UNIVERSITÉ DE
SHERBROOKE

Faculté des sciences

Rapport final du projet de session IGL591

Auteur	*Matricule*
RAMI ABDELlAH	19 143 062

Sous la direction de :
Djemel Ziou

28 avril 2020

Table des matières

1 Introduction

Dans le cadre du projet de session du cours IGL591, j'ai eu pour mission de travailler sur un robot à 4 roues dans l'objectif d'implémenter dessus un programme lui permettant d'avancer automatiquement dans un champ d'arbres de sapin, et ceci en détectant la bonne trajectoire à suivre et en évitant de toucher les arbres.

Suite à des problèmes lié au fonctionnement du robot, ainsi qu'à la complexité de la mission, le but du projet s'est transformé à la recherche d'un modèle qui sur entrée d'une image d'un champ d'arbres de sapin, il distingue dans cette images ces arbres pour que le robot puisse les éviter.

Ce modèle sera utilisé par la suite par d'autres personne travaillant sur le même projet pour dessiner le bon chemin à suivre par le robot sur entrée des images en temps réel prise par la caméra du robot.

Ainsi, ce projet s'inscrit parfaitement dans le domaine de ma formation, car c'est un problème qui pourra être traité par les techniques de l'intelligence artificielle.

2 Description du problème

Le problème qu'on va essayer de résoudre est que, étant donné une image qui contient des arbres est un chemin à travers ces arbres, il nous faut savoir distinguer les pixels qui appartiennent à l'un des arbres présents dans l'images, des pixels qui ne le sont pas, qu'on va appelé des pixels d'arrière plan.

Ce type de problème s'appelle la détection d'objets dans les images. Il existe plusieurs et techniques permettant de résoudre cette problématique avec plus ou moins de précision selon la complexité de la tâche.

Après des recherche sur l'état de l'art de ces techniques, j'ai décidé de tester dans mon projets deux techniques. La première s'appelle **le modèle génératif**, qui est une technique utilisant les probabilités pour déterminer l'appartenance d'un objet dans une image à une classe donnée. Pour étudier cette technique je me suis basée deux articles [1],[2] qui utilise des méthode similaires à ce modèle.

Pour la deuxième méthode j'ai choisis un modèle bien plus complexe, il s'agit d'un réseau de neurones profond de type **CNN** spécialisé dans le traitement des images en se basant sur l'article [3].

3 Démarche scientifique

Dans ce chapitre je vais essayer de donner une idée sur le méthode de travail de chacune des deux méthodes avec lesquelles je vais travailler :

3.1 le modèle génératif

Le modèle génératif est un modèle qui utilise les probabilités pour déterminer les classes auquel appartient une donnée [4], ici un pixel, donnée en se basant sur la couleur de ce pixel. C'est à dire, que le modèle est une fonction qui prend en entrée la valeur de la couleur d'un pixel est donne en sortie les probabilités de l'appartenance de cette valeur de couleur à chacune des classes étudiés.
La valeur de la couleur est prise comme un trip-let $x = [x_1, x_2, x_3]$ ou chacun des xi correspond à un champ de couleur dans le format **RGB**, ainsi que chacun des xi est compris entre 0 est 255.
Le probabilité calculé par la fonction du modèle pour une donnée $[x_1, x_2, x_3]$ est comme suit, pour chacune des classes y_i de notre problème on calcule :

$$P(X = [x_1, x_2, x_3]|Y = y_i)$$

Ce terme représente donc la probabilité d'avoir la donnée $x = [x_1, x_2, x_3]$ sachant que la classe de cette donnée est y_i.
Cette probabilité égale à :

$$P(X = x|Y = y_i) = \frac{P(X = x, Y = y_i)}{P(Y = y_i)}$$

Où :

$$P(Y = y_i) = \frac{Card(y_i)}{Card(\Omega)}$$

Ce qui signifie que la probabilité d'une classe est le nombre de valeurs x appartenant à cette classe sur le nombre totale de valeurs dans l'image. Puis on a :

$$P(X = x, Y = y_i) = \frac{Card(x, y_i)}{Card(\Omega)}$$

Ce qui signifie que la probabilité d'avoir à la fois cette donnée et cette classe est égale au nombre de valeurs dans l'image qui sont égale à x est qui appartiennent à y_i divisé par le nombre total de valeurs dans l'image.
Après le calcule de ces probabilité, on prédit que la classes auquel appartient le pixel en entrée est celle qui a la plus grande valeur de probabilité.

3.2 CNN

Le CNN (Convolutional neural network) est un réseau de neurones, généralement de type profond, qui est spécialisé dans les problèmes de traitement d'images [5].
Les réseau CNN utilisent des filtres, qui sont des matrices contenant plusieurs nombres réels, qui lorsque appliqué à une image en entrée permettent d'identifier différentes structures. Le contenu de ces filtres est appris automatiquement par le réseau de neurones pour détecter les structures qui vont nous servir dans notre problème pour détecter l'appartenance de chaque

région dans l'image à une des classe du problème.
La sortie de ce type de modèle est appelé un masque. En général le masque est une image de même taille que l'image d'entrée mais à la place de chaque pixels on trouve une couleur qui correspond à sa classe. Ainsi, dans un problème à deux classes, le masque de sortie sera une image ne contenant que deux couleurs. Le processus de classifier chacun des pixels de l'image est appelé la segmentation.
Il existe plusieurs structures différentes de réseaux de type CNN. La structure est souvent choisie en rapport à la difficulté de la tâche à accomplir, où on va choisir des réseaux bien complexes et profond pour les tâches les plus dures.
La structure que j'ai choisi pour mon projet est le **"UNet"**. Ce type de structure est souvent utilisé pour des tâches nécessitant un segmentation précise et rapide. Donc c'est le bon modèle à utiliser pour notre problème puisque le but final est de faire la détection d'arbres en temps réel sur des images collectée à partir de la caméra de notre robot, d'où la nécessité de la rapidité du modèle pour que le robot n'ait pas à attendre les résultat avant de continuer ses autres tâches.

4 Préparation des données

Pour pouvoir utilisé chacun des deux modèles décrit ci-dessus il faut les entraîner sur des données réels pour qu'ils comprennent la tâche qu'on veut résoudre dans notre problème.
A cause de la grande différence entre les deux modèles, il a fallu une préparation de données différente pour chacun des deux :

1. **Le modèle génératif :**
   Pour ce modèle il a fallu faire le tour sur différentes images réelles contenant des champs d'arbres de sapins, puis prendre la valeur de couleur de plusieurs pixels dans l'image et spécifier la classe de chacun de ces pixels. Pour ce modèle j'ai défini trois classe qui sont [**arbre , tronc, arrière plan**], où arrière plan contient tout ce qui n'appartient pas à l'arbre ou à son tronc.
   Pour faire cette tâche j'ai utilisé un outil de traitement d'images **ImageJ**[6], qui permet de sélectionner un pixels dans l'image et de donner les valeurs **RGB** du pixel sélectionner.

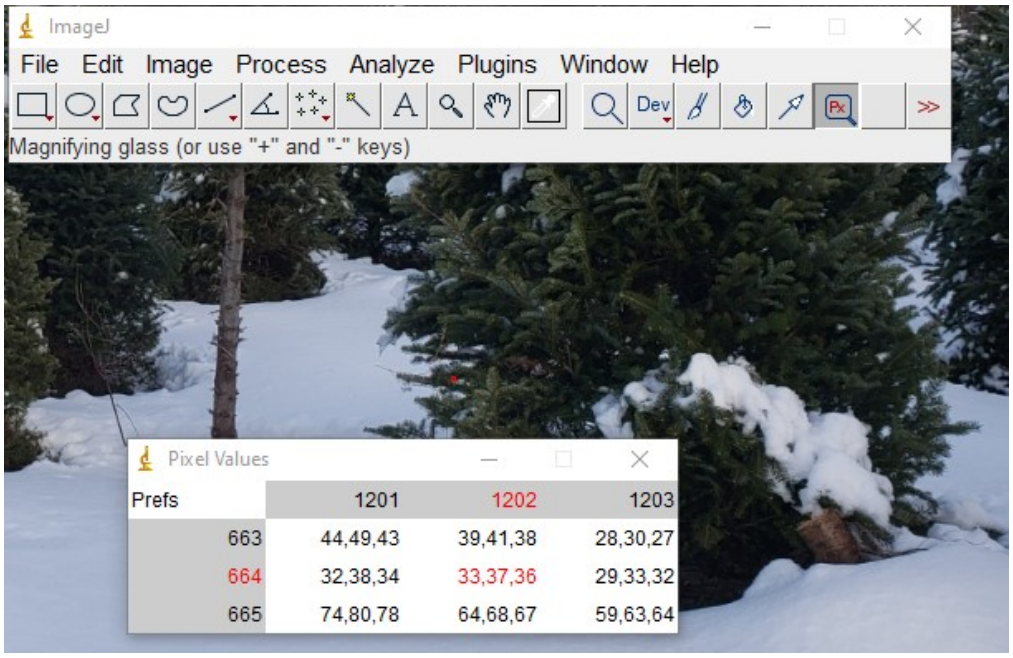

FIGURE 1 – Exemple de valeur de pixel en utilisant ImageJ

2. **CNN :**
   Pour le modèle CNN, l'entraînement du modèles ce fait en lui donnant en entrée plusieurs images contenant des champs d'arbres de sapins, et puis il faut lui donner aussi à quoi ressemble la sortie qu'il doit prédire. Donc la préparation des données consiste à créer manuellement les masques de sortie de chacune des images de notre ensemble de données d'apprentissage.
   Pour faire cette tâche j'ai eu recours à l'application web **"LabelMe"**[7] qui s'agit d'un outil d'annotation d'images en ligne, cette outil à était créer par le laboratoire de l'intelligence artificielle de l'université **"MIT"**.

FIGURE 2 – Image original de celle utilisée à droite

FIGURE 3 – Exemple de masque créer en utilisant LabelMe

5 Implémentation des modèles

Pour implémenter le modèle génératif, j'ai utilisé une bibliothèque du langage de Python **"PlantCV"**[8], qui est spécialiser dans la détection des plantes et des arbres dans les images en utilisant ce modèle. La fonction utilisée est **"naive_bayes_classifier"** qui prend en comme paramètres l'image à classifier ainsi qu'un chemin du fichier contenant les distributions de probabilités de chaque classe. Ce dernier est générer à partir des données décrit dans la partie préparation des données en utilisant un script python sous le nom **plantcvtrain.py** qui fait aussi partie de la même bibliothèque.
Puis pour implémenter le réseau de neurones UNet, j'ai utilisé la bibliothèque de Python **"PyTorch"**[9] qui est un framework d'apprentissage machine open source très utilisé dans la plupart des projets implémentant des réseaux de neurones. Les valeurs qui ont donné les meilleurs résultats sont un nombre d'epochs égale à 10, un "learning rate" de valeur 0.001 et un "optimizer" de type "Adam".

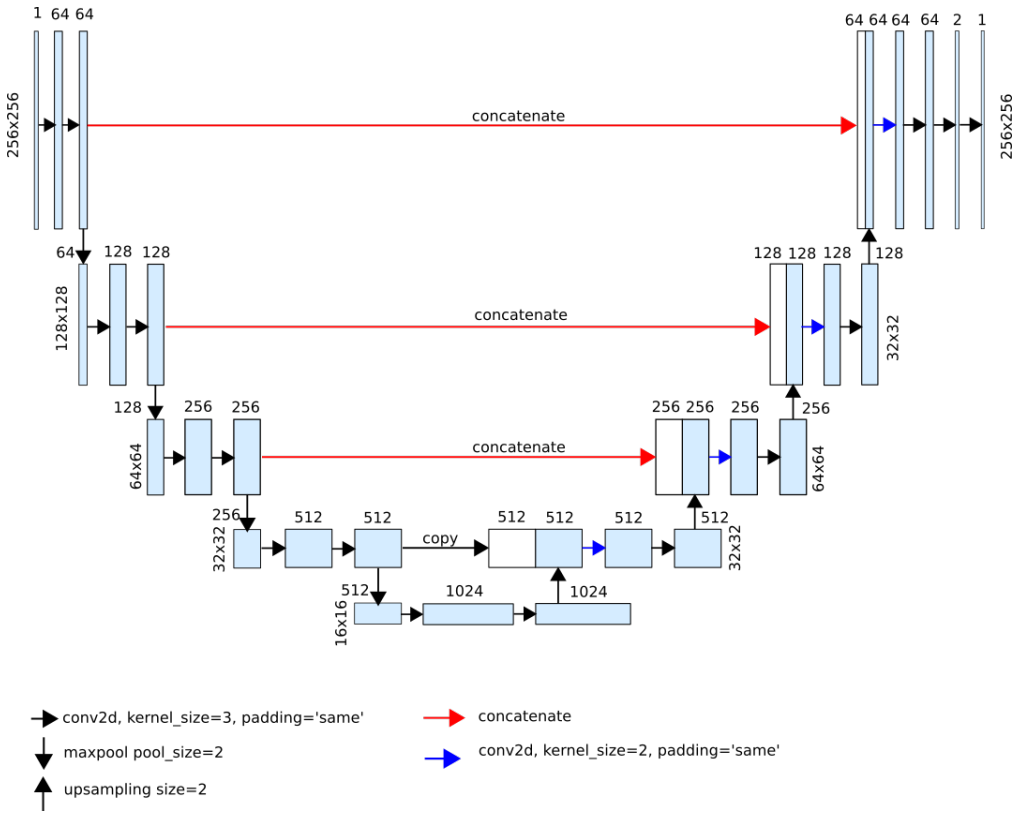

FIGURE 4 – Structure du réseau UNet utilisé

6 Présentation des résultats

Après entraînement de chacun des deux modèles, j'ai fait des prédictions avec des données non utilisée dans l'entraînement pour pouvoir juger correctement le succès ou non du modèle à réaliser la tâche voulu.

6.1 Modèle génératif

Pour l'entraînement du modèle génératif j'ai utilisé 17 images, et pour juger sa performance j'ai vu comment il performe sur ces 17 images ainsi que sur 4 autres images qui ne font pas partie des images de l'entraînement.

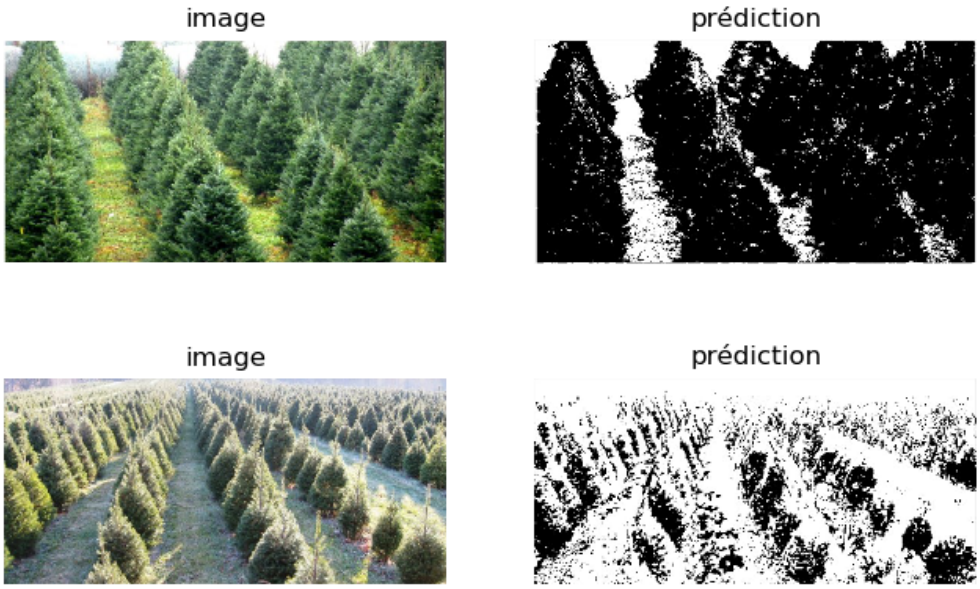

FIGURE 5 – Prédictions du modèle génératif du deux images

Dans la figure 5 j'ai choisis deux images avec leurs prédictions par le modèle génératifs, où j'ai choisi la première parmi ceux sur les quelles le modèle a donné d'assez bon résultats et la deuxième parmi ceux que le modèle a donnée de moins bon résultats.
Donc on voit que dans la première image le modèle a bien su différentier entre les arbres et le sentier à travers eux. Or dans la deuxième image on voit qu'il y beaucoup de zone qui ont la couleur blanche dans la prédiction, alors que dans l'image original ces zones appartiennent à une des arbres. Donc c'est une mauvaise prédiction puisque qui va fort probablement résulter à ce que le robot entre en collision avec l'un des arbres.
Après analyse des résultats, j'ai constaté que ce modèle est vulnérable à différents bruits, par exemple dans la deuxième image de la figure ci-dessus, la prédiction est mauvaise à cause de la lumière du soleil qui a affecté la couleur des arbres, et ainsi puisque le modèle se base sur la couleur du pixel pour déterminer son appartenance à une classe, il a fait une mauvaise classification sur les pixels ayant des couleurs brillantes.
Une autre vulnérabilité est quand l'image n'est pas assez éclairer et que la couleur du sentier entre les arbres est assombris par les ombres des arbres, le modèle fait souvent erreur et considère que ce sentier appartient à la classe 'arbre'.

6.2 CNN UNet

Pour ce modèle, l'entraînement c'est fait avec 20 images au total dont 16 pour l'apprentissage et 4 pour le test.

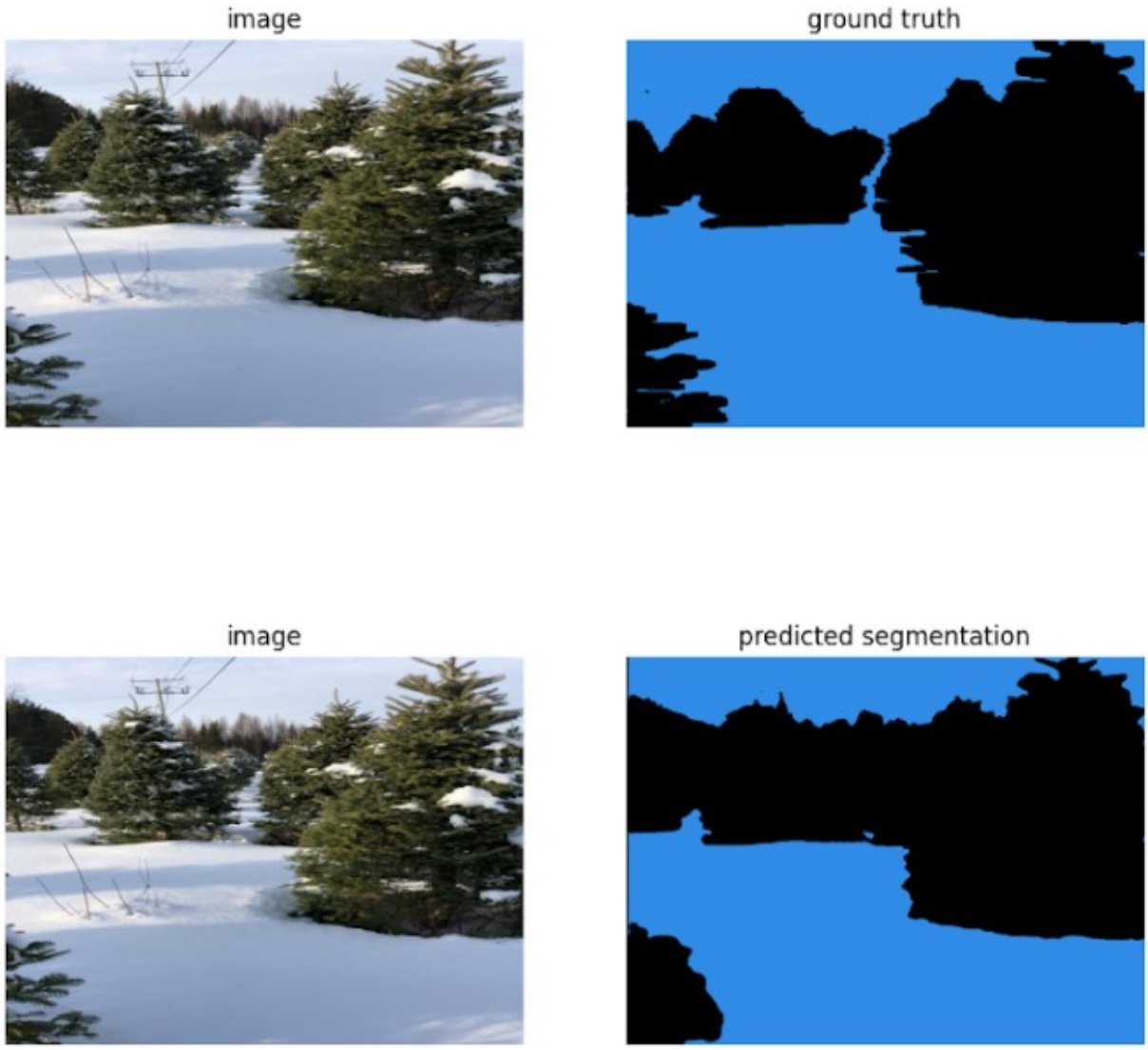

FIGURE 6 – Image réel, son masque crée manuellement et le masque prédit par le modèle

Dans la figure 6 on trouve à gauche l'image utilisé pour la prédiction, à droite en haut, on trouve le masque de cette image qui a était crée manuellement en utilisant l'outil LabelMe, puis à droite en bas en trouve le masque prédit par notre réseau UNet après entraînement.
On voit que la prédiction n'est pas parfaite, surtout que le réseau n'a pas pu identifier le petit sentier à travers les arbres, mais sa revient à sa petite taille.
Un avantage qu'on remarque dans la prédiction de ce modèle comparé au dernier modèle, est que les deux classes prédite sont facilement distinguable, alors que dans la prédiction du modèle génératif on trouve nécessairement dans chacune des deux classes (blanc ou noir) des points de l'autre classes. Ces petits bruits peuvent causer des problèmes au bon fonctionnement du robot.
Autre point important, est que ce genre de modèle de réseau de neurones nécessite un grand

nombre de données pour pouvoir donnée de bon résultats, et vu que dans mon problème le nombre de données disponible était limité, alors j'ai pas pu obtenir les meilleurs résultats par ce modèle.

7 Comparaison des modèles

Pour pouvoir comparer les résultats, j'ai refait l'entraînement des deux modèles sur le même jeux de données composé de 20 images pour l'apprentissage et 5 images pour le test.

Après préparation de ces données pour chacun des deux modèles. Le modèle CNN a pris 15 minutes pour faire l'entraînement, mais dans la prédiction il est assez rapide vu qu'il prend qu'environ 1 seconde pour générer le masque d'une image en entrée.

Alors que pour le modèle génératif, il n'y a pas de processus d'entraînement séparées, pour faire une prédiction le modèle a besoin de l'image d'entrée, et des valeurs des probabilités calculées en utilisant les 20 images de l'entraînement lors de la préparation des données. Cette prédiction prend bien plus de temps que le modèle CNN vu qu'on moyenne le modèle prend 30 secondes pour faire la prédiction.

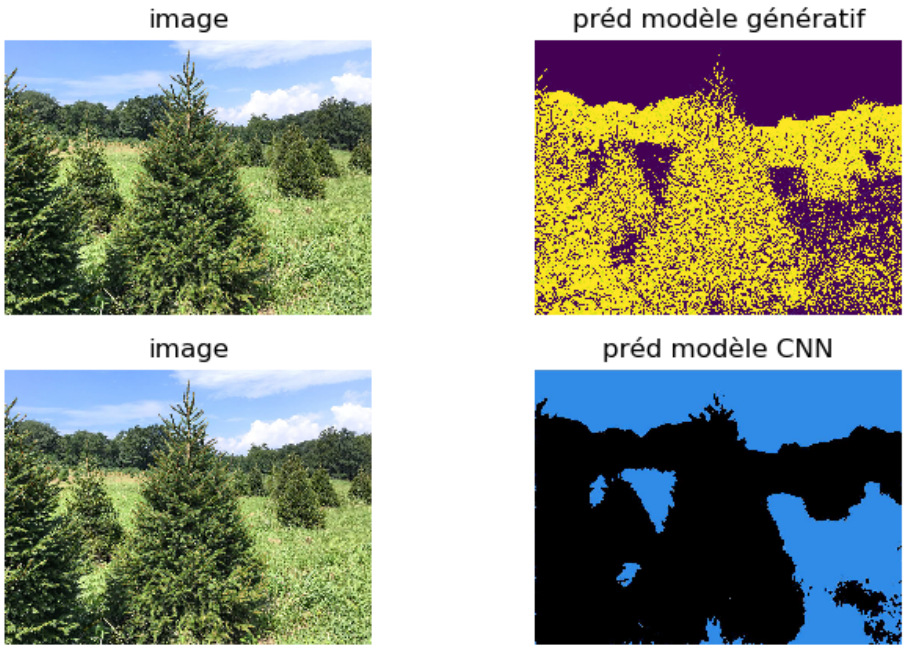

FIGURE 7 – Prédiction des deux modèles sur une même image pour comparer

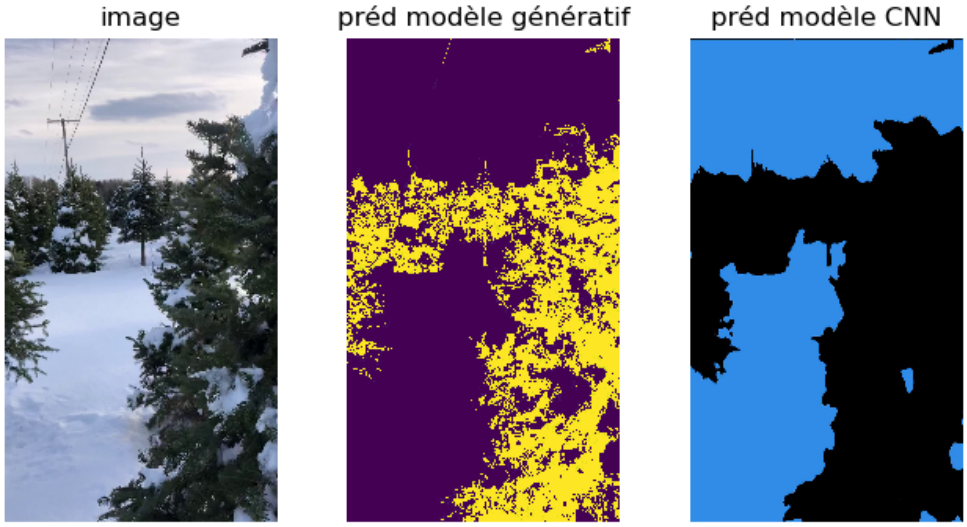

FIGURE 8 – Prédiction des deux modèles sur une même image pour comparer

On remarque que sur l'image avec la neige de la figure 8 les deux modèles performe d'une façon un peu similaire avec pas trop de bruit dans la prédiction, alors que dans la première image de la figure 7, le modèles CNN performe un peu mieux vu que la surface entre les arbres à une seule couleur unie mais pour le modèle génératif cette surface contient quelque point de bruits.

8 Conclusion et perspectives

En conclusion le travail fait constitue un avancement dans la mission du projet mais pas un succès absolu. Les deux modèles utilisées n'ont pas donnée des résultats assez bon pour qu'il soient utilisé sur le robot sans risque. Ainsi, comme perspective pour la suite du projet, il faudra tester d'autres modèles, ainsi que utilisé d'autres données plus variés pour s'assurer que les modèles fonctionnent bien dans les différentes circonstances.

Références

[1] **Xiaogang Wang** (2016), « Deep Learning in Object Recognition, Detection, and Segmentation », Foundations and Trends in Signal Processing : Vol. 8 : No. 4, pp 217-382.
http://dx.doi.org/10.1561/2000000071

[2] **Liu, Lanlan & Muelly, Michael & Deng, Jia & Pfister, Tomas & Li, Li-Jia.** « Generative Modeling for Small-Data Object Detection. , » (2019).
https://arxiv.org/pdf/1910.07169.pdf

[3] **Jiang, Xue-Hong & Feng, Hui-Li & Dong, Yi-Jie.** « Application of Neural Network in Image Detection of Illegal Billboards ». 10.2991/assehr.k.200207.003.(2020).

[4] **Ng, Andrew Y. ; Jordan, Michael I.** (2002). « On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes » Advances in Neural Information Processing Systems.
http://robotics.stanford.edu/ ang/papers/nips01-discriminativegenerative.pdf

[5] « Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation ». DeepLearning 0.1. LISA Lab. Retrieved 31 August 2013.
http://deeplearning.net/tutorial/lenet.html

[6] **Image processing and analysis in Java**
https://imagej.nih.gov/ij/

[7] **LabelMe : online annotation tool to build image databases for computer vision research**
http://labelme.csail.mit.edu/Release3.0/

[8] **PlantCV : open-source image analysis software package targeted for plant phenotyping**
https://plantcv.readthedocs.io/

[9] **PyTorch : An open source machine learning framework**
https://pytorch.org/