Projet 2A
Système

Reconnaissance de place vide dans un parking à partir d'images prises depuis un drone

Chahma Sirine
Marchenoir Paul

CentraleSupélec

Campus de Rennes
2018-2019

Tables des matières :

I. Contexte du projet et application :

Les gens passent en moyenne 7 à 8 minutes à la recherche d'une place de stationnement pendant les heures de pointe. Cela représente environ 30% du trafic des flux urbains et contribue à la congestion du trafic pendant les heures de pointe. Ce problème est donc à l'origine d'impacts négatifs à la fois sur la vie personnelle des gens, sur leur santé (à cause du stress qui s'en suit) et sur l'environnement. Des systèmes d'information et de guidage sur le stationnement (IGP) ont donc été mis en place pour résoudre, ou au moins soulager ce problème et économiser du temps et des efforts pour trouver un espace de stationnement vacant. Ils ont le potentiel de réduire la congestion dans les zones surpeuplées en fournissant des indications en temps réel sur l'occupation des places de stationnement. À ce jour, ces systèmes sont pour la plupart mis en œuvre pour les environnements intérieurs ou au moins privés, en utilisant des techniques coûteuses basées sur des capteurs, et sont très peu généralisés. En conséquence, avec la demande croissante de systèmes IGP dans les environnements extérieurs, les méthodes de détection peu coûteuses basées sur l'analyse d'images grâce à des réseaux de neurones, sont devenus un centre de recherche et développement récemment.

Les systèmes IGP existants sont classés en quatre catégories, sur la base des méthodes de détection:

1) systèmes basés sur des compteurs,
2) systèmes basés sur des capteurs câblés,
3) systèmes basés sur des capteurs magnétiques sans filet
4) systèmes basés sur des images ou des caméras.

Les systèmes basés sur des compteurs reposent sur des capteurs à l'entrée et à la sortie parkings. Les systèmes basés sur des compteurs peuvent uniquement fournir des informations sur le nombre total de places vacantes plutôt que de guider les conducteurs vers l'emplacement exact des places de stationnement, et de tels systèmes ne peuvent pas être appliqués aux aires de stationnement présents dans les rues et aux espaces de stationnement résidentiels. Les systèmes à capteur magnétique câblé et magnétique sans fil reposent sur des capteurs à ultrasons, infrarouges ou magnétiques sans fil installés sur chaque parking. Cependant, de telles méthodes nécessitent l'installation de capteurs coûteux, en plus des unités de traitement et des émetteurs-récepteurs pour les technologies sans fil.
Les systèmes basés sur des capteurs jouissent d'un haut degré de fiabilité, mais leurs coûts d'installation et de maintenance élevés limitent leur utilisation pour des applications larges. Comparées aux systèmes basés sur des capteurs, les technologies basées sur des caméras sont relativement rentables, car les deux fonctions de surveillance générale et de détection d'occupation de parking peuvent être effectuées simultanément.

L'utilisation des systèmes IGP basés sur des caméras présente trois avantages par rapport aux autres systèmes existants. Premièrement, aucune infrastructure supplémentaire n'est requise, à condition que l'installation soit équipée de caméras de surveillance couvrant les places de stationnement. Deuxièmement, les systèmes basés sur des caméras fournissent l'emplacement exact des espaces de stationnement vacants qui est une exigence pour la navigation des véhicules aux places de stationnement

vacantes. Troisièmement, les méthodes basées sur une caméra sont hautement applicables aux places de stationnement dans les rues et résidentielles (possibilité de mettre des caméras dans les rues, ou encore d'utiliser un drone si la qualité d'image est suffisante et si on en a l'autorisation). Ainsi, un système reposant sur la reconnaissance d'image pourrait servir à une grande surface (si elle crée une application qui lui est propre), la rendant alors plus attirante auprès des clients car ils savent qu'ils ne perdront pas de temps à chercher une place. Ce système pourrait aussi servir à la population de manière individuelle pour faciliter le stationnement dans les rues si des caméras sont disponibles.

Cette étude présente un cadre de détection d'occupation de stationnement robuste utilisant la reconnaissance d'images par apprentissage automatique. Nous avon d'abord créé et étudié un réseau de neurones fully connected, puis un CNN (convolutional neural network) profond, tous les deux ayant le rôle de classificateur binaire permettant de détecter l'occupation des places de parking extérieures à partir d'images. Le classificateur a été formé et testé à partir d'une base de données (PKLot) formée d'images de places de parking vides et occupées, ayant des éclairements et des conditions météorologiques différents.

La détection d'occupation de stationnement par image implique essentiellement la détection d'objets de véhicules dans des espaces de stationnement. Les réseaux de neurones apprennent des fonctionnalités décrivant de manière optimale le contenu de l'image. Cela a été montré que les réseaux de neurones, et en particulier les CNN, à l'aide de jeux de données d'image volumineux, offrent une performance remarquable dans une variété de reconnaissance d'images et les tâches de détection d'objet. Les caractéristiques extraites par un CNN peuvent être utilisées directement pour former un classificateur pour la détection de l'occupation du stationnement dans une séquence d'images de vidéosurveillance. Nous avons donc comparé l'utilisation de deux types de réseaux de neurones différents, un réseau de neurone fully connected, ainsi qu'un CNN, pour extraire les caractéristiques et former un classifieur à partir d'un jeu de données de stationnement disponible au public. Nous avons ensuite essayé d'améliorer nos résultats obtenus en appliquant des filtres aux images de notre base de données. Finalement, nous avons vérifié et testé notre algorithme sur un cas pratique.

II. Cahier des charges :

Le client doit pouvoir utiliser des images reçues (provenant d'un drone ou d'une caméra) pour localiser des places de parking. Il aura pour cela une application pour faire fonctionner le drone : géolocalisation des places vides, et/ou pilotage manuel du drone. Dans un premier temps nous essaierons de localiser les places de parking libres sur une image de parking fixe. Nous utilisons un drone pour pouvoir rendre mobile la solution apportée à la problématique.

Nous allons mettre en place un système qui permet de détecter la disponibilité de places de parking en temps réel grâce à un algorithme de machine learning. Le traitement de l'image sera réalisé en Python à partir de la bibliothèque Tensorflow.

II.1) Etapes du projet :

- traitement des images par filtrage
- algorithme de machine learning pour reconnaître les images

II.2) Domaines concernés :

- système embarqué
- machine learning, réseaux de neurones fully connected et convolutif
- traitement et filtrage des images

II.3) Expérience :

Le cadre expérimental comprend trois étapes principales :

1) formation d'un classifieur binaire à l'aide des fonctionnalités extraites par le réseau de neurones (fully connected ou convolutif) à partir d'une base de données

2) évaluation de la précision de la classification par validation croisée sur le jeu de données

3) améliorer ces résultats en appliquant des filtres sur les images.

Notre algorithme de traitement de données a été testé sur des images contenant une place de parking (libre ou occupée). Pour l'évaluation, nous utilisons quatre mesures : la véracité ainsi que l'entropie croisée de nos prédictions d'un côté pour la base de données servant à l'entraînement, et de l'autre d'une base de données servant à la validation de nos résultats.

III. Machine learning :

La réalisation d'un algorithme d'apprentissage automatique s'est effectuée en plusieurs étapes. Il nous a fallu :

- Trouver les images nécessaires à l'apprentissage de notre réseau de neurones
- Créer la base de donnée qui servira d'entrée à nos algorithmes
- Créer un Algorithme de reconnaissance d'image Fully Connected
- Créer un algorithme de reconnaissance d'image convolutif

Avant de réaliser un algorithme d'apprentissage automatique, nous avons d'abord dû trouver une base de données assez complète, pertinente et importante pour pouvoir entraîner notre algorithme. Nous avons trouvé une base de données de 4000 images environ sur PKLot qui étaient triées (séparées entre les places de parking vides et celles occupées), mais cette base de données n'était pas suffisante (nous voulions au moins 7000 images). Nous avons donc trouvé une autre base de données de 3000 images, qui n'était pas triée cette fois-ci, mais qui avait l'avantage de regrouper des images prises dans des conditions météorologiques différentes. Nous avons donc dû trier cette base de données à la main.

Après avoir constitué une base de données suffisante composée d'images, nous avons dû travailler cette base de données pour qu'elle soit adaptée à notre futur algorithme d'apprentissage automatique. Nous avons alors réalisé le programme "create_training_data_train" pour créer une base de données adaptée pour réaliser un algorithme fully connected de classification, ainsi que le programme "create_training_data_CNN" pour réaliser un algorithme convolutif de reconnaissance d'image. Ces deux algorithmes prennent en argument l'emplacement fichier des images de la base de données, et renvoient 2 fichiers "X.pickle" et "Y.pickle" qui contiennent respectivement les images filtrées et toutes de même dimension (150*150 ici), ainsi qu'un vecteur contenant une indication sur l'occupation de la place (0 si la place est libre et 1 si la place est occupée). La différence entre les 2 algorithmes est que dans "create_training_data_train", les pixels des images sont des flottants alors que dans "create_training_data_CNN", les pixels sont des listes de 1 élément (car l'image est en noir et blanc, il y aurait 3 éléments si l'image était en couleur).

Une fois la base de données réalisée, nous avons créé plusieurs algorithmes d'apprentissage automatique. Nous avons tout d'abord réalisé un algorithme reposant sur un réseau de neurones fully connected. Ces réseaux de neurones prennent en entrée un vecteur contenant autant de données qu'il y a de pixels dans l'image. Le réseau est ensuite constitué de plusieurs couches de neurones, reliés aux neurones de la couche précédentes par des poids et des biais. La dernière couche de neurones est constituée de deux neurones indiquant la probabilité que la place soit occupée pour le premier, et celle pour que la place soit libre pour le deuxième. La fonction Softmax est appliquée à cette dernière couche de neurones pour que l'un des deux neurones contienne une valeur proche de 1, et pour que l'autre contienne une valeur proche de 0.

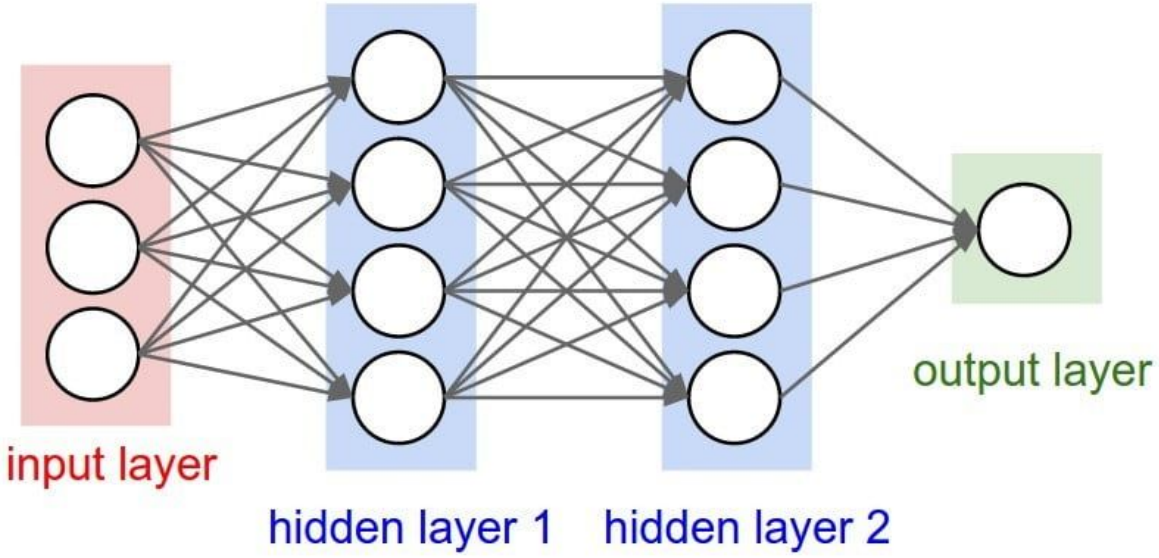

Figure 1 : principe de fonctionnement d'un réseau de neurones fully connected

Nous avons commencé par réaliser un réseau de neurones reposant sur l'API de haut niveau Keras de la bibliothèque Tensorflow, ce qui facilite beaucoup la programmation du réseau de neurones. Nous entraînons notre algorithme sur une base de données d'environ 5500 images, et gardons 1500 images de côté pour valider notre modèle. En effet, le risque est que notre algorithme apprenne uniquement à reconnaître les images de notre base de données, mais ne puisse pas reconnaître d'autres images de place de parking avec ou sans voiture. Nous obtenons alors un algorithme très satisfaisant, avec une précision pour le jeu de validation de plus de 99%.

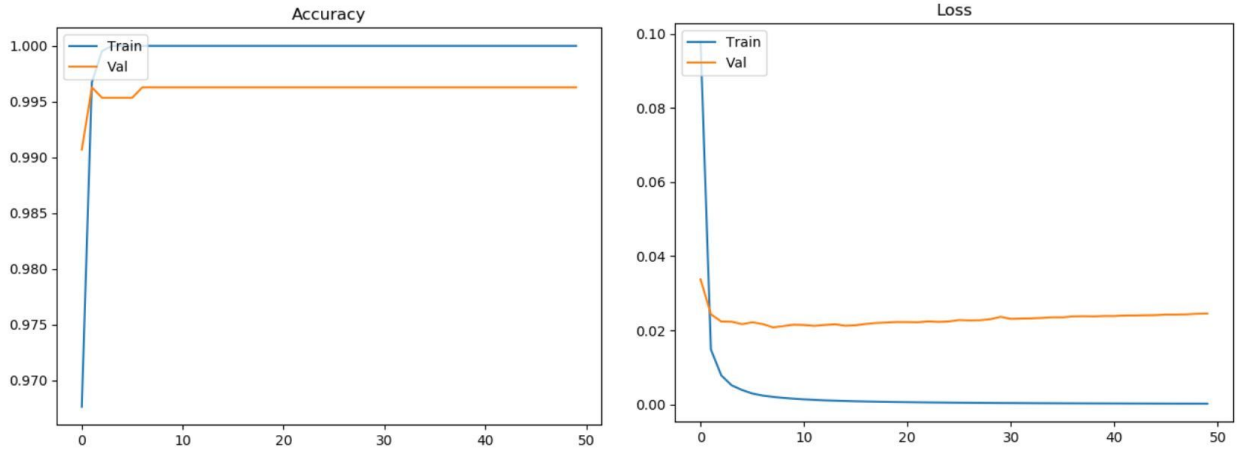

Figure 2 : Accuracy et entropie croisée de notre algorithme fully connected

Cependant, nous remarquons qu'à partir d'un certain moment, l'entropie croisée de notre base de données servant à la validation augmente. C'est ce qui est appelé "Overfitting" ou "surapprentissage". Pour palier à ce problème, nous avons ensuite voulu réaliser un programme reposant moins sur des fonctions pré-écrites. Nous avons alors essayé d'écrire un programme dans lequel nous définissons nous même nos matrices contenant nos biais et nos poids, pour ainsi pouvoir plus facilement les maîtriser. En effet, une des solutions pour palier au problème du surapprentissage est de réduire le nombre de neurones dans notre réseau de manière aléatoire. En effet, le phénomène d'"overfitting" se manifeste lorsqu'il y a trop de degrés de liberté dans notre réseau de neurones. Réduire le nombre de neurones permet de diminuer le nombre de degrés de

liberté, et réduit donc l'overfitting. Cependant, nous avons rencontré plusieurs problèmes pour réaliser ce programme. Les types utilisés (tensors), ne présentent pas les mêmes attributs que la plupart des types que l'on utilise, et cela a été très compliqué de concilier ces types avec notre programme. Nous avons réussi à faire fonctionner ce programme, mais nous obtenions une précision de 70%, ce qui n'est pas suffisant pour notre problème.

Nous avons ensuite réalisé un algorithme reposant sur un réseau de neurones convolutif. Les réseaux de neurones convolutifs, contrairement aux réseaux de neurones fully connected, prennent en entrée l'image dans sa dimension originale (pour nous une matrice de 150*150). Chaque neurone ne prend en compte qu'une petite portion de l'image (qui est définie dans les paramètres de la couche de neurones). Par ailleurs, les poids reliants chaque portion de même taille de l'image à un neurone sont identiques pour tous les neurones d'une même couche. Ainsi, il y a autant d'inconnues (les poids) que de pixels dans la portion de l'image étudiée par un neurone. Cette opération est réitérée pour que toutes les portions de l'images soient parcourues, et de manière à créer plusieurs couches de neurones. Ainsi, un réseau de neurone convolutif transforme une matrice à 3 dimensions en une autre matrice à 3 dimensions. On peut ensuite réaliser cette opération plusieurs fois et ainsi modifier de plus en plus la taille de la matrice obtenue. Finalement, il ne suffit plus que d'aplatir la dernière matrice obtenue et de réaliser une dernière couche de neurones pour n'avoir plus qu'à la fin 2 neurones, comme dans un réseau de neurones fully connected.

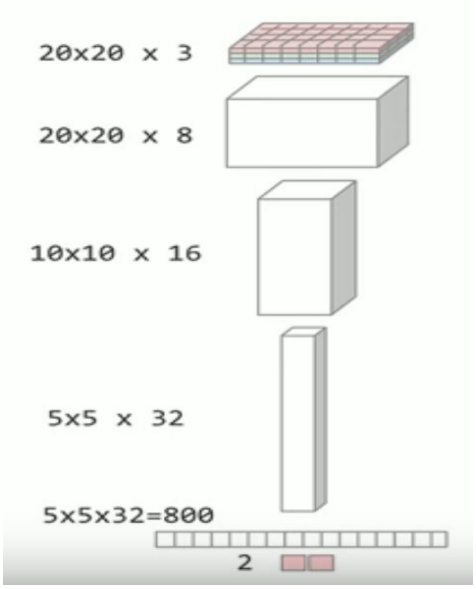

Figure 3 : principe de fonctionnement d'un réseau de neurones convolutif

Ce programme s'est avéré être beaucoup plus lent et moins efficace que le programme fully connected. En effet, nous obtenons une accuracy inférieure à 90%.

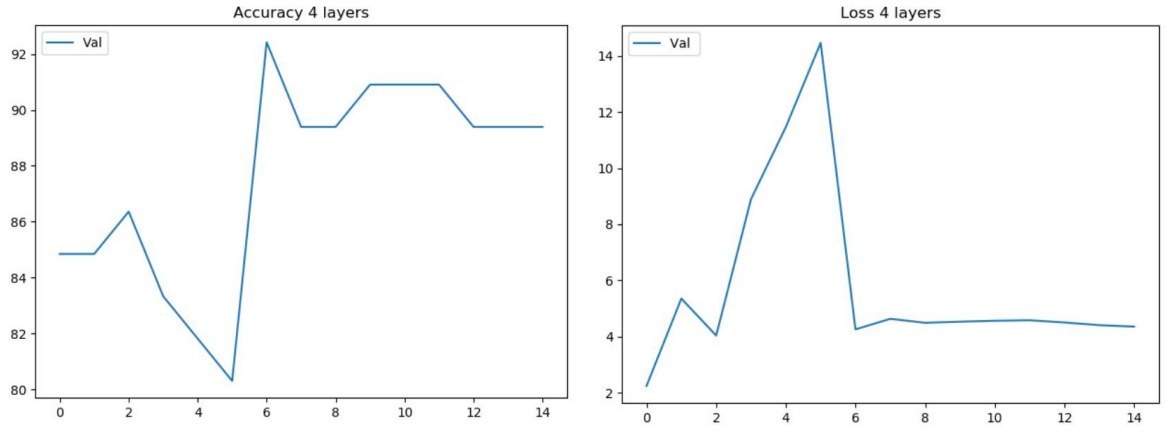

Figure 4 : Accuracy et entropie croisée de notre algorithme convolutif

Cela est dû au faite qu'un algorithme convolutif est adapté pour un apprentissage sur des bases de données beaucoup plus importantes que la nôtre.

Au regard des résultats obtenus, nous avons décidé de garder notre algorithme fully connected pour la suite de notre projet.

Nous avons ensuite entraîné cet algorithme de réseau de neurones fully connected sur la même base de données à laquelle nous avions préalablement appliqué plusieurs filtres que nous avions réalisés.

IV. Traitement des images :

Nous avons étudié les différents filtres qu'il est possible d'appliquer sur une image. Nous avons décidé que les filtres les plus adaptés pour mettre en avant les voitures présentes dans les places de parking étaient :

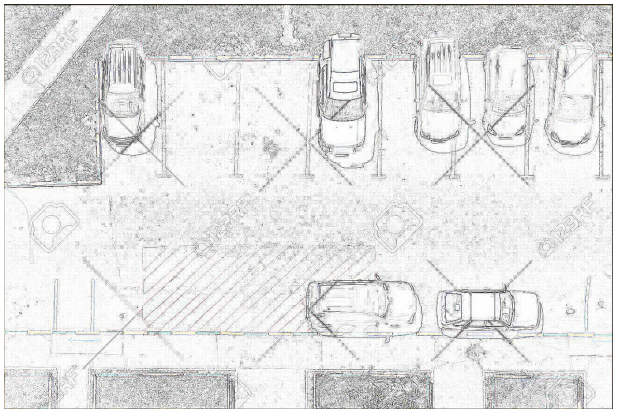

Figure 5 : Par contour d'image

Figure 6 : Par filtrage passe-haut

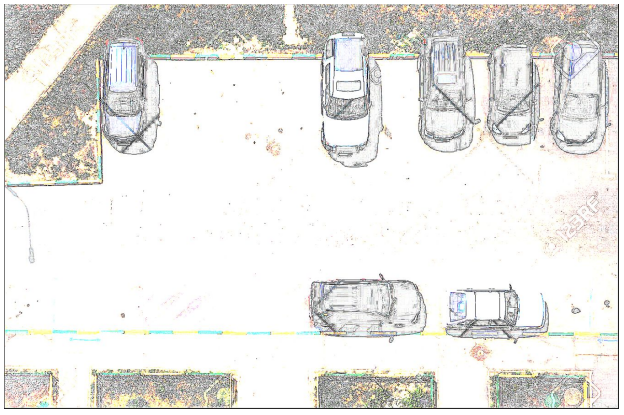

Figure 7 : Par contour d'image et filtrage passe haut

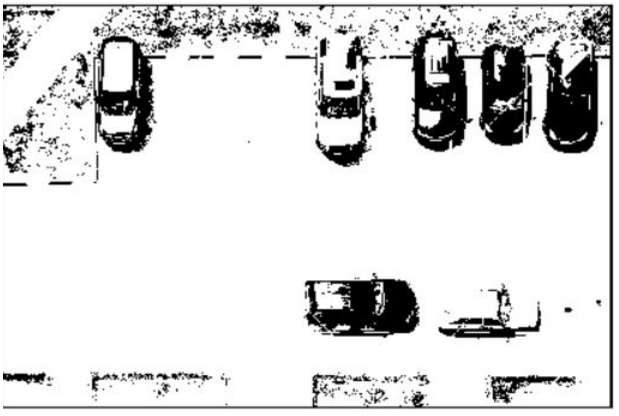

Figure 8 : Par filtrage passe-haut et Binaire

IV.1) Combinaison du traitement d'image et du machine learning :

Nous avons ensuite essayé d'améliorer les résultats de notre réseau de neurones fully connected en appliquant différents filtres sur notre base de données et en observant alors l'accuracy et l'entropie croisée obtenues.

Nous obtenons les résultats suivants :

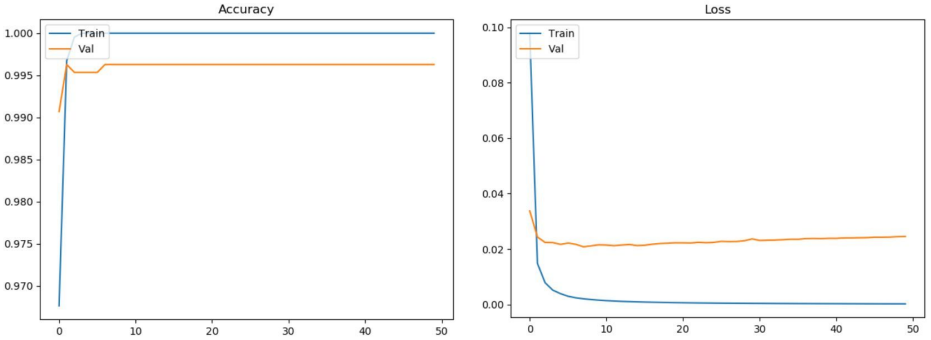

Figure 9 : Accuracy et entropie croisée de notre algorithme fully connected sans filtre sur la base de données

Nous appliquons tout d'abord un filtre passe haut sur notre base de données. On remarque que l'accuracy est maintenant de 99,8%, ce qui est supérieur à ce qu'on avait obtenu sans filtre. De plus, l'entropie croisée a été divisée par 2.

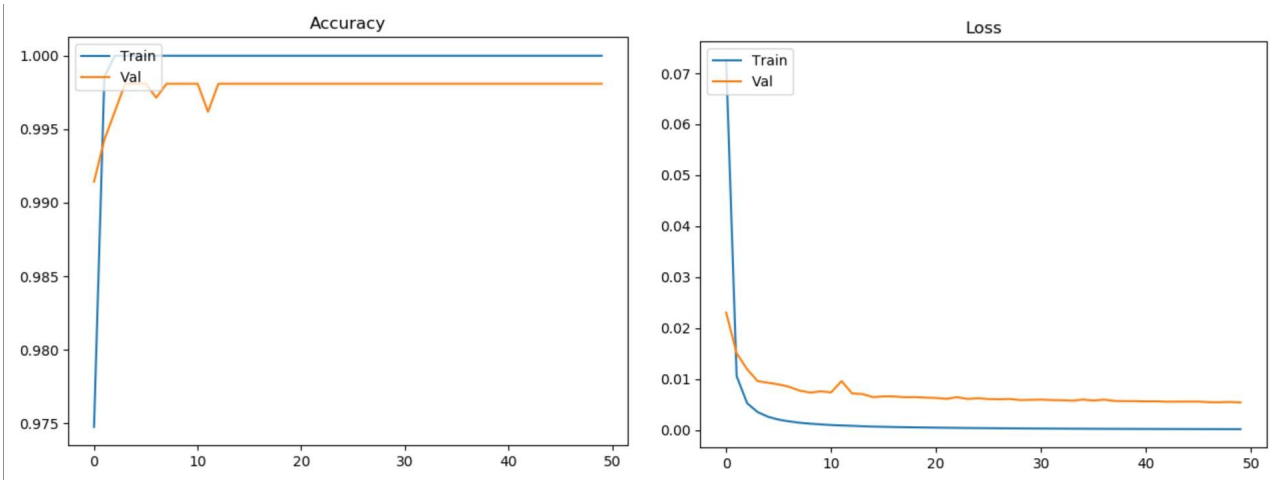

Figure 10 : Accuracy et entropie croisée de notre algorithme fully connected avec un filtre passe haut sur la base de données

Nous appliquons ensuite un filtre passe haut ainsi qu'un filtre binaire sur les images de notre base de données. On remarque que cette fois-ci l'accuracy varie beaucoup en fonction de l'époque d'entraînement à laquelle on s'intéresse. De plus, l'accuracy atteint au maximum 96%, ce qui est inférieur aux résultats obtenus précédemment. Ce filtre n'est donc pas un filtre à conserver pour notre étude.

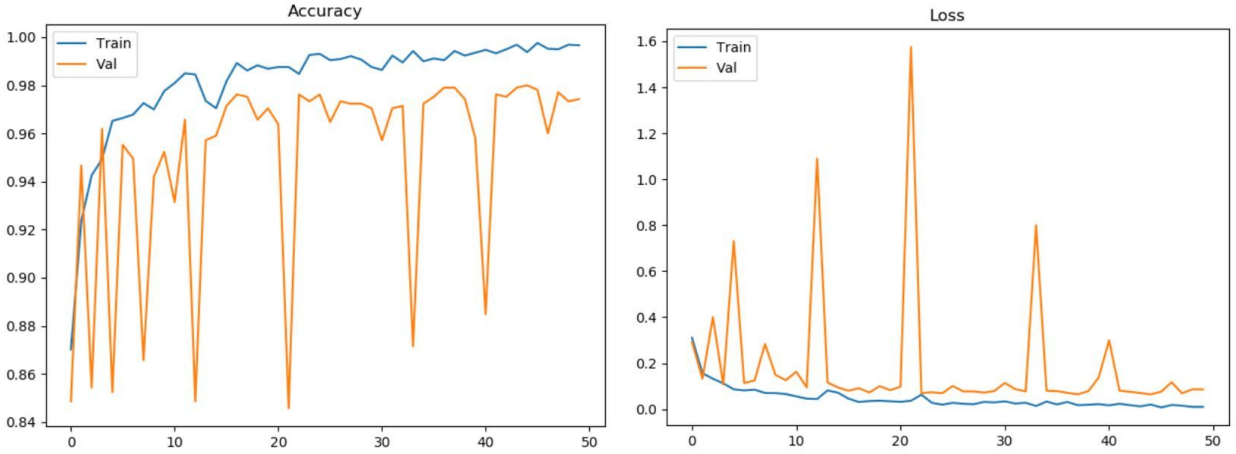

Figure 11 : Accuracy et entropie croisée de notre algorithme fully connected avec un filtre passe haut et un filtre binaire sur la base de données

Nous essayons finalement un filtre passe haut et un filtre contour d'image sur la base de données. On remarque cette fois-ci encore de bons résultats : l'accuracy est de 99,2% environ est l'entropie croisée est de 2%, ce qui est très satisfaisant.Cependant, ces résultats restent inférieurs à ceux obtenus lorsque que la base de données avait été filtrée avec un filtre passe haut simple.

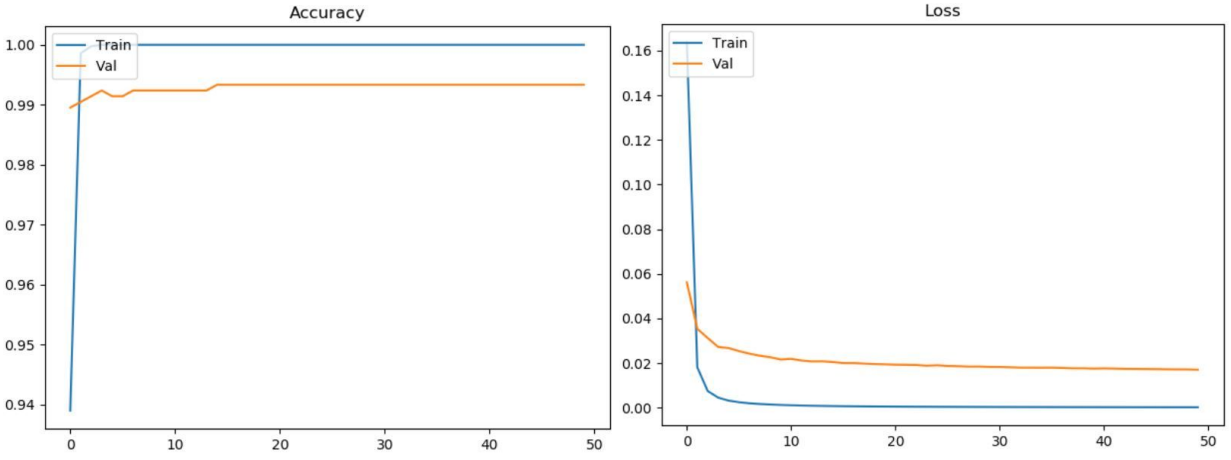

Figure 12 : Accuracy et entropie croisée de notre algorithme fully connected avec un filtre passe haut et un filtre contour d'image sur la base de données

Ainsi, l'application d'un filtre passe haut sur notre base de données semble améliorer les résultats de notre réseau de neurones : nous allons donc conserver ces paramètres pour la suite du projet.

V. Cas d'étude :

Nous avons voulu tester notre réseau de neurones sur l'image de parking suivante :

Figure 13 : Parking pour le cas d'étude

Pour notre cas d'étude, nous sommes partis d'une image de parking supposée fixe pour traduire le fait que ce parking nous est familier. Nous avons commencé par travailler cette image pour la diviser en sous image. Ces sous-images sont de la taille d'une place de parking. Nous avons ainsi séparé les différentes places de parking de cette image à la main. Nous avons ensuite trié les images pour supprimer celle qui ne sont pas des places (par exemple les routes etc.). Nous obtenons les images suivantes :

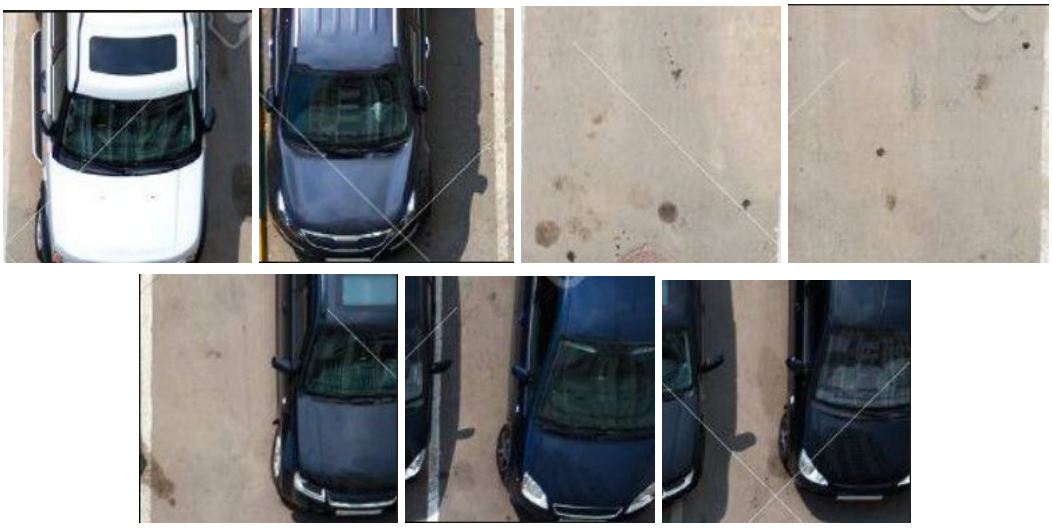

Figure 14 : Places de parking séparées utilisées pour l'étude de cas

Nous avons ensuite appliqué un filtre passe haut aux différentes places de parking obtenues. Les images alors obtenues ont été mises comme entrées pour notre réseau de neurones. Les résultats obtenus en sortie sont les suivants (prévu correspond à ce qui a été prédit par réseau de neurones, et voulu correspond à ce que l'on attendait) :

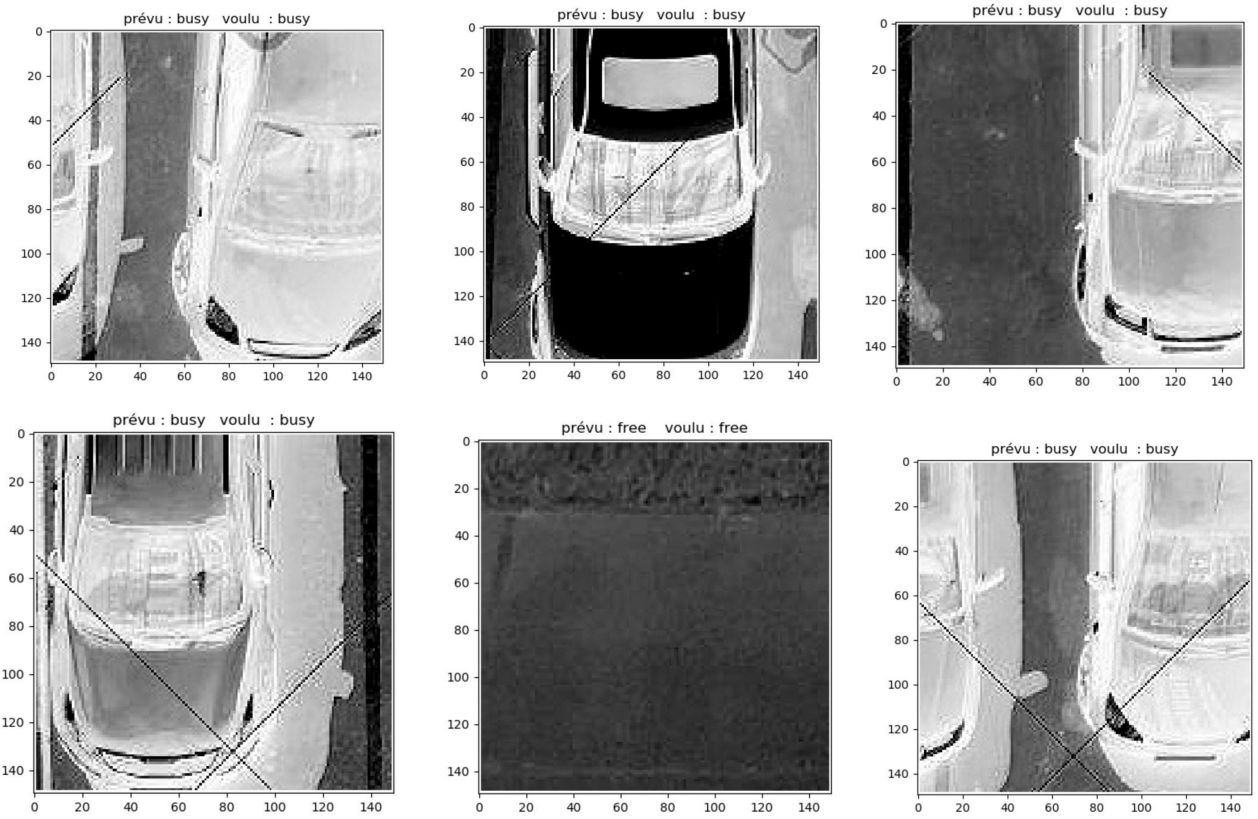

Figure 15 : Prédiction des images de notre cas d'étude par notre algorithme

On remarque que toutes les images ont été prédites de la bonne manière. Ainsi, notre programme présente des résultats satisfaisants.

VI. Conclusion et suite du projet :

Ainsi, nous avons réussi à réaliser un algorithme d'apprentissage automatique faisant la différence entre une place de parking vide et une place de parking occupée. Nous avons réalisé plusieurs type de réseaux de neurones (fully connected et convolutif) afin de comparer leurs résultats et de choisir le plus pertinent. Nous avons ensuite testé plusieurs types de filtrages sur les images de notre base de données pour voir si cela avait un impact sur les performances du réseau de neurones. Nous en sommes venu à la conclusion que la combinaison la plus performante était un réseau de neurones full connected et un filtrage passe haut sur les images de la base de données.

La beauté de notre projet réside dans le fait qu'un tel cadre pourrait être mis en œuvre dans l'optique d'être vendu au grand public, mais aussi à des entreprises. On pourrait imaginer créer une application mobile qui est soit rattachée à une grande surface pour indiquer les places disponibles dans son parking avec des images prises à partir d'une caméra fixe, soit pour une utilisation plus personnelle dans des espaces disposants de caméras, ou ne disposants pas de caméra en utilisant des images prises à partir d'un drone appartenant à l'utilisateur. Notre application ne nécessiterait donc aucune formation et peut commencer à fonctionner dès la minute de l'installation.

Cependant, pour un système IGP pratique, plusieurs aspects du cadre proposé peuvent être améliorés. Tout d'abord, le modèle n'était pas formé ou testé dans des conditions de faible luminosité telles que la nuit, ce qui peut limiter son utilité et la rendre moins persuasive pour une utilisation commerciale future. Deuxièmement, en pratique, il faudrait être capable de détecter les zones prédéfinies des places de stationnement automatiquement plutôt que d'identifier manuellement les limites. Troisièmement, le cadre devrait être testé sur des images de surveillance en temps réel pour examiner l'applicabilité de la caméra en direct. Quatrièmement, l'ambiguïté causée par l'occupation partielle des places de stationnement peut être améliorée par une dynamique méthode de segmentation.

VII. Bibliographie :

http://lisa.ulb.ac.be/index.php/Programmation_en_Python :
Répertorie les différentes applications pour le traitement d'images

http://www.tangentex.com/TraitementImages.htm :
Rudiments de traitement d'images avec Python

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html :
Tutoriel Open-CV

https://docs.opencv.org/3.2.0/d7/d4d/tutorial_py_thresholding.html :
Documentation sur le filtrage par thresholding

http://effbot.org/imagingbook/imageops.htm :
Module ImageOps

https://www.youtube.com/watch?v=hP7Ac8S9Tgs&list=PLpEPgC7cUJ4byTM5kGA0Te1jUeNwbSgfd
Formation à Tensorflow 2.0 et présentation des réseaux de neurones avec Keras

https://www.youtube.com/watch?v=u4alGiomYP4&t=945s
Réalisation d'un réseau de neurones fully connected sans Keras

http://cnrpark.it/
Base de données utilisée

VIII. Codes utilisés et mode d'emploi :

Pour faire tourner l'algorithme d'apprentissage automatique fully-connected (Tensorflow 2.0) :

- Télécharger la base de données à partir du lien suivant : http://cnrpark.it/
- Lancer le programme "create_training_data_train" en oubliant pas de modifier le chemin d'accès vers la base de données et d'appliquer un filtrage choisi par vos soins et changeable
- Lancer le programme "Keras"

Pour faire tourner l'algorithme d'apprentissage automatique sans passer par l'API Keras (Tensorflow 1.13) :

- Télécharger la base de données à partir du lien suivant : http://cnrpark.it/
- Lancer le programme "create_training_data_train" en oubliant pas de modifier le chemin d'accès vers la base de données
- Lancer le programme "Tensorflow" (ne fonctionne pas très bien, à améliorer)

Pour faire tourner l'algorithme d'apprentissage automatique convolutif (Tensorflow 1.13) :

- Télécharger la base de données à partir du lien suivant : http://cnrpark.it/
- Lancer le programme "create_training_data_CNN" en oubliant pas de modifier le chemin d'accès vers la base de données
- Lancer le programme "CNN"

Pour réaliser notre cas d'étude :

- Faire tourner l'algorithme d'apprentissage automatique fully-connected
- Séparer les places de parking de l'image voulue à la main à partir de l'algorithme "rogner"
- Créer la base de données contenant les images des places auxquelles a été appliqué un filtrage choisi par nos soins pour maximiser l'accuracy du réseau de neurones (ici un filtre passe-haut) grâce à l'algorithme "create_training_data_parking"
- Lancer l'algorithme "notre_test" qui va alors appliquer le réseau de neurones fully connected réalisé précédemment aux quelques places de parking que nous venons d'extraire et qui va prédire leur occupation ou non.