REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES
--T L E M C E N--

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Mémoire de fin d'étude

Pour l'obtention du diplôme de Master

Filière : Automatique
Spécialité : Automatique

Présenté par : BERREZOUG Lotfi
BENYAGOUB Ali

Thème

Utilisatition des Algorithmes de L'Intéligence Artificielle appliquer à l'Aide à la Conduite

Soutenu publiquement, le 08 / 07 / 2021 , devant le jury composé de :

M. BRAHAMI Mostapha Anwar	MCB	ESSA. Tlemcen	Président
M. ABDELLAOUI Ghouti	MCB	ESSA. Tlemcen	Directeur de mémoire
M. ABDI Sidi Mohammed	MCB	ESSA. Tlemcen	Examinateur 1
Mme. NEDJAR Imene	MCB	ESSA. Tlemcen	Examinateur 2

Année universitaire : 2020/2021


بسم الله الرحمن الرحيم

Remerciements

Nous tenons premier lieu remercier Dieu tout puisant de nous avoir accordé la force et le courage de mener ce travail à terme.

Nous tenons à adresser nos sincères remerciements à notre encadreur de thème le Dr.ABDELLAOUI Ghouti, Enseignant à l'école supérieure en sciences appliquées Tlemcen pour sa disponibilité, pour sa lecture , suggestion et remarques et surtout pour sa confiance sans limite mise en mois tout au long de ce projet de recherche.

Également du président de jury **BRAHAMI Mostapha Anwar** pour l'intérêt qu'il a porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par ses propositions.

Encore, de remercier respectueusement les membres de mon jury d'examen **Monsieur ABDI Sidi Mohammed** et **Madame NEDJAR Imene**, qui ont accepté d'examiner cet humble travail que j'espère à la hauteur de mes ambitions bien modestes.

Nous remercions particulièrement nos famille pour leur soutien moral tout au long de ce travail, merci de nous avoir encouragé, et cru en nous.

Nous remercions également nos amis nos collègues de travail, qui nous ont encouragé tout au long de ce projet et nous ont beaucoup aidé.

ملخص

من أجل تسهيل عملية القيادة وتجنب حوادث الطرق التي أصبحت متزايدة في الآونة الأخيرة ، والتي أصبحت خطرة على حياة الإنسان بشكل عام وعلى حياة الأطفال بشكل خاص ، يولي الإنسان قدرا كبيراً من الاهتمام في البحث عن حل مناسب للخروج من هذه المشكلة أو على الأقل لتقليلها. لذا فإن من بين الحلول هو عبور مجال رؤية الكمبيوتر الذي ندرسه ونستخدمه لتحديد هذا النوع من الصور.

يتمثل عملنا في تنفيذ تطبيق يستطيع معرفة إشارات المرور من الصورة. بعد خطوة معالجة مسبقة ، موضحة بالتفصيل في فصل "التصميم والتطوير" ، نستخدم شبكة عصبية باستخدام الوسيلة **keras** التي نجحت في تحديد علامات حدود السرعة وعرضها.

الكلمات المفتاحية : Python , Tensorflow , keras , L'apprentissage en profondeur (Deep learning) . Réseau de neurone (neuron network)

Résumé

Afin de faciliter le processus de conduite et d'éviter les accidents de la route, qui sont devenus incrémentales ces derniers temps, et qui sont devenus dangereux pour la vie humaine en général et pour la vie des enfants en particulier, donc les chercheurs font une grande attention dans la recherche d'une solution appropriée pour sortir de ce problème ou au moins de le réduire. Alors la solution est de franchir le domaine de la vision par ordinateur que nous étudions et utilisons pour identifier ce type d'images.

Notre travail consiste à implémenter une application bureau permettant d'extraire des panneaux de signalisation à partir d'une image. Après une étape de prétraitement, expliquée en détail dans le chapitre "Conception et Développement", nous utilisons un réseau de neurone a l'aide de la librairie keras qui a réussi à identifier les panneaux de limitation de vitesse et les affichés.

Mots clé : Tensorflow, keras, L'apprentissage en profondeur (Deep learning), Réseau de neurone (neuron network), Python.

Abstract

In order to facilitate the process of driving and avoid road accidents, which have become incremental in recent times, and which have become dangerous for human life in general and for the lives of children in particular, human being makes a great deal of attention in finding an appropriate solution to get out of this problem or at least to minimize it. The solution was to break into the field of computer vision that we are studying and using to identify this type of image in order to offer a set of driving assistance tools.

Our work consists in implementing application to extract traffic signs from an image. After a pre-processing step, explained in detail in the chapter "Design and Development", we use a neural network using the keras library which has succeeded in identifying speed limit signs and displaying them.

key words : Tensorflow, keras, L'apprentissage en profondeur (Deep learning), Réseau de neurone (neuron network), Python.

Table des matières

Table des figures

Introduction générale

La détection et la reconnaissance de la signalisation verticale est une problématique majeure dans l'analyse de scènes routières par traitement d'images. Les applications sont nombreuses, telles que le calcul d'itinéraire avec estimation des temps de parcours, le développement d'outils pour la gestion et la maintenance du patrimoine routier, les systèmes d'aide à la conduite en temps réel ou, en lien avec la robotique, l'automatisation des véhicules ou encore le mise au point d'une nouvelle génération d'outils multimedia sur le web pour la navigation 3D géographique.

Quelles que soient les applications, les méthodes de détection et de reconnaissance se heurtent aux difficultés classiquement rencontrées en analyse d'images en environnement non contrôlé. Même si la qualité des images s'est largement améliorée ces dernières années avec l'apparition de capteurs numériques toujours plus performants, les variations d'éclairement engendrent des modifications de couleur apparente, des ombres portées, des réflexions ou des contrejours. Par ailleurs, des distorsions géométriques ou des rotations peuvent apparaître suivant l'angle d'observation et l'échelle des panneaux. Leur aspect peut également varier en fonction de leur état d'usure et d'éventuelles salissures ou dégradations. Enfin la scène routière en elle-même, de par sa complexité, est source de nombreux distracteurs et d'occultations partielles des objets. Les algorithmes développés pour la détection et l'identification de la signalisation verticale doivent répondre de manière robuste aux diverses perturbations observées et tenir compte de la variabilité d'aspect des panneaux.

Cependant, les panneaux de signalisation sont des objets manufacturés et normalisés.

En tant que tels, ils présentent donc un certain nombre de caractéristiques géométriques, colorimétriques et pictographiques permettant de les distinguer de leur environnement. Dans cet ouvrage, nous utiliserons les réseaux de neurone artificiel.

Le chapitre 1 présente un état de l'art qui contient quelque généralité sur la détection des panneaux de signalisation routière, et le domaine d'application de cette méthode, et des méthodes de détection.

Dans le chapitre 2 nous présentons les types d'image et le système de traitement d'image et les différentes domaines d'application.

Et dans le dernier chapitre nous donnons les différentes types de panneaux de signalisation, ensuite les outils pour développé notre détection, et enfin la méthode de développement et la création du réseau de neurone, et faire des tests.

Chapitre 1

État de l'art sur la détection des panneaux de signalisation

Introduction

Ces dernières années, de nombreux systèmes d'assistance avancés aide à la conduite (ADAS) ont été développés afin de réduire le nombre d'accidents sur les routes. Cette technologie est mise en œuvre dans la plupart des nouveaux modèles de voitures de toutes les marques et permet au conducteur de connaître l'état de la route d'une manière précise. Ces systèmes permettent aussi d'avertir le conducteur de l'état des routes, signaler les collisions possibles, détecter les obstacles sur la route et départs de voie ou détecter et communiquer au conducteur les panneaux de signalisation qui se trouvent sur la route.

En ce qui concerne les systèmes de détection et de reconnaissance du signal (RSR), c'est un système qui agit comme un observateur infatigable en combinaison avec d'autres systèmes de voiture pour assurer la sécurité des passagers du véhicule. C'est généralement un avertissement automatique qui va avertir le conducteur en émettant un bip et en montrant directement la limitation du chemin via un ordinateur de bord. Ce type de technologie n'est pas un standard puisque chaque fabricant lui donne des caractéristiques.

Cette détection et reconnaissance de panneaux de signalisation est possible grâce à l'utilisation de caméras vidéo qui sont normalement situés sur le pare-chocs avant ou sur le pare-brise du véhicule afin d'enregistrer les images de la route.

Une fois le signal vidéo capturé, c'est l'ordinateur qui est responsable de la lecture du signal et de son interprétation. Le système doit être suffisamment robuste et rapide pour avertir le conducteur en temps réel. Avec cela, le conducteur peut consulter en cas de dépassement de la limitation de la route pour augmenter la sécurité et oubliez les amendes éventuelles si vous suivez les instructions affichées par le système.

Actuellement, il y a des voitures qui ont déjà été munies par ces systèmes comme nous l'avons dit auparavant et il y a aussi des applications qui peuvent être utilisées via le mobile.

1.1 Définition

Comme son nom l'indique, la reconnaissance des panneaux est un équipement automobile qui lit et interprète les panneaux de signalisation sur la route, afin d'en informer le conducteur au cas où il n'aurait pu les voir. Les panneaux de limitation de vitesse et d'interdiction de dépassement sont notamment concernés [1].

1.2 Principe

La reconnaissance des panneaux fonctionne grâce à une caméra installée derrière le rétroviseur intérieur. Elle détecte un panneau situé à gauche, à droite ou au-dessus de la route et le compare avec une base de données interne. Une fois le panneau reconnu, le conducteur est averti de la situation grâce à un visuel sur le GPS ou l'instrumentation [1].

1.3 État actuel

Dans la liste d'équipements d'un véhicule, la reconnaissance des panneaux est très souvent associée à d'autres aides à la conduite qui pourraient en bénéficier : le régulateur de vitesse adaptatif, l'alerte de survitesse, l'alerte de franchissement de ligne... Comme ces systèmes, elle se propage petit à petit mais reste un équipement "haut de gamme" [1].

1.4 Évolution

La reconnaissance des panneaux fait partie des fonctions de la voiture autonome, capable de rouler sans l'intervention d'un conducteur. Cette dernière reconnait les panneaux de signalisation, tout en analysant la position des autres véhicules afin de prendre des décisions. Le tout, bien entendu, à une vitesse de calcul extrêmement élevée [2].

1.5 Anecdote

Le conducteur peut généralement paramétrer la manière dont son système de reconnaissance de panneaux affiche ses informations : emplacement, taille, type d'alerte... À noter que chaque information disparait progressivement au fur et à mesure que l'on s'éloigne du panneau concerné dans le but de conserver un affichage bien à jour.

1.6 Système RSR intégré dans les voitures

Les fabricants qui ont actuellement intégré ces systèmes dans leur véhicules sont nombreux tels que : Audi, BMW, Citroën, Ford, Honda, Infiniti, Jaguar, JEEP, Land Rover, Lexus, Mercedes, Nissan, Opel, Peugeot, Porsche, Renault, Toyota, Volkswagen et Volvo, ci-dessous un exemple d'application TSR dans une voiture de la marque Opel (Figure 1.1) [2].

Figure 1.1: Exemple d'application RSR dans une voiture de la marque Opel

En ce qui concerne les systèmes RSR intégrés dans les véhicules, il existe deux types d'applications de la technologie RSR:

1. Application passive: Consiste à informer le conducteur par le biais de pictogrammes ou des sons qu'il entre dans une zone qui a une nouvelle limitation indiquée pour un feu de circulation. Dans ce cas, ce sera le conducteur qui prendra la décision de respecter le signal ou non [2].

2. Application active: Consiste à intervenir automatiquement sur la voiture quand il détecte le signal. Par exemple, si le conducteur conduit à une vitesse excessif et le système RSR détecte un signal d'arrêt et la voiture n'interprète pas que le conducteur a l'intention de s'arrêter, l'ordre de freiner sera directement dirigé vers la voiture pour éviter un éventuel accident [2].

1.7 Applications mobiles avec RSR

Les applications mobiles offrent une application passive de la technologie RSR et qui n'ont aucun type de communication avec l'ordinateur du véhicule dans le cas où la voiture en a un. Plusieurs applications permettent de détecter et de reconnaître les panneaux de circulation tels que myDriveAssist ou Skylten EU. L'application la plus précise et professionnelle est myDriveAssist, qui permet de détecter les limitations de vitesse et les panneaux d'interdiction utilisant simplement l'appareil photo Smartphone il permet également d'avertir le conducteur dans le cas où il circule trop rapide grâce à l'utilisation permanente du signal GPS (Figure 1.2) [2].

Figure 1.2: Application mobile myDriveAssist

Cette application peut être téléchargée pour le système d'exploitation Android et pour IOS. Elle peut également être utilisée dans plusieurs pays et permet la mise à jour des limites de vitesse par les utilisateurs que le mobile ne détecte pas en raison du mauvais état de quelques signes [2].

Parmi les inconvénients de l'utilisation de cette application que la qualité dépend de la caméra du Smartphone et que la batterie de l'appareil se décharge rapidement [2].

1.8 Travaux de recherche

Dans cette section, nous discuterons brièvement des lignes générales qui suivent les principaux travaux de recherche sur la reconnaissance des panneaux de signalisation routière.Un grand nombre d'articles sur ce sujet [21] ont été publiés. Ces systèmes comportent deux parties fondamentales : la détection des signaux dans l'image et la reconnaissance dans une base de données [2].

La plupart des articles scientifiques examinés utilisent la couleur comme l'un des principales fonctionnalités pour détecter les panneaux de circulation à côté des fonctions basée sur la forme.

Pour la détection et la segmentation de la couleur, la grande majorité des articles coïncidaient avec l'utilisation de seuils dans un espace colorimétrique HSV (Tonalité, Saturation et Valeur), car il est plus intuitif que les autres et nous permet également de dépendre moins de luminosité ou d'ombres de l'image.

Le seuillage sur cet espace peut être plus pratique dans certaines situations et généralement moins sensible aux changements dans les conditions d'éclairage [2].

En ce qui concerne la reconnaissance des signaux, il a été observé que la méthode SVM (Support Vector Machine) est une méthode d'apprentissage automatique très efficace dans ce cas.

Cette méthode présente de nombreux avantages tels qu'il existe une multitude d'implémentations déjà disponibles, l'inconvénient de cette méthode est que peut être très complexe et coûteuse avec un nombre élevé de classes.

L'autre méthode de classification, qui a fait l'objet de recherches approfondies dans les années passé de reconnaissance, est les réseaux de neurones. Cette méthode a démontré ses atouts pour des tâches telles que la classification, la localisation et la détection des images [2].

Nous pouvons dire que c'est la méthode la plus puissante innovant pour réaliser des algorithmes de reconnaissance en vision artificielle. Le principal inconvénient réside dans la nécessité d'un matériel très puissant, actuellement il existe des techniques novatrices de reconnaissances d'objet de formes libres ont été développées.Ces méthodes se basent sur l'extraction des points

caractéristiques et le calcul de descripteurs invariants pour permettre une reconnaissance robuste et fiable de toute forme géométrique. Tel que : SURF, SIFT ... etc. Chacun des articles publiés sur ce sujet possèdent leurs propre approche qui diffère de l'autres soit pour les étapes de la partie de détection ou la partie de reconnaissance.Un seul article [26] ressemble d'une manière générale, à notre approche mais il ya des différences dans les détails des étapes.Il est important de mentionner que le code source des applications existantes n'est pas disponible, aussi une haute sécurité couvre les mémoires contenant ce genre de logiciel, ça peut aller jusqu'à l'auto-effaçage en cas de tentative d'accès à cette mémoire.En plus, l'exclusivité de notre application est qu'elle est dédiée au code de la route de l'Algérie [2].

1.9 Les voitures Tesla

Tesla déploie une nouvelle mise à jour pour ses véhicules. Et parmi les nouveautés : la détection des panneaux de limitation de vitesse qu'elle va pouvoir prendre en considération et un carillon dès que c'est à vous d'y aller.

Une voiture autonome rassurante, c'est quand même le rêve de chacun. Pour cela, il faut qu'elle soit capable de comprendre le mieux possible son environnement. Les différents capteurs peuvent détecter des éléments alentour et adapter la conduite du véhicule aux possibles dangers. Mais jusqu'à présent, la lecture des panneaux indicateurs était plus difficile [2].

Tesla a déployé sa nouvelle mise à jour 2020.36. Celle-ci améliore notamment la gestion de la vitesse du véhicule électrique avec l'ajout d'une fonctionnalité appelée "Amélioration de l'assistance de vitesse". Le fabricant américain explique se servir "des caméras de votre voiture pour détecter les panneaux de limitation de vitesse afin d'améliorer la précision des données de limitation de vitesse sur les routes locales".

Une fois détectés, les panneaux de limitation de vitesse sont affichés dans la visualisation de conduite et utilisés pour définir l'avertissement de limitation de vitesse associé. Les paramètres de l'assistance de vitesse peuvent être ajustés depuis l'onglet Commandes du tableau de bord (Figure 1.3) [2].

Figure 1.3: Le tableau de bord d'une voiture Tesla

UN CARILLON POUR SIGNALER QUE LE FEU EST VERT Les feux de signalisation vont également être mieux pris en compte. Un "carillon de signalisation de feu vert" retentira lorsque vous pouvez redémarrer ou pour vous signaler que le véhicule vous devançant redémarrer, à condition que "le régulateur de vitesse ou le pilote automatique ne soit pas actif". Tesla prévient que c'est un avertisseur via une notification et que cela ne doit pas remplacer la prise de décision du conducteur ni remplacer sa responsabilité d'observation de son environnement.

Jusqu'à présent, vous deviez utiliser les molettes du volant pour modifier ou réinitialiser le régulateur de trafic. Grâce à la mise à jour, il suffira d'appuyer sur le compteur de vitesse ou sur le bouton du panneau de limitation de vitesse à l'écran.

Le déploiement a commencé et concernera progressivement tous les modèles [2].

1.10 Méthodes de détection

1.10.1 Sélection colorimétrique

Dans de nombreux cas, une première étape de détection consiste à identifier les pixels d'une image en couleurs correspondant aux caractéristiques colorimétriques du panneau. Il est alors nécessaire de fixer un ou des seuils au-delà duquel le pixel est classé comme appartenant au panneau ou non. De nombreux auteurs travaillent dans l'espace colorimétrique HSV et la segmentation est le plus souvent effectuée par seuillage de la composante de teinte (Hue), insensible aux

changements de luminance. Cette approche est utilisée par pour la sélection de panneaux rouges. Pour cette même catégorie de panneaux, une alternative, tout aussi efficace, consiste à considérer la composante normalisée R/(R+V+B), également indépendante des variations lumineuses ou, de manière similaire, en prenant en compte les rapports R/B et V/B. Dans le cadre d'applications en temps réel, l'espace colorimétrique YUV, obtenu à partir de transformations linéaires de l'espace RVB est utilisé par. Dans ces mêmes travaux, une correction chromatique est initialement effectuée sur toute l'image. Le coefficient de correction est obtenu en se basant sur les valeurs chromatiques des pixels de la chaussée, théoriquement grise (R, V et B égaux). Des espaces plus complexes comme CIE1 Lab ou CIECAM97 sont aussi utilisés. Quelles que soient les méthodes, le choix des seuils est délicat. Ils sont le plus souvent fixés à partir des normes concernant les panneaux ou à partir de méthodes empiriques. On peut noter cependant les travaux de, proposant un outil de seuillage adaptatif fondé sur l'algorithme d'Otsu et ceux de proposant un seuillage dynamique dans l'espace HSV [2].

L'utilisation de la couleur permet de sélectionner rapidement des zones d'intérêt dans l'image. Cependant elle présente l'inconvénient de focaliser rapidement la recherche sur une catégorie de panneaux (rouge, bleu, vert ou jaune). D'autre part, pour certaines signalisations, comme les panneaux blancs (fin d'interdiction), la couleur n'est pas une information suffisamment discriminante pour être utilisée [2].

1.10.2 Caractéristiques géométriques

Certaines équipes se basent sur l'analyse des caractéristiques géométriques des contours extraits de l'image de luminance. Dans le cas de la détection de panneaux triangulaires, une méthode consiste à filtrer les segments, issus de la détection de contours, en fonction de leurs pentes et de leurs longueurs et à vérifier si les segments restants appartiennent à un même triangle équilatéral.Une méthode intéressante proposée dans permet de calculer pour un objet donné des mesures de ressemblances aux formes de types cercle, rectangle et triangle [2].

Quelques auteurs proposent d'utiliser les algorithmes de vote, comme la transformée de Hough, la transformée de Hough circulaire, la transformée chinoise, la transfor- mée en symétrie

radiale et sa version étendue à la détection des polygones.

Une autre approche repose sur la recherche d'un modèle simplifié du panneau dans une image candidate. Les techniques de template matching peuvent alors être mises en œuvre sur une carte de distance aux contours calculée à partir de l'image en niveaux de gris. Une autre possibilité revient à ajuster un modèle simplifié du panneau dans une image candidate. L'optimisation de cet ajustement est ensuite réalisée par des algorithmes génétiques [2].

En s'affranchissant de l'information couleur, les algorithmes proposés sont susceptibles de sélectionner tous types de panneaux, y compris les blancs. Ainsi propose de détecter les panneaux de fin de limite de vitesse en recherchant les transitions "clair-sombre-clair" et en appliquant ensuite la transformée de Hough circulaire autour des zones d'intérêt. Cependant, L'efficacité de ces méthodes reste conditionnée à la qualité de l'extraction de contours et donc à l'amplitude et l'orientation des gradients. Par ailleurs, ces algorithmes conduisent généralement à des temps de calcul plus importants que lorsque la couleur est utilisée [2].

1.10.3 Combinaison de critères colorimétriques et géométriques

La stratégie la plus utilisée consiste à combiner les informations couleurs et géométriques, peuvent ainsi être utilisés comme une première étape pour la détection de panneaux de signalisation. A l'issue de cette pré-détection, une analyse géométrique des composantes connexes permet d'affiner la sélection des objets. Il est ainsi possible d'éliminer rapidement les objets selon leurs tailles ou la valeur de la compacité, définie par le rapport entre l'aire de l'objet et son périmètre au carré . Toutefois des déscripteurs plus élaborés sont souvent nécessaires pour effectuer une sélection plus fine. Le choix s'oriente le plus souvent vers la caractérisation du contour de la forme en utilisant par exemple les descripteurs de Fourier obtenus par la transformée de Fourier normalisée de la signature de l'objet. La circularité des objets rouge peut également être contrôlée selon l'approche définie dans . La distance des bords du contour de l'objet aux bords de sa boite englobante de l'objet est également une caractéristique intéressante qui peut permettre de discriminer cercles, triangles et rectangles.

La sensibilité de ces approches face à la variabilité des situations (occultations, changement

d'aspect...) représente une difficulté majeure.

A partir de l'image issue du filtre colorimétrique, les modèles d'ajustement de forme peuvent également s'appliquer en utilisant les algorithmes génétiques, les algorithmes de recuit simulé ou les algorithmes de sélection clonale ou d'essaims de particules.Il y a deux systèmes de détection de panneaux triangulaires par sélection colorimétrique et ajustement d'un modèle géométrique (RANSAC ou algorithme génétique) [2].

L'efficacité de ces méthodes est liée aux performances de la segmentation colorimétrique. Le regroupement des pixels en composantes connexes peut en particulier s'avérer problématique puisqu'il conditionne la forme de l'objet à analyser par la suite. Une sur-segmentation entraîne une division de l'objet d'intérêt en différentes parties. Au contraire, une sous-segmentation présente un risque de fusion de plusieurs composantes. On peut cependant penser que les mé- thodes de détection robustes, notamment vis-à-vis des occultations, sont susceptibles d'être robustes vis-à-vis d'une segmentation colorimétrique, que l'on pourrait qualifier de moyenne. Une alternative est d'introduire plus profondément l'information de couleur dans les méthodes géométriques par exemple en sélectionnant ou en pénalisant les contours en fonction des cou- leurs voisines [2].

1.10.4 Méthodes avec apprentissage

Certains travaux de détection sont basés sur les méthodes de classification qui consistent à comparer le vecteur de caractéristiques d'une observation à un vecteur de référence. Pour obtenir ces références, un apprentissage est classiquement réalisé sur un ensemble d'images représentatives. Différentes techniques d'apprentissage peuvent être utilisées, les plus connues étant Adaboost et SVM (Support Vector Machine). Moins classique, une méthode non super- visée, basée sur l'analyse des changements statistiques temporels des objets d'intérêt dans les séquences (apparition, grossissement, disparition). Les histogrammes multi-dimensionnels (de mesures de couleur ou de forme) entre deux images successives sont ainsi comparés et les objets sont localisés par rétro-projection d'histogramme. Dans ces travaux , une approche de détection par apprentissage est également présentée à partir de représentations globales de l'apparence de l'objet. En particulier, des détecteurs robustes, fondés sur l'hypothèse que l'image observée est une occurrence bruitée

d'une image d'apprentissage, sont proposés [2].

Enfin, les algorithmes de type Adaboost, dans lesquels un modèle de classification dit "fort" est construit à partir d'une combinaison de classifieurs dits "faibles" peuvent être développés dans ce cadre. Le modèle Adaboost en cascade défini par : considère un jeu de caractéristiques basé sur les ondelettes de Haar. Il est utilisé comme détecteur de panneaux sur des images en niveaux de gris. Notons cependant quelques différences entre ces deux approches. En effet, le détecteur opère sur l'ensemble de l'image alors qu'une va- riante du jeu d'ondelettes de Haar, appelée dipôles dissociés. Une dernière approche consiste à prendre en compte l'information couleur en intégrant dans le détecteur sept images monochromes (les composantes R, V , B, les composantes R,V ,B normalisées et l'image de luminance).Un autre exemple de détecteur fondé cette fois sur un apprentissage par SVM et utilisant des caractéristiques de forme et de couleurs est décrit. Les performances de ces méthodes restent conditionnées à la qualité de l'apprentissage effectué au préalable [2].

1.11 Méthodes de reconnaissance

L'étape de détection permet de générer une liste d'objets représentant potentiellement des panneaux de signalisation. La phase de reconnaissance consiste à identifier l'objet candidat parmi un ensemble de référence comprenant les panneaux à répertorier, ou à rejeter l'objet candidat si aucun modèle ne correspond. Parmi ces méthodes, de nombreux auteurs font appel aux techniques classiques de corrélation en comparant les objets candidats à chaque modèle de référence. En particulier la fonction ZNCC (Zero mean Normalized Cross-Correlation) permet de rester invariant aux changements d'illumination. Les objets candidats sont ré-échantillonnés pour être remis à l'échelle des panneaux de référence et la corrélation est effectuée le plus souvent sur l'ensemble du panneau. La corrélation peut également être réalisée en comparant le pictogramme extrait de l'objet candidat à une base de pictogrammes ou grâce à un système de reconnaissance de chiffres pour les panneaux de limite de vitesse [2].

Les objets sélectionnés par le détecteur présentent parfois une déformation géométrique due

à l'effet de perspective. Cette distorsion peut compliquer l'appariement de l'objet à une référence de la base, construite à partir d'images fronto-parallèles. Pour pallier cette difficulté, il est possible de rectifier l'image candidate, correspondant à la boite englobante de la composante connexe détectée, pour la remettre dans une géométrie. Des outils invariants comme les descripteurs SIFT.

Les autres méthodes choisies pour la reconnaissance nécessitent un apprentissage. Elles sont souvent basées sur des réseaux de neurones, et en particulier, le perceptron multi-couches. Les objets sont classés parmi trois catégories "panneau de limite de vitesse", "autres panneaux de signalisation" et "rejet". Plus rarement, d'autres types de réseaux de neurones ont été considérés dans le cadre de cette application. Citons les réseaux de fonction à base radiale et les ré- seaux ART (Adaptive Resonance Theory). Les méthodes de classification comme Adaboost et SVM peuvent également être envisagées dans l'étape de reconnaissance. Cette phase d'identification peut être effectuée par les méthodes bayésiennes, soit par estimation d'une dis- tribution gaussienne en incluant une information de suivi temporel, soit à partir d'un modèle de type fenêtre de Parzen à base d'un noyau de Laplace en utilisant les moments géomé- triques comme attributs. Enfin, de façon similaire aux outils de détection robuste définis dans la section précédente, un algorithme de reconnaissance robuste a été développé et appli- qué à l'identification de panneaux de signalisation [2].

1.12 Évaluation des performances

Dans la littérature, les auteurs s'attachent le plus souvent à détailler l'aspect méthodologique du système de détection et de reconnaissance de panneaux. Les outils d'évaluation utilisés et l'analyse des performances y sont présentés de manière succincte. De plus, l'évaluation concerne le plus souvent l'algorithme dans son ensemble, c'est-à-dire à la fin de la dernière étape de traitement (détection et reconnaissance) et les performances des étapes intermédiaires (le détecteur seul par exemple) ne sont pas décrites [2].

1.12.1 Outils d'évaluation

Pour évaluer les performances d'un algorithme de détection et de reconnaissance de panneaux, les résultats en sortie de l'algorithme sont en général comparés à une vérité-terrain, dans laquelle les panneaux de signalisation sont relevés par un opérateur humain. On peut ainsi facilement calculer un taux de détections correctes (TDC) et un taux de fausses alarmes (TFA) correspondant aux objets détectés et/ou reconnus par l'algorithme ne correspondant pas à la vérité-terrain.

Suivant les auteurs et les applications, les définitions de VP, N, FP et NbImg peuvent varier. Le plus généralement, dans le cadre de la détection de panneaux, on comptabilise le nombre d'objets correctement détectés : VP est ainsi le nombre de vrai positif. Cependant, lorsque les images sont acquises de manière successive dans une séquence par un véhicule en mouvement, un panneau apparaît sur plusieurs images et la possibilité de le détecter et de l'identifier est multiple. Ainsi, le nombre N peut être choisi comme le nombre de panneaux différents vus dans la séquence, ou alors comme le nombre total d'apparitions de panneaux dans une image de la séquence. Le deuxième cas est plus sévère car les meilleures performances sont obtenues seulement lorsque toutes les apparitions d'un panneau sont détectées.Les résultats sont présentés en distinguant ces deux définitions. D'autres auteurs présentent et comparent les résultats en fonction du nombre de répétition des détections d'un même panneau pris dans des images successives. FP correspond le plus souvent au nombre total d'objets détectés à tort comme panneau et NbImg est le nombre total d'images de la séquence. On remarque que ce taux de fausses alarmes peut ainsi être supérieur à 100 pour 100 (dans ce cas, le nombre moyen de fausses alarmes par image est supérieur à 1). Cette possibilité est assez inhabituelle dans la littérature.FP peut également correspondre au nombre d'images contenant au moins une fausse alarme. Dans ce cas, le taux maximum est obligatoirement inférieur ou égal à 1 [2].

A partir de ces résultats, deux outils sont classiquement utilisés pour comparer les méthodes et optimiser les paramètres :

- La courbe "Caractéristiques Opérationnelles du Récepteur (COR)" correspond aux va- riations du taux de détections correctes en ordonnée et du taux de fausses alarmes en abscisse en fonction des valeurs d'un paramètre. Dans les cas simples, une courbe située au-dessus d'une autre correspond à la méthode la plus efficace mais le plus souvent, les courbes se croisent et l'analyse est plus complexe. Elle dépend en particulier du taux de détections correctes minimum acceptable ou du taux de fausses alarmes maximum toléré.

- Le coefficient de similarité de Dice (DSC) également appelé F-mesure, est utilisé en complément de la courbe COR

  Il permet notamment de déterminer la valeur optimale d'un paramètre, c'est-à-dire, ce- lui offrant le meilleur « compromis » entre détections correctes et fausses alarmes. Par ailleurs, la largeur du pic de la courbe du DSC renseigne sur la sensibilité du détecteur par rapport à la valeur du paramètre choisi. Plus le pic est étroit, plus la valeur du paramètre va influer sur l'efficacité du détecteur [2].

1.12.2 Analyse des performances

Les travaux présentés ci-dessus donnent des résultats sur des bases comprenant un nombre relativement faible d'images. Quelques travaux de recherche proposent des résultats sur des bases de données de taille importante. Ainsi, dans le cadre d'un projet industriel avec Daimler-Benzr, présente une évaluation systématique en termes de détections correctes et fausses alarmes sur une grande base de données, comprenant plus de 20000 images. Plus récemment, un algorithme de pré-détection de panneaux de danger et d'interdiction a été évalué sur une base de 26000 images, soit 175 kilomètres de route. Citons également les travaux portant sur un système de détection de panneaux de fin d'interdiction. L'évaluation a été effectuée sur une base de 10000 images contenant 37 panneaux de fin de limite de vitesse différents.Les auteurs proposent d'évaluer de façon distincte la phase de détection et d'évaluation sur une base de 9500 paires d'images stéréoscopiques. Enfin, proposons une méthodologie d'analyse quantitative (à partir de courbes Precision-Recall, variante des courbes COR) et qualitatives (à partir de critères ergonomiques notamment) des performances

de systèmes de reconnaissance de panneaux de signalisation [2].

Les performances des différents algorithmes peuvent difficilement être comparées entre elles. En effet, les outils d'évaluation, les bases de test et les objectifs diffèrent selon les pu- blications. Par ailleurs, les auteurs étudient en général la signalisation verticale de leurs pays et on constate que les caractéristiques des panneaux diffèrent selon le pays. Cependant,Les performances de trois algorithmes de détection de panneaux sont comparées sur une même base d'images de scènes routières urbaines [2].

1.13 Systèmes opérationnels

D'une manière générale, les projets industriels sur la problématique des aides à la conduite sont peu nombreux (ou peu divulgués). Notons néanmoins les travaux au sein du consortium automobile allemand Daimler-Benzr portant sur les panneaux circulaires et triangulaires. Depuis 2008, deux autres constructeurs (Opelr et BMWr) commercialisent, sur un de leur véhicule, des systèmes de détection et reconnaissance de panneaux par vision. Ces système se présentent sous forme d'alerte en temps réel au conducteur. Celle-ci porte uniquement sur les panneaux de limite de vitesse et d'interdiction de dépasser. Les méthodes sous-jacentes et les limites de ces systèmes restent confidentielles.

Les autres systèmes opérationnels concernent le relevé automatique de la signalisation verticale pour la gestion du patrimoine. Dans ce cas, la contrainte temps réel n'est plus indis- pensable. Les outils de détection et de reconnaissance sont intégrés dans la chaîne complète de traitement qui comprend l'acquisition des données, et l'analyse de l'image. Dans plusieurs cas, l'acquisition est effectué par des véhicules équipés de caméras dans le visible et de capteur LASER. Pour la société Viametrisr 2.

A l'inverse, la société Geo3Dr 3 effectue une détection manuelle ou automatisée à partir de données LASER précédant une étape d'identification par analyse d'images, par comparaison du candidat détecté avec une base de modèles. Les données LASER donnent également accès à une mesure de rétro-réflexion des panneaux. Dans le projet RSR 4, l'équipe de l'Institut National

d'Optique (INO) du Canada a développé un système à partir de caméras infrarouge exploitant les propriétés de rétro-réflexion des panneaux lors de la phase de détection [2].

Une approche similaire est utilisée dans le projet VISUALISE (VISUAL Inspection of Signs and panEls), porté par les sociétés Euroconsult r 5 et 3M-Spain 6 et l'université d'Alcala (Espagne). Dans ce projet, on remarque que les relevés sont effectués de nuit.

La problématique est abordée de façon différente dans le projet Mobvisr 7. En effet, l'acquisition de la signalisation verticale est effectuée par un PDA équipée d'un appareil photo et d'un GPS. L'image obtenue est ainsi fortement focalisée sur le panneau. L'automatisation du relevé consiste uniquement à identifier le panneau forcément présent dans l'image [2].

Conclusion

L'analyse de la signalisation verticale de police constitue un problème relativement classique de détection et reconnaissance des formes. Ce domaine, fortement relié à l'analyse de données, à la vision par ordinateur et à l'intelligence artificielle, se caractérise par une grande variété de méthodes. On retrouve donc naturellement cette variété dans la littérature spécialisée sur l'analyse de la signalisation. On peut néanmoins identifier deux tendances.

La première est celle des chaînes de traitement bottom-up, fondées sur une succession d'étapes : extraction rapide de primitives, puis filtrage des détections selon des critères géométriques ou photométriques, abstraction de la représentation et analyse des formes extraites. Les progrès ré- cents de la qualité des images numériques et l'évolution de l'informatique ont permis de réelles avancées en termes de performances et remis au goût du jour certaines méthodes robustes, mais coûteuses. Ainsi, les techniques de vote ont connu un regain au début de la dernière décennie, notamment autour de la Transformée en Symétrie Radiale et de sa version adaptée aux poly- gones.

La seconde tendance a suivi les progrès des méthodes de détection et reconnaissance des formes. Non sans un certain effet de mode, des techniques telles que les réseaux de neurones, les mo- dèles déformables, les modèles d'apparence, les Machines à Vecteurs Support (SVM) ou les méta-classifieurs de type Ada-Boost ont successivement été employées. Cette évolution des

méthodes s'est également accompagnée d'un accroissement de la qualité et des performances.

Les techniques que nous proposons dans ce document sont représentatives de ces deux tendances, et nous semblent être au niveau de l'état de l'art actuel dans le domaine. Nous pouvons néan- moins nuancer cette affirmation en soulignant que certains systèmes commerciaux ne sont pas documentés et que les évaluations dont ils ont fait l'objet ne sont pas publiées. Toutefois, il est peu probable que les techniques employées diffèrent de beaucoup de celles proposées dans la littérature.

La question de l'évaluation est centrale dans la mise au point des systèmes de détection et reconnaissance des formes. De ce point de vue, on ne peut que regretter l'absence de bases de données publiques de référence permettant une évaluation comparée des performances des différents systèmes. Il n'existe d'ailleurs pas, à notre connaissance, d'étude publiée proposant une comparaison quantitative des performances d'un ensemble représentatif de méthodes exis- tantes. La comparaison de trois méthodes de détection proposée dans le présent ouvrage, sur une base de données réelle qui sera prochainement mise en ligne constitue un premier pas dans cette direction.

Chapitre 2

Généralités sur le traitement d'images

Introduction

Le traitement d'image est un domaine très large qui a connu, et qui a assisteé un développement important depuis quelques dizaines d'années.

Alors, le traitement d'image est l'ensemble des techniques qui sont appliqué à l'image numérique pour améliorer ou d'en extraire des informations.

Donc, dans ce chapitre, nous abordons les notions de base nécessaires et des techniques de traitement d'images. Tels que : la définition d'image, les types et caractéristiques d'image, les opérations ou les traitements possibles appliquer sur l'image et enfin le domaine d'application.

2.1 Définition de l'image

Une image est une représentation visuelle de description d'un objet ou personne par dessin, peinture, photographie, films,...etc. C'est aussi derrière l'affichage sur écran c'est un ensemble organisé d'informations, acquise, créée, traitée ou stockée sous forme binaire (suite de 0 et de 1). Ce qui nécessite sa numérisation (Figure 2.1).

Figure 2.1: Image numérique

2.2 Image numérique (numérisée)

L'image numérique est l'image dont la surface est divisée en éléments de taille fixe appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs. La numéri-

sation d'une image est la conversion de celle-ci de son état analogique en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques f(x,y), comme la montre la figure 1.2 où (x,y) coordonnées cartésiennes d'un point de l'image. f(x, y) : niveau d'intensité. La valeur en chaque point exprime la mesure d'intensité lumineuse perçue par le capteur (Figure 2.2) [1].

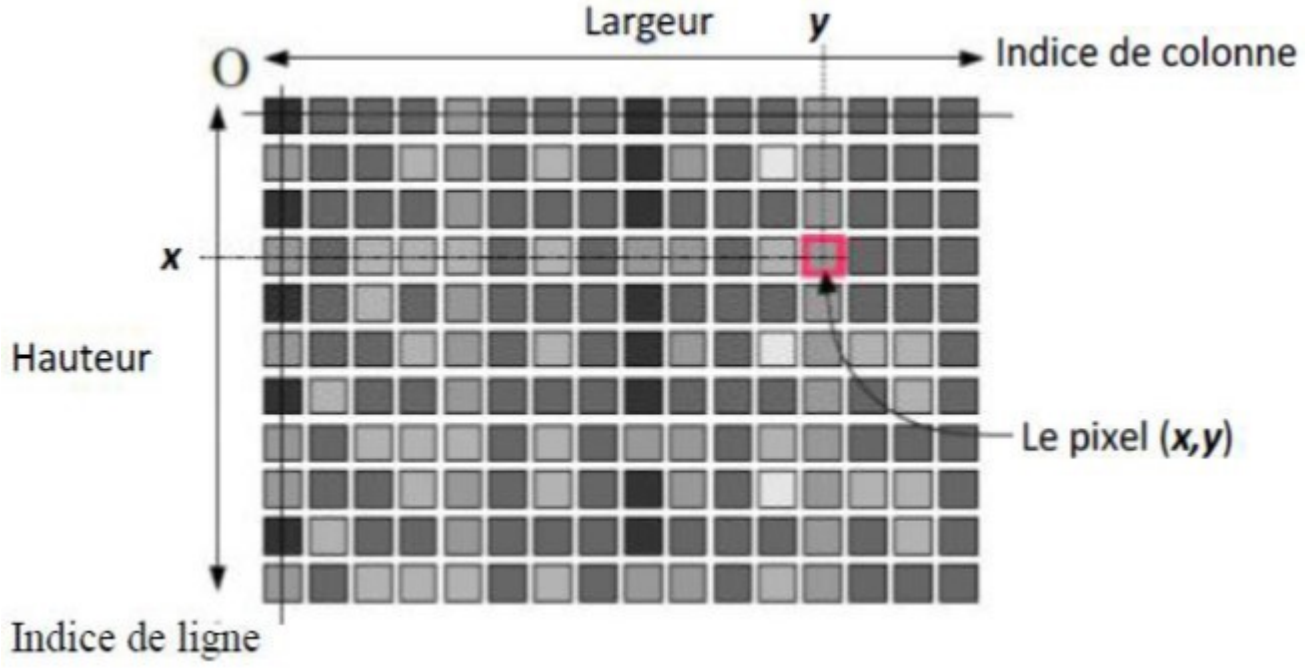

Figure 2.2: Représentation d'image numérique

2.3 Caractéristiques d'une image numérique

Comme nous l'avons déjà dit, l'image est un ensemble organisé d'informations qui contient les caractéristiques suivantes :

2.3.1 Pixel

Une image numérique est constituée d'un ensemble de points appelés pixels (abréviation de Picture Elément) pour former une image. Le pixel représente ainsi le plus petit élément constitutif d'une image numérique. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image. Par exemple, peut être affichée comme un groupe de pixels (Figure 2.3) [1].

Figure 2.3: Représentation en groupe de Pixel d'une image

2.3.2 La résolution

La résolution est un terme souvent confondu avec la "définition", détermine par contre le nombre de points ou pixels par unité de surface, exprimé en points par pouce (PPP, en anglais DPI pour Dots Per Inch), un pouce représentant 2.54 cm (Figure 2.4) [1].

Figure 2.4: Représentation de Résolution d'une image

Formule de conversion cm, Pixel, Pouce

$$\left.\begin{array}{l} 1\ cm = 37.79527559055\ pixel \\ 100\ pixel = 2.646\ cm \\ 1\ pouce = 2.54\ cm \\ 1\ cm = 0.3937\ pouce \end{array}\right\} \quad \text{alors 1 pouce} = 72 \text{ pixel} \tag{2.1}$$

2.3.3 Dimension

Nous appellons définition, le nombre de points (pixel) constituant l'image, c.à.d. sa (dimension informatique). Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image (Figure 2.5) [1].

Figure 2.5: Représentation de dimension d'une image

2.3.4 La texture

Une texture est une région dans une image numérique qui a des caractéristiques homogènes. Ces caractéristiques sont, par exemple, un motif basique qui se répète. La texture est composée

de Texel, l'équivalent des pixels.

2.3.5 Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l' éclairage des dispositifs optiques et électroniques du capteur (Figure 2.6) [1].

(a) Image sans bruit (b) Image avec bruit

Figure 2.6: Image sans et avec bruit

2.3.6 La luminance

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface, pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet [1].

Une bonne luminance se caractérise par :

- Des images lumineuses (brillantes).
- Un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir; ces images entraînent des pertes de détails dans les zones sombres ou lumineuses (Figure 2.7).
- L'absence de parasites.

Figure 2.7: Image représente l'éffet de luminance

2.3.7 Histogramme

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (couleur) dans l'image. Pour diminuer l'erreur de quantification, pour comparer deux images obtenues sous des éclairages différents, ou encore pour mesurer certaines propriétés sur une image. Il permet de donner un grand nombre d'informations sur la distribution des niveaux de gris (Couleur) et de voir entre quelles bornes est repartie la majorité des niveaux de gris (couleur) dans les cas d'une image trop claire ou d'une image trop foncée (Figure 2.8) [1].

Figure 2.8: Image avec histogramme

2.3.8 Le 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 zones d'images. Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport :

$$C = \frac{L_1 - L_2}{L_1 + L_2} \tag{2.2}$$

2.4 Types d'images

Nous distinguons deux types d'images :

2.4.1 Images matricielles

Dans la description que nous avons faite jusqu'à présent des images nous avons utilisé une matrice. Nons disons alors que l'image est matricielle ou en anglais bitmap. Ce type d'image est adapté à l'affichage sur écran mais peu adapté pour l'impression car bien souvent la résolution est faible (couramment de 72 à 150 ppp pour les images sur Internet).

2.4.2 Images vectorielles

Le principe des images vectorielles est de représenter les données de l'image à l'aide de formules mathématiques. Cela permet alors d'agrandir l'image indéfiniment sans perte de qualité et d'obtenir un faible encombrement [1].

Par exemple, pour décrire un cercle dans une image, il suffit de noter la position de son centre et la valeur de son rayon plutôt que l'ensemble des points de son contour. Ce type est généralement obtenu à partir d'une image de synthèse créée par logiciel (exemple : Autocad) et non pas à partir d'un objet réel. Ce type est donc particulièrement adapté pour le travail de redimensionnement d'images, la cartographie ou l'infographie [1].

2.5 Codages des couleurs

Nous avons vu qu' une image apparait comme une matrice où chaque case contient des nombres associés à une couleur. Usuellement on distingue 3 grands types de couleurs pour une image numérique :

- Le noir et blanc.
- Les niveaux de gris.
- La couleur.

Ces types sont généralement à choisir lors d'une numérisation par scanner ou lors de la configuration d'un appareil photographique .

2.5.1 Image noir et blanc

Le noir et blanc est le plus simple. Le contenu de chaque case de la matrice est soit un 0 (noir) soit 1 (blanc). Le nombre de couleurs n'est que de 2 et le rendu de l'image le moins performant mais parfois suffisant dans le cadre par exemple de documents scripturaux (Figure 2.9) [1].

Figure 2.9: Comparaison du volume d'une image pour différents codages

2.5.2 Niveaux de gris

Le codage dit en niveaux de gris permet d'obtenir plus de nuances que le simple noir et blanc. Il offre des possibilités supplémentaires pour coder le niveau de l'intensité lumineuse. La couleur est codée souvent sur un octet soit 8 bits ce qui offre la possibilité d'obtenir 256 niveau de gris (0 pour le noir et 255 pour le blanc). On peut aussi le faire avec 16 niveaux de gris (4 bits) (Figure 2.10) [1].

Figure 2.10: Image en niveaux de gris

2.5.3 Image couleur

Principe

La couleur d'un pixel est obtenue, comme le ferait un peintre, par le mélange de couleurs fondamentales. Il ne s'agit pas ici de décrire toutes les techniques utilisées. Nous allons décrire un des principes les plus couramment utilisé qui est celui de la synthèse additive [1].

Codage RVB

Le principe consiste à mélanger les 3 couleurs : rouge, vert et bleu (noté RVB ou RGB en anglais). A l'aide de ces 3 couleurs, on obtient toute une palette de nuances allant du noir au blanc. A chaque couleur est associé un octet (donc 256 niveaux de luminosité) de chacune des

couleurs fondamentales (Figure 2.11) [1].

Rouge	Vert	Bleu	Couleur
0	0	0	Noir
0	0	1	Nuance de noir
255	0	0	Rouge
0	255	0	Vert
0	0	255	Bleu
128	128	128	Gris
255	255	255	Blanc

Figure 2.11: Principe codage de la couleur

Un pixel "couleur" est alors codé avec 3 octets et on a alors la possibilité d'obtenir 224 possibilités de couleurs soit de l'ordre de 16 millions de couleurs différentes (Figure 2.12) [1].

Figure 2.12: Valeurs RVB d'un pixel

Question comment convertit-on une image couleur en une image en niveau de gris?

$$NdG = \frac{R+G+B}{3} \tag{2.3}$$

$$NdG = \frac{\log(R)+\log(G)+\log(B)}{3} \tag{2.4}$$

$$NdG = 0.30R + 0.59G + 0.11B \tag{2.5}$$

2.6 Formats d'image

Lors de son enregistrement, une image est stockée suivant un format d'image précis. Ce format doit permettre de stocker l'information de l'image avec un minimum de perte d'informations. Il existe ainsi différents formats qui pourront favoriser soit la conservation de la qualité soit la diminution de la taille du fichier informatique [1].

Le tableau suivant donne les principales caractéristiques des principaux standards utilisés (Figure 2.13):

Format	Type	Compression données	Nombre couleurs	Affichage progressif	Usage
BMP	matriciel	non	de 2 à 16 millions	non	Image non dégradée ; Taille fichier importante.
JPG	Matriciel	oui	16 millions	oui	Taux de compression réglable ; Perte de qualité.
GIF	Matriciel	oui	De 2 à 256 couleurs	oui	Pas de perte de qualité ; Usage pour Internet.
TIFF	Matriciel	oui	16 millions	non	Pas d'usage Internet
PNG	Matriciel	oui	de 2 à 16 millions	Oui	Recommandé pour Internet
SVG	Vectoriel	oui	16 millions	non	Usage cartographie, animations

Figure 2.13: Principaux formats d'image

2.7 Le système de traitement d'images

Nous pouvons appliquer plusieurs traitements sur l'image numérique, la figure suivante résume l'ensemble de ces traitements (Figure 2.14).

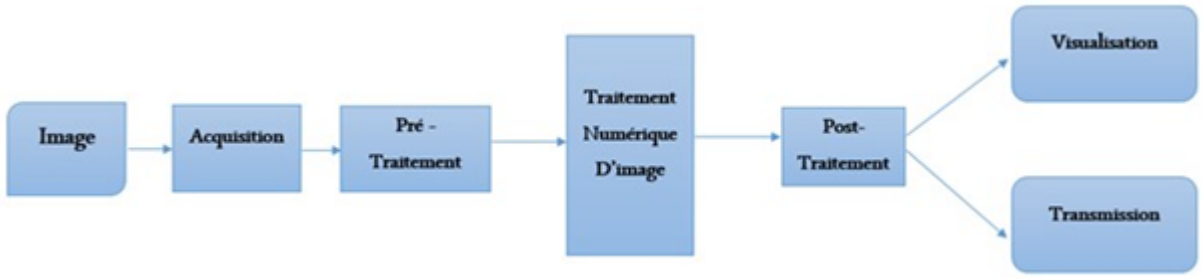

Figure 2.14: Schéma d'un système de traitement d'images

2.7.1 Acquisition

Pour pouvoir manipuler une image sur un système informatique, il est avant tout nécessaire de lui faire subir une transformation qui la rendra lisible et manipulable par ce système. Le passage de cet objet externe (l'image d'origine) à sa représentation interne (dans l'unité de traitement) se fait grâce à une procédure de numérisation (échantillonnage, quantification). On utilise plus couramment des caméras vidéo, des appareils photos numériques. En médecine, on utilise des imageurs IRM, TEP, scanner X, écho doppler, échographie, scintigraphie etc... [1]

2.7.2 Filtrage

Est une opération qui consiste à réduire le bruit contenu dans une image au moyen d'algorithmes provenant des mathématiques par l'utilisation de méthodes d'interpolation ou de la morphologie mathématique [1].

Nous pouvons diviser les filtres en deux grandes catégories :

1- Filtres linéaires

Le filtre est dit linéaire si la valeur du nouveau pixel est une combinaison linéaire des valeurs des pixels du voisinage.

Il y a plusieurs types de filtrage linéaire nous distinguons :

Filtre passe-bas (lissage)

Ce filtre n'affecte pas les composantes de basse fréquence dans les données d'une image, mais doit atténuer les composantes de haute fréquence. L'opération de lissage est souvent utilisée pour atténuer le bruit et les irrégularités de l'image.

Filtre passe-haut (accentuation)

Le renfoncement des contours et leur extraction s'obtiennent dans le domaine fréquentiel par l'application d'un filtre passe-haut. Le filtre digital passe-haut a des caractéristiques inverses du filtre passe-bas. Ce filtre n'affecte pas les composantes de haute fréquence d'un signal, mais doit atténuer les composantes de basse fréquence [1].

Filtre passe-bande (différentiation)

Cette opération est une dérivée du filtre passe-bas. Elle consiste à éliminer la redondance d'information entre l'image originale et l'image obtenue par filtrage passe-bas. Seule la différence entre l'image source et l'image traitée est conservée [1].

Filtre directionnel

Dans certains cas, nous cherchons à faire apparaître des détails de l'image dans une direction bien déterminée. Pour cela, on utilise des filtres qui opèrent suivant des directions (horizontales, verticales et diagonales) [1].

* Quelque méthode de filtrage linéaire :

- Filtre moyenneur.
- Filtre gaussien.

Le principale inconvénient des filtres linéaire est que la réduction de bruit s'accompagné d'un étalement des transitions entre régions. Ce problème peut être surmonté par l'utilisation des filtres non linéaires.

2- Filtres non linéaire

Le filtrage non-linéaire est une opération qui remplace la valeur de chaque pixel par une combinaison non-linéaire des valeurs de ses pixels voisins, ce type de filtre pallie les inconvénient majeur des filtres linéaires dont la présence des valeurs aberrantes même après filtrage et la mauvaise conservation des transitions. On trouve aussi dans les filtres non linéaire les deux types de filtrages : le Filtre passe-bas et le Filtre passe-haut [1].

* Quelque méthode de filtrage non linéaire :

- Filtre médian.
- Le filtre de Canny.
- Le filtre de laplacien.
- Le filtre de gradient.
- Les filtres de PREWITT, SOBEL, FREEMAN, ET KIRSCH.
- Symmetric Nearest Neighbor .
- Nagao.

2.7.3 Segmentation

A partir d'une image numérique, il convient de d'extraire les informations pertinentes en regard de l'application concernée, les traiter puis les interpréter. Le terme générique d'analyse d'images désigne l'ensemble de ces opérations.

En analyse, d'images on distingue les traitements de bas-niveau et ceux de haut niveau. Cette distinction est liée au continu sémantique des entités traitées et extraites de l'image. Les traitements de bas-niveau opèrent, en général, sur les grandeurs calculées à partir des valeurs attachées à chaque point de l'image sans faire nécessairement la liaison avec la réalité qu'elles représentent. A l'opposé, les traitements de haut-niveau s'appliquent à des entités de nature

symboliques associées à une représentation de la réalité extraite de l'image; ils sont relatifs à l'interprétation et à la compréhension de l'image. La segmentation d'image est un traitement de bas-niveau qui consiste à créer une partition d'une image A en sous-ensembles Ri appelés régions, tel que [1]:

$$\forall i,\ \mathcal{R}_i \neq \emptyset \tag{2.6}$$

$$\forall i, j\ i \neq j,\ \mathcal{R}_i \cap \mathcal{R}_j = \emptyset \tag{2.7}$$

$$\mathcal{A} = \cup \mathcal{R}_i \tag{2.8}$$

Une région est un ensemble connexe de pixels ayant des caractéristiques communes (intensité, texture,...) qui les différencient des pixels des régions voisines. Dans le cas des images couleur ces caractéristiques sont déterminées à partir des composantes colorimétriques des pixels. Les connaissances utilisées sont les plus souvent du domaine de l'image numérique et du traitement du signal, donc sémantiquement assez pauvres.

Les différentes méthodes de segmentation

La segmentation définie par l'ensemble des pixels representent en deux méthodes, soit par : approche région de la segmentation ou bien par les contours de la région approche contour de la segmentation. Ces deux approches sont duales du fait que chaque région possède un contour et qu'un contour délimite forcément une région (Figure 2.15) [1].

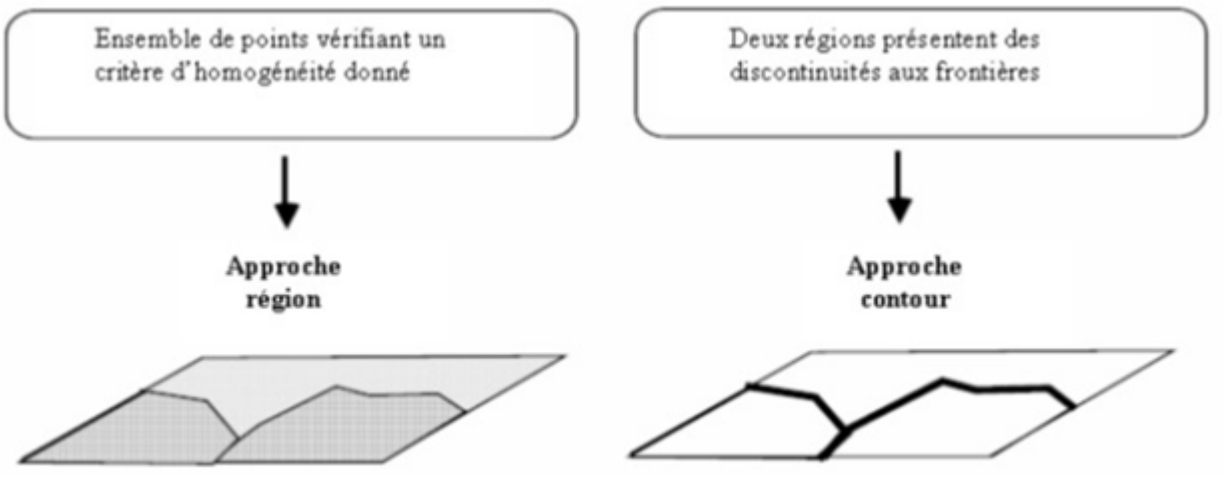

Figure 2.15: approche région et approche contour

a - Approche "régions"

En approche région, l'affinité des points connexes est favorisée. Cela peut être vu comme une technique contextuelle. Les points connexes ayant des propriétés semblables (attributs) : intensité de gris, couleur, texture, vont être réunis dans le même ensemble.

b - Approche "contours"

Cette approche, est une technique non contextuelle qui ignore les rapports pouvant exister entre les régions de l'image. On regroupe les pixels suivant un attribut global. Elle comprend les techniques de détection de contours, mais les contours obtenus ne conduisent pas toujours directement à la partition recherchée. En effet, les pixels contours mis en évidence pour une forme, généralement ne sont pas connexes. Il faut alors appliquer des algorithmes de fermeture de contours. Ce n'est qu'après fermeture que les régions apparaissent, déterminées par l'intérieur des contours [1].

2.8 Domaines D'application

Le traitement d'images possède l'aspect multidisciplinaire. On trouve ses applications dans des domaines très variés tels que:

2.8.1 Imagerie aérienne et spatiale

Dans laquelle les traitements concernent l'étude des images satellites, l'analyse des ressources terrestres, la cartographie automatique, les analyses météorologiques [1].

2.8.2 Technologies biomédicales

Nous trouvons des utilisations de cette technique dans l'échographie, la résonance magnétique nucléaire, ainsi que dans le domaine de la reconnaissance automatique des cellules ou de chromosomes [1].

2.8.3 La robotique

Qui connaît actuellement le plus grand développement et dont les tâches usant de l'imagerie sont principalement l'assemblage (pièce mécanique, composants électroniques,...), le contrôle de qualité, ainsi que la robotique mobile [1].

2.8.4 La télésurveillance

Exemple, radar automatique : recherche en temps réel d'un véhicule par reconnaissance de son immatriculation parmi un flot de véhicules circulant sur le boulevard périphérique par caméra fixe.

2.8.5 Simulation et contrôle de processus

Nous trouvons des utilisations de cette technique dans les cours de pilotage et le contrôle des panneaux.

Aussi : L'astronomie, la chimie, la physique nucléaire (identification de trajectoires de particules), l'armement (guidage de missiles, la détection des couleurs + reconnaissance des formes pour les panneaux de signalisation routière c'est le but de notre sujet de projet fin d'étude), métiers du spectacle, les télécommunications (TV, radio, vidéo, publicité,...), l'architecture, l'imprimerie,... [1]

Conclusion

Dans ce chapitre, nous avons présenté une masse importante d'informations concernant les méthodes utilisées dans le domaine de traitement d'image. Dans ce qui suit,nous allons parler sur les outils utilisés pour réaliser notre projet.

Chapitre 3

Conception et développement

Introduction

Dans ce chapitre nous allons présenter les différentes types de panneaux de signalisation routière et leurs caractéristiques, ensuite on va voir l'outils de développement keras pour notre détection qui est une librairie de tensorflow utilisé pour la création des réseaux de neurone, et enfin la méthode de développement et la création du réseau de neurone et l'implémentation détaillée de l'application en expliquant les étapes de notre système, et faire des tests.

3.1 Les panneaux de signalisation routière

La signalisation routière constitue un élément fondamental de tout système de circulation. Les panneaux de signalisation routière sont généralement implantés de part et d'autres sur nos routes, indiquent les règles de la circulation établies pour permettre aux véhicules et aux piétons de se déplacer en toute sécurité sur les routes.

Les usagers de la route doit d'être bien sensés de la signalisation routière qui demeure aujourd'hui un des éléments phares de la prévention routière. Grâce aux signaux routière, les conducteurs sont informés des règles d'avertissements sur les dangers pouvant apparaître sur la route ou tout type d'informations intéressant le conducteur, parmi les quelles on peut citer ces exemples (Figure 3.1), (Figure 3.2), (Figure 3.3) [2]:

Figure 3.1: Exemples des dangers pouvant être signalés grâce aux signaux routière

Passer à droite

Rambouillet

Figure 3.2: Exemples des indications fournies par les panneaux sur les directions à suivre

Interdit moins de 7,5 t

Interdit Camion

Céder le passage

Figure 3.3: Exemples des informations liées à l'usage de la route que les panneaux peuvent indiquer

Il y a beaucoup de panneaux de signalisation sur les routes, nous avons présenté les différents types les plus importants qui existent actuellement :

3.1.1 Panneaux d'interdiction ou de restriction

Ces types de signaux interdisent ou limitent certaines actions à ceux qui les trouvent devant dans la direction de leur marche et de l'endroit où ils se trouvent. Ces signes sont circulaires et avec un bord rouge. Comme montre la figure suivante (Figure 3.4) [2]:

Figure 3.4: Signaux d'interdiction ou de restriction

3.1.2 Panneaux d'avertissement de danger

Les signes de danger ont pour mission d'indiquer la nature d'un danger, leur objectif se conformer aux règles de comportement et d'éviter les chocs éventuels lors de la conduite. Sa forme est triangulaire avec un bord rouge, comme indique la figure suivante (Figure 3.5) [2]:

Figure 3.5: Signaux d'avertissement de danger

3.1.3 Panneaux de priorité

Ce sont destinées à informer les usagers de la route aux règles de priorité spéciales aux intersections ou aux passages étroits. À l'intérieur de cette classe, nous pouvant trouver deux des signes les plus importants qui existent "stop" et "Cédez le passage". Comme vous pouvez le voir, ils n'ont pas de formulaire ou une couleur spécifique (Figure 3.6) [2].

Figure 3.6: Signaux de priorité

3.1.4 Panneaux de fin de l'interdiction ou de la restriction

Ces panneaux de fin d'interdiction ou de la restriction signalent que l'interdiction ou de la limitation est terminée. Ils sontronds et blancs avec une ligne diagonale noire (Figure 3.7) [2].

Figure 3.7: Signaux de fin d'interdiction ou de restriction

3.1.5 Panneaux des obligations

Ils sont destinés à obliger les usagers à respecter certaines prescriptions et certaines règles à partir du niveau du panneau ou devant celui-ci. Ces panneaux peuvent donc indiquer non seulement une direction que l'usager devra suivre, mais aussi une vitesse minimum à respecter, ou encore une voir obligatoire pour certains types d'usagers .Ils sont ronds avec un couleur bleu (Figure 3.8) [2].

Figure 3.8: Signaux d'obligations

3.2 Outil de développement

3.2.1 Présentation de la librairie keras

Keras est une bibliothèque logicielle open source qui fournit une interface Python pour les réseaux de neurones artificiels et d'apprentissage automatique. Keras agit comme une interface pour la bibliothèque TensorFlow [3].

Jusqu'à la version 2.3, Keras pris en charge plusieurs backends, y compris tensorflow , Microsoft Cognitive Toolkit , Théano et PlaidML . Depuis la version 2.4, seul TensorFlow est pris en charge. Conçu pour permettre une expérimentation rapide avec les réseaux de neurones profonds,

il se concentre sur la convivialité, la modularité et l'extensibilité. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), et son principal auteur et mainteneur est François Chollet, un Googleingénieur. Chollet est également l'auteur du modèle de réseau de neurones profonds XCeption.

Keras contient de nombreuses implémentations de blocs de construction de réseau de neurones couramment utilisés tels que des couches, des objectifs , des fonctions d'activation , des optimiseurs et une multitude d'outils pour faciliter le travail avec les données d'image et de texte afin de simplifier le codage nécessaire à l'écriture de code de réseau neuronal profond. Le code est hébergé sur GitHub et les forums de support de la communauté incluent la page des problèmes GitHub et un canal Slack [3].

En plus des réseaux de neurones standard, Keras prend en charge les réseaux de neurones convolutifs et récurrents . Il prend en charge d'autres couches utilitaires courantes telles que l' abandon , la normalisation par lots et la mise en commun .

Keras permet aux utilisateurs de produire des modèles profonds sur des smartphones (iOS et Android), sur le Web ou sur la machine virtuelle Java . Il permet également l'utilisation de la formation distribuée de modèles d'apprentissage en profondeur sur des clusters d' unités de traitement graphique (GPU) et d' unités de traitement tensoriel (TPU) [3].

3.2.2 Installation et configuration de keras

Sous Linux:

Keras est un framework d'apprentissage en profondeur Python, vous devez donc avoir installé python sur votre système.

Dans Ubuntu, python est inclus par défaut, nous vous recommandons d'avoir la dernière version de python, c'est-à-dire python3. Pour vérifier si python 3 est installé sur votre système ou non [4]:

1. Ouvrez votre terminal (Ctrl + Alt + T):
2. Tapez:

```
$python3 -version
```

Vous devez obtenir la sortie de la version de python 3 (Figure 3.9) [4].

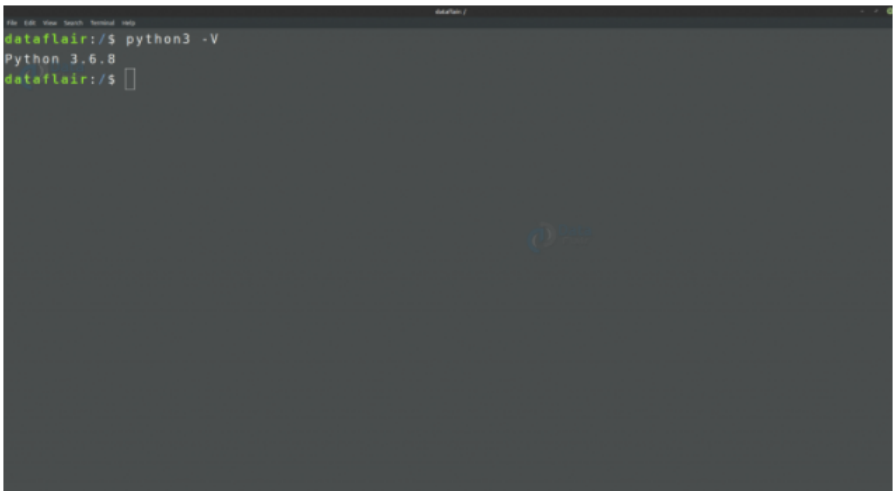

Figure 3.9: La version de python

Si python 3 n'est pas installé sur votre système, suivez les étapes ci-dessous [4]:

1. Ajoutez PPA en exécutant la commande suivante dans le terminal.

```
$sudo add-apt-repository ppa:jonathonf/python-3.6
```

2. Vérifiez les mises à jour et installez python 3.6.

```
$sudo apt-get update
$sudo apt-get install python3.6
```

3. Vérifiez à nouveau la version python 3 [4].

Maintenant, puisque vous avez python 3, nous allons installer Keras.

Avant d'installer Keras, nous devons installer l'un de ses moteurs principaux, à savoir Tensorflow, Theano ou Microsoft CNTK. Nous vous recommandons d'installer Tensorflow (Figure 3.10) [4].

Installez Tensorflow à partir de PyPI :

```
$pip3 install tensorflow
```

Figure 3.10: Installation de tensorflow

Maintenant, faisons l'installation de Keras (Figure 3.11) [4]:

Installez Keras depuis PyPI :

```
$pip3 install Keras
```

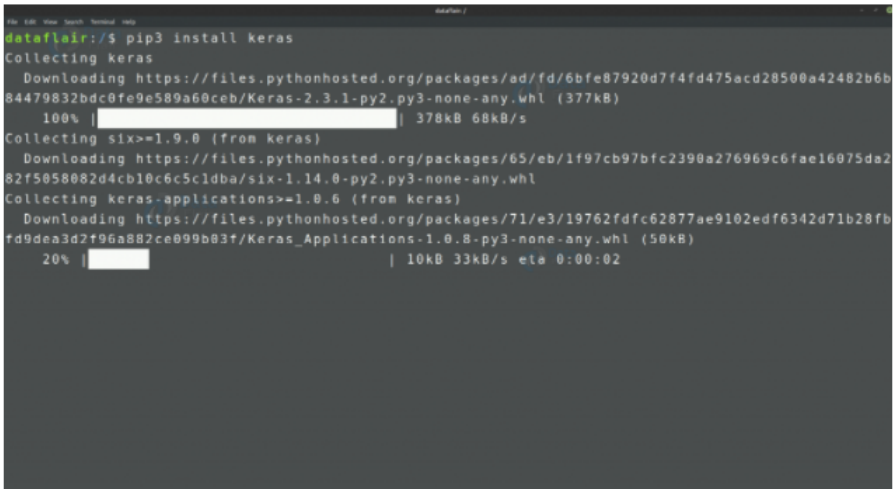

Figure 3.11: Installation de keras

Ou alors,

Installez Keras depuis Github (Figure 3.12) [4]:

1- Clonez d'abord le dépôt:

```
$git clone https://github.com/keras-team/keras.git
```

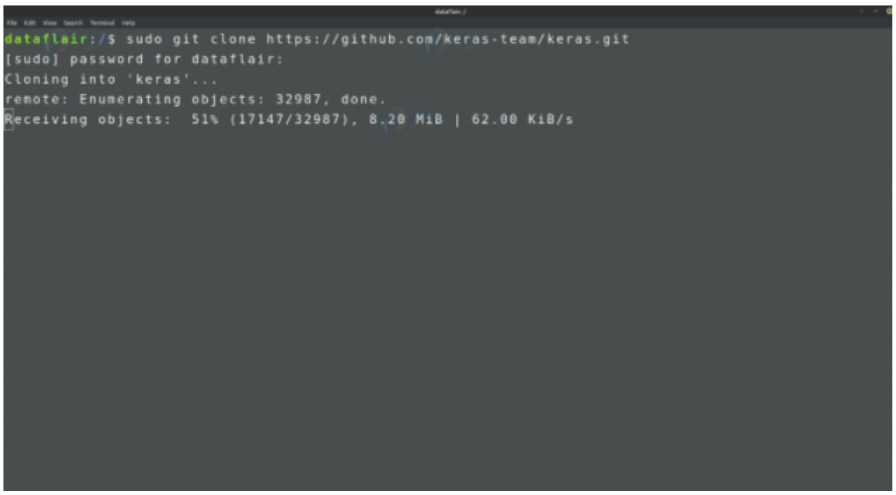

Figure 3.12: Installation de keras depuis Github

2- Allez dans le dossier keras:

```
$cd Keras
```

3- Exécutez la commande d'installation [4]:

```
$sudo python3 setup.py install
```

Keras est installé avec succès (Figure 3.13) [4].

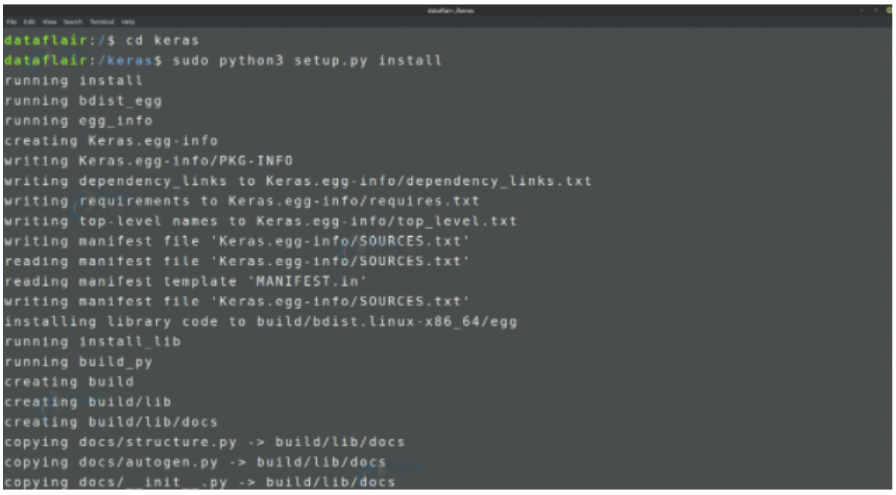

Figure 3.13: Voir l'installation de keras

Sous Windows:

Avant d'installer Keras, vous devez avoir Python installé sur votre système. Nous vous recommandons d'avoir la dernière version de python (python 3.5+).

Pour vérifier la version de python installée sur votre système.

1. Ouvrez cmd.
2. Tapez:

```
$python --version
```

Il vous montrera la version python (Figure 3.14)

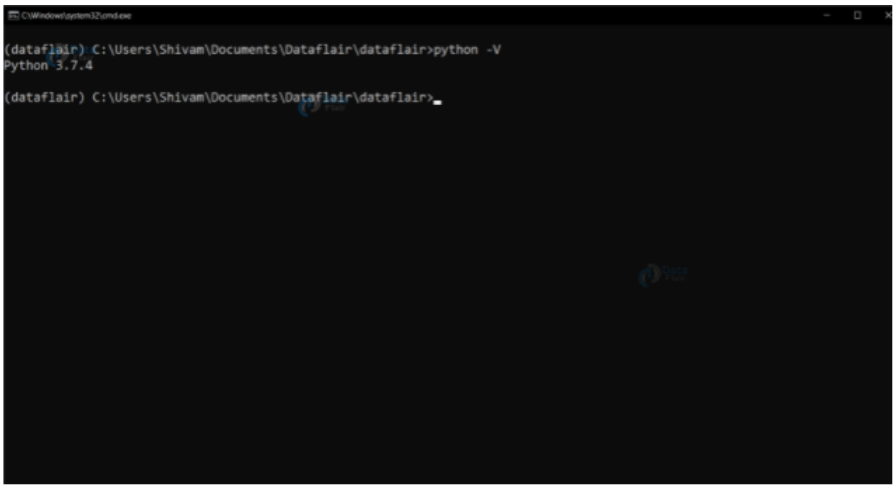

Figure 3.14: La version de python

Si vous n'avez pas installé python sur votre système ou si vous disposez d'une version inférieure de python [4]:

1. Accédez à la page de téléchargement sur python.org .
2. Cliquez et sélectionnez la dernière version de python pour Windows.
3. Allez au bas de la page et sélectionnez le programme d'installation exécutable Windows x86-64 pour 64 bits ou le programme d'installation exécutable Windows x86 pour PC 32 bits (Figure 3.15).

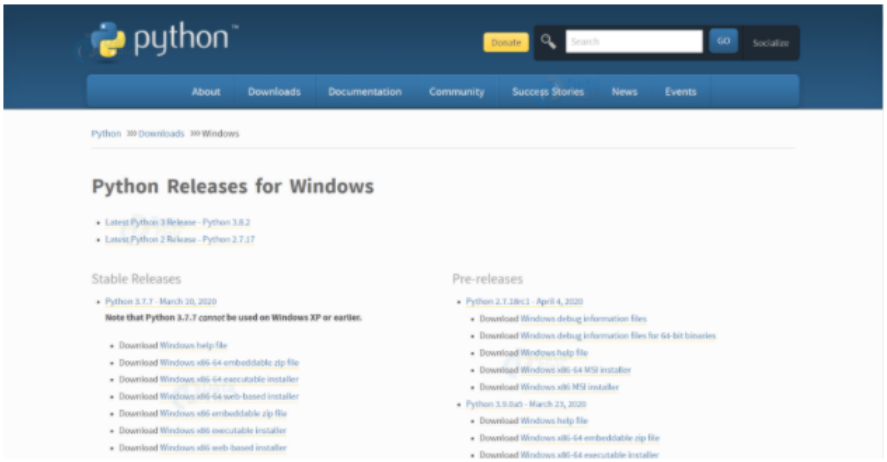

Figure 3.15: La page et sélectionnez le programme d'installation

4. Après avoir téléchargé le programme d'installation, exécutez-le en double-cliquant dessus.

5. Vérifiez à nouveau la version de python sur votre cmd.

Maintenant que vous avez python 3, avant d'installer Keras, vous devez installer l'un de ses moteurs principaux, c'est-à-dire Tensorflow, Theano ou Microsoft CNTK. Nous vous recommandons d'installer Tensorflow [4].

Installez Tensorflow à l'aide du gestionnaire de packages pip3 (Figure 3.16):

```
$pip3 install tensorflow
```

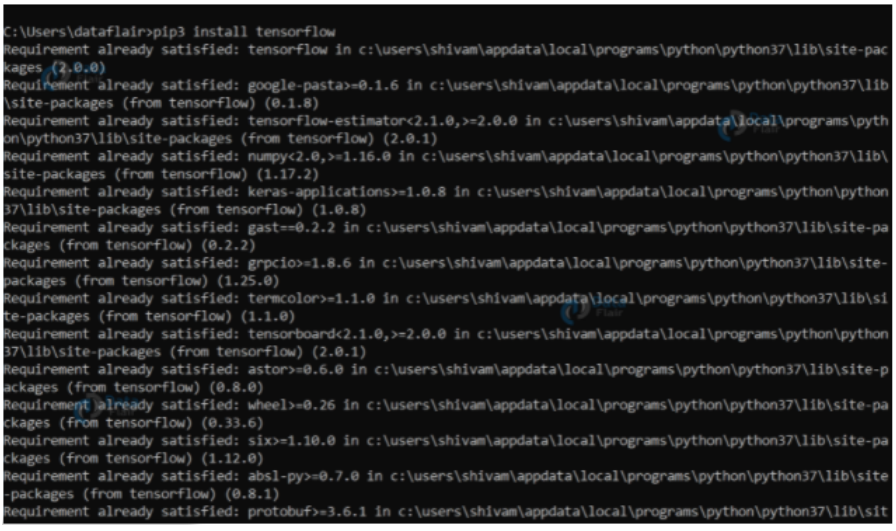

Figure 3.16: Installation de tensorflow

Installez maintenant Keras (Figure 3.17).

Installez Keras depuis PyPI :

```
$pip3 install Keras
```

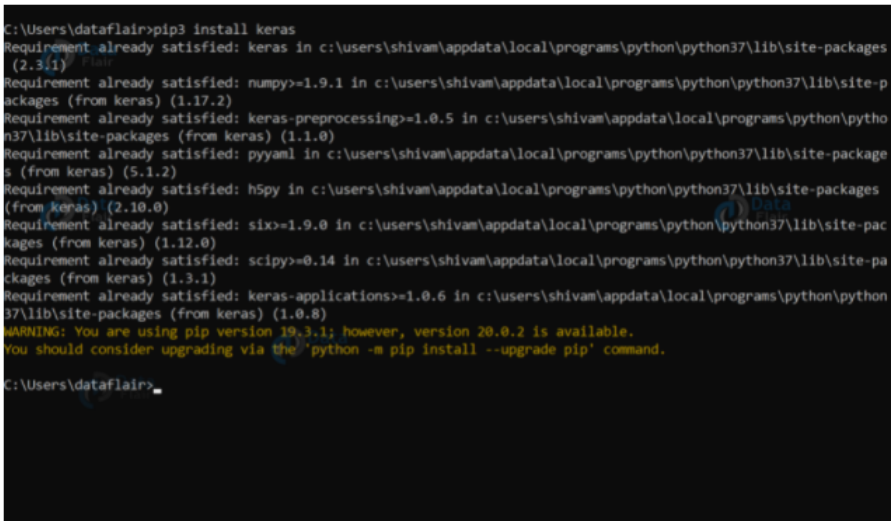

Figure 3.17: Installation de keras

Installez Keras depuis Github (Figure 3.18) [4]:

Clonez le dépôt git :

```
$git clone https://github.com/keras-team/keras.git
```

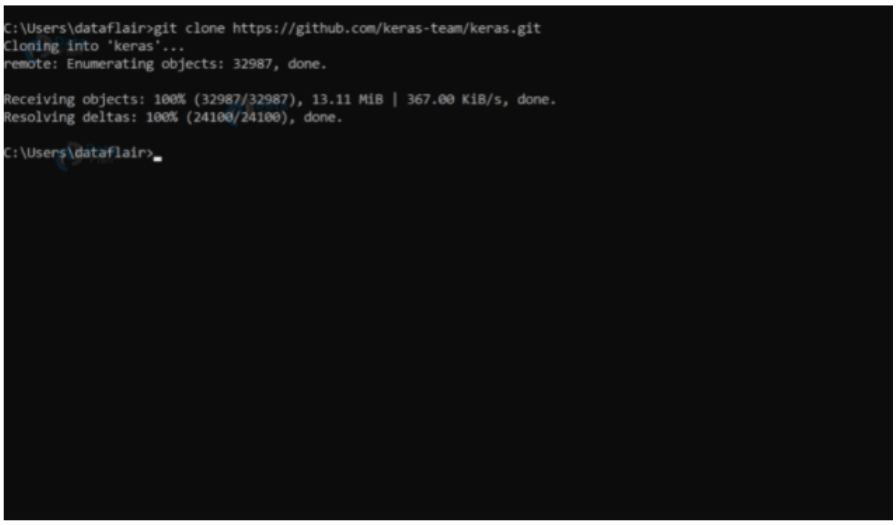

Figure 3.18: Installation de keras depuis Github

Allez dans le dossier keras (Figure 3.19).

```
$cd keras
```

Exécutez la commande d'installation :

```
$python3 setup.py install
```

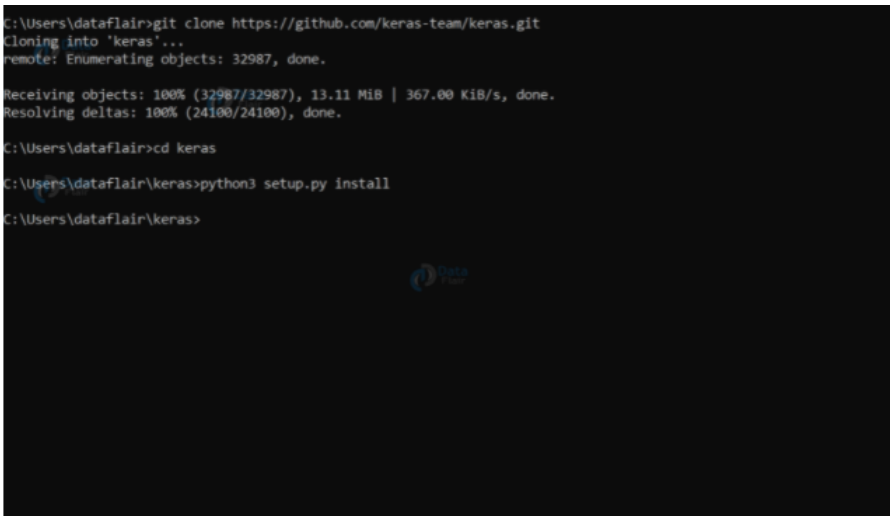

Figure 3.19: Installation de python 3

3.3 Le réseau de neurone avec keras

À l'heure actuelle, vous connaissez peut-être déjà l'apprentissage automatique, une branche de l'informatique qui étudie la conception d'algorithmes capables d'apprendre [5].

Maintenant, l'apprentissage en profondeur, un sous-domaine de l'apprentissage automatique qui est un ensemble d'algorithmes inspirés de la structure et de la fonction du cerveau. Ces algorithmes sont généralement appelés réseaux de neurones artificiels (ANN). L'apprentissage en profondeur est l'un des domaines les plus en vogue de la science des données avec de nombreuses études de cas qui ont des résultats étonnants en robotique, en reconnaissance d'images et en intelligence artificielle (IA) [5].

Keras est l'une des bibliothèques Python les plus puissantes et les plus faciles à utiliser pour développer et évaluer des modèles d'apprentissage en profondeur ; Il enveloppe les bibliothèques

de calcul numérique efficaces Theano et TensorFlow. L'avantage de ceci est principalement que vous pouvez commencer avec les réseaux de neurones de manière simple et amusante [5].

3.3.1 Présentation des réseaux de neurones artificiels

Avant d'approfondir Keras et comment vous pouvez l'utiliser pour démarrer avec l'apprentissage en profondeur en Python, vous devriez probablement savoir une chose ou deux sur les réseaux de neurones.

Les réseaux de neurones ont trouvé leur inspiration et leur biologie, où le terme "réseau de neurones" peut également être utilisé pour les neurones. Le cerveau humain est alors un exemple d'un tel réseau de neurones, qui est composé d'un certain nombre de neurones [6].

Et, comme vous le savez tous, le cerveau est capable d'effectuer des calculs assez complexes, et c'est de là que vient l'inspiration pour les réseaux de neurones artificiels. Le réseau dans son ensemble est un puissant outil de modélisation [6].

Perceptrons

Le réseau de neurones le plus simple est le "perceptron", qui, dans sa forme la plus simple, est constitué d'un seul neurone. Tout comme les neurones biologiques, qui ont des dendrites et des axones, le neurone artificiel unique est une structure arborescente simple qui a des nœuds d'entrée et un seul nœud de sortie, qui est connecté à chaque nœud d'entrée. Voici une comparaison visuelle des deux (Figure 3.20) [6]:

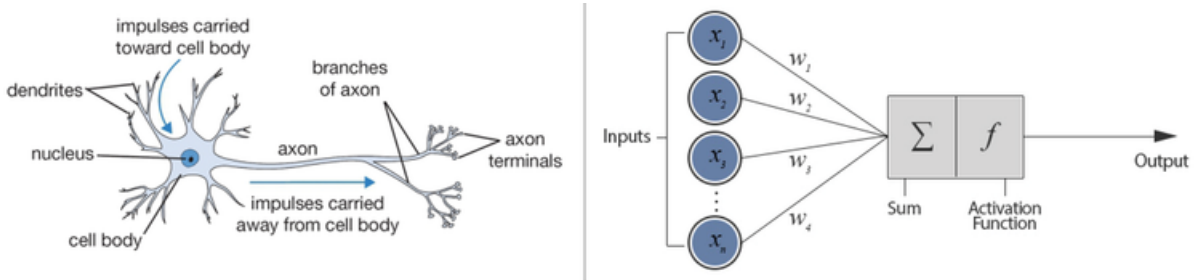

Figure 3.20: Neurone biologique contre réseau de neurones artificiels

Comme vous pouvez le voir sur la photo, les neurones artificiels comportent six composants.

De gauche à droite, ce sont [6]:

1- Nœuds d'entrée . En l'occurrence, chaque nœud d'entrée est associé à une valeur numérique, qui peut être n'importe quel nombre réel. N'oubliez pas que les nombres réels constituent le spectre complet des nombres : ils peuvent être des nombres positifs ou négatifs, entiers ou décimaux.

2- Connexions . De même, chaque connexion qui part du nœud d'entrée a un poids qui lui est associé, et cela peut également être n'importe quel nombre réel.

3- Ensuite, toutes les valeurs des nœuds d'entrée et les poids des connexions sont rassemblées : elles sont utilisées comme entrées pour une somme pondérée :

$y = f(\sum_{i=1}^{D} w_i * x_i)$

Ou, autrement dit, $y = f(w_1 * x_1 + w_2 * x_2 + ...w_D * x_D)$.

4- Ce résultat sera l'entrée d'une fonction de transfert ou d'activation . Dans le cas le plus simple mais trivial, cette fonction de transfert serait une fonction identité, $f(x) = x$ ou $y = x$. Dans ce cas, x est la somme pondérée des nœuds d'entrée et des connexions. Cependant, tout comme un neurone biologique ne se déclenche que lorsqu'un certain seuil est dépassé, le neurone artificiel ne se déclenche également que lorsque la somme des entrées dépasse un seuil, disons par exemple 0. C'est quelque chose que vous ne pouvez pas retrouver dans une fonction identitaire ! La manière la plus intuitive que l'on puisse imaginer consiste à concevoir un système comme celui-ci :

$f(x) = 0$ si $x < 0$

$f(x) = 0.5$ si $x = 0$

$f(x) = 1$ si $x > 0$

Bien sûr, vous pouvez déjà imaginer que la sortie ne sera pas une ligne lisse : ce sera une fonction discontinue. Parce que cela peut causer des problèmes dans le traitement mathématique, une variante continue, la fonction sigmoïde, est souvent utilisée. Un exemple de fonction sigmoïde que vous connaissez peut-être déjà est la fonction logistique. L'utilisation de cette fonction permet d'obtenir un résultat beaucoup plus fluide !

5- En conséquence, vous avez le nœud de sortie , qui est associé à la fonction (telle que la fonction sigmoïde) de la somme pondérée des nœuds d'entrée. Notez que la fonction sigmoïde est une fonction mathématique qui donne une courbe en forme de "S"; Vous en saurez plus à ce sujet plus tard.

6- Enfin, le perceptron peut être un paramètre supplémentaire, appelé biais , que vous pouvez considérer comme le poids associé à un nœud d'entrée supplémentaire qui est fixé en permanence à 1. La valeur de biais est critique car elle permet de décaler la fonction d'activation vers le gauche ou à droite, ce qui peut faire un déterminer le succès de votre apprentissage.

Notez que la conséquence logique de ce modèle est que les perceptrons ne fonctionnent qu'avec des données numériques. Cela implique que vous devez convertir toutes les données nominales dans un format numérique.

Maintenant que vous savez que les perceptrons fonctionnent avec des seuils, l'étape de leur utilisation à des fins de classification n'est pas si lointaine : le perceptron peut convenir que toute sortie au-dessus d'un certain seuil indique qu'une instance appartient à une classe, tandis qu'une sortie en dessous du seuil peut faire en sorte que l'entrée soit membre de l'autre classe. La ligne droite où la sortie est égale au seuil est alors la frontière entre les deux classes [6].

Perceptrons multicouches

Les réseaux de perceptrons sont des perceptrons multicouches, et c'est ce que ce tutoriel va implémenter en Python avec l'aide de Keras ! Les perceptrons multicouches sont également connus sous le nom de « réseaux de neurones à action directe ». Comme vous l'avez en quelque sorte deviné maintenant, ce sont des réseaux plus complexes que le perceptron, car ils se composent de plusieurs neurones organisés en couches. Le nombre de couches est généralement limité à deux ou trois, mais théoriquement, il n'y a pas de limite !

Les couches agissent beaucoup comme les neurones biologiques dont vous avez parlé ci-dessus : les sorties d'une couche servent d'entrées pour la couche suivante.

Parmi les couches, vous pouvez distinguer une couche d'entrée, des couches cachées et une couche de sortie. Les perceptrons multicouches sont souvent entièrement connectés. Cela signifie

qu'il existe une connexion entre chaque perceptron d'une couche spécifique et chaque perceptron de la couche suivante. Même si la connectivité n'est pas une exigence, c'est généralement le cas.

Notez que bien que le perceptron ne puisse représenter que des séparations linéaires entre les classes, le perceptron multicouche surmonte cette limitation et peut également représenter des limites de décision plus complexes [6].

3.3.2 Construire un réseau de neurone avec keras

Le but de ce projet est de programmé un réseau de neurone qui va détecté les panneaux de signalisation routière plus précisament les panneaux de limitation de vitesse.

1- construction de la base de donnée

La dataset est construi de deux parties, la première pour l'entrainement et la deuxième pour le test.

Lorsque nous faissons de la reconnaissance d'image par apprentissage profond, le plus important c'est d'avoir une grande base de données, c'est à dire avoir le plus d'images possible. Comme la écrit Jean-Claude Heudin (auteur de "Comprendre le deep learning") : "Ce n'est pas forcément celui qui a le meilleur algorithme qui gagne, c'est celui qui a le plus de données".

Il faut voir les CNNs (Convolutionnal Neural Network) comme un humain qui n'aurait aucune connaissance a priori (mais vraiment aucune !). Nous voulons que cette personne apprenne à reconnaître une forme, un objet, quelque chose. Pour cela on va lui montrer un maximum d'images représentant l'objet à reconnaître (ou autre). Idéalement ces images doivent toutes être différentes afin de généraliser au maximum ce que l'on souhaite reconnaître.

Lorsque que nous créons un modèle de réseaux de neurones convolutifs ou tout simplement quand on utilise un modèle déjà existant, nous devons entraîner ce modèle avec des données afin qu'il puisse apprendre et être capable de reconnaître un objet ou quelque chose en particulier. Pour cela on utilise une base de donnée contenant toutes nos images.

Une partie de cette base de données servira à entraîner le modèle CNN, c'est ce qu'on appelle "la base de données d'entraînement". La partie de la base de données restante sert pour

évaluer le modèle, on l'appelle la "base de données de validation" (Figure 3.21).

Ce qui est important de savoir c'est que toutes les images que nous utilisons doivent êtres préalablement étiquetées, c'est à dire que chacune des images que nous utilisons doit appartenir à une classe en particulier.

Par exemple si nous souhaitons reconnaître des voitures, des vélos et des motos, cela nous donne 3 classes. Chaque image de notre base de données doit appartenir à une de ces trois classes. C'est ce que l'on appelle l'apprentissage supervisé [6].

Figure 3.21: l'architecture de la base de donnée

2- Prétraitement des données

Tout d'abord, nous devons nettoyer l'ensemble des données que nous utiliserons pour le modèle d'apprentissage.

Nous importons les bibliothèques nécessaires:

```
import sys
import os
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense, Activation
from keras.layers.convolutional import Convolution2D, MaxPooling2D
```

```
from keras import callbacks
from keras.utils import to_categorical
import time
import PIL
```

Nous définions les caractéristiques suivantes :

- Le nombre d'époques.
- Nombre de couches.
- Types des couches.
- Nombre de neurones dans chaque couche.
- Fonctions d'activation de chaque couche.
- Taille des entrées et des sorties...

```
epochs = 20
img_width, img_height = 150, 150
batch_size = 32
samples_per_epoch = 1000
validation_steps = 300
nb_filters1 = 32
nb_filters2 = 64
conv1_size = 3
conv2_size = 2
pool_size = 2
classes_num = 8
lr = 0.0004
```

Nous importons ensuite l'ensemble des données sur lequel allos travailler travailler sous forme de data-frame:

```
train_data_path = 'data/train'
validation_data_path = 'data/test'
```

3- Construction du modèle d'apprentissage profond

Nous créons une instance du Sequential() de la bibliothèque Keras, celle-ci superpose un ensemble de couches pour en créer un seul modèle. nous lui passe en paramètre une liste de couches qu'on souhaite utiliser pour notre modèle.

```
model = Sequential()
```

Maintenant, nous allons faire une couche de convolution 2D:

Keras Conv2D : est une couche de convolution 2D, cette couche crée un noyau de convolution qui est le vent avec des couches d'entrée qui aident à produire un tenseur de sorties.

Noyau : dans le traitement d'images, le noyau est une matrice de convolution ou des masques qui peuvent être utilisés pour le flou, la netteté, le gaufrage, la détection des contours, etc. en effectuant une convolution entre un noyau et une image.

```
model.add(Convolution2D(nb_filters1, conv1_size, conv1_size,
   padding ="same", input_shape=(img_width, img_height, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))
```

Le paramètre Conv2D obligatoire correspond au nombre de filtres à partir desquels les couches convolutives apprendront.

C'est une valeur entière qui détermine également le nombre de filtres de sortie dans la convolution.

Ici, nous apprenons un total de 32 filtres, puis nous utilisons Max Pooling pour réduire les dimensions spatiales du volume de sortie.

En ce qui concerne le choix de la valeur appropriée pour no. de filtres, il est toujours recommandé d'utiliser des puissances de 2 comme valeurs.

Le paramètre d'activation de la classe Conv2D est simplement un paramètre de commodité qui vous permet de fournir une chaîne qui spécifie le nom de la fonction d'activation que vous souhaitez appliquer après avoir effectué la convolution.

```
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))
```

Si vous ne spécifiez rien, aucune activation n'est appliquée et n'aura aucun impact sur les performances de votre réseau de neurones convolutifs.

4- Compilation du modèle

L'étape suivante est la compilation du modèle. Pour ce faire, il faut choisir :

La fonction de perte : utilisée pour diminuer l'érreur. Plus l'érreur est faible plus le modèle est précis.

L'optimiseur : aide à obtenir de meilleurs résultats pour la fonction de perte.

Métriques : utilisées pour évaluer le modèle.

Dans le code suivant nous avons utilisé la fonction loss comme étant l'érreur quadratique moyenne avec l'optimiseur RMSprop qu'on lui donne un taux d'apprentissage de 0,0004. Et pour la métrique nous avons utilisé l'erreur absolue moyenne qui va avec la fonction loss.

```
model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.RMSprop(lr=lr),
              metrics=['accuracy'])
```

5- Entrainement du modèle

Dans ce qui suit, nous entraînons le modèle sur le jeu de données train, et faire la validation.

L'entraînement d'un CNN consiste à déterminer et à calculer empiriquement la valeur de chacun de ses poids. Le principe est le suivant : le CNN traite une image (de la base de données

d'entraînement) et en sortie il fait une prédiction, c'est-à-dire qu'il dit à quelle classe il pense que cette image appartient. Sachant qu'on connaît préalablement la classe de chacune des images d'entraînement, on peut vérifier si ce résultat est correcte.

En fonction de la véracité de ce résultat, on met à jour tout les poids du CNN selon un algorithme qui s'appelle la rétropropagation du gradient de l'erreur. Lors de la phase d'entraînement du modèle, le processus expliqué ci-dessus est répété plusieurs fois et avec la totalité des images de la base de données d'entraînement. Le but étant que le modèle classifie au mieux ces données [7].

Lorsque le modèle a fini de mettre à jour ses poids, on évalue le modèle en lui présentant la base de données de validation. Il classe toutes ces images (qui sont des images que le modèle n'a jamais vues) et on calcule son taux de bonne classification, c'est ce qu'on appelle la précision du modèle [7].

```
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_path,
```

```
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='categorical')

print(train_generator.classes)
```

La fonction fit-generator() affiche la valeur de la fonction loss et la métrique pour chaque époque.

```
model.fit_generator(
        train_generator,
        steps_per_epoch=samples_per_epoch,
        epochs=epochs,
        validation_data=validation_generator,
        callbacks=cbks,
        validation_steps=validation_steps,
        )
```

TensorBoard : Est le kit de visualisation de TensorFlow, il fournit les solutions de visualisation et les outils nécessaires aux tests de machine learning (Figure 3.22) [8]:

- Suivi et visualisation de métriques telles que la perte et la justesse.
- Visualisation du graphe de modèle (opérations et couches).
- Affichage d'histogrammes de pondérations, de biais ou d'autres Tensors au fur et à mesure de leur évolution.
- Projection de représentations vectorielles continues dans un espace à plus faible dimension.
- Affichage d'images, de texte et de données audio.
- Profilage de programmes TensorFlow.

Et bien plus encore.

```
"""
Tensorboard log
"""
log_dir = './tf-log/'
tb_cb = callbacks.TensorBoard(log_dir=log_dir, histogram_freq=0)
cbks = [tb_cb]
```

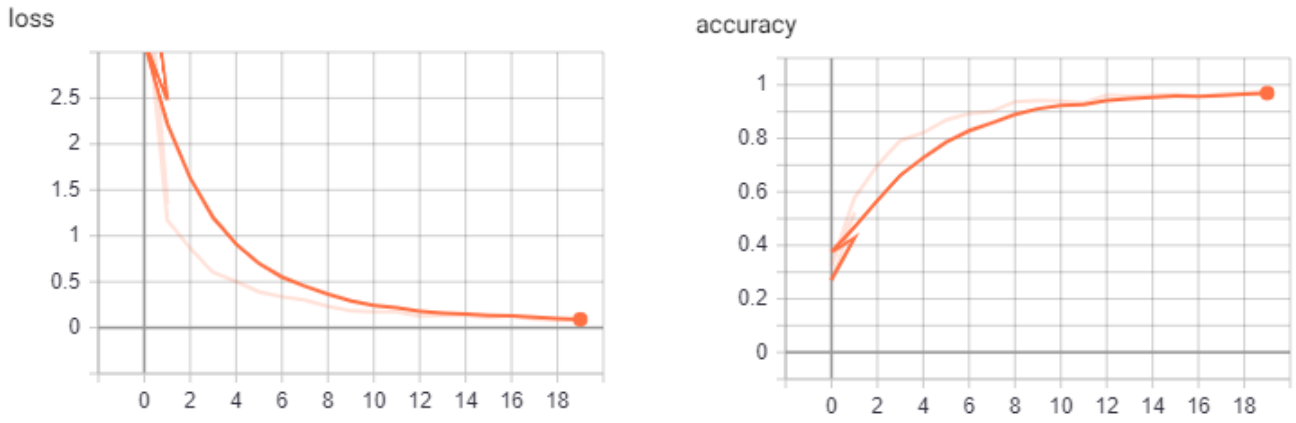

Figure 3.22: L'evolusion de l'érreur et de la précision pendant l'entrainement

Keras fournit une API pour vous permettre d' enregistrer votre modèle dans un fichier.

Le modèle est enregistré au format de fichier HDF5 qui stocke efficacement de grands tableaux de nombres sur le disque. Vous devrez confirmer que la bibliothèque Python h5py est installée. Il peut être installé comme suit [9]:

```
$sudo pip install h5py
```

Vous pouvez enregistrer un modèle Keras adapté dans un fichier à l'aide de la fonction save() sur le modèle.

Ce fichier unique contiendra l'architecture et les poids du modèle. Il comprend également la spécification de l'algorithme de perte et d'optimisation choisi afin que vous puissiez reprendre l'entraînement [9].

```
target_dir = './models/'
if not os.path.exists(target_dir):
  os.mkdir(target_dir)
model.save('./models/model.h5')
model.save_weights('./models/weights.h5')
```

Nous pouvons calculer le temps d'éxécution par les instruction suivante:

```
end = time.time()
dur = end-start


if dur<60:
    print("Execution Time:",dur,"seconds")
elif dur>60 and dur<3600:
    dur=dur/60
    print("Execution Time:",dur,"minutes")
else:
    dur=dur/(60*60)
    print("Execution Time:",dur,"hours")
```

L'entrainement prendra des heures, et ce la dépend de la quantité de la dataset (Figure 3.23).

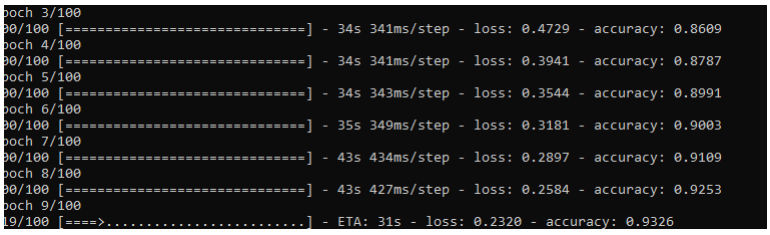

Figure 3.23: L'entrainement du modèle

Voici un schéma qui montre tout les étapes suivi lors de l'entrainement du modèle (Figure 3.24):

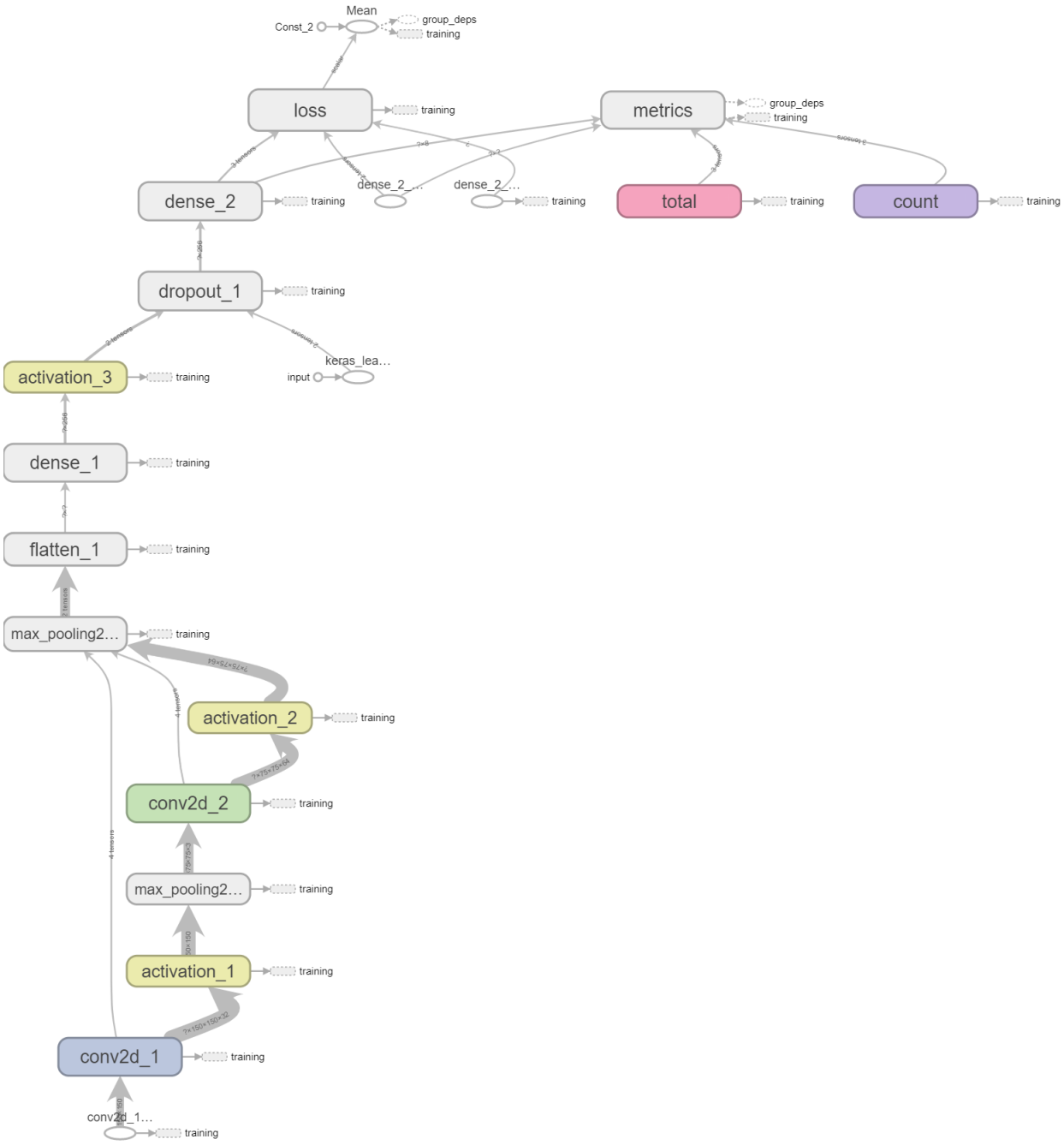

Figure 3.24: Le schéma final de l'entrainement du modèle

3.3.3 Le test du modèle entrainer

chargement du modèle

L'entrainement évalue un modèle simple sur l'ensemble de données des Indiens Pima. Le modèle est ensuite converti au format JSON et écrit dans model.json dans le répertoire local. Les pondérations du réseau sont écrites dans model.h5 dans le répertoire local [9].

Les données de modèle et de poids sont chargées à partir des fichiers enregistrés et un nouveau modèle est créé. Il est important de compiler le modèle chargé avant de l'utiliser. C'est pour que les prédictions faites à l'aide du modèle puissent utiliser le calcul efficace approprié du backend Keras [9].

Le modèle est évalué de la même manière en imprimant le même score d'évaluation.

Votre modèle enregistré peut ensuite être chargé ultérieurement en appelant la fonction load-model() et en passant le nom du fichier. La fonction renvoie le modèle avec la même architecture et les mêmes poids [9].

Dans ce cas, nous chargeons le modèle, résumons l'architecture et l'évaluons sur le même ensemble de données pour confirmer que les poids et l'architecture sont les mêmes.

```
#Define Path
model_path = './models/model.h5'
model_weights_path = './models/weights.h5'
test_path = 'data/alien_test'


#Load the pre-trained models
model = load_model(model_path)
model.load_weights(model_weights_path)
```

L'exécution de l'exemple charge d'abord le modèle, imprime un résumé de l'architecture du modèle, puis évalue le modèle chargé sur le même ensemble de données.

Faire les prédictions

Une fois que vous avez choisi et adapté un modèle final d'apprentissage en profondeur dans Keras, vous pouvez l'utiliser pour faire des prédictions sur de nouvelles instances de données [10].

Nous pouvons prédire des quantités avec le modèle de régression finalisé en appelant la fonction predict() sur le modèle finalisé [10].

La fonction predict() prend un tableau d'une ou plusieurs instances de données [10].

```
def predict(file):
  x = load_img(file, target_size=(img_width,img_height))
  x = img_to_array(x)
  print(x.shape)
  x = np.expand_dims(x, axis=0)
  print(x.shape)
  array = model.predict(x)
  result = array[0]
  print(result)
  answer = np.argmax(result)
  if answer == 0:
    print("Predicted: 5 km/h")
  elif answer == 1:
    print("Predicted: 15 km/h")
  elif answer == 2:
    print("Predicted: 30 km/h")
  elif answer == 3:
    print("Predicted: 40 km/h")
  elif answer == 4:
    print("Predicted: 50 km/h")
  elif answer == 5:
```

```
      print("Predicted: 60 km/h")
    elif answer == 6:
      print("Predicted: 70 km/h")
    elif answer == 7:
      print("Predicted: 80 km/h")

    return answer
```

Maintenant il faut Parcourir le répertoire pour chaque image et affiché le résultat :

```
for i, ret in enumerate(os.walk(test_path)):
  for i, filename in enumerate(ret[2]):
    if filename.startswith("."):
      continue

    print(ret[0] + '/' + filename)
    result = predict(ret[0] + '/' + filename)
    print("")
```

Nous pouvons calculer le temps d'éxécution par les instruction suivante :

```
end = time.time()
dur = end-start

if dur<60:
    print("Execution Time:",dur,"seconds")
elif dur>60 and dur<3600:
    dur=dur/60
    print("Execution Time:",dur,"minutes")
else:
```

```
dur=dur/(60*60)

print("Execution Time:",dur,"hours")
```

Nous allons choisir quelques plaques pour testé (Figure 3.25).

Figure 3.25: Les plaques utilisé pour faire les tests

Voici les résultats des tests (Figure 3.26).

```
data/test11/1.png
(150, 150, 3)
(1, 150, 150, 3)
[1. 0. 0. 0. 0. 0. 0. 0.]
Predicted: 5 km/h

data/test11/10.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 0. 0. 0. 1. 0. 0.]
Predicted: 60 km/h

data/test11/11.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 0. 0. 0. 1. 0. 0.]
Predicted: 60 km/h

data/test11/12.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 0. 0. 0. 0. 1. 0.]
Predicted: 70 km/h

data/test11/13.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 0. 0. 0. 0. 0. 1.]
Predicted: 80 km/h

data/test11/14.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 0. 0. 0. 0. 0. 1.]
Predicted: 80 km/h

data/test11/2.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 1. 0. 0. 0. 0. 0. 0.]
Predicted: 15 km/h

data/test11/3.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 1. 0. 0. 0. 0. 0.]
Predicted: 30 km/h

data/test11/4.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 1. 0. 0. 0. 0. 0.]
Predicted: 30 km/h

data/test11/5.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 1. 0. 0. 0. 0. 0.]
Predicted: 30 km/h

data/test11/6.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 0. 1. 0. 0. 0. 0.]
Predicted: 40 km/h

data/test11/7.png
(150, 150, 3)
(1, 150, 150, 3)
[0.000000e+00 0.000000e+00 0.000000e+00 1.000000e+00 0.000000e+00
 2.295082e-15 0.000000e+00 0.000000e+00]
Predicted: 40 km/h

data/test11/8.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 0. 0. 1. 0. 0. 0.]
Predicted: 50 km/h

data/test11/9.png
(150, 150, 3)
(1, 150, 150, 3)
[0. 0. 0. 0. 1. 0. 0. 0.]
Predicted: 50 km/h
```

Figure 3.26: Le résultat des tests

Conclusion

Dans ce chapitre nous avons présenté les démarches suivi pour l'élaboration d'un outil de détection et de reconnaissance des panneaux de signalisation, notre système inclut les panneaux de limitation de vitesse, l'utilisation de l'outil keras nous permet d'obtenir de bon résultat.

Conclusion générale

Dans les systèmes modernes de sécurité des véhicules, de nombreuses technologies pourraient nous aider à conduire et informer le conducteur en cas de danger.

La capacité à surveiller en permanence les panneaux de signalisation de restrictions et les avertissements sur la route, conduit à ce que le conducteur est souvent distrait du contrôle du véhicule. Ainsi, cela augmente les risques d'accident.

La solution consiste à développer activement des systèmes de détection et de reconnaissance des panneaux de signalisation pour informer le conducteur.

Les systèmes existants sont des systèmes informatiques complexes, trop coûteux à mettre en œuvre, et leur code ni gratuit ni disponible. Pour réduire les coûts de mise en œuvre de ce type de systèmes, il est nécessaire de mettre au point des applications de détection et de reconnaissance des panneaux de signalisation.

Pour valider notre travail, nous avons développé une application de détection et de reconnaissance des panneaux de signalisation avec l'outil keras à base de réseaux de neurones, les résultats obtenues ont été jugés très satisfaisants vue les métriques calculées à la fin de chaque apprentissage.

Ce travail implique plusieurs perspectives, qui peuvent se présentées sous forme de d'autre projets de fin d'études et même des sujet de doctorat vue l'importance de la thématique dans la vie quotidienne du citoyen. Parmi les perspectives qui peuvent être une continuation pour ce projet nous pouvons cité: l'implémentation de la solution sur un système embarqué implanté au bord

du véhicule, améliorer la solution pour donner des résultats a partir d'une vidéo capturée lors du déplacement de véhicule.

Reférences

[1] D. Boukhlouf, "Résolution de problèmes par écosystèmes: Application au traitement d'images," Ph.D. dissertation, Université Mohamed Khider-Biskra, 2005.

[2] P. Foucher, "Détection et reconnaissance de la signalisation verticale par analyse d'images," 2010.

[3] N. Ketkar and E. Santana, *Deep learning with python*. Springer, 2017, vol. 1.

[4] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[5] F. Chollet, "Building autoencoders in keras," *The Keras Blog*, vol. 14, 2016.

[6] G. Dreyfus, J. Martinez, M. Samuelides, M. Gordon, F. Badran, S. Thiria, and L. Hérault, *Réseaux de neurones*. Eyrolles Paris, 2002, vol. 39.

[7] C. Clark and A. Storkey, "Training deep convolutional neural networks to play go," in *International conference on machine learning*. PMLR, 2015, pp. 1766–1774.

[8] T. Hope, Y. S. Resheff, and I. Lieder, *Learning tensorflow: A guide to building deep learning systems*. " O'Reilly Media, Inc.", 2017.

[9] J. Brownlee, *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery, 2016.

[10] J.-C. Kim and K. Chung, "Prediction model of user physical activity using data characteristics-based long short-term memory recurrent neural networks." *KSII Transactions on Internet & Information Systems*, vol. 13, no. 4, 2019.

[11] J. Candido, "Dog breed classification and visualization."

[12] M. A. H. Tuhin, T. Pramanick, H. K. Emon, W. Rahman, M. M. I. Rahi, and M. A. Alam, "Detection and 3d visualization of brain tumor using deep learning and polynomial interpolation," in *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. IEEE, 2020, pp. 1–6.

[13] E. Gibson, W. Li, C. Sudre, L. Fidon, D. I. Shakir, G. Wang, Z. Eaton-Rosen, R. Gray, T. Doel, Y. Hu *et al.*, "Niftynet: a deep-learning platform for medical imaging," *Computer*

methods and programs in biomedicine, vol. 158, pp. 113–122, 2018.

[14] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," *arXiv preprint arXiv:1611.00791*, 2016.

[15] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[16] N. K. Manaswi, "Understanding and working with keras," in *Deep Learning with Applications Using Python*. Springer, 2018, pp. 31–43.

[17] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural Networks*, vol. 71, pp. 1–10, 2015.