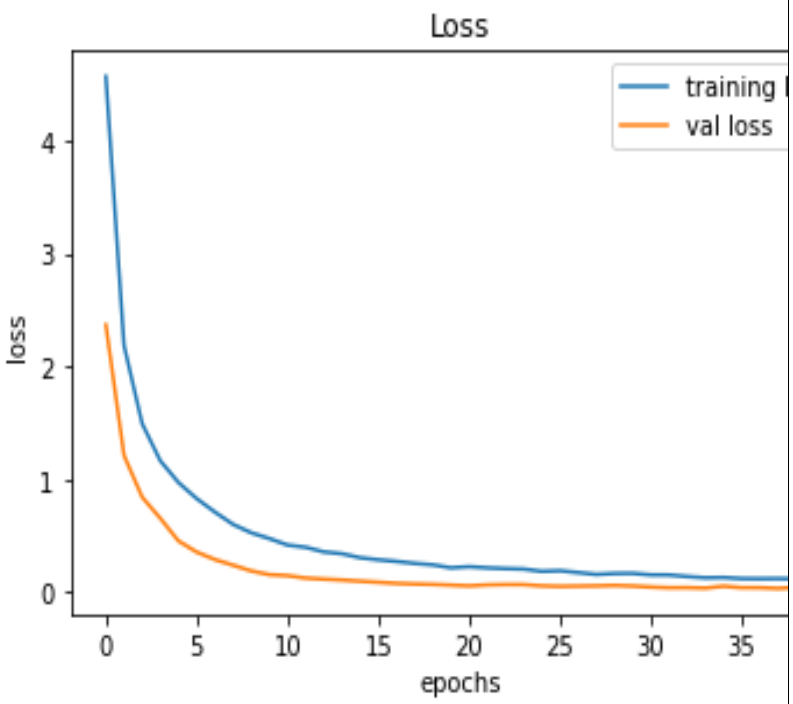
Scenario 9				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	5.2545	0.0835	3.0836	0.2448
02	2.6941	0.3178	1.6659	0.6021
03	1.7348	0.5479	0.8659	0.8073
04	1.1330	0.6947	0.4922	0.8915
05	0.7775	0.7865	0.3096	0.9366
06	0.5822	0.8415	0.1977	0.9603
07	0.4750	0.8706	0.1674	0.9686
08	0.3765	0.8974	0.1267	0.9728
09	0.3194	0.9121	0.1059	0.9783
10	0.2878	0.9229	0.0884	0.9816
11	0.2452	0.9336	0.0704	0.9844
12	0.2162	0.9398	0.0698	0.9851
13	0.2068	0.9431	0.0565	0.9879
14	0.1802	0.9495	0.0514	0.9889
15	0.1668	0.9541	0.0503	0.9893
16	0.1534	0.9581	0.0505	0.9865
17	0.1389	0.9607	0.0461	0.9898
18	0.1340	0.9627	0.0373	0.9904
19	0.1233	0.9643	0.0374	0.9912
20	0.1210	0.9668	0.0356	0.9922
21	0.1220	0.9657	0.0326	0.9923
22	0.1018	0.9714	0.0298	0.9930
23	0.1003	0.9720	0.0349	0.9915
24	0.0958	0.9732	0.0296	0.9927
25	0.0959	0.9741	0.0271	0.9935
26	0.0833	0.9766	0.0240	0.9945
27	0.0749	0.9792	0.0225	0.9944
28	0.0790	0.9784	0.0233	0.9950
29	0.0729	0.9791	0.0215	0.9945
30	0.0718	0.9801	0.0266	0.9926

Tableau 28. Débogage scenario 9

Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
09		
	0.9801 0.9926	0.0718 - 0.0266
	Tableau 29. Résultat Scénario 09	

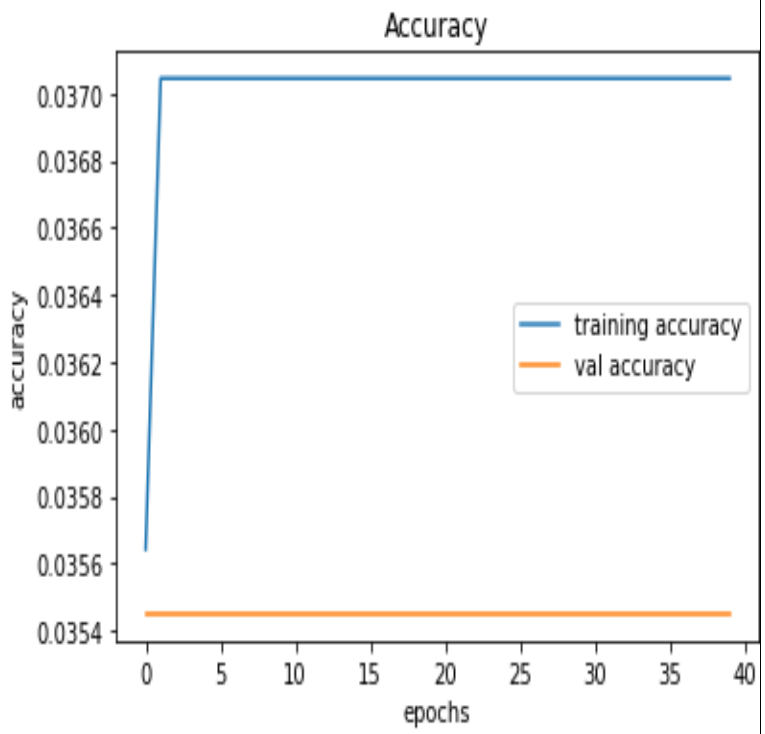
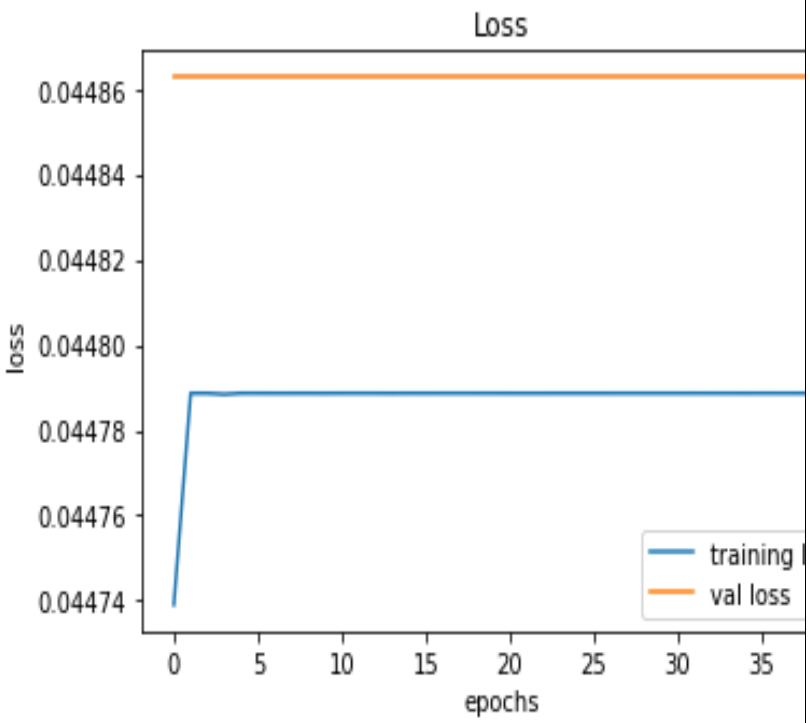
Scenario 10				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	4.5691	0.1383	2.3679	0.4768
02	2.1845	0.4386	1.2096	0.7618
03	1.4911	0.5936	0.8404	0.8486
04	1.1622	0.6799	0.6522	0.8813
05	0.9695	0.7249	0.4517	0.9009
06	0.8277	0.7594	0.3537	0.9243
07	0.7087	0.7926	0.2874	0.9378
08	0.5994	0.8225	0.2376	0.9471
09	0.5247	0.8442	0.1864	0.9575
10	0.4734	0.8587	0.1531	0.9635
11	0.4158	0.8740	0.1444	0.9645
12	0.3959	0.8821	0.1231	0.9689
13	0.3550	0.8931	0.1135	0.9726
14	0.3388	0.8993	0.1057	0.9713
15	0.3032	0.9100	0.0952	0.9774
16	0.2852	0.9142	0.0859	0.9782
17	0.2707	0.9193	0.0757	0.9818
18	0.2531	0.9240	0.0704	0.9816
19	0.2405	0.9278	0.0667	0.9829
20	0.2141	0.9351	0.0606	0.9853
21	0.2238	0.9329	0.0540	0.9860
22	0.2138	0.9358	0.0625	0.9833
23	0.2070	0.9372	0.0655	0.9830
24	0.2031	0.9396	0.0648	0.9827
25	0.1827	0.9450	0.0540	0.9865
26	0.1880	0.9429	0.0497	0.9857
27	0.1719	0.9473	0.0518	0.9852
28	0.1554	0.9527	0.0540	0.9855
29	0.1633	0.9517	0.0581	0.9823
30	0.1648	0.9497	0.0526	0.9850
31	0.1513	0.9544	0.0436	0.9892
32	0.1505	0.9546	0.0374	0.9899
33	0.1367	0.9593	0.0381	0.9894
34	0.1258	0.9616	0.0338	0.9915
35	0.1279	0.9616	0.0510	0.9847
36	0.1186	0.9638	0.0378	0.9892
37	0.1175	0.9645	0.0376	0.9892
38	0.1186	0.9628	0.0309	0.9906
39	0.1163	0.9653	0.0394	0.9880
40	0.1129	0.9659	0.0314	0.9913

Tableau 30. Débogage scenario 10

Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
10		
	0.9659 0.9913	0.1129 - 0.0314
	Tableau 31. Résultat Scénario 10	

Scenario 11				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	0.0447	0.0356	0.0449	0.0355
02	0.0448	0.0370	0.0449	0.0355
03	0.0448	0.0370	0.0449	0.0355
04	0.0448	0.0370	0.0449	0.0355
05	0.0448	0.0370	0.0449	0.0355
06	0.0448	0.0370	0.0449	0.0355
07	0.0448	0.0370	0.0449	0.0355
08	0.0448	0.0370	0.0449	0.0355
09	0.0448	0.0370	0.0449	0.0355
10	0.0448	0.0370	0.0449	0.0355
11	0.0448	0.0370	0.0449	0.0355
12	0.0448	0.0370	0.0449	0.0355
13	0.0448	0.0370	0.0449	0.0355
14	0.0448	0.0370	0.0449	0.0355
15	0.0448	0.0370	0.0449	0.0355
16	0.0448	0.0370	0.0449	0.0355
17	0.0448	0.0370	0.0449	0.0355
18	0.0448	0.0370	0.0449	0.0355
19	0.0448	0.0370	0.0449	0.0355
20	0.0448	0.0370	0.0449	0.0355
21	0.0448	0.0370	0.0449	0.0355
22	0.0448	0.0370	0.0449	0.0355
23	0.0448	0.0370	0.0449	0.0355
24	0.0448	0.0370	0.0449	0.0355
25	0.0448	0.0370	0.0449	0.0355
26	0.0448	0.0370	0.0449	0.0355
27	0.0448	0.0370	0.0449	0.0355
28	0.0448	0.0370	0.0449	0.0355
29	0.0448	0.0370	0.0449	0.0355
30	0.0448	0.0370	0.0449	0.0355
31	0.0448	0.0370	0.0449	0.0355
32	0.0448	0.0370	0.0449	0.0355
33	0.0448	0.0370	0.0449	0.0355
34	0.0448	0.0370	0.0449	0.0355
35	0.0448	0.0370	0.0449	0.0355
36	0.0448	0.0370	0.0449	0.0355
37	0.0448	0.0370	0.0449	0.0355
38	0.0448	0.0370	0.0449	0.0355
39	0.0448	0.0370	0.0449	0.0355
40	0.0448	0.0370	0.0449	0.0355

Tableau 32. Débogage scenario 11

Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
11		
	0.0370 0.0355	0.0448 - 0.0449
	Tableau 33. Résultat Scénario 11	

Scenario 12				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	3.2815	0.2524	1.4024	0.6994
02	1.3179	0.6407	0.6039	0.8683
03	0.7646	0.7817	0.3007	0.9292
04	0.5092	0.8539	0.1810	0.9564
05	0.3968	0.8862	0.1304	0.9680
06	0.3153	0.9090	0.1189	0.9699
07	0.2670	0.9227	0.0896	0.9770
08	0.2385	0.9296	0.0744	0.9837
09	0.2092	0.9382	0.0628	0.9842
10	0.1776	0.9483	0.0532	0.9867
11	0.1616	0.9529	0.0484	0.9884
12	0.1418	0.9587	0.0485	0.9866
13	0.1365	0.9599	0.0417	0.9892
14	0.1238	0.9638	0.0318	0.9925
15	0.1275	0.9639	0.0395	0.9898
16	0.1077	0.9702	0.0343	0.9913
17	0.1061	0.9688	0.0406	0.9895
18	0.0967	0.9703	0.0257	0.9932
19	0.1098	0.9681	0.0298	0.9923
20	0.0914	0.9732	0.0303	0.9921
21	0.0891	0.9733	0.0297	0.9922
22	0.0823	0.9748	0.0320	0.9922
23	0.0918	0.9734	0.0315	0.9917
24	0.0832	0.9761	0.0276	0.9934
25	0.0798	0.9767	0.0221	0.9945
26	0.0916	0.9738	0.0282	0.9934
27	0.0752	0.9779	0.0286	0.9916
28	0.0861	0.9752	0.0230	0.9929
29	0.0923	0.9741	0.0237	0.9946
30	0.0798	0.9760	0.0258	0.9926
31	0.0805	0.9771	0.0215	0.9941
32	0.0660	0.9804	0.0204	0.9955
33	0.0675	0.9799	0.0311	0.9907
34	0.0740	0.9784	0.0258	0.9927
35	0.0785	0.9762	0.0244	0.9927
36	0.0833	0.9767	0.0228	0.9946
37	0.0765	0.9776	0.0266	0.9926
38	0.0694	0.9800	0.0227	0.9943
39	0.0706	0.9784	0.0277	0.9915
40	0.0657	0.9806	0.0412	0.9894

Tableau 34. Débogage scenario 12

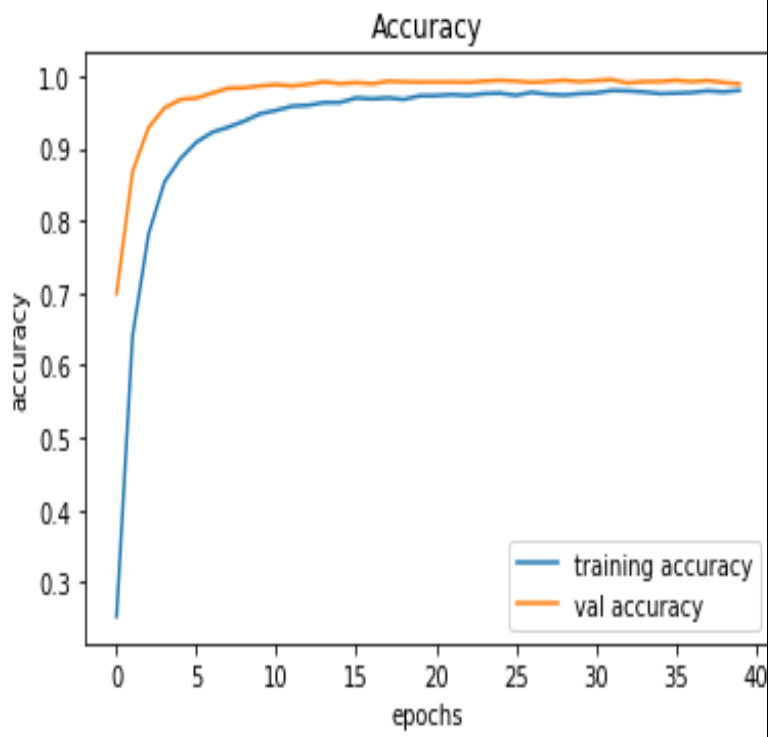
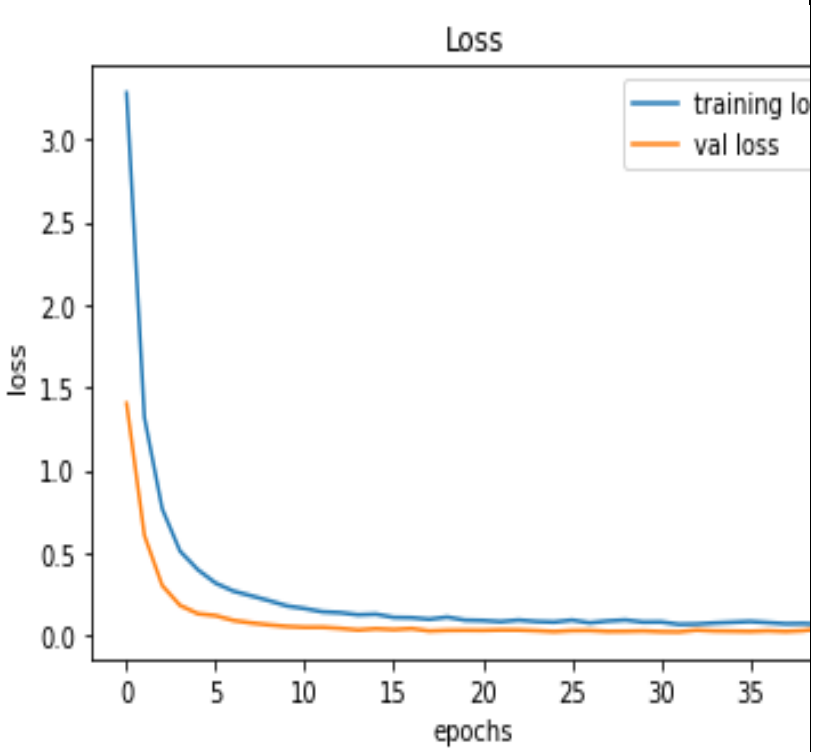
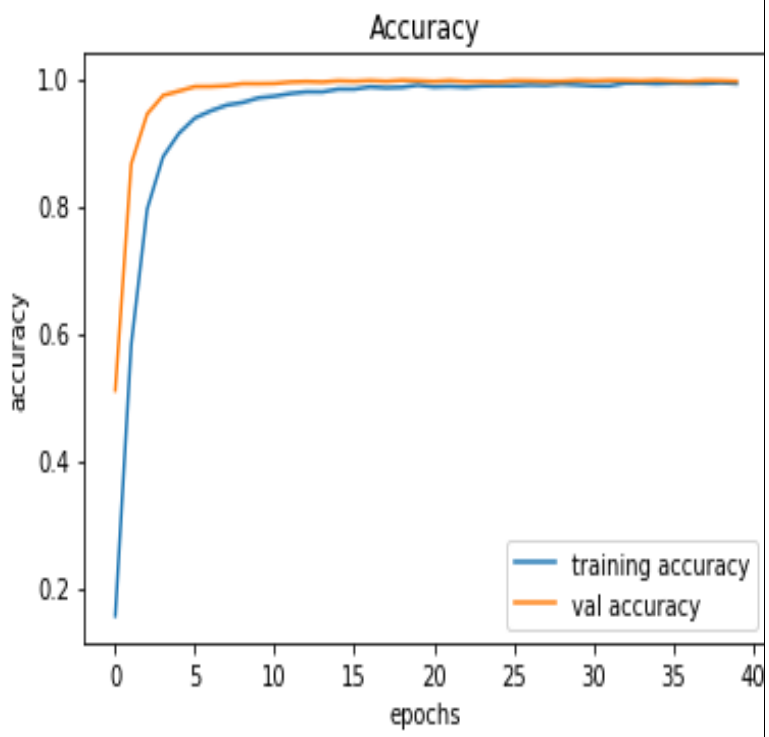
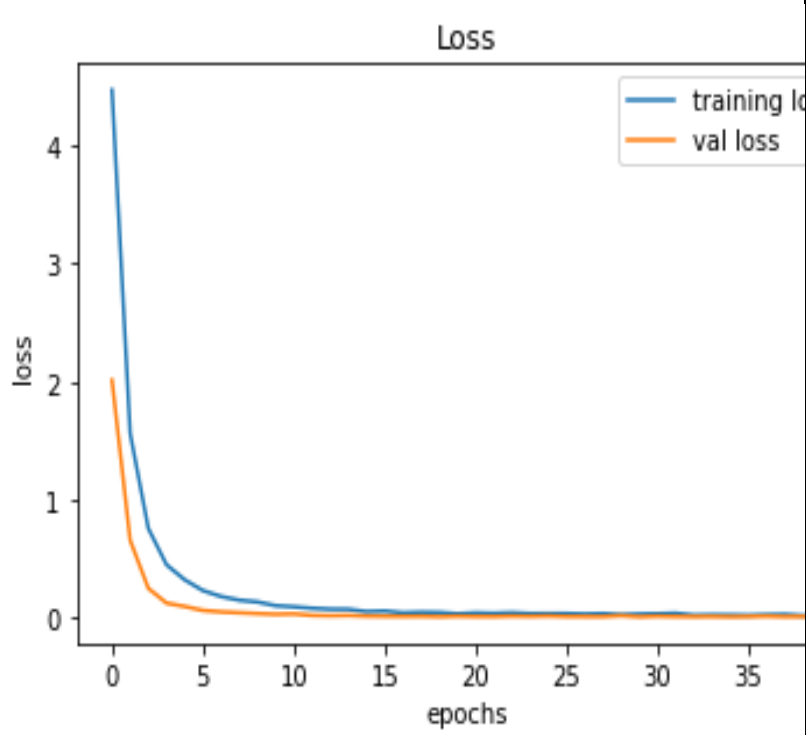
Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
12		
	0.9806 0.9894	0.0657 - 0.0412

Tableau 35. Résultat Scénario 12

Scenario 13				
Epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
01	4.4686	0.1561	2.0138	0.5106
02	1.5625	0.5818	0.6580	0.8661
03	0.7586	0.7951	0.2522	0.9445
04	0.4515	0.8775	0.1273	0.9737
05	0.3247	0.9142	0.0989	0.9806
06	0.2335	0.9383	0.0656	0.9876
07	0.1834	0.9495	0.0531	0.9878
08	0.1522	0.9586	0.0458	0.9890
09	0.1372	0.9626	0.0390	0.9922
10	0.1057	0.9699	0.0331	0.9921
11	0.0962	0.9725	0.0359	0.9925
12	0.0836	0.9768	0.0234	0.9948
13	0.0760	0.9794	0.0206	0.9955
14	0.0749	0.9792	0.0229	0.9948
15	0.0556	0.9836	0.0177	0.9967
16	0.0601	0.9836	0.0165	0.9959
17	0.0453	0.9872	0.0156	0.9971
18	0.0497	0.9858	0.0164	0.9959
19	0.0477	0.9866	0.0133	0.9974
20	0.0371	0.9901	0.0171	0.9967
21	0.0444	0.9872	0.0149	0.9957
22	0.0409	0.9881	0.0145	0.9971
23	0.0461	0.9868	0.0179	0.9954
24	0.0387	0.9888	0.0161	0.9957
25	0.0394	0.9892	0.0189	0.9950
26	0.0391	0.9889	0.0147	0.9968
27	0.0350	0.9903	0.0142	0.9966
28	0.0386	0.9898	0.0136	0.9959
29	0.0289	0.9917	0.0215	0.9959
30	0.0348	0.9904	0.0124	0.9972
31	0.0374	0.9889	0.0172	0.9966
32	0.0417	0.9886	0.0143	0.9973
33	0.0255	0.9930	0.0129	0.9972
34	0.0279	0.9931	0.0140	0.9966
35	0.0271	0.9921	0.0114	0.9974
36	0.0244	0.9930	0.0139	0.9964
37	0.0287	0.9926	0.0171	0.9957
38	0.0316	0.9925	0.0127	0.9972
39	0.0218	0.9938	0.0141	0.9968
40	0.0282	0.9923	0.0158	0.9959

Tableau 36. Débogage scenario 13

Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
13		
	0.9923 0.9959	0.0282 - 0.0158
	Tableau 37. Résultat Scénario 13	

Epoch	loss	accuracy	val_loss	val_accuracy
01	3.7636	0.2087	1.5357	0.6275
02	1.2587	0.6537	0.4589	0.9176
03	0.5876	0.8370	0.1894	0.9649
04	0.3927	0.8930	0.1343	0.9809
05	0.2886	0.9238	0.0915	0.9828
06	0.2478	0.9312	0.1020	0.9787
07	0.1984	0.9446	0.0618	0.9850
08	0.1627	0.9550	0.0540	0.9898
09	0.1458	0.9583	0.0425	0.9908
10	0.1335	0.9627	0.0376	0.9920
11	0.1063	0.9697	0.0336	0.9935
12	0.0983	0.9716	0.0320	0.9920
13	0.0937	0.9735	0.0280	0.9930
14	0.0860	0.9752	0.0310	0.9927
15	0.0846	0.9763	0.0238	0.9949
16	0.0771	0.9771	0.0301	0.9918
17	0.0742	0.9791	0.0256	0.9936
18	0.0669	0.9805	0.0286	0.9932
19	0.0601	0.9818	0.0211	0.9946
20	0.0548	0.9829	0.0280	0.9936
21	0.0610	0.9816	0.0259	0.9943
22	0.0543	0.9842	0.0220	0.9944
23	0.0476	0.9857	0.0168	0.9958
24	0.0450	0.9863	0.0192	0.9958
25	0.0578	0.9831	0.0305	0.9926
26	0.0552	0.9828	0.0201	0.9953
27	0.0486	0.9854	0.0201	0.9950
28	0.0447	0.9865	0.0212	0.9953
29	0.0444	0.9866	0.0221	0.9950
30	0.0433	0.9867	0.0260	0.9945
31	0.0429	0.9868	0.0207	0.9953
32	0.0412	0.9880	0.0203	0.9964
33	0.0405	0.9884	0.0191	0.9958
34	0.0390	0.9884	0.0147	0.9959
35	0.0378	0.9886	0.0220	0.9959
36	0.0422	0.9878	0.0187	0.9952
37	0.0396	0.9881	0.0161	0.9960
38	0.0400	0.9884	0.0183	0.9954
39	0.0414	0.9882	0.0179	0.9960
40	0.0408	0.9879	0.0201	0.9940
41	0.0392	0.9888	0.0171	0.9955
42	0.0391	0.9878	0.0195	0.9955
43	0.0451	0.9865	0.0172	0.9962
44	0.0366	0.9893	0.0164	0.9964
45	0.0314	0.9909	0.0177	0.9962
46	0.0339	0.9896	0.0173	0.9960
47	0.0431	0.9885	0.0216	0.9953
48	0.0354	0.9899	0.0199	0.9954
49	0.0295	0.9914	0.0182	0.9967

Tableau 38. Débogage scenario 14

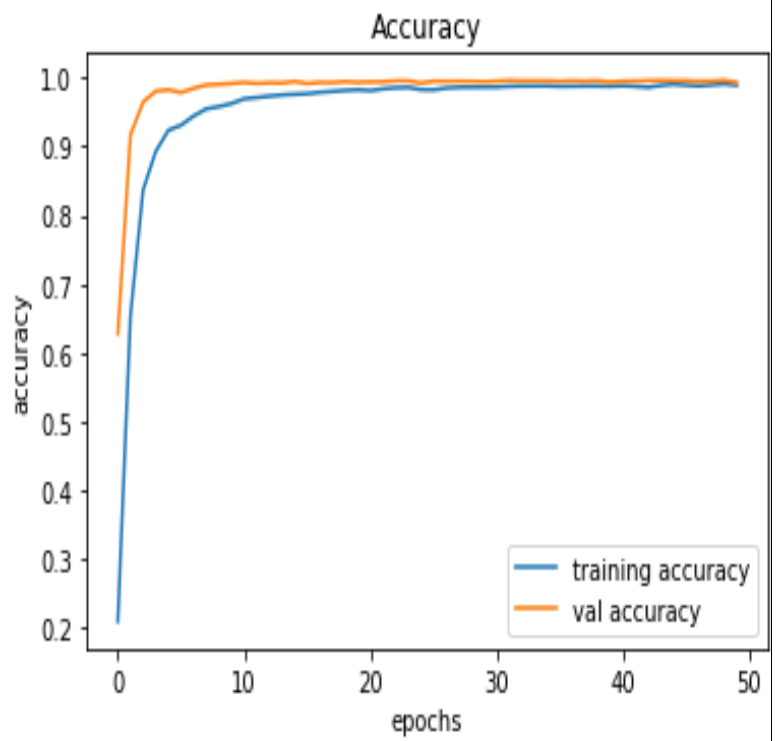
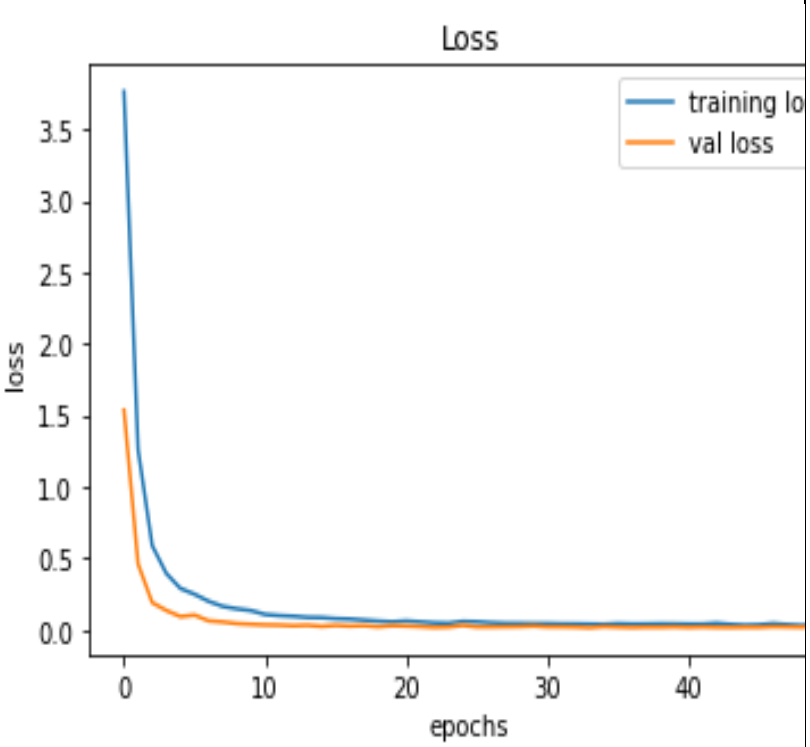
Scénario	Accuracy (apprentissage-validation)	Loss (apprentissage-validation)
14		
	0.9914 0.9967	0.0295 - 0.0182

Tableau 39. Résultat Scénario 14

ALGORITHEM 1 : CNN

Variables
data *est de type* ***LISTE[]***
labels *est de type* ***LISTE[]***
Début

Initialisation des variables *: attribuer la valeur 43 a la variable* ***classes***
Initialisation des variables *: attribuer la valeur du chemin a la variable* ***cur_path*** *en utilisant os.getcwd()*
for i in range (classes) *// pour la récupération des images et de leurs étiquettes*
    path ⟵*jointer le lien du projet avec l'emplacement du dataset en utilisant* ***join(string,string,int)***
    images ⟵*returner liste des noms des fichiers dans un repertoire en utilisant* ***os.listdir(path)***
    Afficher(i) *// pour tester l'importation des images selon les dossier du dataset (43 dossiers)*
    for a in images
        ouvrir tous les images en utilisant ***Image.open(path)***
        *redimmensionner l'image avec 30*x*30 en utilisant* ***resize(30,30)***
        ajouter l'image a la liste ***data[]*** *en utilisant* ***append(object)***
        ajouter la valeur du variable ***i*** *a la liste* ***labeles[]*** *en utilisant* ***append(object)***
        afficher message d'erreur dans le cas de la male prétairetement d'image par classes
    end
end
convertir les listes ***data[]*** *et* ***labels[]*** *en liste de numpy en utilisant* ***np.array(liste[])***
affichage des listes
deviser ***data[]*** *entre entrainement et test en utilisant*
// train_test_split(*arrays, test_size, train_size, random_state, shuffle=True, stratify)
affichage le tableau x_train //tableau d'image à former
affichage le tableau y_train //tableau d'ID de la classe correspondante
conversion des labels en un seul encodage a chaud en utilisant ***to_categorical(y,classes)*** *// hot encoding*
construire le model //Figure 22
compiler le model en utilisant ***compile(optimize,loss,matrice)***
fixer le nomber des epochs en utilisant ***fit(x,y,batch_size,num_epoch,validation_data)***
enregistrer le model en utilisant ***save(file_name)***
afficher la configuration utilisé en utilisant ***summary()*** *// car en vas utiliser plusieurs scénarios*
afficher les courbes des résultats(accuracy et loss) en utilisant ***plot(args,scaley,data)***
lire le fichier ***Test.csv*** *en utilisant* ***read_csv(path)*** *// tester accuracy on test data set*
labels[]⟵*attribuer les valeurs récupéré du champ* ***ClassId*** *a partir du fichier* ***Test.csv***
imgs[]⟵*attribuer les valeurs récupéré du champ* ***Path*** *a partir du fichier* ***Test.csv***
for img in imgs
    Ouvrir img en utilisant ***Image.open(path)***
    *redimmensionner l'image avec 30*x*30 en utilisant* ***resize(30,30)***
    convertir la liste ***data[]*** *en liste de* ***numpy*** *en utilisant* ***np.array(liste[])***
end
x_test ⟵*attribuer la liste data déjà converté en* ***numpy arrary***
generer la classe prediction en utlisant ***model.predict_classes(x_test)***
afficher le score du l'accuracyen utlisant ***accuracy_score(labels,pred)***
Fin algorithm

Figure 28. Pseudo code Algorithme CNN

ALGORITHEM 2 : TEST

Début

importer le model déjà entrainé en utilisant ***load_model(path)***

//*attribuer les valeurs du dictionnaire*

Classes { *1 :'Limite de vitesse (20km/h)',*

2:'Limite de vitesse (30km/h)',

3:'Limite de vitesse (50km/h)',

4:'Limite de vitesse (60km/h)',

5:'Limite de vitesse (70km/h)',

6:'Limite de vitesse (80km/h)',

7:'Fin de la limite de vitesse (80km/h)',

8:'Limite de vitesse (100km/h)',

9:'Limite de vitesse (120km/h)',

10:'Dépassement Interdit',

11:'Aucun véhicule passant de plus de 3,5 tonnes',

12:'Droit de passage à l intersection',

13:'Route prioritaire',

14:'Rendement',

15:'Stop',

16:'Aucun véhicule',

17:'Véhicules> 3,5 tonnes interdits',

18:'Entrée interdite',

19:'Mise en garde générale',

20:'Virage dangereux à gauche',

21:'Virage dangereux à droite',

22:'Double virage',

23:'Route cahoteuse',

24:'Route glissante',

25:'La route se rétrécit à droite',

26:'Travaux routiers',

27:'Des signaux de trafic',

28:'Piétons',

29:'Enfants traversant',

30:'Traversée de vélos',

31:'Méfiez-vous de la glace/neige',

32:'Traversée d animaux sauvages',

33:'Vitesse finale + dépassement des limites',

34:'Tourner à droite',

35:'Tourner à gauche',

36:'Droit seulement',

37:'Allez tout droit ou droit',

38:'Allez tout droit ou à gauche',

39:'Restez à droite',

40:'Restez à gauche',

41:'Rond-point obligatoire',
42:'Fin de non-passage',
43:'Fin sans dépassement véh> 3,5 tonnes' ***}***

initialiser l'interface graphique en utilisant ***tkinter***

Func classification(*path****)***

ouvrir image en utilisant Image.open(path)

*redimmensionner l'image avec 30*x*30 en utilisant* ***resize(30,30)***

insérer axe qu'apparaitre a la position de l'axe de la liste utilisant ***expand_dims(list[],axis)***

créer une liste en numpy en utlisant numpy.array(liste)

afficher les nouveaux dimmension de l'image (matrice)

generer la classe prediction en utlisant ***model.predict_classes(liste[0])***

attribuer la resultat préduction avec notre dictionnaire utilisant ***classes(predection+1)***

afficher le nom de la classe

relier l'affichage avec l'interface

Fin

Func button_classifier(*path****)***

déclarer le boutton

configurer le boutton

emplacement du boutton dans l'interface graphique

Fin

Func upload()

afficher la fenetre de dialogue pour sélectionner une photo pour la classer

ouvrir l'image

afficher l'image

envoyer le lien de l'image au button de classification en utlisant la fonction qu'on developé

Fin

déclarer un button et le relier avec la fonction upload()

Fin algorithem

Figure 29. Pseudo code Algorithme TEST