Département technique de Virton
Master en Sciences de l'ingénieur industriel
Orientation automatisation

Systèmes intelligents

Création d'une porte intelligente pour chien de race

Slama Rim

Florian Alonso – Christophe Humblet - Arnaud Schmadtke
Année académique 2019-2020

Table des matières

1 Introduction

Dans le cadre du cours de Systèmes Intelligents de Master 1, nous avons dû trouver une idée de projet mettant en œuvre l'intelligence artificielle, et plus particulièrement une branche de celle-ci : la "Computer Vision" ou vision par ordinateur. Ce travail permettra d'exploiter les connaissances acquises lors du cours de microprocesseur (B3) et des cours de modélisations de processus et de systèmes intelligents (M1). Afin de réaliser ce projet, deux éléments devront être utilisés pour répondre aux attentes. Notre prototype devra donc comporter l'utilisation d'un Raspberry, essentiellement pour la partie consacrée à l'intelligence artificielle, ainsi qu'un Arduino pour tout ce qui concerne les fonctionnalités du projet.

Notre projet consistait dans un premier temps à réaliser une chatière intelligente afin de permettre aux propriétaires de chats de ne laisser rentrer que leur animal de compagnie. Cependant, après réflexion, notre projet a légèrement été modifié. Nous avons donc fait une trappe intelligente pour chien au lieu d'une chatière. Par conséquent, le principe reste similaire, mais la demande est assez différente puisque ce produit n'existe pas encore sur le marché. Cela permet d'ajouter une plus-value à notre produit.

L'objectif est de détecter la présence d'une race de chien spécifique avec la caméra afin que la trappe se déverrouille et que le chien puisse entrer dans la maison. La détection d'autres animaux laisserait la trappe fermée par une serrure et un signal sonore serait émis par un buzzer. Par exemple, si un chat se présente à la porte, un son strident sera produit pour faire fuir celui-ci. Si un chien d'une autre race est devant le volet, il ne pourra pas entrer non plus. L'intérêt du projet est donc de personnaliser la trappe pour chien afin qu'elle ne corresponde qu'aux chiens appartenant à une famille. En plus de différencier les chiens des autres animaux, il sera donc nécessaire de reconnaître précisément une ou plusieurs races de chiens. Un écran affichera également diverses informations telles que des éléments d'accueil pour le chien de la famille ou, au contraire, des messages d'intrusion en association avec le buzzer.

Dans ce rapport, vous trouverez donc une explication des démarches mises en place afin d'accomplir nos objectifs. En effet, nous abordons les recherches et tests effectués pour concevoir notre idée. Nous expliquons aussi les méthodes de détection et les algorithmes liés à la reconnaissance d'objets (animaux dans notre cas). Les méthodes utilisées et testées pour notre application sont le "Supervised Learning" (Apprentissage Supervisé) avec le "SVM, support vector machine" (Machines à Vecteur Support) et le "Deep Learning" (Apprentissage profond) avec un "CNN, Convolutional Neural Networks" (Réseau de neurones convolutif). Pour utiliser ces algorithmes, il est également important de mettre en place et récupérer des "Datasets", bases de données. Ensuite, nous passons à la phase de réalisation du projet à savoir la réalisation de la maquette avec l'implémentation des différents éléments présents dessus (la caméra de reconnaissance et le système de la porte intelligente). Pour finir, nous abordons les tests effectués ainsi que les améliorations qui peuvent être apportées à notre prototype.

2 Gestion de projet

Afin de réaliser ce travail, il est essentiel d'avoir une méthode de travail adéquate. L'objectif de cette section est de présenter la gestion mise en place lors de ce projet. Cette section va être divisée en différentes points. Chaque section représente les outils qui nous ont permis d'avoir une gestion optimale du projet.

2.1 Outil de communication

Dans un projet de groupe, il est essentiel de communiquer entre chaque membre surtout lorsqu'il n'est plus possible d'effectuer des réunions en présentiel. Ainsi, nous avons utilisé la plateforme "Teams" pour échanger des informations tout au long du travail. Cela permettait de faire des réunions pour discuter sur l'avancement du travail ainsi que pour la mise en commun et l'approbation des tâches effectuées. En effet, nous avons effectué des réunions pour avoir une gestion précise de notre production finale. Ces dernières avaient un ordre du jour précis comme par exemple la mise en commun des parties que nous avons travaillées à domicile, réaliser des parties du travail tous ensemble ou encore soumettre des idées nouvelles. Ces réunions permettent d'accomplir les taches fixées dans une date limite que nous avons choisie via le "Gantt".

2.2 Planification – Gantt

Afin d'optimiser au mieux notre temps de travail, nous nous sommes divisés l'ensemble du projet entre les membres de notre groupe et nous avons fixés des dates limites pour chaque étape. Cela s'est fait par le biais d'un outil de planification, le diagramme de "Gantt". Celui-ci est un outil utilisé en gestion de projet, qui permet de visualiser dans le temps les diverses tâches composant un projet. Il était mis à jour régulièrement, en fonction de notre état d'avancement. Notre projet s'est déroulé en plusieurs étapes afin de valider pas à pas nos objectifs :

- Premièrement, nous nous sommes penchés sur la reconnaissance entre un chat et un chien.
- Ensuite, nous avons ajusté nos résultats pour détecter la présence d'un chien par rapport aux autres animaux.
- Enfin, pour rendre cette trappe intelligente pour chiens personnalisable, nous avons travaillé sur la reconnaissance d'une race précise de chien que pourrait posséder une famille. La reconnaissance précise d'un ou plusieurs chiens, de races différentes, a été une option supplémentaire qui sera expliquée dans la partie consacrée à l'amélioration du projet.
- Concernant les fonctionnalités, nous avons géré l'actionnement d'une serrure, l'affichage des informations sur un écran LCD et l'utilisation d'un buzzer pour émettre des sons répulsifs. Certaines options pourront également être ajoutées dans les améliorations telle que l'affichage des heures d'entrée et de sortie du chien.

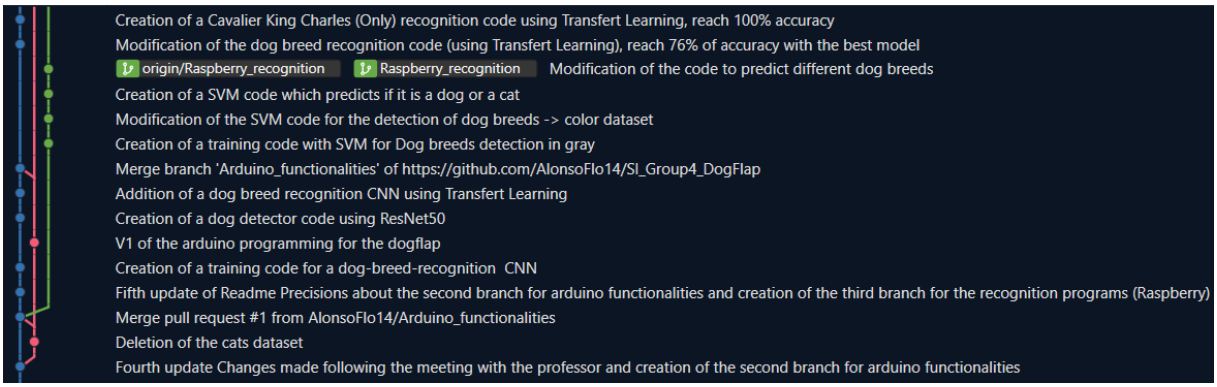
La partie concernant l'intelligence artificielle, reconnaissance est gérée par le Raspberry. Pour comparer divers algorithmes, nous avons essayé l'algorithme supervisé SVM spécialisé dans le traitement d'images. En parallèle, nous avons réalisé un CNN pour pouvoir comparer l'efficacité des programmes. Le réseau de neurones étant plus puissant, nous avons travaillé uniquement sur cet algorithme à la fin du travail pour l'intégrer au Raspberry. Concernant les fonctionnalités du projet, les éléments sonores, les affichages, le

verrouillage/déverrouillage ont été gérés par un Arduino. Toutes ces étapes ont été reprises dans le Gantt présent en annexe.

Au niveau de la répartition du travail, Humblet Christophe s'est consacré à la programmation de l'Arduino alors qu'Alonso Florian et Schmadtke se sont penchés sur la partie consacrée à l'intelligence artificielle. Dans un premier temps, Florian a réalisé le SVM pendant qu'Arnaud a testé un CNN. En second lieu, après comparaison des premiers résultats obtenus, ils se sont consacrés à l'amélioration et l'implémentation du CNN dans le Raspberry.

2.3 Gestion des fichiers - Github.

Afin de gérer les différents fichiers créés tout au long du projet, un outil très utilisé dans le monde de la programmation est "Github". Cette plateforme propose un hébergement pour des codes open source et est étroitement lié à "Git" puisque ce dernier est le système de contrôle de version, tandis que "GitHub" est le service qui héberge les dépôts "Git" et aide les utilisateurs à collaborer pour le développement de logiciels. Cet outil permet de suivre la progression d'un projet de programmation. En effet, à chaque modification du programme, il est possible de consulter les changements apportés aux codes et ainsi choisir de revenir ou non à des versions antérieures. Pour cela, il est important de maitriser cet outil et d'être structuré lors de l'ajout, modification de codes en effectuant des commentaires adéquats. Il est également intéressant de l'associer à un logiciel d'interface graphique, "Sourcetree". Le dépôt et rapatriement de fichier peut ainsi directement s'effectuer par son intermédiaire au lieu d'utiliser un invite de commande ou directement le site Internet "Github". Son rôle est donc bien d'avoir une représentation graphique de l'avancement du projet.



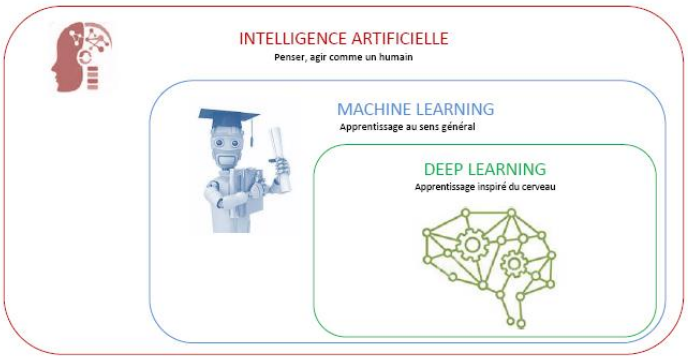
Visualisation du projet dans Sourcetree

Comme nous pouvons sur l'image ci-dessus, plusieurs branches ont été créés afin que chaque membre du groupe puisse travailler de façon indépendante. La branche principale a été consacrée principalement à la réalisation des algorithmes utilisant le CNN ainsi qu'à l'actualisation du fichier détaillant les étapes du projet. A la fin du projet et après fusion des branches, cette branche maître rassemblera l'ensemble des fichiers des autres branches créées. Ainsi, une branche a été consacrée aux fonctionnalités du projet (haut-parleurs, écran, ...) qui sont gérées par un Arduino. La troisième branche contient également les programmes de reconnaissance qui seront gérés par le Raspberry. L'intérêt de créer cette branche parallèlement à la branche maître réside, comme nous l'avons dit, lors de la fusion. En effet, la branche principale est également dédiée aux programmes d'IA (détection, ...) et il est donc possible que certains fichiers soient communs aux deux branches. Ainsi, lors de la fusion, il sera facile de voir les différences entre les fichiers en question et de choisir le fichier que nous souhaitons conserver.

A la fin du projet, il est ainsi possible de suivre toutes les étapes qui ont été réalisées au fur et à mesure. Cela permet à un utilisateur quelconque de reprendre facilement en main le projet en visualisant les démarches qui ont été faites. De plus, un fichier a été créé et reprend le résumé du projet avec le détail de chaque phase. Pour finir, un fichier "Readme" est présent pour que toute personne utilisant notre projet sache le faire fonctionner. Le lien renvoyant sur notre Github se trouve dans la bibliographie.

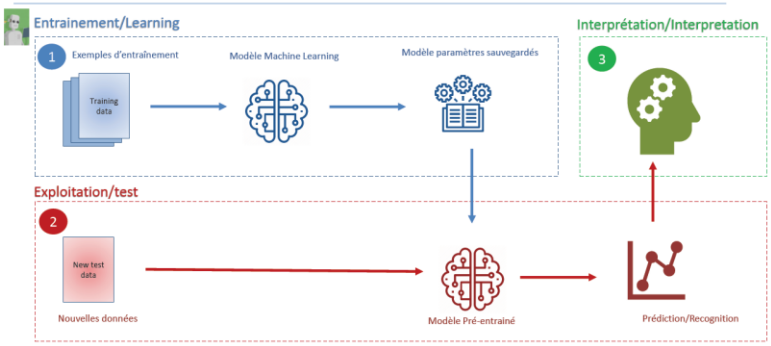
3 Recherches et tests

Comme vous l'aurez vite compris, il est bien question d'exploiter le domaine de l'intelligence artificielle pour ce projet puisque nous devons analyser et transformer les données reçues d'une caméra en des actions intelligentes.



Domaine de l'IA

Il existe une multitude d'algorithme de "Machine Learning" pour effectuer de la reconnaissance d'images ou vidéos et il nous faut donc en choisir un en vue de réaliser notre projet. Ce "Machine Learning" ou apprentissage automatique en français est un sous-ensemble de l'intelligence artificielle qui utilise souvent des techniques statistiques pour permettre aux ordinateurs d'apprendre. Comme le montre le schéma ci-dessous, la procédure d'apprentissage consiste à rassembler et traiter les données afin que les machines les comprennent. Ensuite, il faut construire le modèle, algorithme basé sur le problème à résoudre. Ce modèle sera ensuite formé avec les données de formation recueillies pour finalement prédire les résultats futurs en utilisant le modèle formé sur des données invisibles.



Processus du "Machine Learning"

Il est donc crucial de choisir un algorithme adéquat car il va déterminer l'efficacité de la reconnaissance de notre dispositif. Nous avons donc réalisé des tests pour en valider un seul.

3.1 La détection

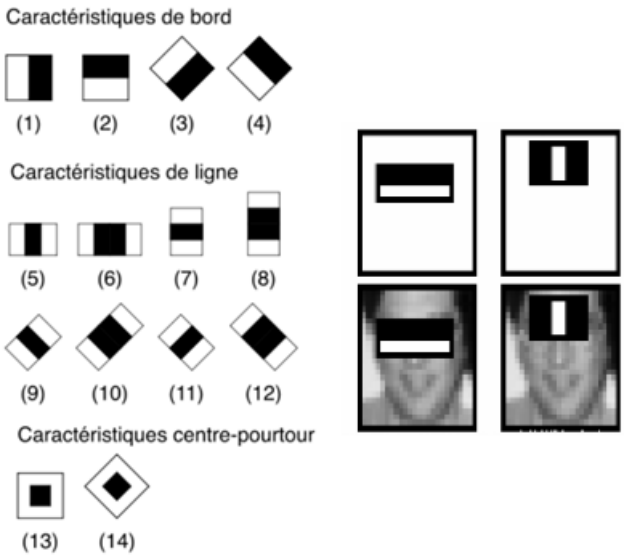
Avant d'essayer de reconnaitre une race de chien, il est important de savoir si un chien est présent devant la caméra. En effet, nous trouverons deux options à la fin de notre algorithme : le chien est un Cavalier King Charles ou un chien de race inconnue. Or s'il n'y a rien en face de l'objectif de la caméra, nous pourrions tromper l'algorithme et lui faire prédire qu'il y a un Cavalier King Charles ou un chien inconnu. Cela poserait un problème pour l'activation des actionneurs. Si l'algorithme prédit en continu alors l'écran LCD afficherait en continu qu'il y a un chien inconnu ou un Cavalier. On peut imaginer que le buzzer sonnerait la plupart du temps.

Nous avons donc besoin de deux détecteurs : un détecteur de chat et un détecteur de chien. Il est facile de réaliser un détecteur de chat en utilisant un "Haar Cascades[1]" entrainé pour reconnaitre la tête de ceux-ci.

3.1.1 Le "Haar Cascade Classifiers"

Le classificateur en cascade de Haar est une méthode publiée en 2001 par Paul Viola and Michael Jones et qui concerne la détection d'objets. Cette méthode entre les caractéristiques de Haar dans une série de classificateurs d'où l'emploi du terme "Cascade" pour identifier les objets dans une image. Ainsi, la détection de visage, de yeux ou même de chats dans notre cas peut facilement être réalisée.

Le but de cette méthode est de parcourir une image à l'aide d'une fenêtre glissante et de déterminer si un visage est présent. L'ensemble de l'image est ainsi parcouru en calculant un certain nombre de caractéristiques dans des zones se chevauchant. Celles-ci sont nommées caractéristiques pseudo-Haar par référence aux ondelettes de Haar.



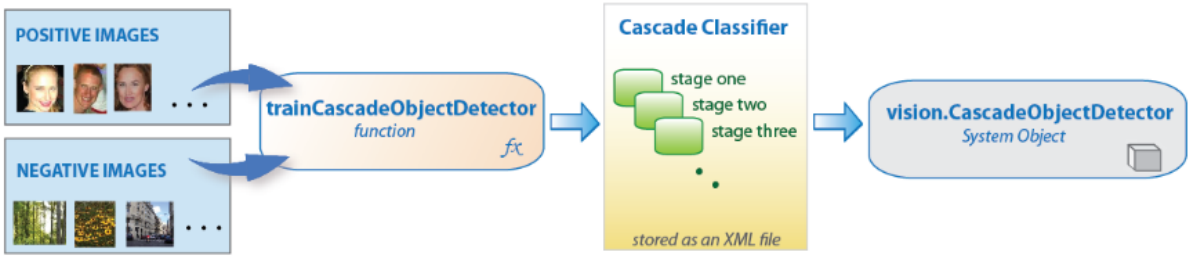
Caractéristiques pseudo-Haar

Pour chaque caractéristique, la somme de pixels délimités par la zone sombre soustrait à la somme des pixels délimités par la zone claire a été calculée. Ainsi, il est clair qu'il est préférable d'appliquer cette méthode sur une photo à nuance de gris.

Les classificateurs sont formés à l'aide de nombreuses images positives et négatives. Ainsi, les caractéristiques recueillies vont être analysées dans des classificateurs l'un à la suite

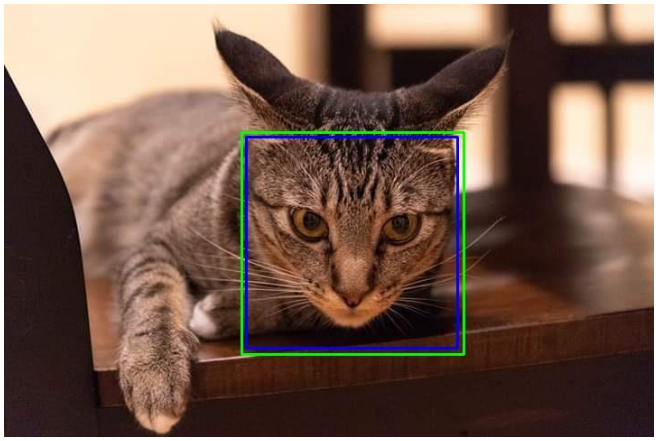
[1] Le terme "Haar" fait référence aux ondelettes de Haar, fonction créée par Alfréd Haar.

de l'autre d'où le terme cascade. Ces classificateurs vont ainsi être appliqués successivement sur une région d'intérêt jusqu'à ce qu'un des stages échoue ou qu'ils soient tous validés. L'idée est de rejeter les zones ne contenant pas l'objet avec le moins possible de calculs. Le premier classificateur est donc le plus optimisé et permet de rejeter rapidement une zone si l'objet recherché ne s'y trouve pas. Si potentiellement l'objet s'y trouve, alors le deuxième classificateur est utilisé et ainsi de suite jusqu'au dernier, afin de déterminer si oui ou non, nous sommes en présence d'un visage ou autre en fonction de l'analyse effectuée. Ainsi, nous pouvons constater que l'entraînement d'un classificateur peut durer longtemps. C'est pourquoi il existe des classificateurs déjà entrainés afin de réaliser la détection souhaitée comme la détection d'une tête de chat dans notre cas.



Haar Cascade Classifiers

Nous avons vu en cours comment tracer un rectangle autour de la tête des chats sur un flux vidéo. On peut alors détourner cette application pour vérifier s'il y a bien une tête de chat dans ce rectangle. Si c'est le cas alors il y a un chat devant la caméra, sinon il n'y en a pas. Cette méthode comporte cependant des lacunes car si le chat est tourné dans l'autre sens, de manière que la caméra ne puisse pas voir sa tête, alors il ne le détectera pas. Cependant, vu que nous voulons empêcher les chats d'entrer dans notre domicile, dans la plupart des cas le chat sera tourné vers la caméra.



Exemple de détection de tête de chat

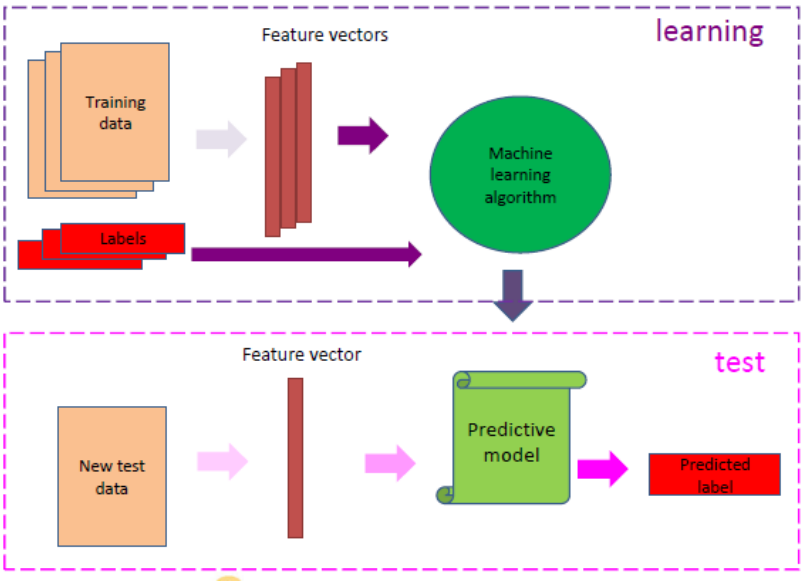
Pour détecter les chiens, c'est une autre paire de manche. En effet, nous n'avons pas trouvé un Haarcascade entrainé pour détecter leur tête. Nous avons donc dû trouver une autre solution. Celle-ci consiste à utiliser un CNN déjà entrainé avec des centaines d'images et durant une longue période : le ResNet50. Celui-ci peut prédire environ 120 races de chien. On peut se servir de cette info pour détecter un chien sur une image. En effet, nous avons accès au dictionnaire de prédiction du ResNet50 et si celui-ci nous retourne une valeur allant de 151 à 268, alors un chien est présent sur cette image. L'avantage de cette méthode est qu'elle nous permet de détecter des chiens de face, de profile et de dos.

3.2 L'algorithme de reconnaissance

3.2.1 Supervised Learning – SVM

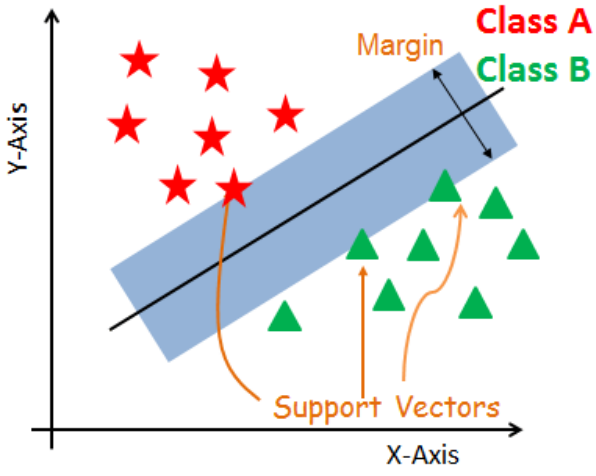
- Point de vue théorique

Contrairement à un apprentissage non-supervisé qui ne nécessite pas l'utilisation d'étiquettes ("Labels") et qui doit trouver le modèle ou la structure sous-jacente dans les données en utilisant que les données d'entrée (X), l'algorithme SVM est une méthode supervisée qui nécessite donc l'utilisation de "labels". Le modèle doit donc apprendre une fonction de mappage entre les données d'entrée et la sortie.



Supervised Learning

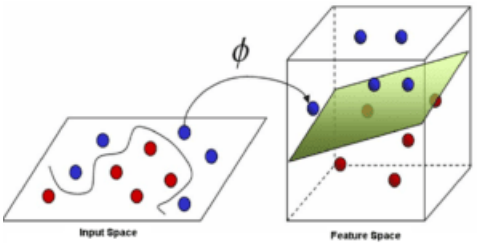
Ainsi, le "Support Vector Machine" ou Machine à Vecteurs de Support en français est un algorithme qui permet de résoudre des problèmes de classification, régression ou détection d'anomalie. Ainsi, dans notre cas, il va nous permettre de séparer nos données en différentes classes au moyen d'une frontière. Cet algorithme va faire en sorte que la distance entre les différents groupes de données et la frontière (aussi appelé l'hyperplan) qui les sépare soit maximale.



Représentation graphique de la méthode SVM

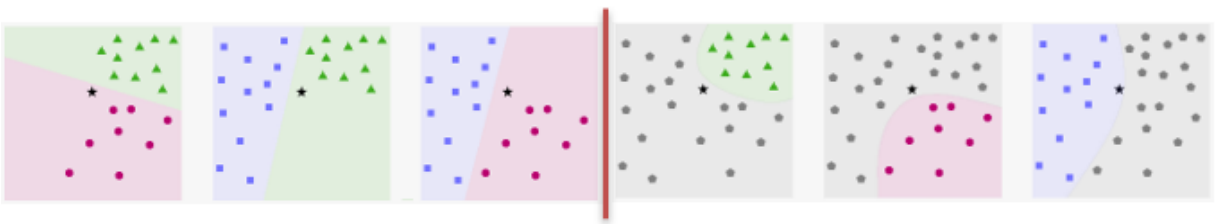
Comme on peut le voir sur cette représentation, cette distance de séparation est aussi appelée "marge" d'où le second nom de cette méthode, "Séparateurs à Vaste Marge". Les vecteurs de support représentent les données les plus proches de la frontière et ce sont ceux-ci qui déterminent donc l'hyperplan. Nous pouvons y voir tout l'intérêt de cette méthode puisque seuls les vecteurs supports vont être déterminants. Les exemples utilisés pour la détermination de l'hyperplan ne sont plus utiles.

Toutefois, comme le montre le graphe, la notion de frontière suppose que les données peuvent être séparées linéairement. Cependant, c'est rarement le cas. Nous nous retrouvons donc avec des "SMV" de type linéaire et d'autres de type non-linéaire. Ainsi, les "SVM" reposent sur l'utilisation de "Kernel", noyaux en français. Ces noyaux sont des fonctions mathématiques non-linéaires qui séparent les données en les projetant dans un espace vectoriel de plus grande dimension tout en utilisant la technique de maximisation. Cette dernière permet une meilleure robustesse du modèle.



Projection dans un espace vectoriel de plus grande dimension

Ces noyaux permettent donc de transformer une entrée en une forme voulue et donc ici, une entrée de faible dimension en un espace de dimension supérieure. Pour cela, plusieurs types de noyau existent comme le noyau linéaire (qui effectue un produit scalaire), le noyau polynomiale (qui généralise le noyau linéaire), le noyau de type radiale gaussien par exemple (qui permet d'atteindre des dimensions infinies), … Chaque noyau offre des résultats différents et il est donc intéressant de choisir celui qui convient au mieux à notre situation en testant ceux-ci. De plus, nous sommes souvent contraints à devoir classer des ensembles multiples. Ainsi, des stratégies multi-classes existent aussi comme le propose "Scikit-learn". Cela consiste à ajuster un classifieur par paire de classes. Au moment de la prédiction, la classe qui a reçu le plus de votes est sélectionnée. Nous retrouvons donc le classifieur "un contre un" ou "un contre le reste". La stratégie "un contre un" est souvent préférée car elle peut être avantageuse pour des algorithmes contenant plusieurs échantillons. En effet, cette méthode est plus lente car elle nécessite d'ajuster les classificateurs mais chaque problème d'apprentissage individuel n'implique qu'un petit sous-ensemble de données alors qu'avec "un contre le reste", l'ensemble de données complet est utilisé "n" fois.



One VS One *One VS All*

Comme le traduit plus explicitement ces images, on peut voir la différence entre les deux méthodes en sachant toujours que la classe qui a reçu le plus de votes est sélectionnée (la classe rose pour le "One VS One" alors que ça sera la bleue pour le "One VS All").

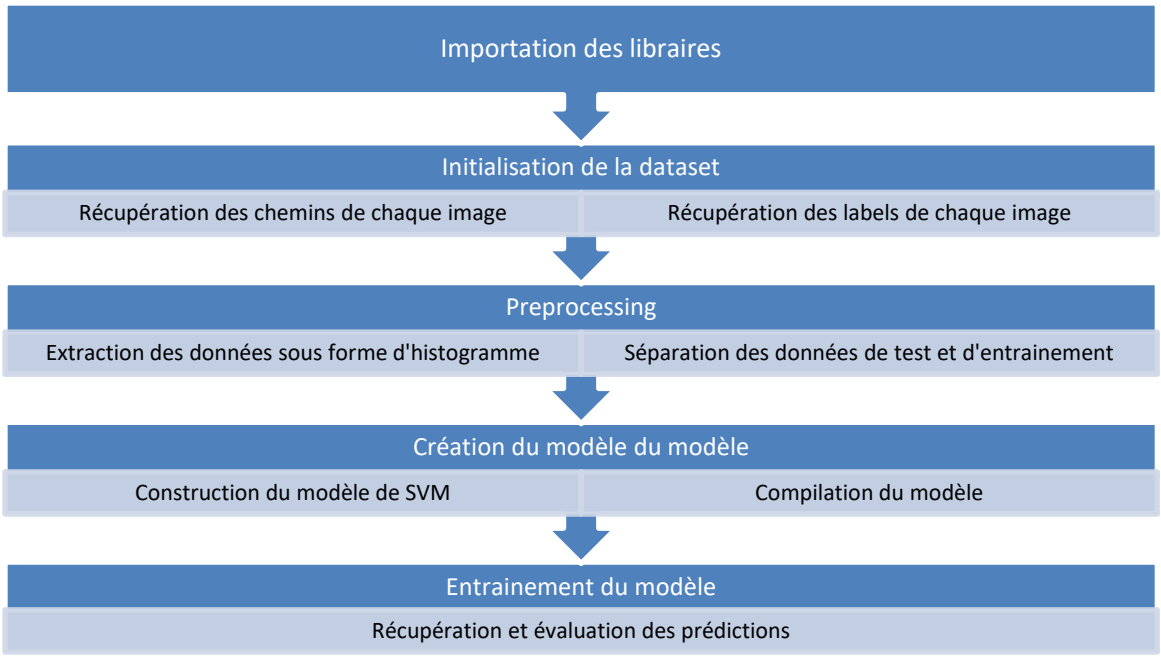
Pour conclure, le SVM est souvent utilisé pour les problèmes de classification de texte, attribution de catégorie, ... et également pour les problèmes de reconnaissance d'image. Cependant, lorsqu'on fait référence à la reconnaissance d'image, on parle plus particulièrement de reconnaissance de forme et de couleur. Ce qui, comme nous pouvons le constater n'est peut-être pas optimal pour notre situation puisque les couleurs peuvent varier entre une même race et les formes ne sont pas significatives.

Ainsi, bien que les performances du SVM soit en général comparable à un réseau de neurones tout en utilisant des bases de données réduites, ce n'est pas toujours le cas pour la classification d'images. En effet, ce type de classification nécessite une base de données plus fournie et il est donc plus difficile d'atteindre des résultats significatifs puisque cet algorithme est moins efficace sur les bases de données contenant du bruit. Le temps d'entrainement du SVM est également très long pour des bases de données volumineuses. De plus, cette méthode peut être souvent sujet au phénomène d'"Overfitting" (sur-apprentissage) qui dégrade la performance des algorithmes du "Machine Learning". En effet, lors de l'utilisation du SVM avec des noyaux polynomiaux de très haut degré dans le cadre de l'apprentissage d'un problème linéaire ce problème peut survenir. Le phénomène s'explique par le fait qu'un algorithme cherche le modèle qui exprime le mieux la relation entre les données, or parfois cet algorithme sur-apprend. Cela se traduit par de l'"Overfitting". En effet, il se peut que le modèle apprenne à partir des données mais aussi à partir de patterns (motif, structure, …) qui ne sont pas liés au problème et qui sont donc perçus comme du bruit. Une variance très élevée caractérise ce phénomène et cela se traduit par de mauvaises performances ou des taux de prédictions faussés. La régularisation des données est une technique qui permet de réduire l'erreur de type variance et donc ce phénomène. Ainsi, malgré les conclusions que nous venons de tirer, nous avons quand même réalisés et expliqués les tests dans la partie pratique.

- Point de vue pratique

Après avoir acquis ces notions théoriques, nous avons donc essayé de réaliser cet algorithme afin qu'il réponde à notre situation.

Nous avons donc tenté d'entrainer plusieurs types de SVM avec une procédure casi identique pour chaque test effectué.



En premier lieu, nous avons utilisé une "dataset" qui provenait de Kaggle et était composée de 10 222 images de chien de race et on y compte 120 races différentes. Ainsi, après avoir suivi le procédé ci-dessus et après avoir réalisé un entrainement très long suite à la diversité des images, les résultats n'étaient pas concluants puisque nous n'obtenions que 4% de précision quel que soit le noyau utilisé. Ces résultats confirment les propos tenus dans la partie théorique puisque la "dataset" n'est pas adaptée à cet algorithme.

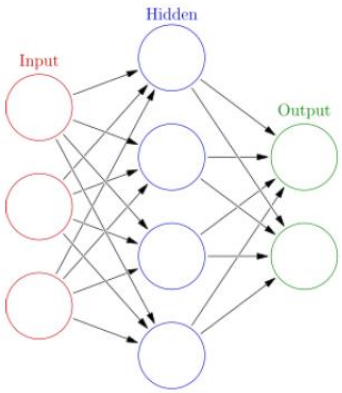
Dès lors, nous sommes repartis sur des bases simplifiées avec un base de données plus claire et un code basique. Ce dernier va permettre de prédire efficacement si nous sommes en présence d'un chien ou d'un chat. Pour se faire, notre base de données provenant aussi de "Kaggle" a été très simplifiée puisqu'elle ne contenait que 2 classes (des chiens et des chats) avec un nombre d'images égal à 8000. Ainsi après avoir effectué toute la procédure, nous avions une prédiction plus fiable de +-70%. C'est donc plus satisfaisant. Ainsi, le programme étant fonctionnel, nous avons apporté quelques modifications pour améliorer ces prédictions tels que l'ajout d'images dans la base de données, la modification de leur taille, ... Ce programme de base a pu ainsi être utilisé pour répondre à la prochaine étape de nos objectifs à savoir la détection des races de chiens.

Nous sommes d'abord partis sur la classification en 2 classes différentes représentant donc 2 races de chiens. Ainsi, nous avons modifié le code afin qu'il puisse prédire différentes races de chiens. Pour ce faire, il a juste fallu modifier la base de données en lui affectant des images de chiens d'uniquement deux races différentes. Le principe a donc été identique à la comparaison entre un chat et un chien. Le pourcentage de prédiction obtenu est cependant descendu d'une dizaine de pourcent dû à une base de données probablement trop petite pour savoir distinguer deux races différentes puisqu'elle contenait 50 images par classe. En effet, les caractéristiques entre deux chiens sont moins flagrantes qu'entre un chat et un chien. Par la suite, nous avons ajouté plus de classes dans la base de données et donc nous avons ajoutés 5-6 races de chiens avec toujours une cinquantaine d'images par classe. Ainsi, après l'ajout de ces classes, l'algorithme parvenait à prédire les 5-6 races de chiens avec une précision de 50%. Cela n'est donc pas encore satisfaisant mais la base du programme était correcte. De plus, il était intéressant d'ajouter que si une race n'est pas reconnue, l'algorithme nous dit que l'espèce est "inconnue" car il n'est pas facile de lui enseigner toutes les races de chiens existantes. Toutefois, après avoir effectué les derniers tests aboutissants à des résultats de 50%, nous avons pu constater, comme les algorithmes étaient réalisent en parallèle, que le CNN était plus performant. Nous nous sommes donc attardés uniquement à l'amélioration du CNN dans la suite du projet et on a donc directement implémenté cette sortie "inconnue" dans le code CNN.

3.2.2 Deep Learning – CNN

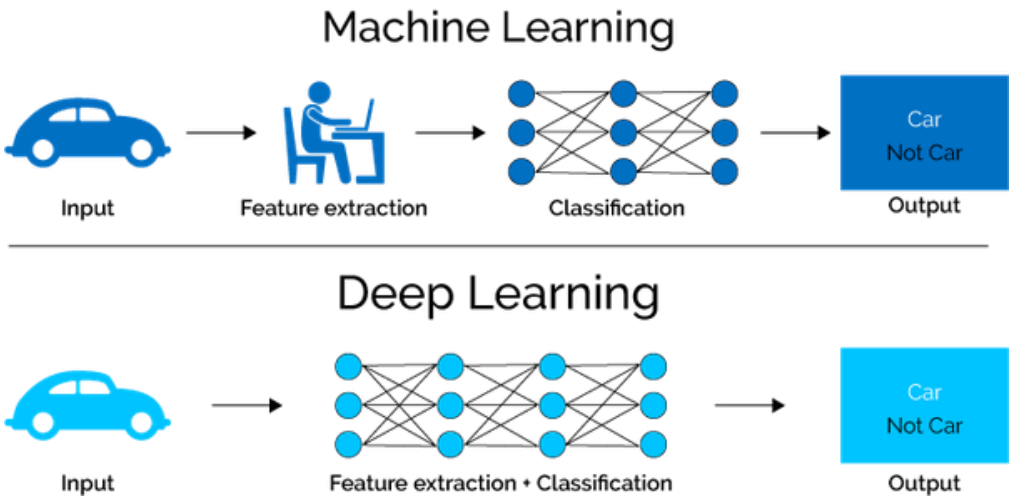
- Point de vue théorique

Durant le cours de Systèmes intelligents, nous avons entendu parler de réseau de neurones ou « Neural Network ». Ceux-ci font partie de la branche de l'intelligence artificielle appelée « Deep learning ». Ce dernier s'intéresse aux algorithmes inspirés de la structure et de la fonction du cerveau appelé réseaux de neurones artificiels. En effet, la structure se compose d'une couche d'entrée, une ou plusieurs couche(s) cachée(s) ainsi qu'une couche de sortie. Chaque couche est ainsi composée d'un certain nombre de neurones reliés entre eux de diverses manières. L'image ci-dessous représente un réseau avec des couches denses puisqu'elles sont toutes complètement connectées entre elles.



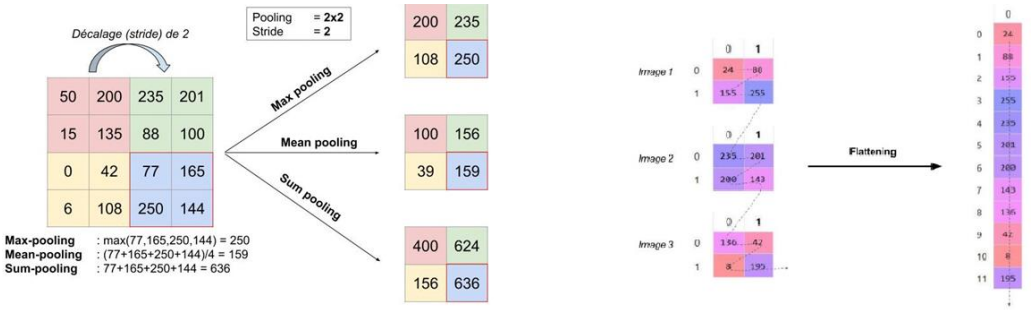
Structure d'un réseau de neurones

Ce type d'apprentissage permet de résoudre des problèmes, peut-être faciles pour les humains, mais difficiles pour les ordinateurs. De plus, grâce à un très grand nombre de paramètres qui s'ajustent automatiquement, ce modèle créera des liens implicites existant dans les données. Les réseaux de neurones sont très intéressants car la partie qui extrait les données fait partie intégrante de leur algorithme.



Différence entre Machine Learning et Deep Learning

Un type particulier de réseau de neurones a attiré notre attention : les "Convolutional Neural Network". Le nom "convolution" vient d'une opération mathématique : le produit de convolution. Les CNN possèdent des couches convolutives qui peuvent détecter des paternes dans les images grâce à des filtres. Pour ce faire, le CNN va convoluer les images qu'il reçoit en entrée avec les filtres. Les paramètres des filtres sont optimisés durant l'entrainement. Pour mieux accentuer les caractéristiques mises en évidence par une couche convolutive, on peut placer à sa sortie une couche "ReLU" qui supprime une partie des valeurs négatives. Il existe d'autres couches qui permettent d'extraire des informations des images, comme les couches de "pooling" qui permette de remplacer un carré de pixels par une valeur unique. Ce procédé permet donc de réduire une image de taille en la lissant. Une autre couche très utilisée est la couche de "flattering" qui permet de mettre bout à bout les pixels résiduels des couches précédentes.

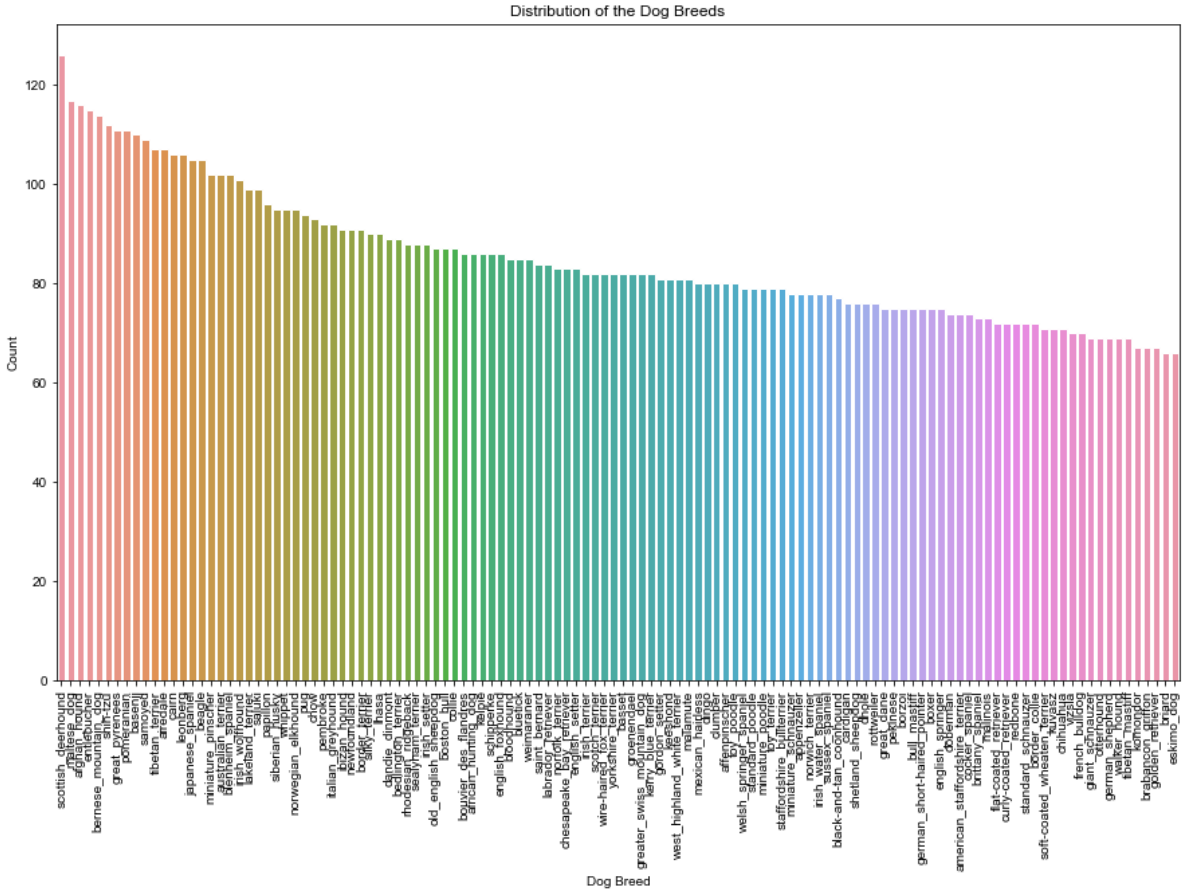


Exemple de pooling et de flattering

Ensuite à la fin du CNN, on trouve des couches "Denses". Celles-ci sont utilisées dans les ANN mais dans le cas des CNN, les neurones sont beaucoup moins nombreux.

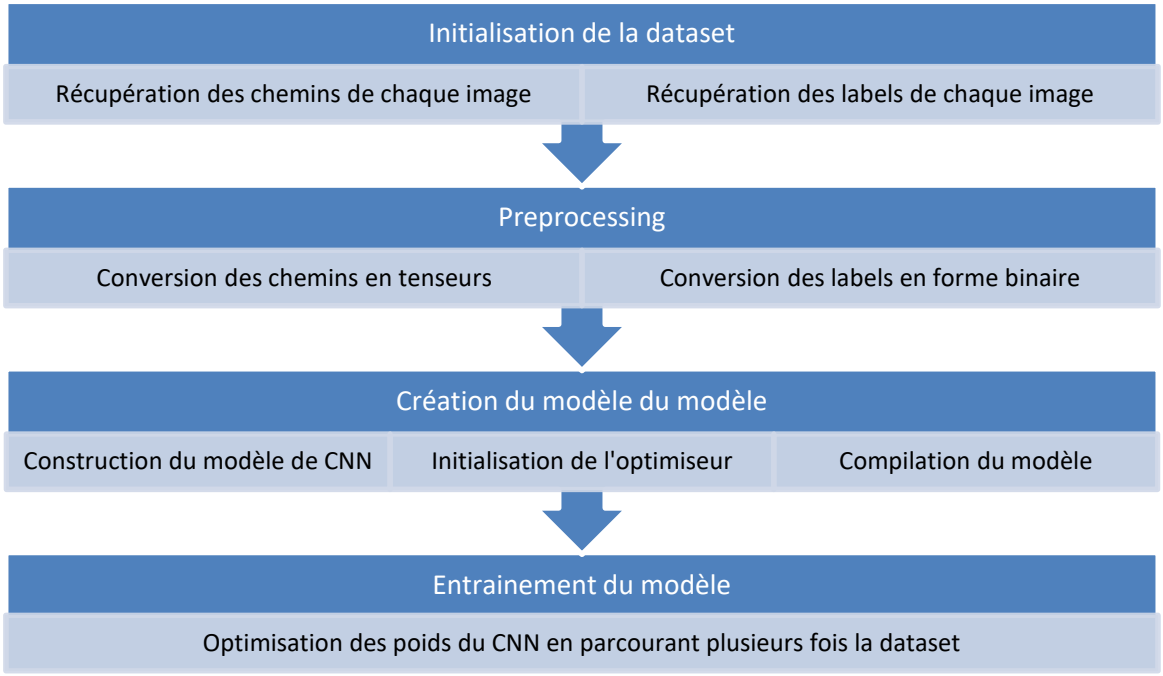
- Point de vue pratique

Les CNN nous intéressent fortement pour la même raison que pour la détection. En effet, le CNN peut analyser une image entière et ne se focalise pas sur la tête d'un chien par exemple. Nous avons donc tenté d'entrainer notre propre CNN en partant de rien. Pour ce faire nous avons utilisé une "dataset" provenant de "Kaggle" (identique au premier essai du SVM) ainsi qu'un article redirigeant vers un "Github". La ""dataset est composée de 10 222 images de chiens de race et on y compte 120 races différentes. Un fichier CSV était disponible ce qui nous a permis d'analyser cette "dataset" et de ressortir le graphique ci-dessous représentant la distribution des images de races de chiens.



Distribution des images de la dataset Kaggle en fonction des races

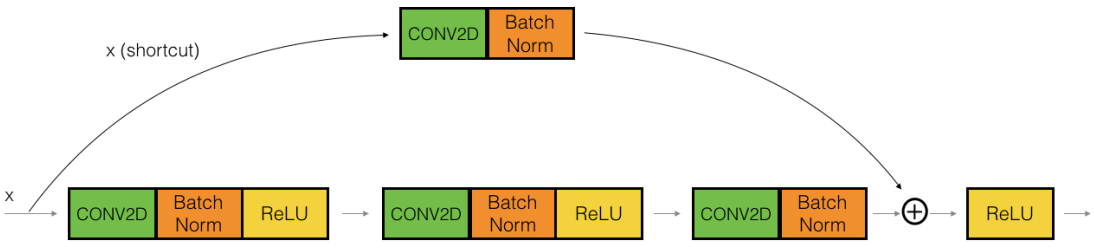
Nous avons procédé comme suit :



Cependant, les résultats n'étaient pas concluants, nous n'avons pas atteint 5% de précision. En effet, les CNN ont besoin d'une grande base de données ce qui augmente la mémoire nécessaire pour faire tourner le programme ainsi que le temps d'entrainement. Il était donc compliqué de continuer sur cette voie car d'une part nous n'avions pas accès à une "dataset" plus conséquente et d'autre part l'entrainement doit prendre le moins de ressources possible car le modèle va être entrainé sur un Raspberry Pi. Nous avons donc continué nos recherches avec deux idées en tête :

1. Avoir une petite "dataset" pour limiter le temps d'entrainement ;
2. Et avoir une précision assez élevée.

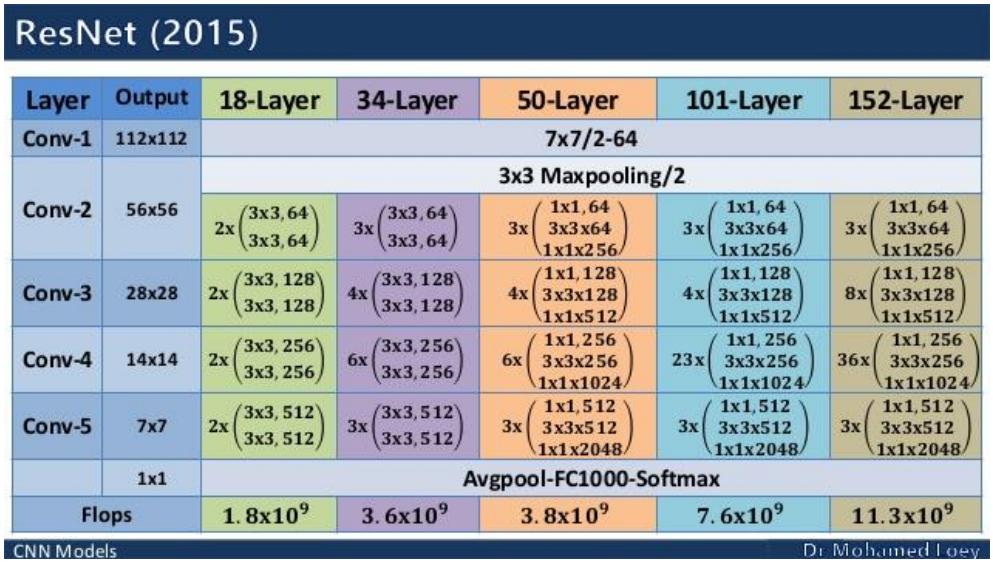
Nous sommes donc tombés sur un article parlant du « Tansfer learning ». Cette méthode d'apprentissage nous permettait de réaliser exactement ce que nous voulions. Mais pour réaliser cela, il nous fallait un CNN déjà pré-entrainé. Heureusement, le ResNet50 que nous avions utilisé pour la détection pouvait nous servir à réaliser cette tâche également.



FIllustration du shortcut

Le ResNet50 est type de CNN qui a gagné le ILSVRC 2015. Un modèle pré-entrainé sur base de la base de données ImageNet peut être téléchargé et est souvent la base pour le

« Transfer Learning ». Son efficacité est basée sur les raccourcis ou « shortcut ». Comme on peut le voir sur l'image ci-dessus, on additionne l'entrée X avec la X(shortcut). Cette technique assure que la couche actuelle soit aussi efficace que la couche précédente et même la couche suivante. Mais il peut arriver que le X(shortcut) n'ait pas la même forme que le X, ce qui poserait un problème. On applique donc sur le raccourci la couche convolutive adéquate pour que les deux X aient la même forme.



ResNet (2015)

Layer	Output	18-Layer	34-Layer	50-Layer	101-Layer	152-Layer
Conv-1	112x112	7x7/2-64				
Conv-2	56x56	3x3 Maxpooling/2				
		$2\times\begin{pmatrix}3\times3,64\\3\times3,64\end{pmatrix}$	$3\times\begin{pmatrix}3\times3,64\\3\times3,64\end{pmatrix}$	$3\times\begin{pmatrix}1\times1,64\\3\times3\times64\\1\times1\times256\end{pmatrix}$	$3\times\begin{pmatrix}1\times1,64\\3\times3\times64\\1\times1\times256\end{pmatrix}$	$3\times\begin{pmatrix}1\times1,64\\3\times3\times64\\1\times1\times256\end{pmatrix}$
Conv-3	28x28	$2\times\begin{pmatrix}3\times3,128\\3\times3,128\end{pmatrix}$	$4\times\begin{pmatrix}3\times3,128\\3\times3,128\end{pmatrix}$	$4\times\begin{pmatrix}1\times1,128\\3\times3\times128\\1\times1\times512\end{pmatrix}$	$4\times\begin{pmatrix}1\times1,128\\3\times3\times128\\1\times1\times512\end{pmatrix}$	$8\times\begin{pmatrix}1\times1,128\\3\times3\times128\\1\times1\times512\end{pmatrix}$
Conv-4	14x14	$2\times\begin{pmatrix}3\times3,256\\3\times3,256\end{pmatrix}$	$6\times\begin{pmatrix}3\times3,256\\3\times3,256\end{pmatrix}$	$6\times\begin{pmatrix}1\times1,256\\3\times3\times256\\1\times1\times1024\end{pmatrix}$	$23\times\begin{pmatrix}1\times1,256\\3\times3\times256\\1\times1\times1024\end{pmatrix}$	$36\times\begin{pmatrix}1\times1,256\\3\times3\times256\\1\times1\times1024\end{pmatrix}$
Conv-5	7x7	$2\times\begin{pmatrix}3\times3,512\\3\times3,512\end{pmatrix}$	$3\times\begin{pmatrix}3\times3,512\\3\times3,512\end{pmatrix}$	$3\times\begin{pmatrix}1\times1,512\\3\times3\times512\\1\times1\times2048\end{pmatrix}$	$3\times\begin{pmatrix}1\times1,512\\3\times3\times512\\1\times1\times2048\end{pmatrix}$	$3\times\begin{pmatrix}1\times1,512\\3\times3\times512\\1\times1\times2048\end{pmatrix}$
	1x1	Avgpool-FC1000-Softmax				
Flops		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

CNN Models — Dr Mohamed Loey

Liste des différentes couches des ResNet

Le « Transfer Learning » consiste à utiliser les couches pré-entrainées d'un CNN existant et y ajouter des couches propres à notre application. Nous avons téléchargé depuis "Github" le ResNet50 ne contenant pas les dernières couches de neurones. En effet, celui-ci s'arrête à la « Bottleneck layer » qui est une couche comportant moins de neurones que la couche d'avant et moins de neurones que la couche d'après d'où son nom (« bottleneck » en anglais signifie « goulot »).

Nous avons utilisé cette partie du ResNet50 pour extraire des informations de l'image que l'on insère en entrée. Ces informations, qui sont l'"output" du Resnet50 incomplet, serviront d'"input" aux couches propres à notre application. Les énormes avantages de cette méthode sont :

1. Nous n'avons pas besoin d'entrainer tout le modèle mais seulement les dernières couches ce qui diminue fortement le temps d'entrainement ;
2. Nous ne perdons pas de précision à cause du faible nombre de couches en sortie car le ResNet50 est assez puissant pour fournir des informations de qualité à nos couches. La précision de notre modèle atteint 76%, ce qui valide ce qui vient d'être dit.

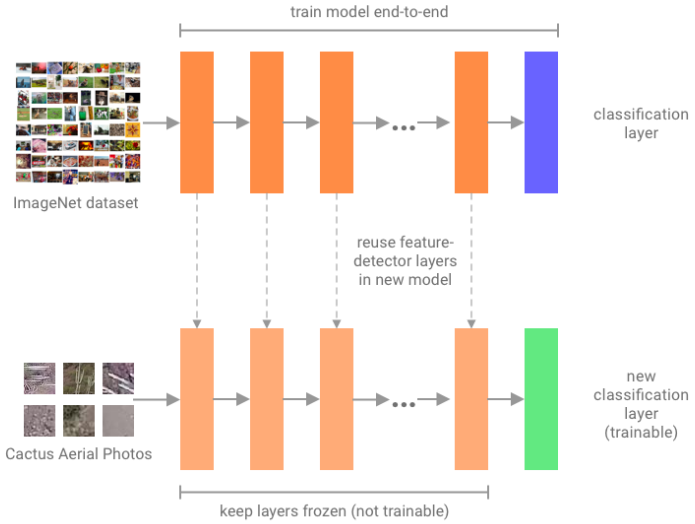
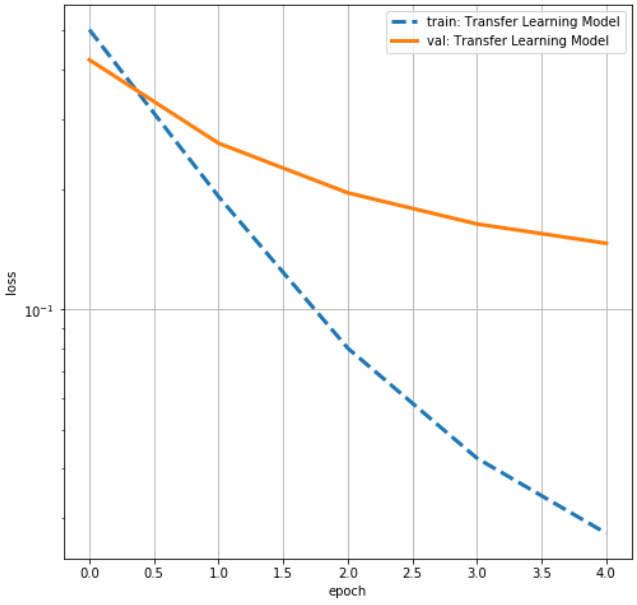


Illustration du principe de Transfer Learning

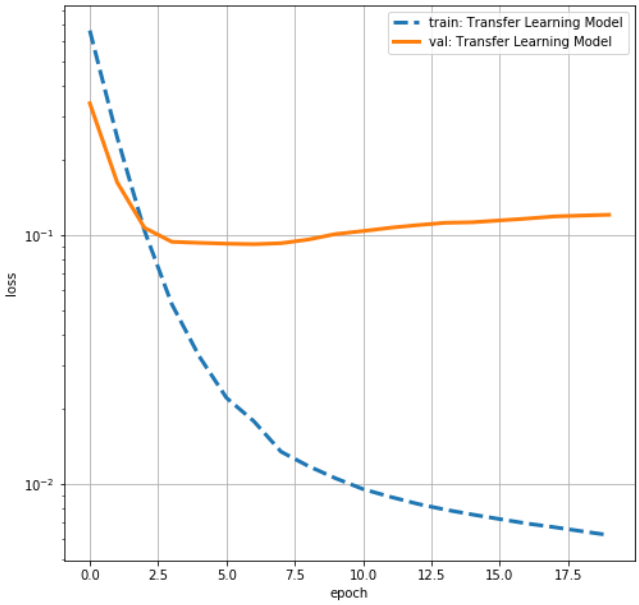
Nous avons ensuite réalisé un code qui répond au mieux à notre application. En effet, nous devons reconnaitre une race, voire deux, et non 120 comme précédemment. Le but ici est de détecter le chien de Florian qui est un Cavalier King Charles et d'indiquer « inconnu » si c'est un chien d'une autre race. Nous avons donc réalisé un petite "dataset" comportant 50 images de Cavalier et 50 images de chiens d'autres races.

Nous avons atteint une précision de 100% sur ce modèle ce qui est parfait. Nous pensons que l'augmentation de précision entre ce modèle et le précédent est due à la réduction du nombre de sortie. En effet, le CNN ne doit plus décider entre 120 races mais entre « Cavalier » ou « Inconnu ». De plus, notre "dataset" est assez variée ce qui renforce l'apprentissage (Il n'est pas question ici d'"overfitting"). En revanche, les CNN demandent beaucoup de ressources ce qui augmente le temps de traitement. Cependant nous avons prôné la précision par rapport à la rapidité du système.



Graphique montrant l'évolution des pertes en fonction des epochs

Le graphe ci-dessus représente les pertes tout au long de l'entrainement et de la validation. On peut remarquer que les pertes au niveau du training ne font que diminuer. On pourrait se dire qu'il faudrait alors continuer indéfiniment pour avoir une perte infiniment petite. Or quand on prête attention à la courbe de validation, on remarque que celle-ci diminue mais avec une pente beaucoup plus faible. Il semblerait même qu'elle s'aplatisse. Or si c'était le cas ça voudrait dire que notre modèle commencerait à « overfitter » ce qui dégraderait les performances de notre modèle. Même si cela avait été le cas, nous ne sauvegardons que le meilleur modèle.



Graphique illustrant l'Overfitting

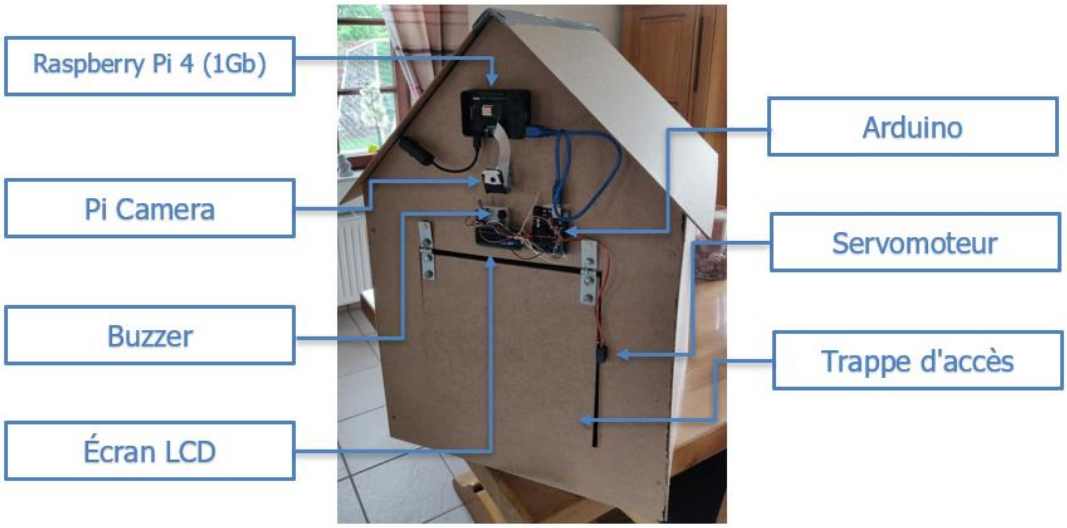
4 Réalisation

Après avoir testé et choisi notre algorithme, nous avons dû construire une maquette en vue de réaliser une vidéo.

4.1 La maquette

Ainsi, le prototype de notre projet contient une partie "mécanique" avec l'assemblage de la maquette. Les composants, même électriques, sont détaillés brièvement sous le schéma suivi des plans électriques avec le câblage des actionneurs.

- **Plan mécanique :**



Maquette de la trappe intelligente pour chiens

- Le raspberry Pi 4 (1Gb) :

Kit Raspberry Pi 4

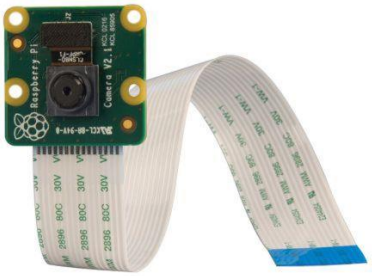
Comme nous l'avons énoncé auparavant, le Raspberry va permettre de faire fonctionner l'algorithme créé pour l'intelligence artificielle. Ainsi, après avoir analysé les données reçues de la caméra, il traitera les informations et enverra celles-ci par le port série à l'Arduino.

- Arduino :

Arduino (Marque : Elegoo R3)

Après avoir récupéré les différents messages envoyés du Raspberry, l'Arduino permettra de faire fonctionner les actionneurs selon l'animal présent devant la trappe.

- Pi Camera :



Pi Camera V2

Il est évident que son but est de filmer les éléments qui se présentent devant la trappe intelligente pour que le Raspberry puisse analyser l'image et agir en conséquence. La caméra détectera soit le chien de Florian, soit un chien de race différente, soit un chat, soit rien du tout.

Chien de Florian (Cavalier King Charles) / Chien d'une autre race / Chat

- Buzzer :

Buzzer

Si la caméra détecte un chat ou un chien de race inconnue, le Raspberry analysera cette information et enverra un message à l'Arduino qui actionnera le buzzer.

- Écran LCD :

Afficheur LCD

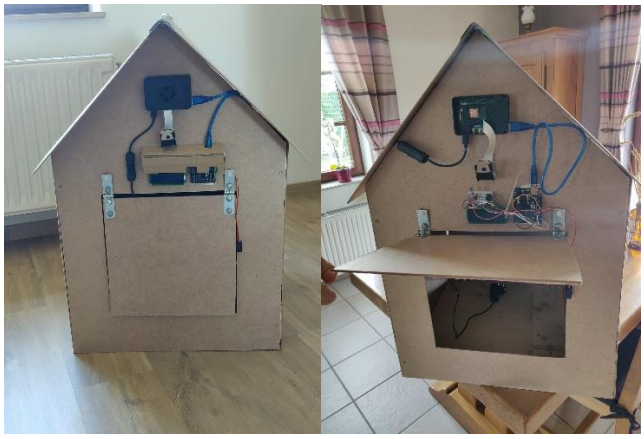
Cet écran permet d'afficher les informations que le Raspberry a détecté. Ainsi, si le chien détecté est un Cavalier King Charles, l'Arduino indiquera un message d'accueil alors qu'il signalera un intrus si tel est le cas devant la trappe.

- Servomoteur :

Servomoteur SG90

Le servomoteur remplace le verrou prévu initialement pour éviter de commander des éléments pendant cette période. Toutefois, son rôle reste identique puisqu'il maintient la porte fermée tant que le Raspberry ne détecte pas le chien de Florian.

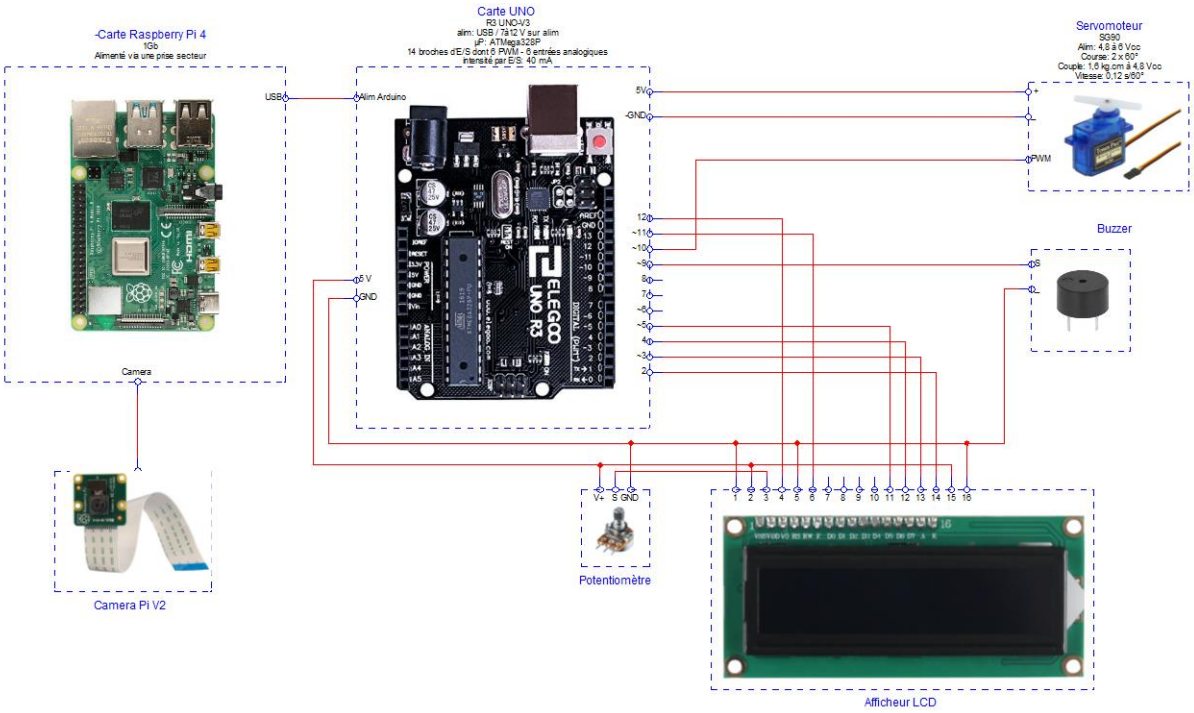
- Trappe d'accès :



Trappe d'accès fermée et ouverte

Elle permet simplement de laisser rentrer le chien lorsque celui-ci est autorisé à passer.

- **Plan électrique :**



Plans électriques

Suite aux composants présents sur cette maquette et en rassemblant tous les éléments qui ont été nécessaires à la réalisation de celle-ci, nous avons pu établir la liste des composants qui nous ont été utiles.

List of materials

	Name	Unit Price (Tax incl.)	Quantity	Price (Tax incl.)	Link	Comment
Components	Servomotor	4,95 €	2	9,90 €	Gotronic	Provided by the school
	Relay	0,90 €	2	1,80 €	Gotronic	Provided by the school
	Diode	0,08 €	4	0,32 €	Gotronic	Provided by the school
	Raspberry Pi 4 - 1 Gb	44,90 €	1	44,90 €	Gotronic	Provided by the school
	Arduino Uno	13,95 €	1	13,95 €	Gotronic	Provided by the school
	Caméra for Raspberry Pi	36,50 €	1	36,50 €	Gotronic	Provided by the school
	Buzzer	1,20 €	1	1,20 €	Gotronic	Provided by the school
	LCD display 2x16	9,90 €	1	9,90 €	Gotronic	Provided by the school
		0,00 €		0,00 €		/
		0,00 €		0,00 €		/
		0,00 €		0,00 €		/
Mechanical	Wood panel	23,99 €	1	23,99 €	Hubo	/
	Camera support	0,00 €		0,00 €		3D printed
		0,00 €		0,00 €		/
			Total =	142,46 €		
			Totalt (with kits, items provided by the school deducted) =	23,99 €		

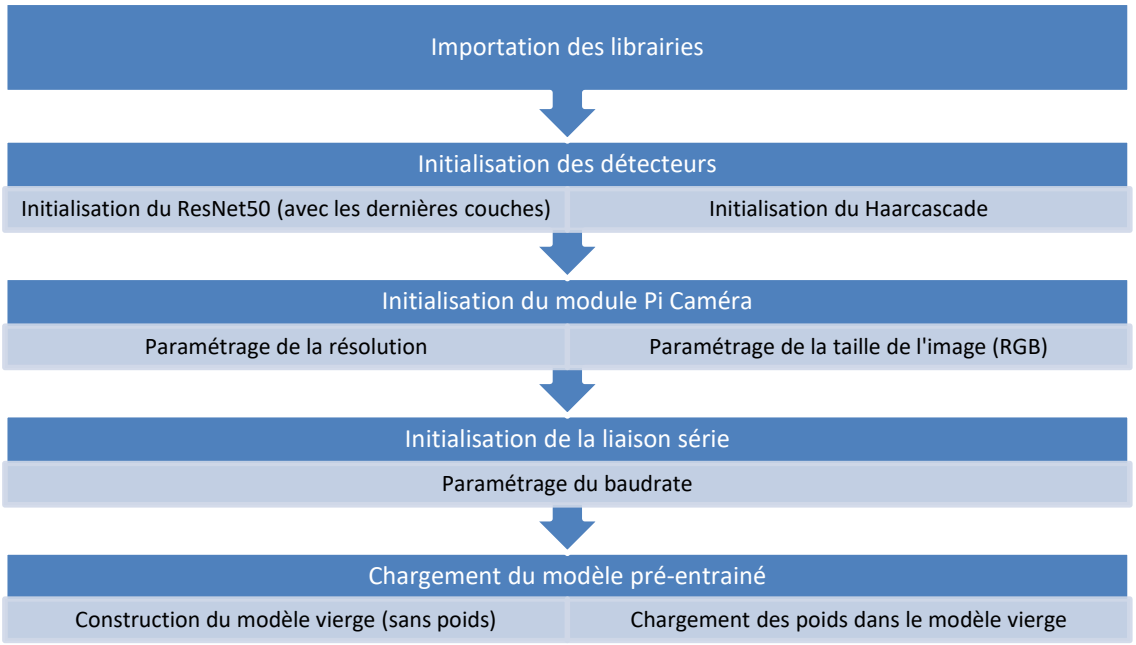
Liste du matériel

Nous pouvons voir que ce prototype atteint un prix de +- 150€ mais que la plupart des éléments qui ont été utilisés nous ont été fournis par l'école. Ainsi, seule la structure en bois a dû être achetée.

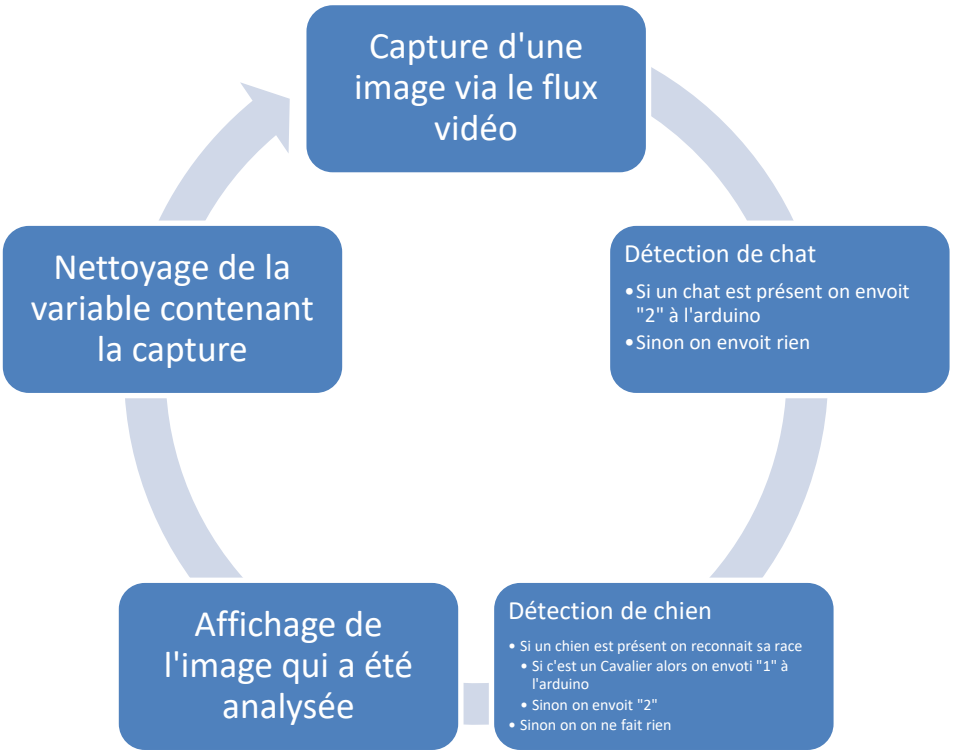
4.1.1 La caméra de reconnaissance

La caméra de reconnaissance est dotée d'un algorithme qui permet de détecter les chats, les chiens et de reconnaitre au moins une race de chien. Pour réaliser cela, nous nous sommes basés sur les codes précédemment réalisés. On peut dès lors diviser le programme en deux parties : l'initialisation et l'analyse.

L'initialisation consiste à (télé)charger les différents fichiers dont l'analyse à besoin ainsi que d'initialiser les variables.



Quand le programme a bien été initialisé, on entre dans une boucle d'analyse. Celle-ci consiste à prendre une capture du flux vidéo de la Pi Camera et de vérifier s'il y a un chat ou un chien. Dans le deuxième cas, on réalise une analyse supplémentaire pour vérifier s'il s'agit d'un Cavalier King Charles ou d'un chien dont la race est inconnue.



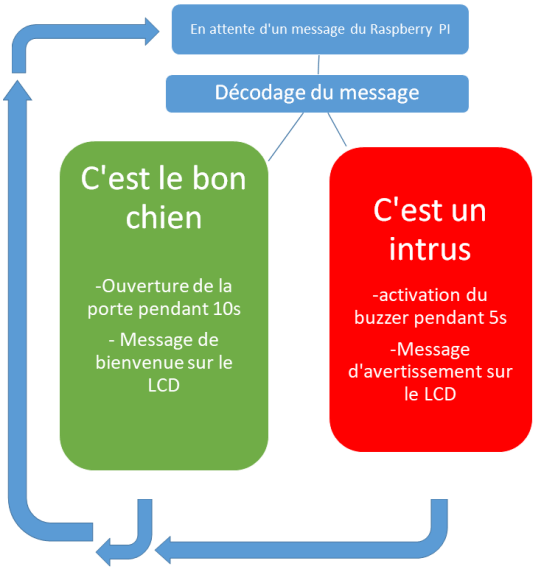
4.1.2 La porte intelligente

Les différents actionneurs s'articulent tous autour d'un Arduino UNO. Celui dispose d'un buzzer, d'un servomoteur et d'un écran LCD en plus d'assurer la communication avec le Raspberry PI.

Le buzzer sert à repousser les individus indésirables devant la porte avec un son aigu de 3khz grâce au signal PWM que l'Arduino lui envoie.

Le servomoteur quant à lui est utilisé pour débloquer la porte, celui-ci tourne d'un angle plus ou moins grand en fonction de la longueur d'impulsion envoyée. Initialement, il était prévu d'utiliser deux solénoïdes lock, mais le kit Arduino n'en disposait pas.

L'écran LCD reçoit des informations venant de l'Arduino, il n'est pas essentiel au fonctionnement du projet, mais permet, lors de présentation, de donner une touche explicative aux personnes visualisant le projet afin de comprendre ce qu'il se passe.



L'Arduino en lui-même ne sait pas ce qui se trouve devant la caméra du Raspberry PI, une communication entre les deux entités a dû être créée. Cette communication utilise un câble USB, qui est donc une communication série. Les données sont converties et découpées en entier de 8 bits sur le Raspberry PI avant d'être envoyées sur l'Arduino via le câble USB pour être recomposées par ce dernier afin d'utiliser les différents actionneurs.

4.2 Les tests

Après avoir réalisé la programmation du Raspberry et de l'Arduino ainsi qu'avoir construit la maquette, il ne restait plus qu'à effectuer les tests. Concernant ceux-ci, ils se sont déroulés en trois phases.

Dans un premier temps, afin de vérifier la fonctionnalité des algorithmes du Raspberry, nous vérifions la prédiction qu'il nous sortait en lui indiquant directement le chemin d'une image présente dans un répertoire précis. Ainsi, les tests étaient très concluants puisque nous atteignons bien une prédiction de 100%.

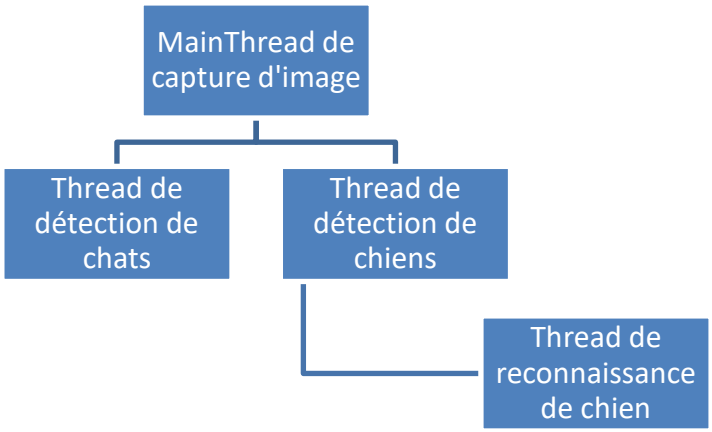
Par la suite, nous avons implémenté la caméra dans le code et nous avons pu tester l'entièreté du projet. Dès lors, la détection et reconnaissance d'un chien devait se faire via l'image renvoyée par la caméra. Pour commencer, nous affichions des images imprimées et fixes devant la caméra pour s'assurer que l'image capturée par la caméra était bonne et que l'algorithme puisse s'effectuer normalement. Les résultats obtenus étaient à nouveau concluants.

Pour finir, nous sommes passés au cas réel en positionnant la maquette dans un pièce quelconque. La caméra filmait donc continuellement la pièce et indiquait bien qu'aucun élément n'était présent devant la caméra. Ensuite, lorsque le chat d'Arnaud s'est présenté devant la porte, la détection s'est correctement effectuée et le signal sonore a retenti. La détection d'un chien s'est également effectuée sans encombre puisque l'algorithme affichait directement qu'un chien était présent. En revanche, il indiquait parfois que le chien était inconnu car l'image captée par la caméra n'était pas très claire. Le chien étant très mobile, il n'était pas toujours évident d'obtenir une bonne capture de l'image. Toutefois, lorsque le chien restait immobile, l'image était prise sans problème et la reconnaissance se réalisait avec succès permettant le déverrouillage de la porte. Nous pouvons ainsi dire que le projet est pleinement fonctionnel malgré quelques petits accros lorsque le chien bouge de trop. Un lien vers une vidéo de présentation de ce projet se trouve sur "Github" dans le fichier résumant le projet.

5 Améliorations

Plusieurs améliorations au sein du projet sont possibles :

- Pour augmenter la vitesse de traitement, un Raspberry disposant de plus de RAM, comme la version à 4Gb peut être une solution ou encore utiliser directement un ordinateur, comme tous les actionneurs sont disponibles via un port USB, l'ordinateur peut directement communiquer via l'Arduino, mais cela augmenterait grandement le coût du projet.
- Toujours pour augmenter la vitesse de traitement, nous aurions pu utiliser le multithreading pour réaliser plusieurs tâches simultanément. Par exemple, les différentes analyses d'images auraient pu se faire simultanément ce qui aurait augmenté le dynamisme de la reconnaissance.



- Pour des raisons d'esthétique, les câbles pourraient être cachés à l'intérieur de la maquette ou dans des goulottes prévues à cet effet.
- De plus, pour une meilleure visibilité lors de présentation en public, l'ajout de led de couleurs, vert et rouge, pourrait être une possibilité : la led verte pour signaler l'ouverture de la porte et une led rouge pour signaler un essai d'intrusion.
- La fermeture de la porte après ouverture se fait actuellement via une temporisation, mais en utilisant un capteur, la fermeture pourrait se faire après l'entrée du chien.
- On peut aussi, pour économiser de l'énergie, utiliser un capteur de mouvement: l'Arduino et le Raspberry PI pourraient être en mode « sleep » et via une impulsion du capteur vers l'Arduino réveiller les différents éléments.
- Il est possible d'augmenter les possibilités en ajoutant de nouvelles races à reconnaître en plus du Cavalier King Charles en réitérant les opérations faites pour programmer celui-ci.
- La reconnaissance d'une deuxième race pourrait aussi se faire directement via le Raspberry. En effet, grâce au "Transfer learning", la "dataset" n'a pas besoin d'être grande ce qui diminue la mémoire à utiliser. On pourrait imaginer un algorithme qui prend des photos du chien lorsqu'on appuie sur un bouton ou si on envoie un signal au Rapsberry. Ces photos seraient stockées dans un dossier portant le nom de la race du chien et les chemins de ces images ainsi que leur label (la race du chien) seraient stockés dans un fichier. Celui-ci regrouperait tous les chemins et les labels de toutes les races de chiens de la maison. Il

suffirait d'initialiser la "dataset" à partir de ce fichier et de réaliser le training avec les nouvelles images.

- D'autres améliorations sont encore possibles, comme l'envoi de message à un broker MQTT pour recevoir une information sur tous les appareils connectés lorsque le chien entre et sort ou si un intrus a essayé d'entrer avec une photo de celui-ci, tout cela est possible si le Raspberry PI est connecté au réseau domestique. Ces données pourraient être utilisées pour avoir des statistiques sur les habitudes des chiens à entrer et sortir de la maison.

6 Conclusion

En conclusion, il nous a été demandé de réaliser une trappe intelligente pour chiens. C'est-à-dire que celle-ci devait détecter les chiens, reconnaitre au moins une race et détecter les chats via une caméra. Comme dit précédemment dans le rapport, nous avons réussi à la réaliser. Nous sommes parvenus à avoir un taux de précision de 100% sur une petite "dataset" grâce au "Transfer Learning". La trappe fonctionne mais elle pourrait être améliorée comme expliqué dans le point 5 de ce rapport. Outre la réalisation, nous avons trouvé ce projet très intéressant car cela nous à permis de se familiariser avec plusieurs choses :

1. Les Raspberry Pi : qui sont des ordinateurs mono-carte plutôt puissants pour leur prix et leur encombrement. Ils peuvent donc être utilisés dans une multitude d'applications ;
2. La programmation en Python : Ce langage de programmation est de plus en plus utilisé car il est ultra polyvalent et s'apprend vite. Il est donc très intéressant de l'apprendre durant notre cursus.
3. Le Deep Learning : On voit de plus en plus d'IA de nos jours et ce fut très intéressant de se plonger le temps d'un projet dans cet univers. Grâce à cela, nous avons de bonnes bases pour entrer dans l'industrie 4.0
4. Github : cette plateforme est extrêmement connue des développeurs. Et c'est compréhensible car au fur et à mesure du projet nous avons compris son utilité.

7 Bibliographie

- GitHub - kairess/dog_face_detector: Detect dog face rect and facial landmarks(6 points) using dlib
- A Dog Detector and Breed Classifier - Towards Data Science
- dog_breed_classifier/dog_app.ipynb at master · HenryDashwood/dog_breed_classifier · GitHub
- Dog Breed Identification us VGG + SVM | Kaggle
- Building powerful image classification models using very little data
- https://github.com/AlonsoFlo14/SI_Group4_DogFlap
- http://www.willberger.org/cascade-haar-explained/
- https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

8 Annexes

- "Gantt"
- La "dataset" utilisée pour notre projet et récupérable via Dog Breed Identification us VGG + SVM | Kaggle
- "Github" contenant :
  - L'ensemble des codes réalisés de la première à la dernière version
  - Un lien vers la vidéo de présentation (https://henallux-my.sharepoint.com/👌/g/personal/etu30602_henallux_be/ERCnO1R38XRGn95g62237gsBmnrNxXemxKM-laoPfos4Xg?e=497sv5)
  - Un "Readme" permettant une prise en main facile du projet
  - Un fichier expliquant les étapes effectuées lors du projet

Gantt diagram

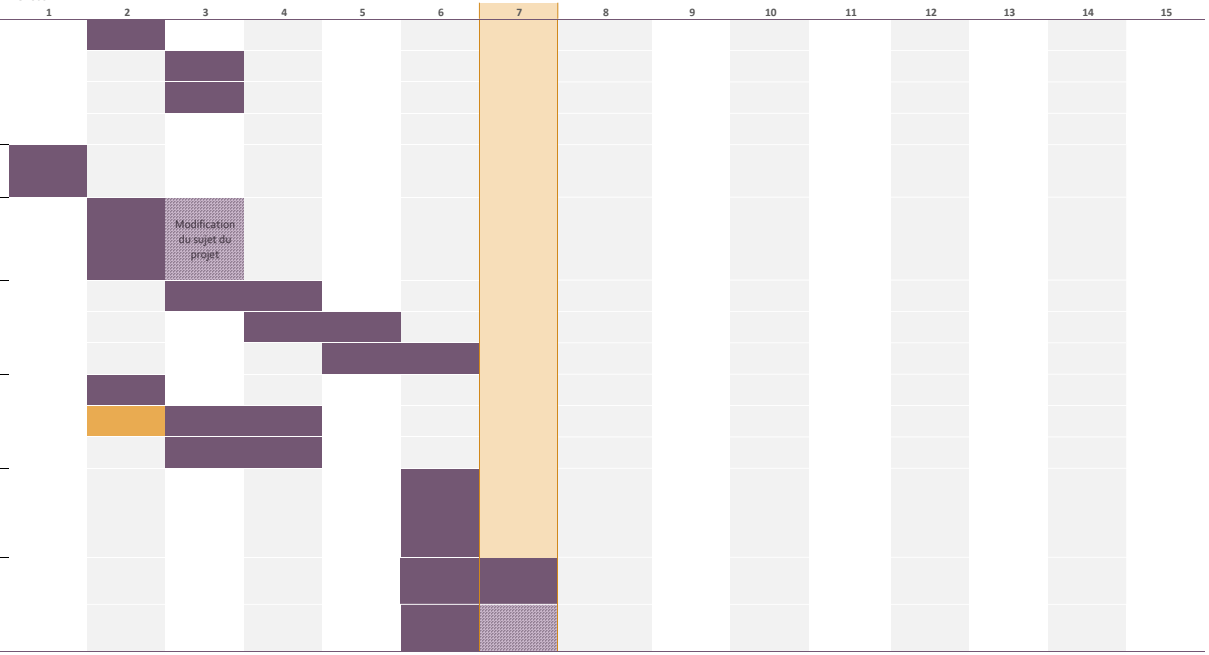
Project planner

Select a period to highlight on the right. A legend describing the graph follows.

Period to highlight: 7

Duration | Real start | % completed | Real (beyond the plan) | % completed (beyond the plan)

Parts	Activities	Concerned persons	Start of the plan	Duration of the plan	Real start	Real duration	% completed
Arduino	Programming of the lock	Christophe	2	1	2	1	100%
	Programming of the LCD screen	Christophe	3	1	3	1	100%
	Programming of the speaker	Christophe	3	1	3	1	100%
	Programming of the date and time feature	Christophe					0%
Dataset	Dataset	Florian & Arnaud	1	1	1	1	100%
Haar-Feature Classifiers	Cat's face detection	Florian & Arnaud	2	2	2	2	95%
CNN	Detection of the dogs	Arnaud	3	2	3	2	100%
	Breed recognition using CNN (from scratch)	Arnaud	4	2	4	2	100%
	Breed recognition using Transfert Learning	Arnaud	5	2	5	2	100%
SVM	Detection between dogs and cats	Florian	2	1	2	1	100%
	Detection between dogs and other animals	Florian	3	2	2	3	100%
	Breed recognition	Florian	3	2	3	2	100%
Implementation on Raspberry	Choice of algorithm ("assembly" of different codes (SVM + CNN))	Florian & Arnaud	6	1	6	1	100%
Administrative	Report	All	6	2	6	2	100%
	Presentation	All	6	2	6	2	90%



1	from	24-03-20	to	31-03-20	
2	from	31-03-20	to	07-04-20	
3	from	07-04-20	to	14-04-20	
4	from	14-04-20	to	21-04-20	
5	from	21-04-20	to	28-04-20	
6	from	28-04-20	to	05-05-20	
7	from	05-05-20	to	12-05-20	