

UNIVERSITE JEAN MONNET
Master 1 Données et systèmes connectés
DSC

RAPPORT DE STAGE
DE RECHERCHE

DÉTECTION DES OBSTACLES SUR LES
RAILS DES TRAMWAYS AUTONOMES

Réalisé par : Fares DABBEK

Encadré par : Fabrice Muhlenbach

Période du projet : 05/2020 jusqu'à 08/2020

Année universitaire : 2019/2020

Table des matières

| | | |
|--------|---|----|
| 1. | Introduction | 4 |
| 1.1. | Contexte du stage..... | 4 |
| 1.2. | Cadre de stage..... | 4 |
| 1.3. | Problématiques | 5 |
| 1.4. | Plan du rapport..... | 5 |
| 1.5. | Planification du projet | 6 |
| 1.5.1. | Trello | 6 |
| 1.5.2. | GanttProject..... | 7 |
| 2. | Matériel et méthodes | 8 |
| 2.1. | Matériel..... | 8 |
| 2.1.1. | Environnement matériel | 8 |
| 2.1.2. | Environnement logiciel | 8 |
| 2.2. | Méthodes | 9 |
| 2.2.1. | Modèles de détection d'objet | 9 |
| 2.2.2. | Modèle de segmentation choisi | 10 |
| 3. | Résultats et Discussion..... | 17 |
| 3.1. | Préparation du Dataset..... | 17 |
| 3.2. | Annotations..... | 18 |
| 3.3. | Entraînement..... | 19 |
| 3.3.1. | L'entraînement du Mask des rails | 19 |
| 3.3.2. | Entraînement des objets sur les rails | 20 |
| 3.4. | Prédiction..... | 21 |
| 3.4.1. | Détection de rail : | 21 |
| 3.4.2. | Détection des objets devant le rail :..... | 23 |

| | |
|---|----|
| 3.5. Combinaison des deux modèles | 24 |
| 4. Conclusion et perspectives | 27 |
| 5. Bibliographie..... | 28 |
| 6. Annexe | 29 |
| 6.1. Annexe 1 (Noms des classes COCO) | 29 |
| Table des figures | 30 |
| Liste des tableaux | 31 |

1. Introduction

1.1. Contexte du stage

De nouveaux types d'accidents surviennent dans le domaine des véhicules malgré l'utilisation de nouvelles techniques pour faire diminuer ces accidents. L'erreur humaine est souvent mise en cause lors d'études statistiques (environ 2/3 des accidents/incidents sont liés aux facteurs humains). Ces derniers doivent donc être pris en compte dans l'analyse des nouveaux systèmes.

Un tramway autonome est un véhicule qui possède des fonctionnalités d'autonomies pour la conduite et la détection des obstacles.

Dans cette optique, le travail consiste à définir un protocole qui permet de gérer la détection des obstacles. Ce protocole s'appuie principalement sur l'utilisation de Deep Learning. Dans ce contexte, j'ai imaginé un système de transport fluide, autonome et original de tramways pour la ville de demain. Ce système de transport est basé sur les tramways qui peuvent détecter automatiquement les obstacles.

1.2. Cadre de stage

Le domaine des véhicules autonomes est un domaine de recherche en pleine croissance.

J'ai choisi ce projet suite à l'annulation de mon stage avec l'entreprise Railenium (spécialisé dans le domaine ferroviaire) à cause de l'épidémie du COVID-19.

Ayant l'occasion de découvrir le monde d'intelligence artificielle lors du premier semestre et le Machine Learning lors du deuxième semestre avec plusieurs séances de TP, j'ai trouvé que le Deep Learning est la suite logique de ça. Je suis passionné par le domaine des véhicules autonomes qui utilise l'application de Deep Learning dans la détection des objets, donc j'ai choisi de poursuivre le travail dans le domaine ferroviaire en choisissant ce sujet innovant.

1.3. Problématiques

Nous pouvons formuler la problématique de notre travail sous forme la question suivante :

- Comment peut-on détecter les obstacles devant le tramway et notifier le conducteur dans cette situation ?



Figure 1 : Présence d'obstacle devant un tramway.

1.4. Plan du rapport

Ce projet de recherche concerne la détection des rails et les localisations des obstacles. Le plan de ce rapport sera organisé comme suit :

Le contexte du stage a abordé la détection des obstacles en utilisant le Deep Learning. Dans la partie 2, je vais discuter des 3 méthodes de détections d'objets (classification, détection d'objets, segmentation) en détaillant les parties matérielles et logicielles .

Un modèle de segmentation est choisi et détaillé dans la partie qui suit : le Mask R-CNN.

Ensuite, dans la partie résultat j'ai expliqué la procédure de projet qui était réalisé en utilisant le réseau de neurones de Deep Learning qui permet de détecter le rail du tramway et un autre réseau pré-entraîné qui permet de détecter les obstacles. Cela permettra par la suite de faire la combinaison entre ces deux parties.

Après je finirai par la conclusion qui va résumer tout mon travail.

1.5. Planification du projet

1.5.1. Trello

J'ai déduit la nécessité de planification d'un programme du travail bien défini et organisé pour le but de réaliser des résultats satisfaisants en utilisant l'outil **Trello**.

Trello est un outil d'organisation collaboratif simple et gratuit qui permet d'organiser les projets en des listes partagées.

Il renseignera sur tous les projets, sur leur état d'avancement et les tâches de chaque membre du projet.

La prise en main de cet outil d'organisation est très simple. Le tableau de bord est composé d'une série de listes correspondant à chaque projet. Chaque liste est composée de plusieurs cartes qu'on peut déplacer d'une colonne à l'autre. Par défaut, chaque tableau est composé de trois colonnes principaux «TO DO », « IN PROGRESS » et « DONE ». Il s'agit d'une méthode universelle d'organisation très efficace. (Trello, s.d.)

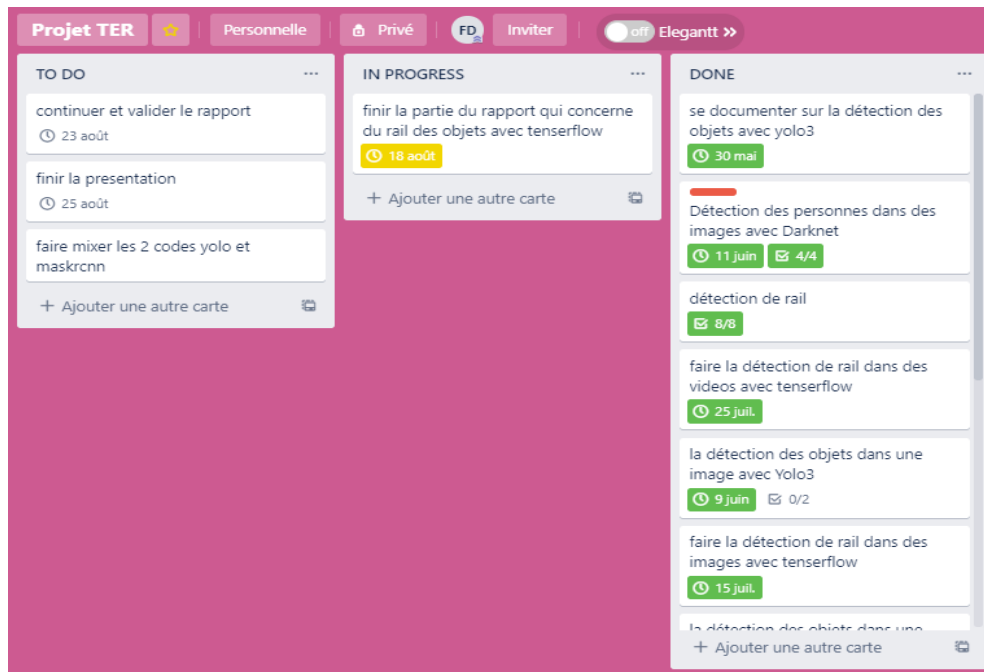


Figure 2 interface de Trello

1.5.2. GanttProject

C'est un outil open source qui permet de faire les planifications des projets en création de diagrammes de Gantt avec les dates de débuts et de fins.

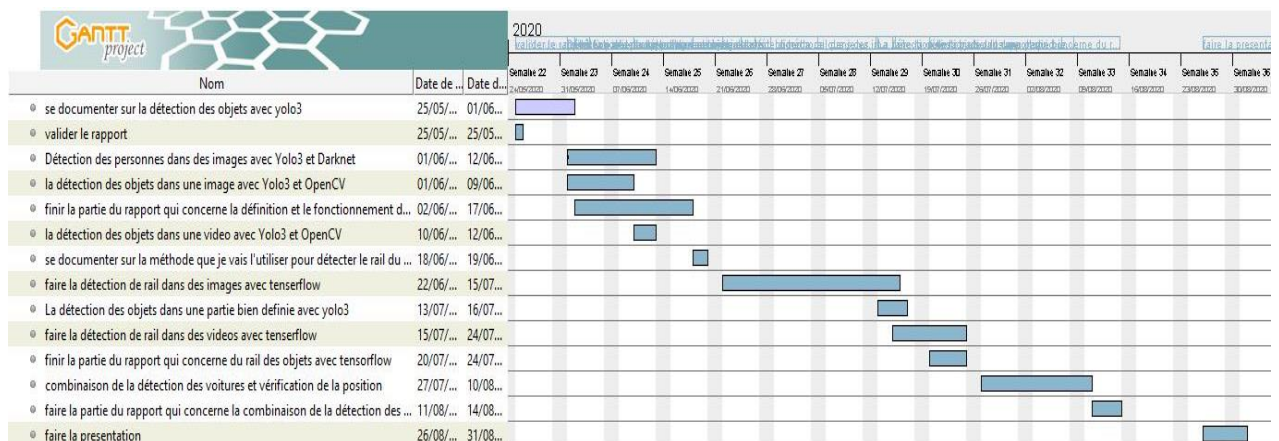


Figure 3 Diagramme de GANTT

2. Matériel et méthodes

2.1. Matériel

2.1.1. Environnement matériel

Google Colab

C'est un outil gratuit de recherche pour l'éducation et la recherche sur l'apprentissage automatique fourni par Google. Il s'agit d'un environnement Jupyter pour ordinateur portable qui permet d'écrire et d'exécuter du code Python dans le navigateur et qui ne nécessite aucune configuration pour être utilisé et d'une source de GPU.

Pour commencer, il vous faut un compte Google.

Au lieu de dépenser des Centaines d'euros pour acheter et installer un GPU, Les spécifications matérielles fournies dans l'environnement GOOGLE COLAB sont :

GPU : 1xTesla K80, a 2496 CUDA cores , 12GB GDDR5 VRAM (compatible avec CUDA)

CPU: 1xsingle core hyper threaded Xeon Processors @2.3Ghz i.e(1 core, 2 threads)

RAM : ~12.6 GB

Disk : ~33 GB

2.1.2. Environnement logiciel

Python

C'est un langage de programmation de haut niveau qui est très utilisé dans la science des données et pour la production d'algorithmes d'apprentissage en profondeur.

TensorFlow :

C'est une bibliothèque de Machine Learning open source Créée en 2011 par Google dédié pour les réseaux de neurones de Deep Learning, amélioré en version 1.0 le mois de février 2017. Cette bibliothèque contient plusieurs algorithmes et modèles

de machine et Deep Learning. Elle permet aussi de faire des calculs mathématiques très complexes et produire des protocoles d'apprentissage qui vont servir à faire l'entraînement et l'exécution des réseaux de neurones.

keras

C'est un API de de Tensorflow écrit en Python qui permet de créer et de faire l'entrainement des modèles de Deep Learning.

2.2. Méthodes

2.2.1. Modèles de détection d'objet

Classification

La classification fait référence à un type de détection dans lequel une image ou une vidéo c'est-à-dire catégoriser l'image ou la vidéo entière dans une classe, dans le but de répondre à la question « Que contient cette image ou cette vidéo ? »

Détection d'objets

La détection d'objets est une technique de vision par ordinateur qui permet d'identifier les objets d'une image ou d'une vidéo. Elle est plus spécifique par rapport à la classification. On applique la classification à des objets distincts dans une image ou une vidéo en utilisant des cadres de délimitation pour nous dire où se trouve chaque objet dans une image / vidéo.

Segmentation

La segmentation est un type d'étiquetage où chaque pixel de l'image est étiqueté.

Ici, des images entières sont divisées en groupes de pixels qui peuvent ensuite être étiquetés dans le but de faire simplifier chaque image ou bien modifier la façon dont une image est présentée au modèle. Cela est fait pour la rendre plus simple à analyser.

Elle est liée à la classification des images et aussi à la détection des objets car ces

deux techniques doivent avoir lieu avant que la segmentation commence et après que l'objet en question est isolé avec une limite.

Le rôle principal de notre modèle c'est d'identifier les objets à partir des données d'entrées donc c'est un apprentissage qui s'appelle supervisé. Donc ce type d'algorithmes d'intelligence artificielle utilise principalement les réseaux de neurones convolutifs (**Convolutional Neural Networks « CNN »**)

2.2.2. Modèle de segmentation choisi

Les modèles de Deep Learning qui sont basées sur la notion des réseaux de neurones gèrent les traitements sur les images convolutives (CNN). Ce principe a connu plusieurs améliorations et évolutions qui ont permis de faire grandir les tâches que ces modèles peuvent les réaliser et aussi d'optimiser son fonctionnement.

Le modèle que j'ai choisi d'utiliser « Le Mask R-CNN » fait une combinaison entre la segmentation et la classification d'objets : c'est le résultat de plusieurs améliorations et combinaisons d'algorithmes sur lesquelles nous allons en parler.

Le Mask R-CNN qui est une partie de famille des modèles R-CNN dont l'objectif est la détection et la localisation de chaque objet dans une image. Il est donc une extension de ce type de modèle de détection d'objets. La question qui se pose c'est que comment ces algorithmes de deep-learning ont-ils été améliorés pour obtenir finalement la version du Mask R-CNN ?

CNN : Réseau de Neurones convolutifs

Les algorithmes de réseaux de neurones ont le même concept de base du réseau de neurones du cerveau humain. Les réseaux de neurones convolutifs (CNN) sont des exemples des algorithmes permettent d'analyser les images. Ce sont des méthodes qui visent à s'approcher de capacités de la vision humaine.

Ces réseaux sont constitués par différentes couches qui sont constituées par plusieurs neurones. Chaque couche reçoit les données de la couche précédente, les traite, après les renvoie vers la couche suivante. Chaque neurone d'une couche est donc relié aux neurones de la couche qui suit. Ces liens sont pondérés par des poids

et, à l'image du signal électrique émis par les neurones biologiques, vont influencer la propagation de l'information à travers le réseau de neurones convolutifs.

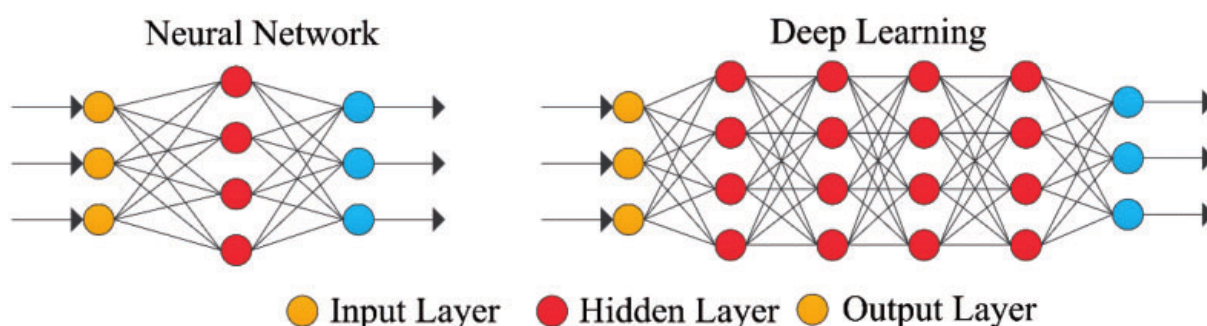


Figure 4 CNN : Réseau de Neurones convolutifs

Ces modèles de réseaux de neurones sont constitués de différents types de couches neuronales, correspondant à plusieurs étapes de l'analyse :

Couche de convolution :

Ces modèles traitent chaque image “ fragment par fragment ”. Après il y a l'étape de comparaison de chaque fragment à des caractéristiques particuliers. La méthode utilisée pour faire la filtration de ces images c'est la « convolution ». Ensuite, on applique des filtres ‘Feature Detector’ qui permettant la détection de caractéristiques spécifiques, et à partir de ça se crée plusieurs Feature Map ou "les cartes des caractéristiques" qui correspondent à ces différents filtres. L'image initiale est donc convertie en plusieurs matrices selon les différentes caractéristiques avec lesquelles on l'a filtrée.

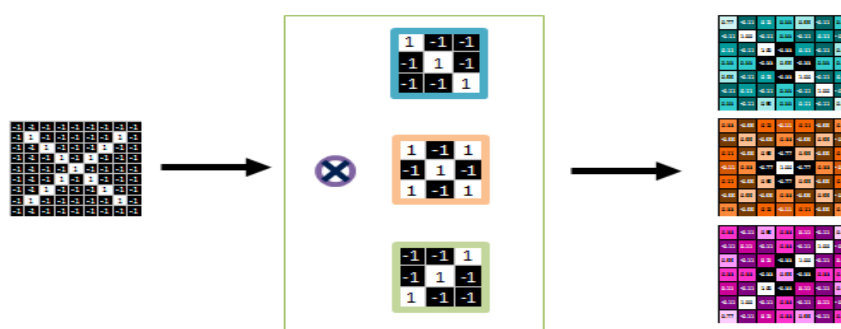


Figure 5 fonctionnement de la couche de convolution

Couche de pooling:

L'étape suivante c'est le pooling, elle consiste à faire diminuer la taille de l'image mais en gardant le maximum d'informations, c'est à dire on réduit le nombre de paramètres descriptifs de chaque image pour ne pas faire le sur-apprentissage. Prenons comme exemple le cas où on veut détecter une personne, savoir que ses oreilles sont situées juste à côté est suffisant donc on n'a pas besoin de faire une localisation très précise de ça.

Pour cela on découpe l'image en des fenêtres carrées généralement de 2*2 pixels sur les Feature Map obtenues par convolution en gardant toujours la valeur maximale qui contient les informations les plus importantes. En sortie, le nombre de feature maps obtenu est le même d'entrée mais plus petit.

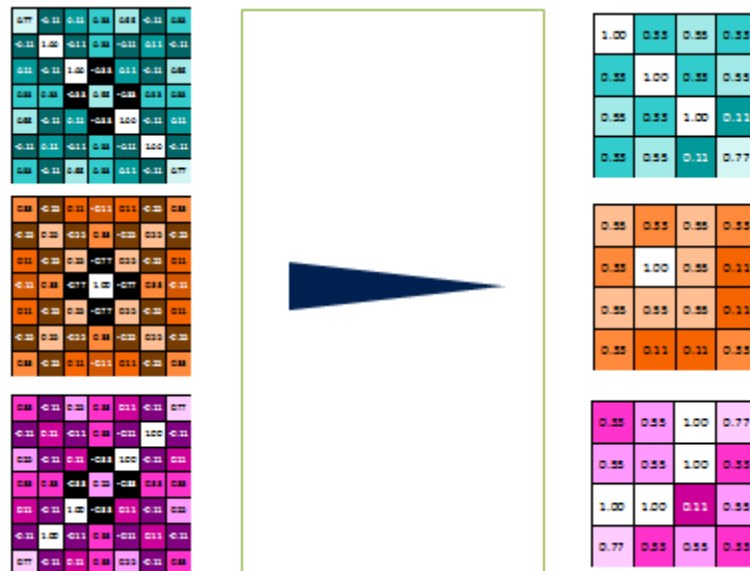


Figure 6 fonctionnement de la couche de pooling

Couche de flattening :

Le but de cette couche c'est obtenir un format de sortie qui est compatible avec les besoins de l'entrée attendue par le réseau de neurones pour cela on transforme la matrice N en un vecteur.

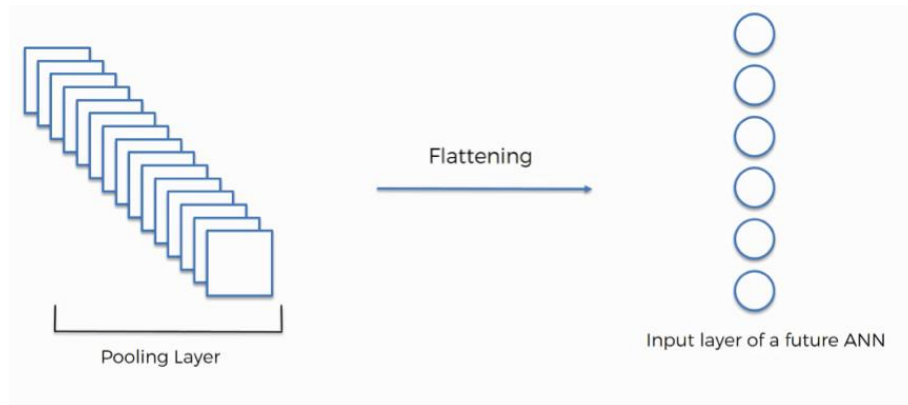


Figure 7 fonctionnement de la couche de flattening

Couche dense (full connection):

C'est la dernière couche de réseau de neurones qui prend comme entrée les résultats de la couche de flattening qui est un vecteur et génère un nouveau vecteur en sortie. Elle renvoie la probabilité de chaque classe de prédiction c'est-à-dire à quelle classe elle appartient.

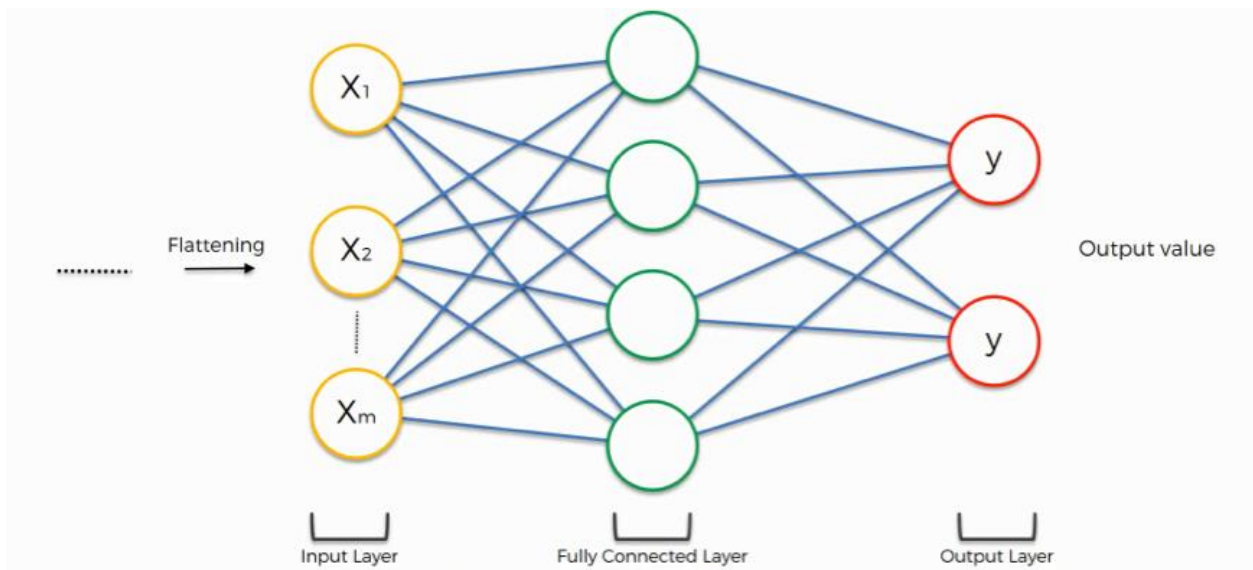


Figure 8 fonctionnement de la couche de dense

R-CNN (Regional Convolutional Neural Network)

Ce modèle prend une image en entrée, il travaille sur des parties qui s'appellent zones d'intérêts qui contiennent les objets à détecter. Pour bien préciser l'objet, le modèle génère des cadres (bounding box). Ces cadres sont placés et classifiés selon la probabilité de détection de l'objet.

Le modèle extrait environ 2000 propositions de régions, après c'est le CNN qui va recevoir ces régions. Le modèle va vérifier si ces régions appartiennent à des classes (qui sont déjà prédéfinies) ou non.

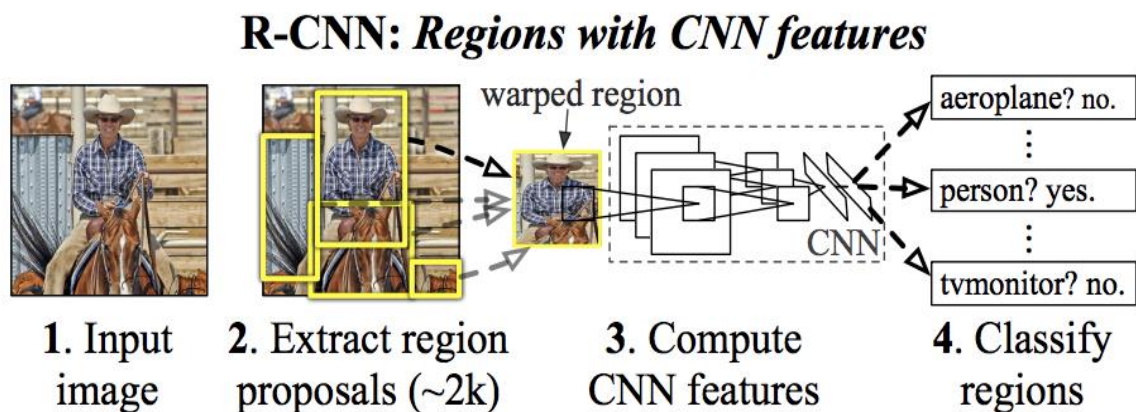


Figure 9 fonctionnement du Mask R-CNN

Fast R-CNN

Ce type de modèle est plus puissant et plus amélioré en termes de temps d'exécution puisque l'étape de convolution est subie sur l'image d'entrée et les propositions de régions d'intérêts qui peuvent contenir l'élément à détecter sont obtenus sur les features map.

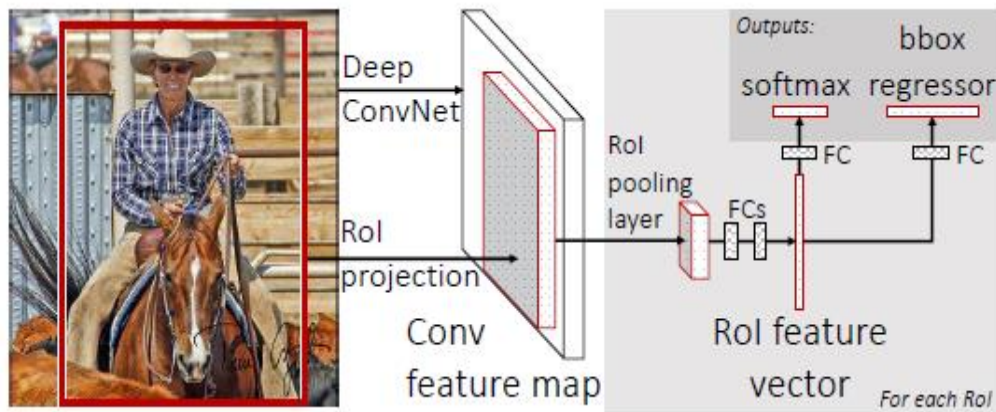


Figure 10 fonctionnement du modèle Fast R-CNN

Faster R-CNN (Fast R-CNN + RPN)

Dans ce type de modèle il y a l'ajout de l'algorithme Region Proposal Networks (RPN) qui permet de générer les propositions de zones directement ce qui rend le temps de produire les zones d'intérêt plus optimisé.

Le modèle Faster R-CNN génère en sortie un label et la bounding box.

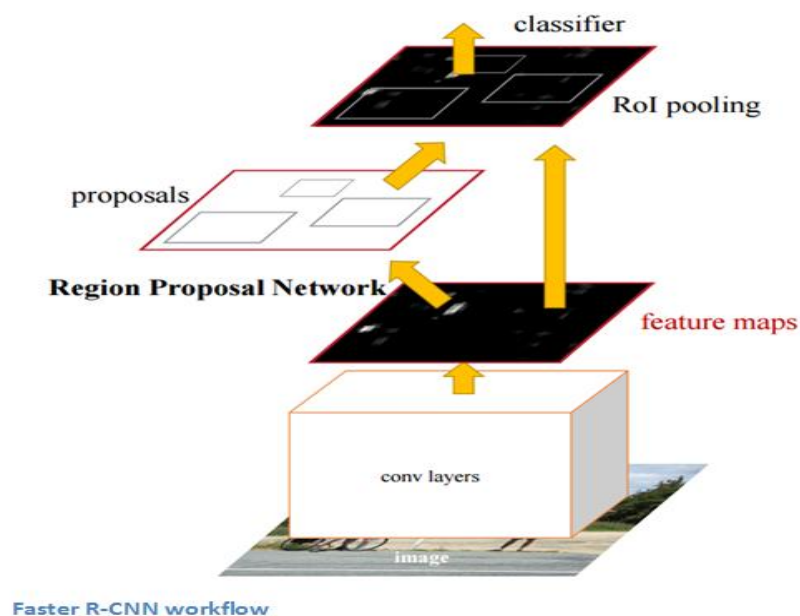


Figure 11 fonctionnement du modèle Faster R-CNN

Mask-RCNN

C'est la dernière version des modèles RCNN qui a été créé par Facebook. Il permet de faire la détection et la classification des objets. L'avantage dans cette version c'est que chaque pixel de l'image sera classé, ce qu'on appelle détection et la segmentation des instances. Il permet aussi d'associer à chaque objet un bounding box et un label même si ces objets appartiennent à la même classe. C'est un fusionnement entre le réseau de proposition de régions (regions proposal network, RPN) et le réseau ResNet101. (Kaiming He, 2018)

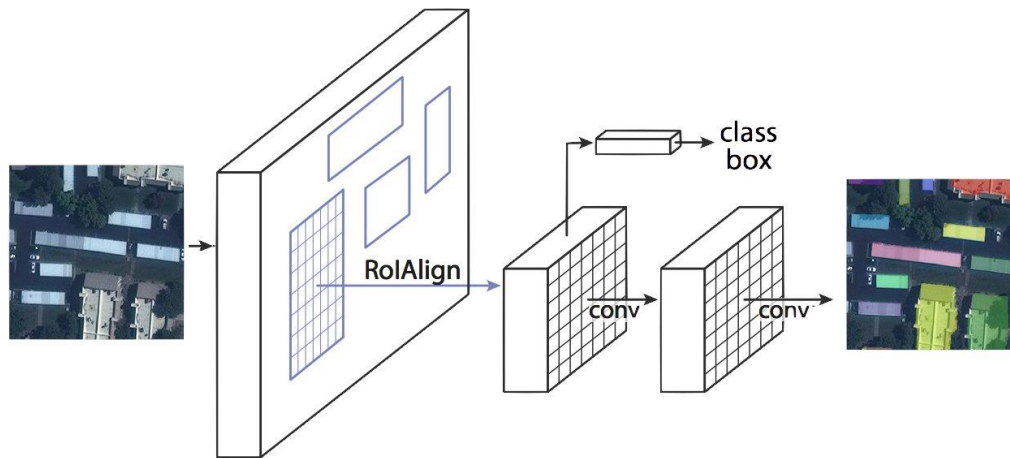


Figure 12 fonctionnement du modèle Mask R-CNN

3. Résultats et Discussion

Cette partie concerne les résultats du travail effectué dans mon projet. La première partie concerne la collecte et l'annotation de la BDD des images de tramway ce qui permettra d'effectuer l'entraînement d'un modèle capable de détecter les rails. La deuxième partie concerne le déploiement d'un modèle de Mask-RCNN pré-entraîné avec la BDD COCO (80 objets) permettant la détection des objets sur les rails du tramway ce qui permettra l'identification de degré de dangerosité de la détection.

3.1. Préparation du Dataset

L'objectif dans cette étape est de construire une base de données des images à partir des vidéos disponibles sur YouTube des tramways qui circulent.

Pour se faire, il faut choisir plusieurs vidéos qui illustrent le cycle de vie d'un tramway dans différentes conditions (jour, nuit, pluie, brouillard...) et extraire des images à partir de ces vidéos pour construire la une Base de données qui devra être divisé en 3 parties : entraînement, test et validation.

Les vidéos utilisées sont extraits de deux chaines Youtube filmant les situations d'anomalie rencontrées par les conducteurs des tramways.




| | | |
|--|--|--|
|  |  |  |
| Extrait de la 1 ^{ère} vidéo utilisée dans l'entraînement https://www.youtube.com/watch?v=DnFc_KGKd4A (Nohavviktor, 2019) | Extrait de la 2 ^{ème} vidéo utilisée dans l'entraînement https://www.youtube.com/watch?v=bLcdm3dWIRc (SHARMA, 2019) | Extrait de la 3 ^{ème} vidéo utilisée dans l'entraînement https://www.youtube.com/watch?v=hUUpxdqNg0M (4K Cabview Holland Dutch Railways, 2019) |

Tableau 1 les vidéos utilisées pour construire la BDD

Données d'entraînement : représente la partie la plus grande de BDD ~80% que le modèle va utiliser durant la phase d'entraînement pour l'apprentissage.

Données de validation : La partie validation contient environ 10% de la base de données des images, après chaque itération d'entraînement cette partie doit être présente. Elle permet de récupérer les caractéristiques de performance pour mesurer la performance de l'entraînement (la spécificité, la sensibilité).

Données de test : Environ 10% de la base de données des images est utilisé dans la partie test qui va permettre d'évaluer le modèle et vérifier l'efficacité de sa prédiction.

3.2. Annotations

L'annotation des données est une étape importante qui sert à montrer au modèle où se trouve l'objet à détecter, dans notre situation ce sont les rails des tramways.

La précision est un facteur essentiel dans l'annotation pour comprendre les images annotées et obtenir des résultats performants.

Cette étape ne peut être faite que par l'humain. Pour cela j'ai utilisé l'outil <http://www.robots.ox.ac.uk/> (Annotation Dataset, s.d.) qui génère un fichier json qui contient les masques annotés des rails devant le tramway.

L'annotation a été focalisé uniquement sur le rail devant le tramway et non pas les autres rails, l'objectif est que le modèle arrive à détecter le rail principal et non pas les autres rails des autres tramways.

Exemple d'annotation :

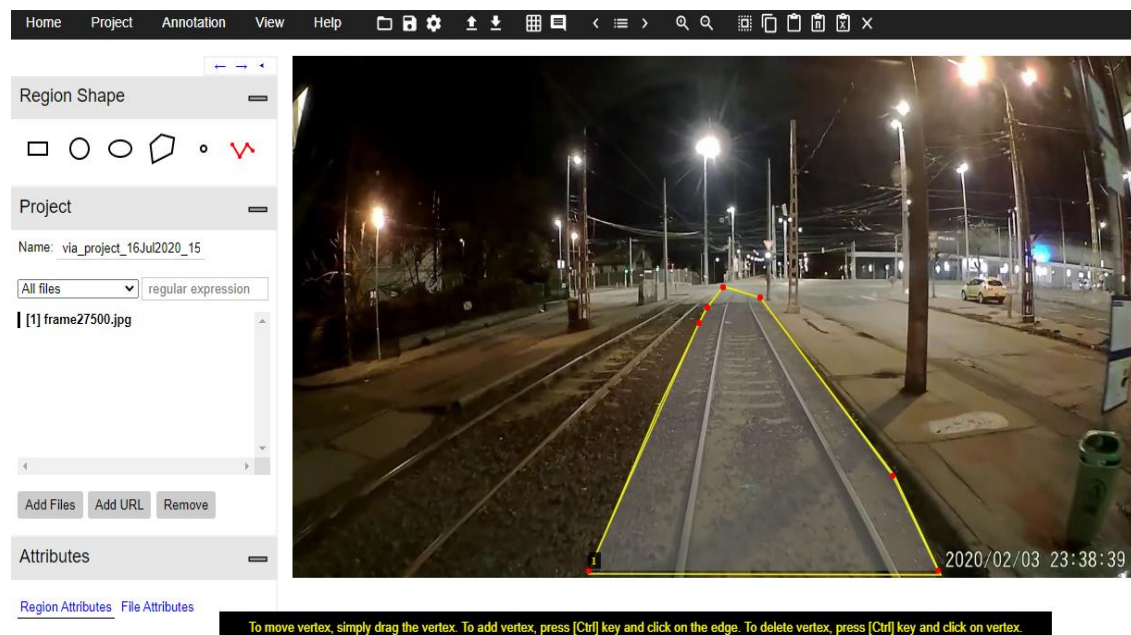


Figure 13 exemple d'annotation

3.3. Entrainement

L'entraînement du modèle consiste à trouver les bons paramètres de fichier de poids « weights » ce qui permettra au modèle de distinguer les pixels appartenant aux rails et les pixels du background. Le modèle utilisera les images annotées dans la phase d'annotation afin d'arriver à retracer correctement les rails sur des images que le modèle n'a jamais vu avant.

3.3.1. L'entraînement du Mask des rails

L'implémentation du modèle Mask-RCNN est disponible sur le repo sur GitHub (https://github.com/matterport/Mask_RCNN, 2017)

Des exemples d'entraînement sur une classe sont disponibles. J'ai travaillé sur l'adaptation du code pour prendre en compte l'entraînement du classe rails déjà annotée.

```

class RailConfig(Config):
    """Configuration for training on the toy dataset.
    Derives from the base Config class and overrides some values.
    """
    # Give the configuration a recognizable name
    NAME = "rail"

    # We use a GPU with 12GB memory, which can fit two images.
    # Adjust down if you use a smaller GPU.
    IMAGES_PER_GPU = 2

    # Number of classes (including background)
    NUM_CLASSES = 1 + 1 # Background + rail

    # Number of training steps per epoch
    STEPS_PER_EPOCH = 100

    # Skip detections with < 90% confidence
    DETECTION_MIN_CONFIDENCE = 0.9

```

Figure 14 les paramètres d'entraînement

Le paramètre le plus important est le nombre des objets à détecter. Dans notre cas on a le background qui un élément par default. Et le l'objet « rail » :

- NUM_CLASSES = 1 + 1 # Background + rail

Dans l'étape de l'entraînement, il y a d'autres paramètres qui sont très importants aussi tels que :

- NAME = "rail" : le nom d'objet à détecter
- STEPS_PER_EPOCH = 100 : qui définit le nombre d'itérations d'entraînement par époques.
- DETECTION_MIN_CONFIDENCE = 0.9 (90%) : qui définit la valeur minium de confiance de prédiction qu'on va la prendre (varie entre 0 et 1)

3.3.2. Entrainement des objets sur les rails

Le paragraphe précédent montre la modification nécessaire pour entrainer un modèle permettant de détecter les rails. Il montre aussi l'utilisation d'un modèle pour détecter les objets devant le rail. Ce modèle est le modèle pré-entraîné sur la BDD COCO de taille presque 250 Mo. Il est disponible sous le nom « mask_rcnn_coco.h5 », ce modèle est déjà entrainé avec 200000 images annotées pour qu'il puisse détecter 80 objets + BG (l'arrière-plan). Les 80 objets sont très variés et il y a beaucoup des objets qu'on peut les retrouver comme obstacle devant le tramway comme :

Personne, vélo, moto, voiture, bus, camion...

La liste complète des 80 classes se trouvent dans [l'Annexe 1](#).

3.4. Prédiction

La prédiction est divisée en 2 parties :

3.4.1. Détection de rail :

Pour obtenir une prédiction performante, j'ai essayé de changer à chaque fois les paramètres principaux tel que : le nombre des epochs (étape d'entraînement), le nombres d'images d'entraînement et le nombres d'images de validation. Ce tableau illustre les informations de chaque modèle utilisé.

Le premier modèle concerne 52 images (42 train, 10 test) de la première vidéo. Le deuxième modèle ajoute à la BDD les images de la deuxième vidéo (47 train, 9 test) et le troisième modèle est basé sur les images des 3 vidéos et j'ai ajouté (140 train et 38 test).

| Numéro modèle | Nombre d'epochs | Images de train ajoutées | Images de val ajoutées | Image de train total | Image de val total |
|----------------------|------------------------|---------------------------------|-------------------------------|-----------------------------|---------------------------|
| 1 | 50 epochs | 42 | 10 | 42 | 10 |
| 2 | 50 epochs | 47 | 9 | 89 | 19 |
| 3 | 100 epochs | 140 | 38 | 229 | 57 |

Tableau 2 La répartition des images de la BDD

Après faire des exemples de détection de rail dans différentes états (matin, nuit) et différentes couleurs de rails en utilisant les 3 modèles pour bien les comparer et distinguer le meilleur entre les 3 on a obtenu ces résultats :

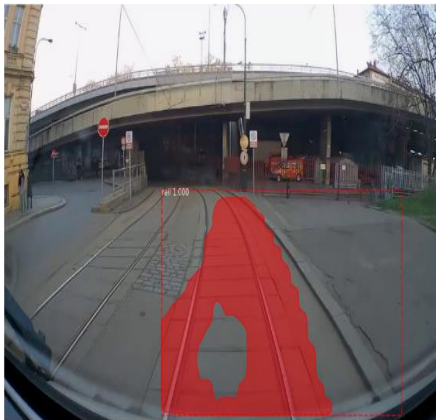





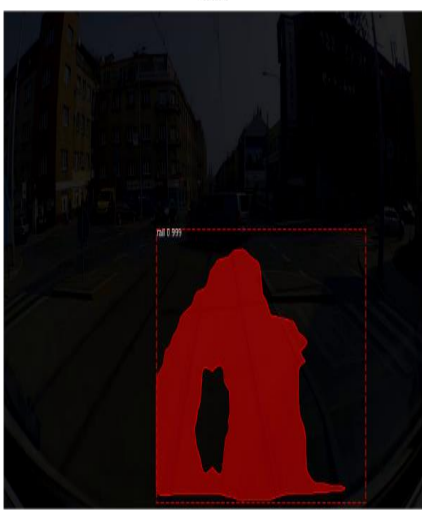
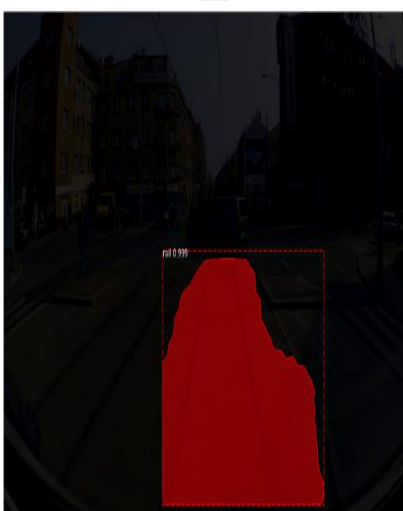

| Model 1 | Model 2 | Model 3 |
|---|---|---|
|  |  |  |
|  |  |  |
|  |  |  |

Tableau 3 La résultat de détection de chaque modèle

A partir de ces résultats on peut conclure que le modèle 1 possède des défauts et n'arrive pas bien à détecter les rails par contre les modèles 2 et 3 sont les meilleurs

mais le 3 est le plus performant et arrive à bien détecter les rails dans les différentes situations. Pour confirmer la performance de ce modèle on focalise aussi sur le paramètre de perte (loss).

- C'est une somme des erreurs commises à chaque epoch, plus cette perte est faible plus que le modèle est plus performant. Cette courbe montre la perte en fonction des epochs de modèle 3 :

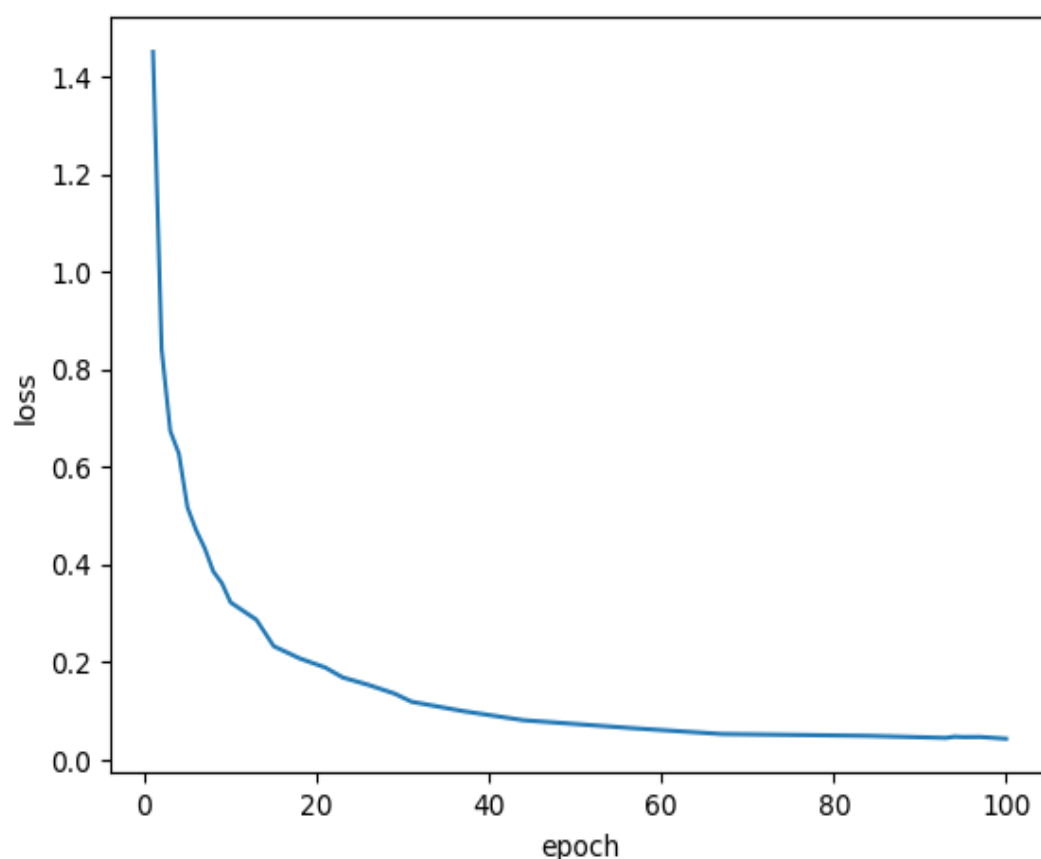


Figure 15 courbe de perte du modèle 4

Comme le montre la courbe la valeur de la perte lors de la dernière epoch est très petite (entre 0.04 et 0.05) ce qui confirme la performance de notre modèle.

3.4.2. Détection des objets devant le rail :

Comme j'ai déjà dit dans la partie d'entraînement, le modèle utilisé pour la détection des objets devant le rail est le modèle pré-entraîné « mask_rcnn_coco.h5 ». Ce modèle était très efficace pour détecter les objets devant le rail exemple voiture, personne, vélo... :

```
frame = display_instances(img, r_coco['rois'], r_coco['masks'],
, r_coco['class_ids'], class_names_coco, r_coco['scores'])
```

La sortie de la prédiction est comme suit :

- Le variable "rois" (regions-of-interest) contient toutes les coordonnées le rectangle qui entoure l'objet qui a été détecté par le modèle ("bbox").
- Le variable masks : sous la forme des matrices qui indique si le pixel fait partie de l'objet détecté ou non si True c'est-à-dire le pixel appartient à l'objet, si False c'est-à-dire le pixel n'appartient pas à l'objet détecté.
- Le variable "class_ids" : Contient l'indice d'objet détecté, dans notre exemple pour la détection de rail l'indice est 1 sinon pour la détection des objets devant le rail l'indice varie entre 1 et 80 selon l'objet qui a été détecté.
- Le variable "scores" c'est un pourcentage qui représente la probabilité que l'objet détecté appartient déjà classe déjà prédéfinie (dans notre cas le minimum c'est 90% pour qu'il soit affiché).

3.5. Combinaison des deux modèles

Cette étape consiste à combiner les deux modèles de détection Mask-RCNN dans le même code. A chaque lecture d'image ou vidéo, le premier modèle entraîné sur la détection des rails effectue la détection et l'affiche sur l'image. La sortie de cette image annotée par le rail est l'entrée au deuxième modèle permettant de faire la détection des autres objets.

Etape de fonctionnement d'algorithme :

Pour arriver à notre objectif c'est-à-dire détecter le rail et les objets devant le tramway, l'algorithme prend une image en entrée et après il va passer par 2 modèles :



Figure 16 image d'entrée

Le modèle de rail : ce modèle va détecter le rail du tramway concerné seulement et applique un bounding box (un rectangle) qui entoure le rail, un masque, un label et aussi la probabilité de prédiction du rail.



Figure 17 la sortie du modèle de détection de rail

Le modèle COCO : vient juste après le modèle de rail. Il prend l'image de sortie du premier modèle en entrée et applique son algorithme de prédiction pour qu'il détecte les objets devant le tramway, aussi comme le modèle de rail il associe à chaque objet un bounding box, un label, et le Mask sur chaque objet et avec sa propre couleur et sa probabilité de prédiction.

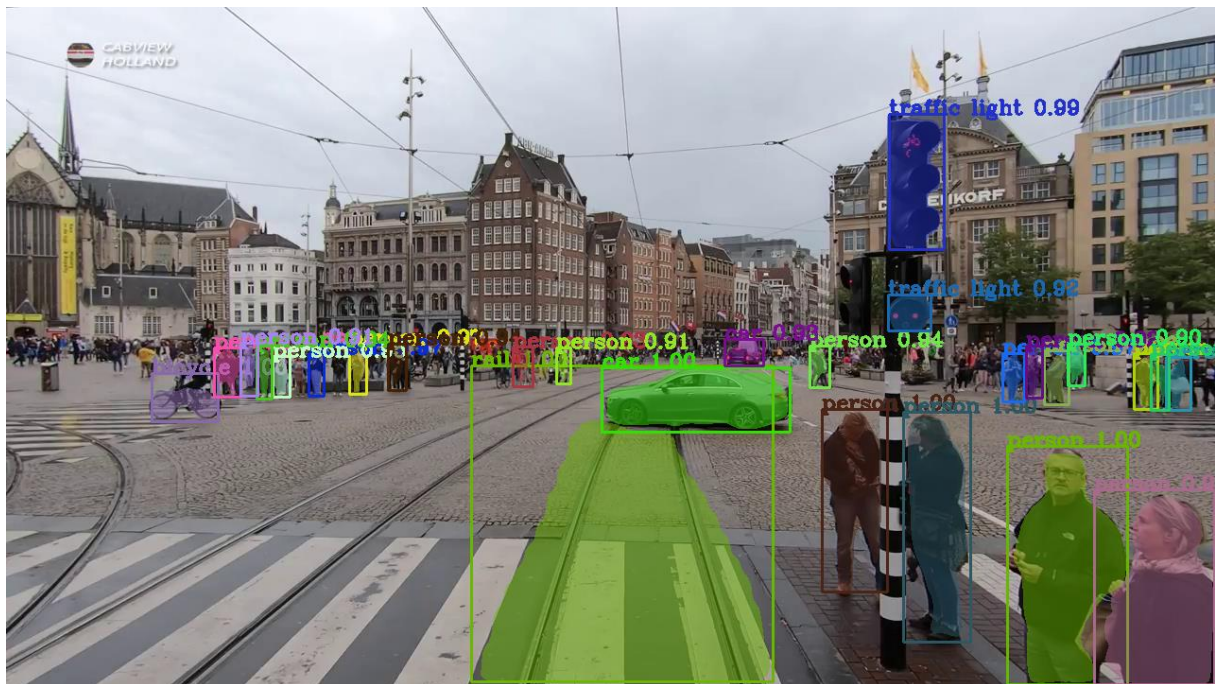


Figure 18 l'image après la détection de rail et d'objets

Cette vidéo d'un cas réel que j'ai publié sur YouTube explique beaucoup mieux mon travail :

https://youtu.be/yky_ORKX7D0

4. Conclusion et perspectives

Dans le cadre de ce projet Ter de master 1, j'ai exploré l'application de l'intelligence artificielle dans le domaine de véhicules autonomes.

Ce projet est une expérience innovante et enrichissante pour mettre en pratique mes connaissances en Deep Learning et explorer les différentes publications et articles d'algorithmes de segmentation se focalisant sur le Mask-RCNN. C'est ce que m'a amené à implémenter un algorithme qui peut détecter les obstacles devant le tramway ce qui représente la brique de la perception du tramway autonome.

Les différentes contributions de ce projet sont :

- Prise en main du modèle de segmentation Mask-RCNN.
- Préparation d'une BDD, annotation et apprentissage d'un modèle permettant de détecter les rails.
- Détections des rails devant le tramway.
- Détections des objets qui se situent devant le tramway en utilisant un modèle pré-entraîné sur la BDD COCO.
- Combinaison des deux modèles pour détecter simultanément les rails et les objets.

Pour un résultat plus précis, la continuité du travail sera de :

- Calculer la distance entre l'objet et le rail du tramway.
- Estimer le degré de dangerosité de l'objet détecté.
- Notifier le conducteur de cette estimation.

Ce projet me permettra par la suite de me spécialiser dans le domaine de Deep Learning. En outre, j'ai beaucoup apprécié le fait de planifier mon travail avec l'outil Trello et réaliser le diagramme de GANTT.

5. Bibliographie

(n.d.). Retrieved from https://github.com/matterport/Mask_RCNN.git

(s.d.). Récupéré sur Annotation Dataset:
http://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html

(s.d.). Récupéré sur Trello: <https://trello.com/>

4K Cabview Holland Dutch Railways. (2019, 08 26). Récupéré sur Amsterdam City Tour CABVIEW HOLLAND [TRAMWAY] Electriscbe Museumtramlijn Amsterdam 18aug 2019: <https://www.youtube.com/watch?v=hUUpxdqNg0M>

A. Garcia-Garcia, S. O.-E.-M.-R. (2017, 04 22). Récupéré sur <https://arxiv.org/pdf/1704.06857.pdf>

https://github.com/matterport/Mask_RCNN. (2017).

Kaiming He, G. G. (2018, janvier 24). *Mask R-CNN*. Récupéré sur Mask R-CNN: <https://arxiv.org/pdf/1703.06870.pdf>

nohabviktor. (2020, 02 26). Récupéré sur Life of a tram driver Part 4: <https://www.youtube.com/watch?v=bLcdm3dWIRc>

Nohavviktor. (2019, 11 4). Récupéré sur Life of a tram driver Part 3: https://www.youtube.com/watch?v=DnFc_KGKd4A

SHARMA, P. (2019, 07 22). Récupéré sur computer-vision-implementing-mask-r-cnn-image-segmentation:
<https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/>

6. Annexe

6.1. Annexe 1 (Noms des classes COCO)

```
class_names = [ 'BG' , 'personne' , 'vélo' , 'voiture' , 'moto' , 'avion' ,  
                'bus' , 'train' , 'camion' , 'bateau' , 'feu de circulation' ,  
                'borne d'incendie' , 'panneau d'arrêt' , 'parcmètre' , 'banc' , 'oiseau' ,  
                'chat' , 'chien' , 'cheval' , 'mouton' , 'vache' , 'éléphant' , 'ours' ,  
                'zèbre' , 'girafe' , 'sac à dos' , 'parapluie' , 'sac à main' , 'cravate' ,  
                'valise' , 'frisbee' , 'skis' , 'snowboard' , 'ballon de sport' ,  
                'cerf-volant' , 'batte de baseball' , 'gant de baseball' , 'skateboard' ,  
                'planche de surf' , 'raquette de tennis' , 'bouteille' , 'verre à vin' , 'tasse' ,  
                'fourchette' , 'couteau' , 'cuillère' , 'bol' , 'banane' , 'pomme' ,  
                «sandwich» , «orange» , «brocoli» , «carotte» , «hot dog» , «pizza» ,  
                'beignet' , 'gâteau' , 'chaise' , 'canapé' , 'plante en pot' , 'lit' ,  
                'table à manger' , 'toilettes' , 'tv' , 'ordinateur portable' , 'souris' , 'télécommande' ,  
                «clavier» , «téléphone portable» , «micro-ondes» , «four» , «grille-pain» ,  
                'évier' , 'réfrigérateur' , 'livre' , 'horloge' , 'vase' , 'ciseaux' ,  
                «ours en peluche» , «sèche-cheveux» , «brosse à dents» ]
```

Table des figures

| | |
|---|----|
| Figure 1 : Présence d'obstacle devant un tramway. | 5 |
| Figure 2 interface de Trello | 7 |
| Figure 3 Diagramme de GANNT | 7 |
| Figure 4 CNN : Réseau de Neurones convolutifs | 11 |
| Figure 5 fonctionnement de la couche de convolution | 11 |
| Figure 6 fonctionnement de la couche de pooling | 12 |
| Figure 7 fonctionnement de la couche de flattening | 13 |
| Figure 8 fonctionnement de la couche de dense | 13 |
| Figure 9 fonctionnement du Mask R-CNN | 14 |
| Figure 10 fonctionnement du modèle Fast R-CNN | 15 |
| Figure 11 fonctionnement du modèle Faster R-CNN | 15 |
| Figure 12 fonctionnement du modèle Mask R-CNN | 16 |
| Figure 13 exemple d'annotation | 19 |
| Figure 14 les paramètres d'entraînement | 20 |
| Figure 15 courbe de perte du modèle 4 | 23 |
| Figure 17 image d'entrée | 25 |
| Figure 18 la sortie du modèle de détection de rail | 25 |
| Figure 19 l'image après la détection de rail et d'objets..... | 26 |

Liste des tableaux

| | |
|---|----|
| Tableau 1 les vidéos utilisées pour construire la BDD | 17 |
| Tableau 2 La répartition des images de la BDD | 21 |
| Tableau 3 La résultat de détection de chaque modèle..... | 22 |