

DÉPARTEMENT DE GÉOMATIQUE APPLIQUÉE
FACULTÉ DES LETTRES ET SCIENCES HUMAINES
UNIVERSITÉ DE SHERBROOKE

**SEGMENTATION SÉMANTIQUE EN TEMPS RÉEL À
PARTIR D'UN NANO-ORDINATEUR : ÉTUDE DES
PERFORMANCES ET DES LIMITES**

*Essai présenté pour l'obtention du grade de Maître en sciences (M.Sc.),
cheminement géodéveloppement durable*

VINCENT LE FALHER

LONGUEUIL
10 Décembre 2021

Remerciements

Afin de mieux apprécier les remerciements que je m'apprête à faire, j'aimerais partager brièvement le cheminement atypique qui m'a amené à cet essai.

En juin 2016, j'entreprenais un changement de carrière dans le domaine de la géomatique en m'inscrivant au microprogramme du 2e cycle en sciences géomatiques à l'Université de Sherbrooke, à temps partiel. Deux années plus tard (2018), je poursuivis avec le diplôme en géomatique appliquée. Ma motivation et ma rigueur dédiées aux études ne diminuèrent pas, conciliant un travail à temps plein, deux jeunes enfants âgés de 6 et 8 ans à cette époque, et l'entretien d'une maison. Réalisant que les perspectives de carrière et les opportunités étaient plus intéressantes avec une maîtrise, surtout avec mon expérience de plus de 20 années en TI, j'ai décidé de continuer et m'inscrire à la maîtrise en sciences géographiques cheminement de type cours en géodéveloppement durable l'automne 2019.

Mon premier remerciement est destiné à la professeure et ancienne directrice du département de géomatique appliquée Lynda Bellalite. J'ai rencontré Mme Bellalite aux journées portes ouvertes au Campus de l'Université de Sherbrooke en 2016 et m'a recommandé de m'inscrire au microprogramme plutôt qu'au diplôme, l'admission étant plus simple et réaliste compte tenu de mon profil atypique et le fait que je devais faire une reconnaissance des acquis (diplômé en France). Ce fut une rencontre clé.

Je voudrais ensuite remercier la professeure Ramata Magagi, anciennement responsable du microprogramme de 2e cycle en sciences géomatiques, qui à la suite d'un échange en 2018 m'a recommandé de ne pas négliger tout ce qui touche les méthodes d'apprentissages automatiques (apprentissage profond, mégadonnées, intelligence artificielle, etc.). Ce sage conseil est la raison pour laquelle je me suis engagé dans ce projet.

J'aimerais remercier le chercheur postdoctoral Étienne Clabaut, d'avoir accepté d'être mon sous-directeur de projet alors que l'essai était déjà complété. Grâce à ses relectures, Mr Clabaut a pu partager des commentaires très perspicaces, constructifs et sincères.

Je tiens finalement à remercier le professeur Mickaël Germain, responsable des cheminements de type cours au 2e cycle, d'avoir accepté d'être mon directeur de projet et m'avoir proposé ce sujet. Je veux souligner le fait que M. Germain a fait preuve d'une grande patience durant tout le déroulement de ce projet. Il a su être disponible et me guider vers l'essentiel grâce à son expérience en matière de rédaction, mais aussi grâce à ses compétences techniques, qui m'ont été très bénéfiques. Nos échanges et nos discussions ont toujours été professionnels, respectueux, calmes et courtois.

Finalement je tiens à remercier ma famille. Durant ces 5 dernières années, leur confiance et leur soutien ont été primordiaux. Cela a été un travail d'équipe et de collaboration, cette maîtrise leur appartient autant qu'à moi.

Table des matières

Liste des figures	III
Liste des tableaux.....	IV
Liste des abréviations.....	V
1 Introduction.....	1
1.1 Mise en contexte	1
1.2 Problématique	2
1.3 Objectifs.....	4
2 Cadre théorique	5
2.1 Revue de littérature	5
2.2 Le nano-ordinateur.....	5
2.3 La segmentation sémantique.....	7
3 Matériel et méthodes.....	9
3.1 Site d'étude	9
3.2 Jeux de données et architectures	12
3.2.1 Données.....	12
3.2.2 Approche prévue pour le traitement des données	12
3.3 Matériel et logiciels	15
3.3.1 Le nano-ordinateur.....	15
3.3.2 Logiciels.....	16
3.4 Méthodologie	21
3.5 Documentation.....	22
3.6 Environnement de travail.....	23
3.6.1 Préparation du nano-ordinateur.....	23
3.6.2 Collecte des données	30
3.6.3 Mise en place des solutions logicielles	30
3.7 Évaluation	33
3.7.1 Les indicateurs de performances.....	33
3.7.2 Les résolutions	33
3.7.3 Médias.....	33
3.7.4 Stratégie de test de l'inférence	34

3.7.5	Stratégie de collecte des indicateurs de performance matérielle	35
3.7.6	Résolutions évaluées	36
3.7.7	Segmentation avec des images.....	36
3.7.8	Segmentation avec des vidéos.....	38
3.8	Réentraînement	39
3.8.1	Choix de l'architecture FCN	41
4	Résultats	42
4.1	Performances matérielles	42
4.1.1	Stockage de données	42
4.1.2	Performances système.....	42
4.2	Performances de l'inférence	53
4.2.1	Images	53
4.2.2	Vidéos	54
4.3	Réentraînement	55
5	Interprétation et discussion des résultats.....	56
5.1	Performances matérielles	56
5.1.1	Stockage de données	56
5.1.2	Performances système.....	56
5.2	Performances de la segmentation	56
5.2.1	Images	56
5.2.2	Vidéos	57
6	Conclusion et recommandations	57
6.1	Objectif principal	57
6.2	Limites	58
6.2.1	Limites matérielles.....	58
6.2.2	Limites applicatives	58
6.3	Optimisation.....	59
6.3.1	Optimisation matérielle.....	59
6.3.2	Optimisation logicielle.....	60
6.4	Documentation.....	61
	Références	62

7	Annexes.....	66
7.1	Exemples de nano-ordinateurs qui supportent les SDK pour l'IA	66
7.2	Résumé des différents tests de configuration du nano-ordinateur avec les disques SSD	
	66	
7.3	Communication avec l'Association des Piétons et Cyclistes du Pont Jacques-Cartier	67

Liste des figures

Figure 1: Segmentation semantic (Wu et al., 2019, p. 1).....	4
Figure 2: Relation entre Intelligence Artificielle, Apprentissage Machine et Apprentissage Profond (Chollet, 2018, p. 4)	8
Figure 3: Vue aérienne du pont Jacques-Cartier (flickr PJCCI)	10
Figure 4: Description de la zone géographique du site d'implémentation : le pont Jacques-Cartier et la piste multifonctionnelle en orange sur le pont (copie-écran OpenStreetMap).	11
Figure 5: Schéma de la configuration de la piste multifonctionnelle (PJCCI)	12
Figure 6: Carte mère Jetson Nano de NVIDIA, représenté avec des Lego pour démontrer sa petite	15
Figure 7: Diagramme de l'architecture du NVIDIA JetPack.....	17
Figure 8: Organigramme de la méthodologie à haut niveau.....	21
Figure 9: Organigramme des détails de la méthodologie pour évaluer les performances	21
Figure 10: Organigramme des détails de la méthodologie pour évaluer les performances après une phase de réentraînement.....	22
Figure 11: Préparation du nano-ordinateur	23
Figure 12: Montage du nano-ordinateur	24
Figure 13: Carte mère, accessoires et périphériques du nano-ordinateur. Les numéros se retrouve au Tableau 3.....	Error! Bookmark not defined.
Figure 14: Éléments pour l'évaluation des performances.....	34
Figure 15: Diagramme d'architecture de la segmentation d'une vidéo	38
Figure 16: Diagramme d'architecture de la segmentation avec la caméra	38
Figure 17: Méthodologie du réentraînement.....	41
Figure 18: Présentation des périodes d'observation des performances système	45

Figure 19 : Diagramme des performances système: la fréquence (haut et bas).....	46
Figure 20 : Diagramme des performances système: la mémoire (haut et bas)	47
Figure 21 : Diagramme des performances système: le I/O total en % de la segmentation (haut et bas).....	48
Figure 22: Diagramme des performances système: le I/O en KBytes de la segmentation (haut et bas).....	49
Figure 23: Diagramme des performances système : le I/O total du disque en KBytes (haut et bas)	50
Figure 24 : Diagramme des performances système : les températures (haut et bas)	51
Figure 25: Diagramme des performances système: la consommation (haut et bas).....	52
Figure 26: (gauche) Image originale (b1-09517); (centre) vérité terrain (GT); (droite) segmentation sémantique générée par l'architecture. Le IoU et le F1 score pour le chemin sont de +80 %.....	53
Figure 27: (gauche) Image originale (b378-61); (milieu) vérité terrain (GT); (droite) segmentation sémantique générée par l'architecture. Le IoU pour le chemin est +69 %.....	53

Liste des tableaux

Tableau 1: Tableau des données	13
Tableau 2: Solutions logicielles de l'essai	17
Tableau 3: Liste du matériel (avec prix en \$CAD).....	25
Tableau 4: Cartes microSD	26
Tableau 5: Résolutions et images par seconde (FPS) qui sont évaluées.....	36
Tableau 6: Classes et palettes de couleur.....	37
Tableau 7: Comparaison des performances du "data read" entre un SDD M.2 NVMe et une microSD	42
Tableau 8: Résolutions et images par seconde (FPS) testées	55
Tableau 9: Comparaison des trois nano-ordinateurs supportant les SDK pour l'IA	66

Liste des abréviations

AM Apprentissage Machine.

AP Apprentissage Profond.

APC-PJC Association des Piétons et Cyclistes du Pont Jacques-Cartier.

CNN Réseau de Neurones Connectés.

CPU Processeur Central.

CSV Valeur Séparée par des Virgules.

FCN Réseau Pleinement Connectés (« Fully Convolutional Network »).

FCNN Réseau de Neurones Entièrement Convolutif (« Fully Convolutional Neural Network »).

FPS Images par Seconde.

GPU Processeur Graphique.

GT Vérité Terrain.

IA Intelligence Artificielle.

IoT Internet des Objets.

IoU Intersection sur Union.

L4T Linux pour Tegra.

ONNX Échange de Réseau Neuronal Ouvert.

PJCCI Les Ponts Jacques Cartier et Champlain Incorporée.

PoE Power over Ethernet.

SDK Kit de Développement.

SSD Disque Dur.

TFLOPS teraFLOPS.

WSL Sous-système Windows pour Linux.

1 Introduction

1.1 Mise en contexte

La compagnie Les Ponts Jacques Cartier et Champlain Incorporée (PJCCI) désire évaluer la mise en service de la piste multifonctionnelle (vélos, piétons, etc.) du pont Jacques-Cartier, à Montréal, durant l'hiver. Pour ce faire, la piste doit rester sécuritaire et dégagée, malgré les évènements météorologiques.

L'Université de Sherbrooke, qui participe à cette initiative, propose de mettre en place sur le pont une plateforme de détection innovatrice qui consiste à installer plusieurs paires d'objets connectés ultralégers et performants (des nano-ordinateurs) à différents endroits du pont. Chacun de ces nano-ordinateurs possède trois différents types de capteurs : vision; son; et météorologiques (température, humidité, etc.). Chaque nano-ordinateur d'une paire perçoit le même environnement, mais d'une perspective différente que son homologue : la caméra pointe vers la même surface, mais d'un autre point de vue; les sons et les données météorologiques sont captés dans le même voisinage. Les données collectées par les capteurs sont traitées en temps réel par des algorithmes de segmentation qui sont adaptés à ce type de problématiques : les réseaux de neurones, du domaine de l'intelligence artificielle (AI). La déduction de l'état de la surface de la piste (sèche, mouillée, glacée, enneigée, etc.) se fait en fusionnant les différentes perceptions (multicibles) de chaque capteur (multicapteurs).

Cet essai se concentre sur le volet vision du projet pour PJCCI. Il s'agit de déployer rapidement, facilement, et en grande quantité (entre 25 et 50) des nano-ordinateurs tout le long de la piste multifonctionnelle du pont. Lors de la mise en service des nano-ordinateurs, leur caméra sera simplement orientée vers la piste multifonctionnelle, et le modèle IA devra détecter automatiquement dès son exécution (inférence), et d'une façon continue (opérationnalisation 24/7), les délimitations (segmentation) de la piste, sans avoir à lui fournir des paramètres ou réglages personnalisés, tels que l'angle de vue, la distance ou la hauteur. Les délimitations de la piste pourront ensuite être transmises à un autre programme installé sur le nano-ordinateur afin de détecter en temps réel ou quasi-temps réel¹, les conditions de la surface de la piste

¹ Temps réel signifie qu'il n'y a pas de délai entre le traitement de la donnée, et la communication du résultat. Quasi-temps réel signifie qu'un faible délai est permis entre l'occurrence de l'évènement et la communication du résultat.

multifonctionnelle : enneigée, mouillée, présence de glace noire, partiellement sèche, etc. Les résultats de la détection seront accessibles ou transmis via un accès à distance aux responsables de PJCCI afin qu'ils puissent prendre les décisions adéquates en matière d'entretien et d'accès.

Pour PJCCI, les avantages d'une telle plateforme seraient multiples, et on peut en énumérer plusieurs, sans se limiter à : contrôler et mesurer l'épandage de sel; surveiller à distance les conditions de la piste multifonctionnelle; éviter le déplacement d'un spécialiste; suivre les effets du gel et du dégel; optimiser les couts des opérations d'entretien (déplacements, quantité); offrir aux usagers des conditions d'accès sécurisées et optimales même en hiver; effets environnementaux atténus; prise de décision et gestion proactive; planification.

D'un autre côté, les défis ne sont pas à sous-évaluer : la détection doit être précise, fiable et consistante, tout cela afin d'assurer aux usagers un service de qualité dans un contexte sécuritaire.

1.2 Problématique

Dans le cadre du projet pour PJCCI, une plateforme technologique sera mise à la disposition des gestionnaires du pont afin de les aider à prendre les décisions les plus responsables et raisonnables possibles. Mais la mise en service d'une solution innovante et fiable, qui concilie des algorithmes d'apprentissage profond, du temps réel, des nano-ordinateurs, et des conditions climatiques variables, est complexe. Dans une certaine mesure, l'essai va contribuer à la recherche de solutions afin de répondre au défi pour le domaine du transport actif et durable d'être soutenu par des solutions technologiques fiables (opérationnelles), l'objectif étant de pouvoir offrir des services de qualité et sécuritaires sur l'ensemble des quatre saisons.

La paramétrisation (des "hyperparamètres") des réseaux de neurones est subtile et intuitive, et nécessite de l'expérience. C'est un processus d'essais-erreurs qui est couteux en temps, et risqué puisqu'il n'y a aucune garantie de succès. La technique d'apprentissage par transfert (*Transfer Learning* en anglais) permet d'hériter d'une architecture qui est déjà entraînée et paramétrée, et de l'adapter à d'autres problématiques, en lui fournissant un plus petit jeu d'images (une centaine) de la nouvelle zone d'étude. Cette technique permet un gain en temps puisque la phase de conception (analyse, architecture, configuration) est raccourcie de façon importante. La problématique pour l'essai est de trouver l'architecture qui est la plus adaptée pour répondre au besoin, et il en existe des milliers (Koh, 2018). La recherche dans la littérature permet heureusement de limiter les choix et donner des pistes (Nguyen et al., 2019; NVIDIA, 2019b; Zheng et al., 2020).

Même si les scores sont satisfaisants lors de la phase de test du modèle, la réalité du terrain peut surprendre. Les tests d'acceptation du modèle doivent se faire en dehors de l'environnement d'entraînement (laboratoire), dans les conditions réelles (luminosité, angle, hauteur, etc.) sur le terrain d'implémentation. Dans le jargon de l'intelligence artificielle et des réseaux de neurones, c'est l'inférence² (Copel, 2016; NVIDIA, 2019b). De plus, le système hôte, dans notre cas le nano-ordinateur NVIDIA Jetson Nano, est conçu avec une architecture matérielle limitée (GPU, CPUs, mémoire, taux de transfert, alimentation).

La segmentation sémantique (Figure 1) est une forme de classification d'image, pixel par pixel, qui tire profit des dernières évolutions de la classification supervisée grâce aux réseaux de neurones pleinement connectés (FCN), et qui peut être réalisée en temps réel avec des nano-ordinateurs (Blanco-Filgueira et al., 2019; Long et al., 2015). Les images doivent être de haute résolution, ce qui nécessite d'avoir à disposition un système informatique capable de fournir une puissance de calcul appropriée, particulièrement pour la manipulation de la mémoire et des nombres flottants pendant l'inférence (Mody et al., 2018). Leur application par des nano-ordinateurs est un défi en raison de la faible consommation d'énergie (Watts) et de la puissance de calcul limité de ces derniers (Copel, 2016).

² Le terme "inférence" est utilisé lorsqu'un modèle, entraîné avec un échantillon de la population, est appliqué pour pouvoir donner une conclusion pour d'autres échantillons de la population (déduire un chien ou un chat sur une image). Le terme "prédiction" est utilisé lorsqu'un modèle, entraîné avec un échantillon de la population, est appliqué pour déduire une valeur selon une ou plusieurs variables (la température selon la localisation et l'altitude).

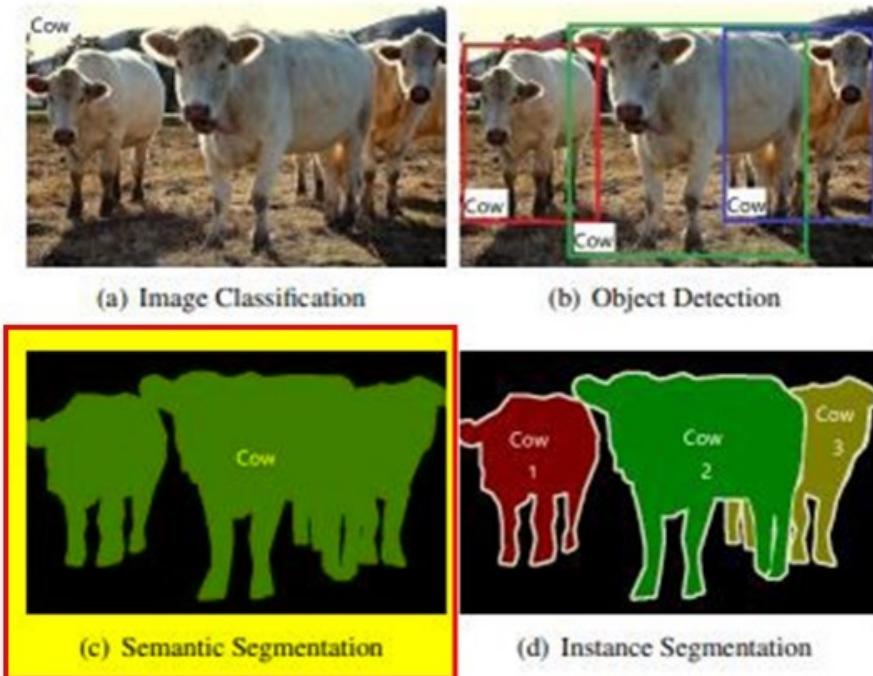


Figure 1: Segmentation semantic (Wu et al., 2019, p. 1)

Il existe différents cadres applicatifs pour l’entraînement de modèles IA, tels que *PyTorch* ou *TensorFlow*. L’inconvénient est d’avoir à installer pour chacun leur propre environnement de développement et d’inférence, ce qui augmente les efforts et les coûts. Le cadre applicatif ONNX a été conçu pour pallier cette contrainte. En effet, il uniformise les architectures des modèles, et simplifie la mise en service grâce à l’installation d’un unique cadre applicatif. NVIDIA fournit avec le Jetson Nano une plateforme applicative qui supporte les modèles convertis au format ONNX, et offre donc une solution supportant l’interopérabilité des modèles IA.

1.3 Objectifs

L’objectif principal de cet essai consiste à étudier la capacité du nano-ordinateur du fabricant NVIDIA, le Jetson Nano (NVIDIA, 2019a), à exécuter, en temps réel, une architecture de réseau de neurones entièrement convolutif (FCNN) entraînée à faire de la segmentation sémantique d’images et de vidéos de haute résolutions.

Le premier objectif spécifique est de déterminer quelles sont les limites de la plateforme, d’un point de vue matériel (GPU, CPUs, mémoire, transfert mémoire, consommation, etc.), mais aussi applicatif, d’un point de vue de l’inférence

Le second objectif spécifique est d'optimiser ou d'adapter la plateforme, d'un point de vue matériel, mais aussi applicatif, afin d'atteindre les meilleures performances et résultats possibles pendant l'inférence.

2 Cadre théorique

2.1 Revue de littérature³

La recherche de références s'est concentrée autour des concepts du sujet de l'essai : la segmentation sémantique, le temps réel, et les nano-ordinateurs. Le premier objectif a été de trouver si des études avaient déjà expérimenté le nano-ordinateur, en particulier pour la segmentation de vidéos en temps réel. Pendant cette recherche, j'en ai profité pour effectuer une révision de l'évolution des réseaux de neurones convolutifs (CNN - Convolutional Neural Network) et des différentes architectures, et chercher d'autres solutions de détection de la route en temps réel grâce au réseau pleinement connectés (FCN - Fully Convolutional Network).

Je me suis intéressé aux références des années les plus récentes, autour de 2020, 2019 et 2018, car les avancées dans le domaine des réseaux de neurones sont rapides. Par curiosité je suis allé aussi parfois voir dans les années bien plus éloignées, comme 1998, où j'ai trouvé un article proposant une solution pour prédire la température de la surface de la route avec des réseaux de neurones. Je n'ai pas pu trouver de références spécifiquement pour la déduction de l'état de la surface (mouillé, gelée, etc.) d'une piste multifonctionnelle (vélo, piéton).

2.2 Le nano-ordinateur

Les nano-ordinateurs et les objets connectés, désignés aussi par l'Internet des Objets ou *IoT*, (Blanco-Filgueira et al., 2019; Sharma et al., 2019) sont le résultat de la miniaturisation des systèmes informatiques. Ils permettent la détection en temps réel à des endroits, dans des situations et dans des conditions qui n'étaient pas envisageables il y a encore 10 ans (Abouzahir et al., 2017; Bernas et al., 2017; Blanco-Filgueira et al., 2019; Zheng et al., 2020). Ils sont utilisés comme système embarqué pour servir les applications dans de nombreux domaines, tel que l'espace et les satellites de télédétection (Xie, 2021) avec la conception de la communication visuelle du paysage

³ La revue de la littérature a débuté en octobre-novembre 2019, c'est-à-dire quelques mois après la disponibilité du nano-ordinateur (juin 2019).

urbain côtier, l'agriculture de précision (Dubey, 2020) avec la détection et l'identification des maladies des plantes agricoles, le développement durable et les villes intelligentes (Catarinucci, 2020) avec un système de gestion des déchets basé sur des services nuagiques et des étiquettes de capteur RFID ultra-basse consommation, et bien d'autres encore, comme les 110 travaux qu'a pu relever Rodriguez-Conde (2021) : détection de véhicules, de piétons et de panneaux dans les moyens de transport intelligent, la navigation dans les chirurgies ouvertes et la détection de blessures et de maladies en médecine, la détection d'obstacles dans un dirigeable sans pilote en défense militaire.

Le nano-ordinateur de cet essai doit être compris comme étant un ordinateur miniature, ayant une taille et des capacités qui lui permettent d'être installé (*embedded system*) dans une voiture, un drone, un tracteur ou être accroché à un poteau. Le terme anglais *On the Edge* (sur le bord), s'y approprie mieux que *IoT*, puisqu'étant sur le terrain il se trouve directement proche des données, ce qui lui donne l'avantage de pouvoir faire des traitements en temps réel. Les premiers systèmes embarqués reconnus comme tels sont ceux installés dans le missile Minuteman (Kilby, 2000) et la navette Apollo (Kilby, 2000). Les avancées technologiques ont permis de les rendre de plus en plus compacts et performants. Les systèmes de la compagnie Campbell Scientific existent depuis les années 1974 et permettent l'acquisition de données à distance. Le système Arduino (Arduino, 2021) est l'un des premiers microprocesseurs à avoir été destinés à la robotique. Le Jetson Nano de NVIDIA (NVIDIA, 2021a) est le dernier né des nano-ordinateurs de la compagnie NVIDIA permettant d'inférer en temps réel des architectures d'intelligence artificielle, sans ajout de périphériques. Du même constructeur, ses grands frères sont le Jetson Xavier (NVIDIA, 2021c) et le Jetson TX2 (NVIDIA, 2021b), plus performants, et donc plus onéreux. Son concurrent direct est le Raspberry Pi (Raspberry Pi Foundation, 2021), mais il nécessite une extension Neural Compute Stick 2 (NCS2) (Intel, 2021) pour l'inférence de modèles IA.

Ce qui caractérise principalement un ordinateur miniature, est le fait qu'il soit assez petit pour pouvoir être embarqué dans un système plus gros, tel qu'un robot ou un drone. Son coût est bas (

Tableau 3), en raison des performances qui sont limitées par une conception répondant à un besoin spécifique. Tous les éléments matériels requis sont contenus sur une même carte. Une fois installé et paramétré, le système se doit d'être fiable et opérationnel sur le long terme. Mais il doit aussi être interchangeable, au besoin, rapidement et facilement. La consommation électrique est faible, entre 5 et 10 W. Étant généralement opérationnel sur le terrain, proche des données, il est responsable d'une tâche bien particulière, qu'il doit accomplir efficacement. Il n'y a généralement pas d'interface utilisateur, et l'accès au système se fait à distance ou via une console. Il est composé de capteurs, au besoin d'une caméra. Le même système peut être déployé en grande quantité, comme dans le contexte de notre essai, où plusieurs paires seront déployées le long de la piste multifonctionnelle; un autre exemple est celui des constellations de nano satellites.

L'annexe 7.1 montre les nano-ordinateurs qui supportent les SDK (Kit de Développement) pour l'IA.

2.3 La segmentation sémantique

L'apprentissage profond est un sous-domaine de celui de l'apprentissage machine qui est lui-même un sous-domaine de celui de l'intelligence artificielle (Figure 2).

Les concepts de l'intelligence artificielle existent depuis les années 1950 (Alom et al., 2018; Chollet, 2018), et ont continué à se développer par vague, jusqu'à leur nouvelle popularité des 15 dernières années. En effet, trois raisons principales ont permis à ce domaine de renaître de nouveau (Chollet, 2018, p. 20) : 1) la capacité et la puissance des machines; 2) des jeux de données plus larges; 3) des algorithmes plus avancés. Les deux moments clés, preuves de cette renaissance, sont : 1) la possibilité d'entrainer des architectures de réseaux de neurones profonds (DNN) (2006) (Alom et al., 2018, p. 6); et 2) l'architecture du réseau de neurones convolutionnels *AlexNet* permet de gagner le challenge *ImageNet* contre les approches traditionnelles (Alom et al., 2018, p. 11).

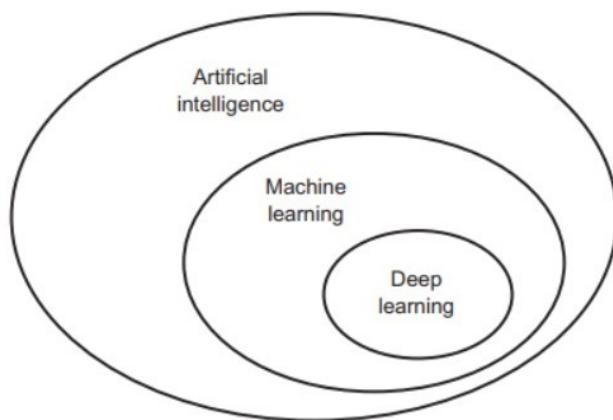


Figure 2: Relation entre Intelligence Artificielle, Apprentissage Machine et Apprentissage Profond (Chollet, 2018, p. 4)

La détection d’objets en temps réel est de plus en plus précise et efficace depuis que les performances des systèmes informatisés permettent l’exécution d’algorithmes exigeants, en majeure partie depuis l’utilisation des processeurs graphiques (GPU) (Beam, 2017; Chong et al., 1992; Dettmers, 2015; Jiaconda, 2019; Kurenkov, 2015; Zheng et al., 2020).

Les réseaux de neurones ont rapidement progressé depuis 2012 (Beam, 2017), permettant d’offrir des alternatives aux solutions de détection et de classifications telles que les algorithmes *SIFT* et *HOG* (Pathak et El-Sharkawy, 2019). Les FCN sont les derniers à avoir émergé et représente l’état de l’art (Zheng et al., 2020) et à profiter au domaine de la vision et de la détection d’objets (Nguyen et al., 2019; Zheng et al., 2020). En 2016 (Alom et al., 2018, p. 14), l’architecture FCN a permis aux tâches réservées à la segmentation d’images d’être plus efficace que les méthodes traditionnelles de la vision par ordinateur. Cette nouvelle méthode s’applique désormais à tous les domaines connexes à l’analyse d’images, tels que l’imagerie médicale, la conduite autonome de véhicules, la robotique, la télédétection d’images satellites, la sécurité par caméra vidéo, l’agriculture de haute précision, etc.. Aujourd’hui, elle peut s’exécuter en temps réel sur des systèmes embarqués proche des données. Des exemples de modèles populaires pour la segmentation sémantique au moment d’écrire ce rapport sont les séries *ResNet* (*ResNet18*, *ResNet50*, *ResNet101*) (He, 2015), qui sont utilisés comme fondation pour construire des modèles plus évolués comme *DeepLabV3* (*TensorFlow*, 2020).

La segmentation sémantique d’images ou de vidéos (figure 1) est une technique de télédétection du domaine de la vision par ordinateur. Elle permet de délimiter (segmenter) différentes parties (sémantique) d’une image. Les méthodes de segmentation ont été améliorées ces dernières années par les récentes avancées dans le domaine de l’apprentissage profond.

Les librairies d’apprentissage profond les plus courants sont identifiées par (Cornioley, 2018). Les plus populaires à ce jour sont *PyTorch*, *TensorFlow* et *Keras* ; elles sont accessibles via le langage de programmation *Python*. *Keras* est une solution intéressante, car elle ajoute une couche d’abstraction à d’autres (*PyTorch*, *TensorFlow* et *Coffee*), et donc est précurseur dans ce domaine

où la simplification et l'accessibilité de la programmation sont recherchées. Une liste plus exhaustive est fournie par le projet communautaire ONNX (Échange de Réseau Neuronal Ouvert)⁴.

ONNX est un projet communautaire qui met à disposition une plateforme applicative permettant de rendre interopérable, pour l'inférence, des architectures de réseaux de neurones conçues avec différentes plateformes applicatives d'apprentissage machine, telles que *PyTorch* et *TensorFlow*. Initiée par Facebook en 2017⁵ et soutenue par l'ensemble des acteurs du domaine (IBM, AWS, Microsoft, NVIDIA, Intel, etc.)⁶, elle est implémentée par NVIDIA dans la solution applicative du Jetson Nano pour l'inférence, et supporte donc des modèles personnalisés, tant qu'ils peuvent être convertis au format ONNX, peu importe la plateforme avec laquelle ils ont été conçus. Peu de références dans la littérature y font référence à ce jour⁷.

3 Matériel et méthodes

3.1 Site d'étude

Le site d'étude se situe dans la ville de Montréal, dans la province du Québec, au Canada, aux coordonnées 45° 31' 18" N, 73° 32' 31" O. Montréal est une île qui est séparée la Rive-Sud par le fleuve Saint-Laurent et de sa rive nord par la rivière des Prairies.

Tel que trouvé dans le rapport détaillé sur le projet pilote d'entretien hivernal de la piste multifonctionnelle du pont Jacques-Cartier (PJCCI, 2018b), la piste multifonctionnelle du pont Jacques-Cartier (Figure 3) relie Montréal intramuros, proche de la station de métro De Lorimier / René Lévesque, et la Rive-Sud, à Longueuil, proche de la station de métro Longueuil et de son terminal de bus. Elle est longue d'une distance de 2,7 km et est située d'un seul côté du pont, côté sud (Figure 4). Elle est surtout utilisée par les cyclistes, et moindrement par les passants (PJCCI, 2018b). Sa configuration est bien particulière (PJCCI, 2018a) (Figure 5) : elle ne longe pas la route adjacente sur toute sa longueur; elle est interrompue par une voie de sortie de l'île Notre-Dame; des chicanes sont disposées à certains endroits; sa largeur varie entre 2,5 m et 1,8 m; elle possède une pente assez prononcée à certains moments; il y a des courbes assez serrées.

⁴ <https://onnx.ai/supported-tools>

⁵ https://en.wikipedia.org/wiki/Open_Neural_Network_Exchange

⁶ <https://onnx.ai/about.html>

⁷ 19 résultats dans SCOPUS le 8 mai 2021

Elle est fermée l'hiver par mesure de sécurité. Elle est ouverte au début du printemps jusqu'au début de l'hiver, lorsque les conditions ne nécessitent pas d'entretien.



Figure 3: Vue aérienne du pont Jacques-Cartier (flickr PJCCI)⁸

⁸ <https://www.flickr.com/photos/pjcci/6830109134>

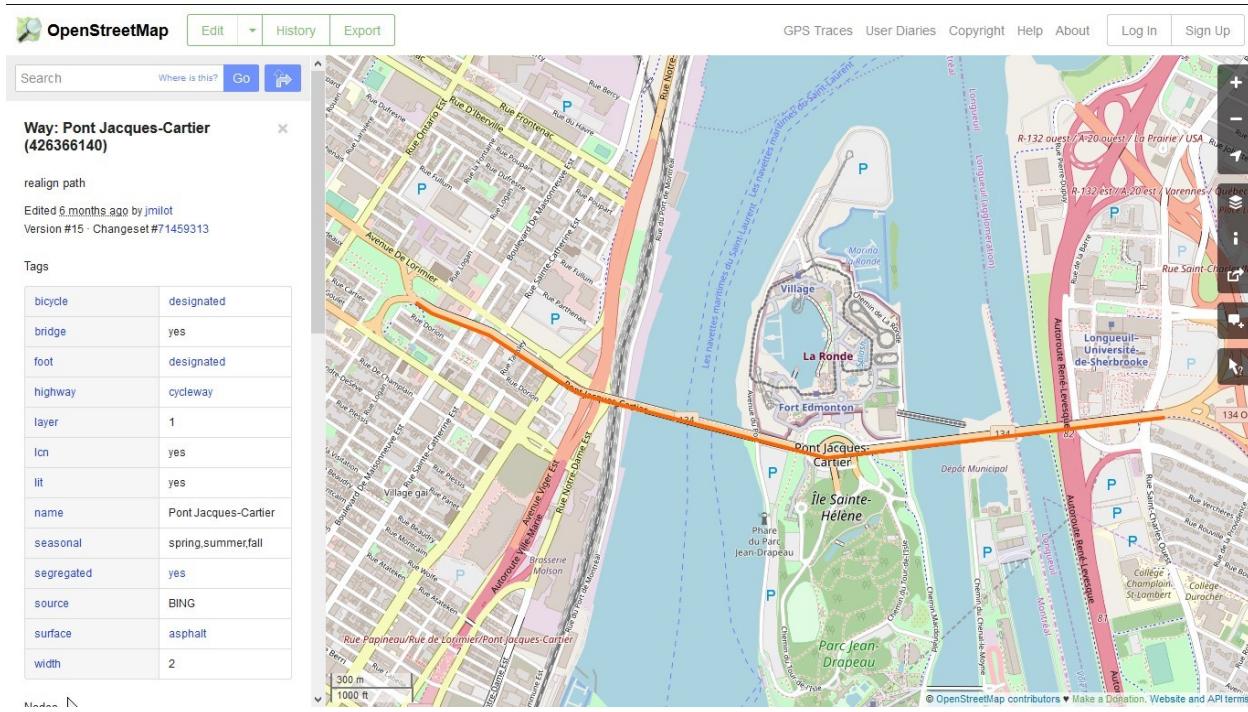


Figure 4: Description de la zone géographique du site d'implémentation : le pont Jacques-Cartier et la piste multifonctionnelle en orange sur le pont (copie-écran OpenStreetMap).

Piste multifonctionnelle du pont Jacques-Cartier

La piste multifonctionnelle du pont Jacques-Cartier est un lien singulier aux défis uniques



Pont urbain à la géométrie atypique et exposé à des conditions météorologiques particulières.

Particularités de la piste et enjeux de sécurité

La combinaison de l'ensemble de ces éléments soulève des enjeux de sécurité accrus en hiver.

Géométrie atypique	Piste surélevée	Complexité des conditions météorologiques du fleuve Saint-Laurent
<ul style="list-style-type: none"> Longueur (2,7 km) Dégagement latéral étroit (de 2,5 m à 1,8 m en hiver) Normes du MTMDET : 3,5 m Pentes longues et abruptes (4,2 %) Virages serrés Enclavement (garde-corps de chaque côté) 	<ul style="list-style-type: none"> Dalle de béton (15 cm) Pas de matériaux en profondeur qui l'isolent et atténuent son refroidissement Plus affectée par les variations météorologiques qu'une piste sur remblai <p>Danger de chute de glace</p> <ul style="list-style-type: none"> Structures métalliques en hauteur (sections 3 et 7) Neige projetée des voies de circulation 	<ul style="list-style-type: none"> L'environnement climatique au-dessus du fleuve Saint-Laurent est unique, rigoureux et changeant Milieu très humide, venteux et changeant propice à la formation imprévisible de glace noire qui affecte directement la qualité de la surface de roulement

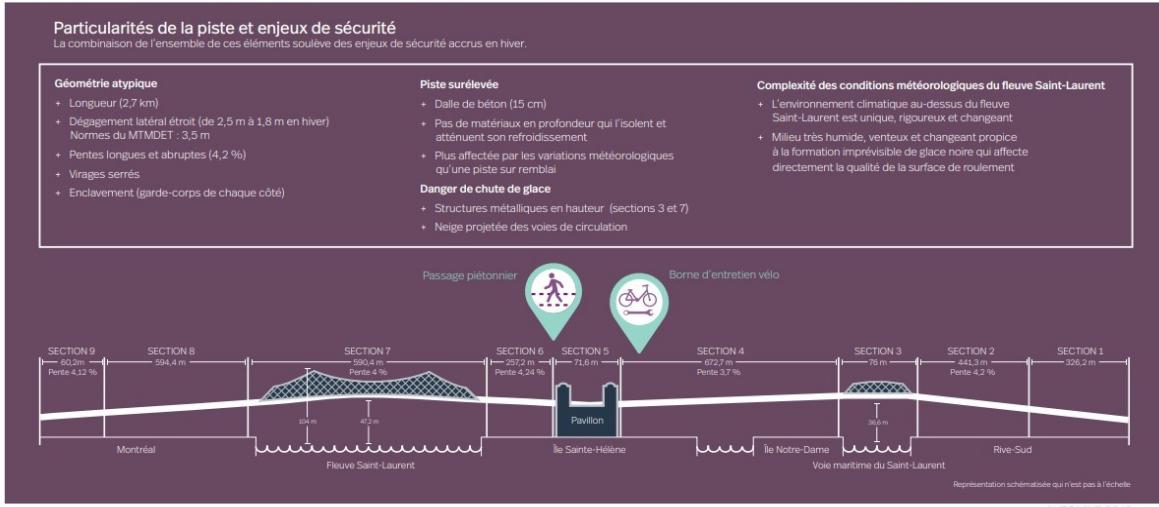


Figure 5: Schéma de la configuration de la piste multifonctionnelle (PJCCI)⁹

3.2 Jeux de données et architectures

3.2.1 Données

Les ressources mises à disposition par le constructeur du Jetson Nano, NVIDIA, font référence à des jeux de données qui sont disponibles publiquement.

En complément des ressources de NVIDIA, deux références scientifiques ont été étudiées, car leurs recherches ont été faites avec le Jetson Nano (Nguyen et al., 2019) et (Zheng et al., 2020). Beaucoup de références ont été publiées ces deux dernières années sur le sujet de la segmentation sémantique, ils existent donc de multiples alternatives inspirantes.

Il existe sur Internet des forums et des blogues dans lesquels des utilisateurs publient leurs expérimentations de la segmentation sémantique en temps réel avec le Jetson Nano (Dustin, 2019), ou plus génériquement la segmentation sémantique; des sites comme *ModelZoo* (*ModelZoo*, 2020) ou *Kaggle* (*Kaggle*, 2020) sont des entrepôts de données et d'architectures FCN prêts à être utilisés; une autre option a été d'effectuer une recherche d'images ou de vidéos de la piste multifonctionnelle du pont Jacques-Cartier via les sites de recherche tels que Google.

L'Association des Piétons et Cyclistes du Pont Jacques-Cartier (APC-PJC) existe depuis de nombreuses années pour promouvoir le transport actif et conserver la piste multifonctionnelle du pont Jacques Cartier ouverte durant l'hiver. Ils fournissent, via leurs sites Internet, des collections de vidéos et d'images qui pourront être utilisées après leur avoir demandé leur autorisation verbale et écrite (Des piétons et cyclistes du pont Jacques-Cartier, 2020).

3.2.2 Approche prévue pour le traitement des données

Il y a deux phases à cet essai : 1) l'inférence avec des modèles déjà disponibles; et 2) l'inférence avec des architectures réentraînées. Les données utilisées pour l'inférence sont des vidéos, et celles pour l'entraînement sont des images. Dans les deux cas, les images pour l'entraînement ou l'inférence doivent être d'une taille bien précise, celles avec lesquelles l'architecture a été, ou sera, entraînées. La résolution et la qualité de l'image vidéo sont nivélées vers le bas afin de déterminer

⁹ https://jacquescartierchamplain.ca/wp-content/uploads/2018/10/IMG_Fiche_piste-multi_pont_JC_FR_vfinale_web_2018-10-10.pdf

la limite inférieure acceptable pour la segmentation la plus efficace et fiable possible. La résolution et le nombre d'images par seconde de la vidéo sont contrôlés par le logiciel ("driver" en anglais) de la caméra, et sont configurables.

- Les vidéos ou les nouvelles images doivent être traitées pour répondre à une certaine taille et résolution requise par l'architecture, tout en conservant une qualité élevée (nombre de pixels, niveaux de couleurs). De nouvelles images pour l'entraînement sont extraites des vidéos, et annotées. Les bibliothèques (par exemple *Keras*, *Pytorch*) offrent l'option d'augmenter automatiquement le jeu de données avec des techniques d'augmentation de données (par exemple la rotation, le redimensionnement, l'effet miroir), ce qui est utile et non négligeable.

Le Tableau 1 synthétise les jeux de données qui ont été découverts pour le cadre de l'essai, incluant la référence à l'architecture du modèle d'apprentissage profond. Ce tableau est complémentaire à celui déjà proposé par NVIDIA¹⁰.

Tableau 1: Tableau des données

Jeu de données	Images / Vidéos	Résolution/s	Architecture	Plateforme	
CamVid	10 minutes	HD	SegNet	Caffe	
		<p><i>SegNet</i> est un réseau qui a été créé pour la segmentation sémantique de vidéos. Il a été entraîné avec le jeu de données de <i>CamVid</i>, qui fournissent des vidéos de la route avec la même perspective que le conducteur du véhicule. Une architecture entraînée est disponible pour le Jetson Nano.</p> <p>Références :</p> <ul style="list-style-type: none"> • https://github.com/alexgkendall/SegNet-Tutorial • https://github.com/PengKiKi/camvid 			
Cityscapes	25 000	512x1024	MFANet		
		<p><i>MFANet</i> (Zheng et al., 2020) est un réseau qui a été créé en 2019 pour la segmentation sémantique sur des appareils tels que le Jetson Nano. Il a été entraîné avec le jeu de données de <i>Cityscapes</i>, qui fournissent des images de scènes urbaines. <i>Cityscapes</i> est un jeu de données qui fournit des images de rues spécifiquement destinées pour la segmentation sémantique. Il peut être utilisé par de nombreux réseaux. Différentes stratégies d'augmentation de données sont utilisées. Des tests ont été faits avec le Jetson Nano.</p>			

¹⁰ <https://github.com/dusty-nv/jetson-inference#semantic-segmentation>

	Contact: leejy@ustb.edu.cn			
	MAVNet (Nguyen et al., 2019) a été entrainé avec deux jeux de données pour être utilisé par le Jetson Nano monté sur des "micro Aerial Vehicles (MAVs)". Référence : https://github.com/tynguyen/MAVNet			
FreiburgForest	15 000	576x320 864x480	AdapNet	TensorFlow
	<i>DeepScene</i> (Valada, 2016) propose plusieurs modèles entraînés avec différents jeux de données, comme <i>Cityscapes</i> , <i>SUN-RGBD</i> , <i>Synthia</i> . Le jeu de données <i>FreiburgForest</i> fournit des images de chemin dans la forêt, qui est destinée pour la segmentation sémantique. L'architecture <i>AdapNet</i> testée durant cet essai a été entraînée avec ce jeu et est disponible en deux résolutions pour le Jetson Nano. Référence : http://deepscene.cs.uni-freiburg.de			
Synthia				
	Les jeux de données <i>Synthia</i> sont composés d'images et vidéos de scènes de rue comme celui de <i>Cityscapes</i> , et qui sont destinés à la segmentation sémantique. Référence : https://synthia-dataset.net			
APC-PJC	313			
	L'Association des Piétons et Cyclistes du Pont Jacques-Cartier a une collection d'images et de vidéos de la piste multifonctionnelle du pont Jacques-Cartier. Ce n'est pas un jeu de données qui est prêt à être utilisé pour l'apprentissage tel quel, il doit être préparé. Mais c'est une source de données qui est importante pour l'essai. Il est envisagé de contacter l'association au besoin afin de leur demander leur collaboration pour la collecte d'autres d'images ou vidéos. Références : <ul style="list-style-type: none">• https://www.flickr.com/photos/pontjacquescartier• http://pontjacquescartier365.com/videos-pont-jacques-cartier			
Images et vidéo sur Internet	Entre 30 et 50			
	Internet est une source de données non négligeable en termes de données. Quelques images et vidéos de la piste multifonctionnelle du pont Jacques-Cartier, autre que celles fournies par L'Association des Piétons et Cyclistes du Pont Jacques-Cartier, sont disponibles. Ce n'est pas un jeu de données qui est prêt à être utilisé pour l'apprentissage tel quel, il doit être préparé. Mais c'est une source de données qui est importante pour l'essai. Référence : https://google.ca			

KITI Road/Lane Detection	289 + 290 images				
	Ce jeu de données contient 289 images d'entraînement et 290 images de tests d'image de routes urbaines. Il existe une grande multitude d'architectures qui sont entraînées avec ce jeu de données. Référence : http://www.cvlibs.net/datasets/kitti/eval_road.php				
Personnel	188 images de 1080x1920 19 vidéos de 30-60 secondes de 1080x1920 et 60FPS				
	Ce jeu de données contient des vidéos de différentes sections de pistes cyclables de mon quartier. Les vidéos ont été prises au mois de mars, dans des conditions ensoleillées, mais avec des endroits de la piste ombragée, sèche ou mouillée, bordée d'herbe ou de neige, parfois avec des passants. 188 images ont été extraites des vidéos.				

3.3 Matériel et logiciels

3.3.1 Le nano-ordinateur

L'objet d'étude de cet essai est le nano-ordinateur Jetson Nano du fabricant NVIDIA (Figure 6). Ce modèle a été choisi, car il a été conçu par la compagnie NVIDIA spécifiquement pour répondre au besoin d'inférence en temps réel sur le terrain, afin d'éviter le transfert de données et le traitement à distance et différé.

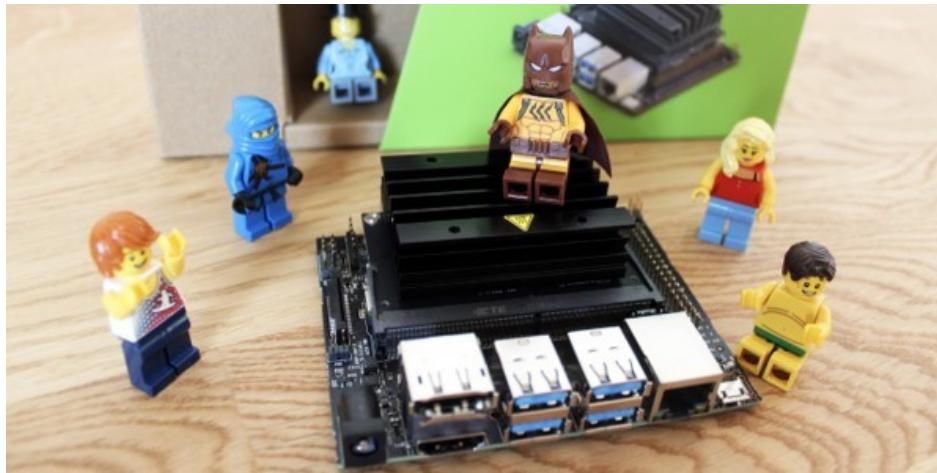


Figure 6: Carte mère Jetson Nano de NVIDIA, représenté avec des Lego pour démontrer sa petite

L'architecture du nano-ordinateur est ARM 64 bits (aarch64), ce qui le limite pour certaines portabilités de bibliothèques, surtout dans le domaine assez restreint de la recherche, ou l'architecture

la plus populaire et portable est x86-64. Il est composé d'un quad-core ARM Cortex-A57 @ 1,43 GHz, qui est conçu pour ce genre de nano-ordinateur, comme le Raspberry Pi. Les performances GPU du Maxwell sont de 128-cores @ 921 MHz, 0.5 TFLOPS (16 FP = 16 bits FP = 2 bytes Floating Points). Par comparaison la PlayStation 4 Pro (2016) supporte +4 TFLOPS. La mémoire est limitée à 4Gb LPDDR4 @ 1.6 GHz. Les autres caractéristiques à considérer sont le port pour une carte microSD, un port Ethernet 10/100/1000Mbs, un port HDMI, un hub USB 4 ports 3.0, un connecteur pour une caméra, et un port PCIe. Le tout tient sur une carte mère d'une taille de 69,6 mm x 45 mm, et consomme entre 5 et 10 W.

3.3.2 Logiciels

De même que pour les périphériques, les solutions logicielles principales qui sont utilisées dans le cadre de l'essai sont résumées dans le tableau suivant (Tableau 2), où il est indiqué leur nom, le type de licence, leur version, leurs rôles et responsabilités, comme pour le système d'exploitation, l'environnement de développement pour l'apprentissage profond, l'inférence, les logiciels de traitements de vidéos et d'images.

Pour tester les performances de la microSD et du disque SDD interne M.2 NVMe, l'utilitaire *hdparm* a été utilisé.

Le SDK qui est utilisé avec le nano-ordinateur est celui fourni par NVIDIA et qui se nomme *JetPack*^{11 12} (Figure 7). La version 4.4¹³ est celle avec laquelle les tests de performance ont été exécutés. Il contient le système d'exploitation *Linux pour Tegra* (L4T)¹⁴ (version L4T 32.4.3), qui est une version de la distribution Linux Ubuntu 18.04 mise à la saveur de NVIDIA. *Jetpack* contient aussi d'autres bibliothèques qui sont nécessaires pour re construire la version ONNX du modèle, tel que *CUDA*, *CuDNN* et *TensorRT*.

¹¹ <https://developer.nvidia.com/embedded/jetpack>

¹² <https://docs.nvidia.com/jetson/jetpack/introduction/index.html>

¹³ <https://developer.nvidia.com/embedded/jetpack-archive>

¹⁴ <https://developer.nvidia.com/embedded/linux-tegra>

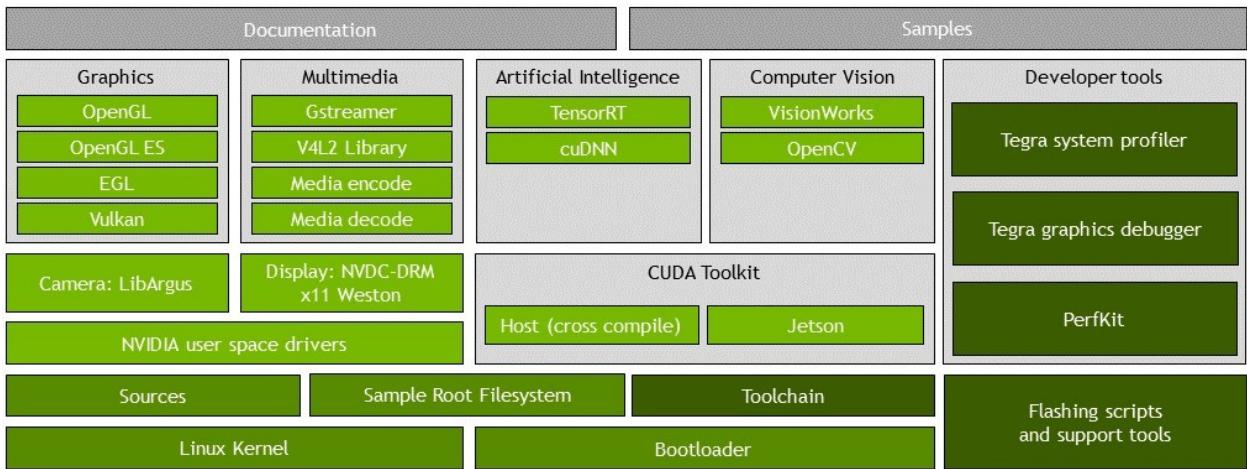


Figure 7: Diagramme de l'architecture du NVIDIA JetPackⁱ

Python et *C++* sont les langages utilisés par la plateforme applicative de *Deep Learning* de NVIDIA. Python est utilisé comme langage accessible et appelle les extensions écrites en *C++* et qui optimisent les accès aux ressources systèmes tels que les CPUs et GPUs, les traitements des images et des vidéos, les boucles et les traitements mémoires intensifs.

La librairie d'apprentissage profond qui est utilisée est *PyTorch*, bonifié avec une version adaptée par NVIDIA de *torchvision*, qui fournit des architectures et des utilitaires pour la vision par ordinateur. Des versions bien spécifiques sont nécessaires et il est important de s'y conformer au risque de tomber dans une investigation bien couteuse en temps et en énergie¹⁵.

Le nano-ordinateur inclut un GPU qui est mis à contribution lors de l'inférence. Le compilateur de NVIDIA pour GPU *CUDA* est nécessaire pour régénérer la version ONNX. La version doit concorder avec la bonne version de *PyTorch*. La version adaptée (fork) de *torchvision* doit être recompilée avec la bonne version de *Pytorch* et *CUDA*.

Enfin pour régénérer la version ONNX lors de la phase de réentrainement, les librairies *TensorRT* et *ONNX* ont été utilisées, en compagnie de l'utilitaire *trtexec* qui permet de valider et tester le fichier ONNX généré.

Tableau 2: Solutions logicielles de l'essai

¹⁵ <https://forums.developer.nvidia.com/t/trying-to-regenerate-onnx-for-jetson-nano/125494?u=vincelf>

Language	Version	Licence	Rôles et responsabilités
JetPack	4.4	NVIDIA	Kit de développement de logiciels incluant le système d'exploitation L4T, et les librairies et utilitaires nécessaires pour l'inférence avec le nano-ordinateur.
L4T	32.4.3	NVIDIA	Le système d'exploitation "Linux pour Tegra" conçu par NVIDIA pour leurs solutions d'inférence légères, comme pour le nano-ordinateur.
Python	2.7	GPL	Language plus accessible que le C++.
C++ (gcc)	7.3.1 ¹⁶	GPL	Certaines extensions du cadre applicatif de NVIDIA pour l'inférence sont écrites en C++, pour des raisons d'optimisation.
pytorch	1.1.0	BSD 3-Clause	Cadre de développement d'application pour l'apprentissage machine et profond.
torchvision	0.0.3	BSD 3-Clause	Branche de torchvision adaptée par NVIDIA ¹⁷ ; Doit être recompilée avec la version de pytorch 1.1.0 et cuda 10.0.
cudaCUDA	10.0	NVIDIA	Compilateur de code C++ pour GPU.
TensorRT	6.0.1.5	NVIDIA	SDK pour générer des modèles au format ONNX, optimisés et interopérables, pour l'inférence.
ONNX	1.7.0	MIT	Librairie qui permet de générer un format interopérable pour l'inférence de modèles d'architecture construits avec différente plateforme applicative d'apprentissage machine (Caffe, PyTorch, TensorFlow, etc.).
trtexec	-	NVIDIA	Utilitaire qui a permis de tester la version ONNX qui a été régénérée.
gstreamer	1.14.5	LGPL	Utilitaire qui a permis d'alimenter l'architecture de la segmentation avec la vidéo.

¹⁶ <https://developer.nvidia.com/embedded/linux-tegra>

¹⁷ <https://github.com/dusty-nv/vision.git> et ensuite branche v0.3.0

Language	Version	Licence	Rôles et responsabilités
v4l2loopback	0.12.5	GPL	Utilitaire qui a permis de créer un matériel vidéo virtuelle permettant de remplacer la caméra, permettant ainsi au modèle d'être alimenté par une vidéo et non la caméra. Une fois installé, le matériel vidéo virtuel est accessible via "/dev/video1", "/dev/video0" étant réservé pour la caméra.
hdparm	9.54	GPL	Utilitaire permettant de tester la capacité de lecture d'une unité de stockage, tel que'un SSD NVMe et différentes cartes microSD.
tegrastats	-	NVIDIA	La commande offre différents indicateurs système tel que l'utilisation des processeurs, la température, la consommation, et qui sont utiles pour observer le comportement du système lors des tests de performance de la segmentation.
free	3.3.12	GPL	La commande offre le statut de la mémoire totale, utilisée, libre, swap, cachée, etc. Elle est utile pour observer le comportement de la mémoire du nano-ordinateur lors des tests de performance de la segmentation.
iotop	0.6	GPL	La commande offre le statut des opérations "I/O" de lecture & écriture sur le disque, totale ou pour le processus de segmentation. Elle est utile pour observer le comportement des opérations sur le disque du nano-ordinateur lors des tests de performance de la segmentation.
segnetconsole	-	NVIDIA	La commande permet de segmenter une image. L'architecture est donnée en argument. Lors de l'essai, celle qui a été évaluée est "fcn-resnet18-deepscene-576x320". Les options qui doivent être utilisées pour que l'image générée soit évaluée doivent être "--visualize=mask --filtermode=point --alpha=0". L'image originale est fournie en avant-dernière place de la commande, et l'image générée en dernière place. Il est possible aussi de démarrer l'inférence avec sa propre architecture grâce aux options "-model", "--prototxt", "--labels", "--colors", "--input_blob" et "--output_blob".

Language	Version	Licence	Rôles et responsabilités
segnetcamera	-	NVIDIA	<p>La commande permet de démarrer la segmentation avec la caméra, ou optionnellement avec un matériel vidéo virtuel (fournie par "v4l2loopback") grâce à l'option "-camera=/dev/video1", comme cela a été le cas durant le projet pour évaluer la segmentation avec des vidéos au lieu de la caméra. L'architecture est donnée en argument. Lors de l'essai, celle qui a été évaluée est "fcnresnet18-deepscene-576x320". La résolution peut être précisée grâce aux options "width" et "height", mais durant l'évaluation les valeurs par défaut ont été conservées (width=1280px et height=720px). Il est possible aussi de démarrer l'inférence avec sa propre architecture grâce aux options "--model", "--prototxt", "--labels", "--colors", "-input_blob" et --output_blob". Il n'est pas possible de conserver la vidéo segmentée, et il n'est pas possible de sauvegarder les images qui sont rafraîchies à l'écran dans la fenêtre XWindow, car le code nécessaire à cette sauvegarde est trop intrusif et impacte négativement les performances de la segmentation et du nano-ordinateur. L'évaluation des performances n'est donc que visuelle, et cette limitation pourrait remettre en question le désir de détecter les délimitations de la piste cyclable en temps réel, puisqu'il n'y a pas de moyen de récupérer le résultat généré.</p>

3.4 Méthodologie

Voici à haut niveau les grandes étapes de cet essai (Figure 8) :

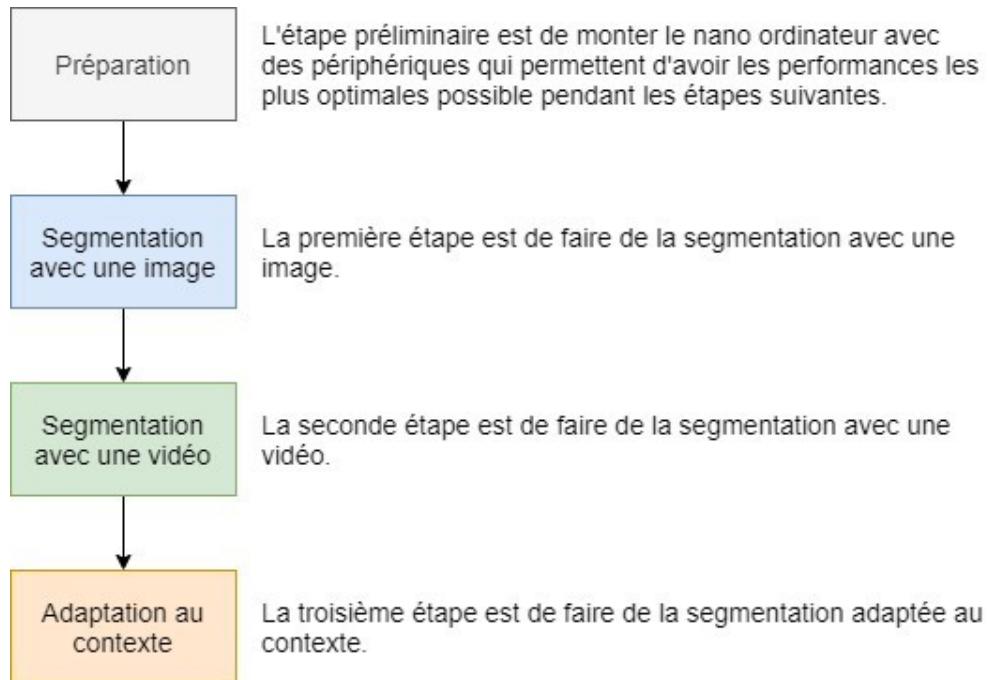


Figure 8: Organigramme de la méthodologie à haut niveau

Pour y parvenir, la méthodologie suivante (Figure 9) a été suivie et permet d'évaluer les performances de base de la segmentation sémantique avec le nano-ordinateur.

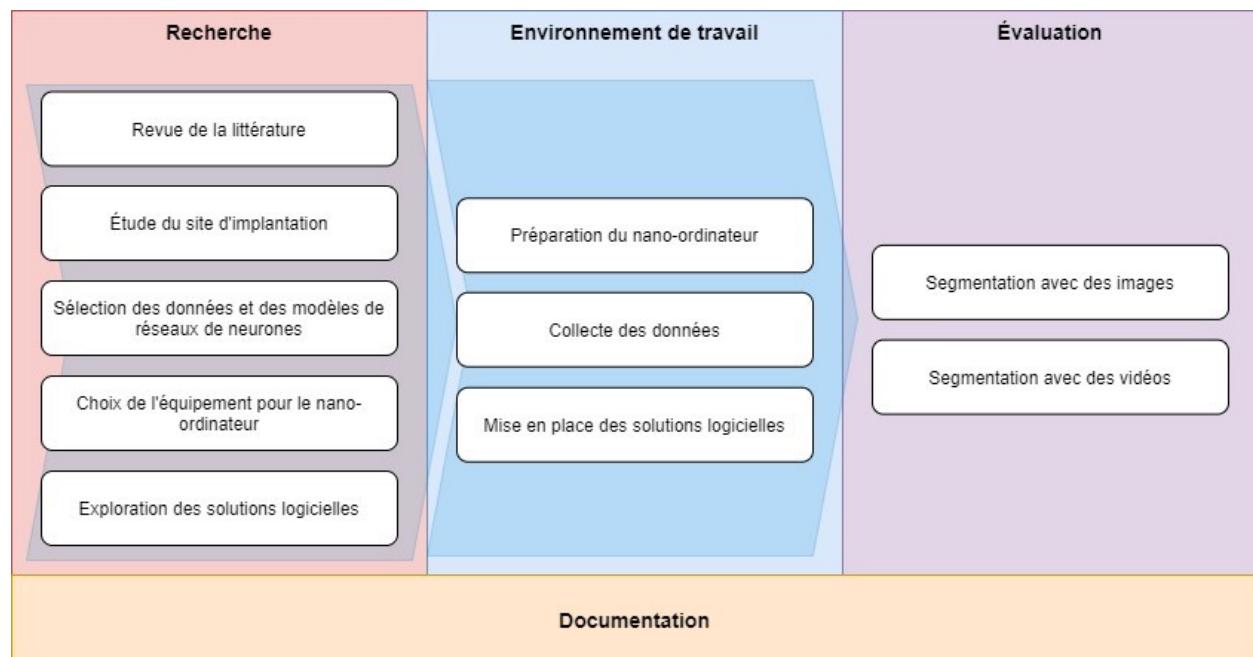


Figure 9: Organigramme des détails de la méthodologie pour évaluer les performances

Si l'évaluation est probante, la méthodologie se verra bonifier par des étapes de réarchitecture et/ou réentraînement (Figure 10).

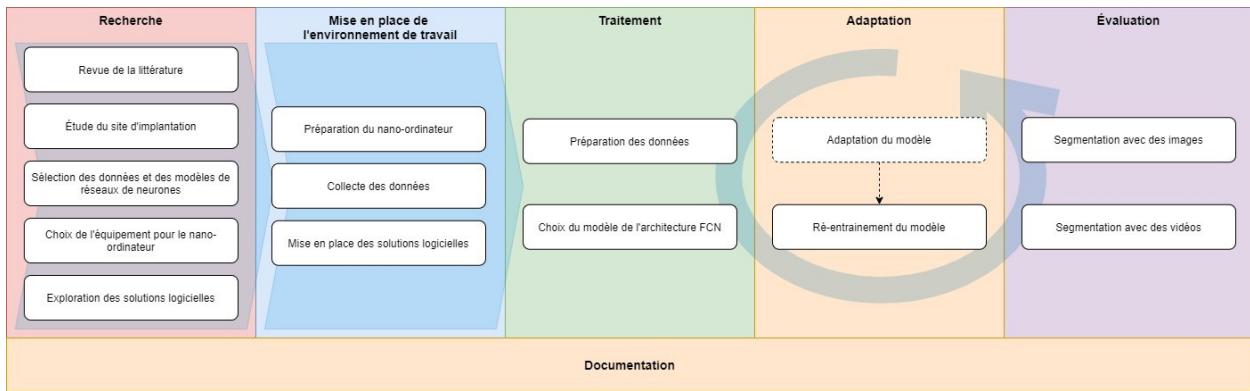


Figure 10: Organigramme des détails de la méthodologie pour évaluer les performances après une phase de réentraînement

Les phases de la méthodologie présentée dans l'organigramme de la figure 10 peuvent être résumées de la façon suivante :

- Recherche des références, des architectures et des données, ainsi que l'équipement pour le nano-ordinateur et des logiciels nécessaires.
- Installation sur le nano-ordinateur le système d'exploitation, l'environnement de développement et de tests pour l'inférence.
- Itération entre les étapes suivantes :
 - Inférence avec le nano-ordinateur en utilisant les architectures et les sources de données sélectionnées.
 - Réentraînement des architectures à différentes résolutions d'images et à la zone d'étude.
 - Traitement des données afin de les adapter au requis des architectures.

3.5 Documentation

La méthodologie a été entièrement documentée pendant tout le déroulement de l'essai. Elle se retrouve pour référence dans un blogue public sur le site de [github.io](https://vince7lf.github.io/)¹⁸. Cette méthodologie de documentation permet de facilement documenter, de ne pas perdre des notes importantes, de suivre le cheminement, de pouvoir retrouver des notes, même si elles ont été effacées ou modifiées, puisque toute modification est sauvegardée dans un repository Git.

¹⁸ <https://vince7lf.github.io/>

Par ailleurs, tous les documents de rédaction, les images, les scripts et code source qui ont été utiles et utilisés durant l'essai ont été géré dans un repository Git public avec *github.com*¹⁹. Ces sources d'information viennent bonifier grandement ce rapport et il est même recommandé de s'y référer pour atteindre un bon niveau de compréhension et avoir accès aux détails.

3.6 Environnement de travail

3.6.1 Préparation du nano-ordinateur

L'organigramme de la Figure 11 présente les activités qui composent la préparation du nano-ordinateur.

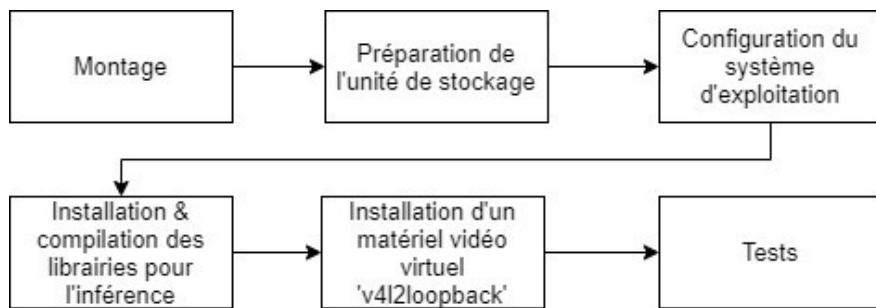


Figure 11: Préparation du nano-ordinateur

a) Montage

Le nano-ordinateur est une carte mère livrée sans aucun périphérique ni même boitier. Vu que les performances logicielles dépendent des performances matérielles, surtout pour une unité telle qu'un nano-ordinateur où les capacités matérielles sont limitées, la première partie de l'essai a été allouée à la sélection des accessoires et périphériques qui vont permettre d'augmenter les performances, protéger et utiliser confortablement le nano-ordinateur.

L'organigramme de la Figure 12 présente les activités qui composent le montage du nano-ordinateur.

¹⁹ <https://github.com/vince7lf/gae724>

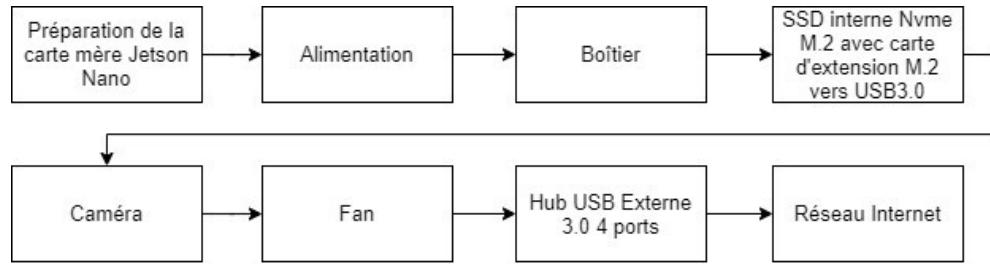


Figure 12: Montage du nano-ordinateur

La **Error! Reference source not found.** présente une photo de la carte mère, les accessoires et périphériques du nano-ordinateur qui a été monté. Chaque élément est identifié par un numéro qui se retrouve dans le Tableau 4.



Figure 13: Carte mère, accessoires et périphériques du nano-ordinateur. Les numéros se retrouvent au Tableau 3.

Tableau 3: Liste du matériel (avec prix en \$CAD). Les numéros se retrouvent à la Figure 13.

1	Boitier face avant	\$27.59	7	M.2 NVME SSD to M2 A/E Key WiFi Port	\$33.29
2	Boitier face arrière	-	8	Carte mère Jetson Nano	\$149.00
3	Caméra Raspberry v2	\$35.98	9	Carte d'extension T100 pour SSD M.2 VVMe	\$45.89
4	Alimentation 5 v 4 A	\$19.58	10	Dongle Wifi USB	\$25.19
5	Carte microSD	\$24.90	11	Ventilateur	\$19.78
6	SSD M.2 NVMe SATA 250GB	\$120.71	12	Hub USB 3.0 externe 4 Ports	\$22.94
Total : \$524.85					

a. Préparation de la carte mère Jetson Nano

Le nano-ordinateur qui est livré dans sa boite est simplement une carte mère. Le système d'exploitation doit être installé. Certaines broches sur la carte mère doivent être activées afin de bénéficier de certains avantages matériels, comme la possibilité de brancher un adaptateur d'alimentation de 5 V 4 A au lieu de l'alimentation micro USB; et d'activer le PoE (*Power-OverEthernet*) afin d'hériter de l'alimentation du câble Ethernet. Aucune autre préparation sur la carte n'est nécessaire.

b. Alimentation

L'alimentation du nano-ordinateur est l'élément matériel le plus important du système. Le besoin en énergie augmente avec les périphériques qui s'accumulent, tel qu'une caméra et un ventilateur. Il est prudent de choisir un adaptateur 5 V 4 A d'un fournisseur recommandé par NVIDIA, car un changement de puissance sensible en entrée impacte le fonctionnement opérationnel du nano-ordinateur.

c. Boitier

Afin de protéger le nano-ordinateur lors des manipulations durant l'essai, et l'utiliser dans les conditions les plus proches de son futur mode d'opération, il a été installé dans un boitier en métal. Le boitier a été choisi en tenant compte qu'une carte d'extension pour un SSD interne est installée, ainsi qu'une caméra et un ventilateur.

d. Unité de stockage

Le nano-ordinateur est conçu pour fonctionner avec un système d'exploitation hébergé sur une carte microSD. Il existe différentes cartes microSD, et certaines sont beaucoup plus performantes que les autres. Malheureusement les cartes microSD ne sont pas destinées à exécuter un système d'exploitation, et leur espérance de vie reste limitée. Étant donné que l'objectif du nano-ordinateur est d'être en service continu à l'extérieur, l'utilisation un disque SSD interne comme alternative semble logique.

i. Carte microSD

Il existe différentes cartes microSD, de multiples constructeurs, et pour différents usages, mais généralement destinées pour stocker des images et vidéos directement par les appareils multimédias. Leur conception est faite pour la manipulation de gros blocs de données, et non des petits fichiers. Trois cartes microSD sont évaluées (

Tableau 4) :

Tableau 4: Cartes microSD

	microSD Samsung EVO 64Gb Plus XC I Grade 3 Class 10
	microSD Samsung EVO 64Gb Select XC I Grade 3 Class 10
	microSD Scan Disk Ultra 32Gb HC I Class 10

ii. Disque SSD

Pour un appareil destiné à être continuellement en service et à l'extérieur, l'unité de stockage doit être non seulement performante, mais aussi endurante. Un disque SSD et une carte microSD sont différents types de matériel pour différents usages. Le disque SSD est plus adapté pour manipuler les petits fichiers et héberger un système d'exploitation. Il est aussi plus résilient à long terme. C'est donc une option qui ne doit pas être négligée dans le contexte de tests de performance, encore plus avec un nano-ordinateur dont les capacités matérielles sont limitées, et qui est un appareil destiné à être continuellement en service et à l'extérieur. Néanmoins, il y a une contrepartie importante dans la situation d'un nano-ordinateur : la consommation d'énergie. Un SSD interne va demander plus d'énergie qu'une carte microSD, et si le nano-ordinateur n'est pas capable de gérer correctement les besoins en énergie de ses extensions matérielles, le SSD interne risque d'échouer en pleine opération et le nano-ordinateur devenir non fonctionnel brusquement.

Un disque SSD interne pour un nano-ordinateur est soit une carte d'extension M.2 NVMe ou SATA, connecté au port PCIe ou USB. Les SSD internes Samsung 970 EVO 250GB NVMe M.2 et Samsung 860 EVO M.2 500GB SATA sont évalués.

À noter qu'une carte microSD est tout de même nécessaire pour *bootstrapper*²⁰ le système d'exploitation.

e. Caméra

L'objectif du nano-ordinateur est d'être utilisé pour détecter continuellement les délimitations de la piste cyclable. Il est évident qu'une caméra doit donc faire partie du système et faire partie de l'évaluation des performances. Néanmoins, durant le déroulement de l'essai, la caméra est peu utilisée. En effet il n'est pas évident d'être dans un mode de développement directement sur le terrain. Un matériel vidéo virtuel (*v4l2loopback*) est utilisé pour simuler la caméra et alimenter l'inférence avec des vidéos préenregistrées, permettant ainsi d'évaluer les performances de l'inférence avec des vidéos, même si d'un point de vue de la performance matérielle l'utilisation n'est pas équivalente. Les performances matérielles de l'inférence en temps réel sont évaluées avec la caméra, même si la vue de la caméra n'est pas la piste cyclable, ce qui n'est pas important pour ce test, peu importe ce qui est détecté.

La caméra qui a été sélectionnée est la version 2 du fournisseur Raspberry Pi. Cette caméra a été éprouvée avec le temps et est performante.

f. Ventilateur

Un système informatique a besoin d'un ventilateur pour évacuer la chaleur produite par ses éléments et ainsi assurer une durée de vie optimale. L'objectif du nano-ordinateur étant d'être opérationnel continuellement, et ses éléments étant contenus dans un boîtier, il est nécessaire d'installer un ventilateur. Le ventilateur est mis en marche par le système selon le besoin, mais dans le cadre de l'essai et des tests il est mis en marche dès que le nano-ordinateur est démarré, afin d'éviter que la chaleur ne s'accumule rapidement.

²⁰ Programme initial permettant de charger le noyau du système d'exploitation lors de la mise sous tension de l'ordinateur (Le Grand Dictionnaire Terminologique de l'Office Québécois de la Langue Française, consulté le 29 mai 2021)

g. Hub USB externe 3.0 4 ports

Le nano-ordinateur comprend un hub USB 3.0 4 ports internes, les 4 ports étant connectées via le même contrôleur. Ce hub consomme de l'énergie pour alimenter les périphériques qui y sont connectés, comme un SSD interne ou un dongle Wifi, et gérer l'échange de données. Afin de minimiser les besoins en alimentation et optimiser le plus possible le transfert de données, la souris, le clavier et le dongle USB ont été branchés à un hub USB 3.0 externe auto alimenté. Malheureusement cette option complexifie le déploiement sur le terrain du nano-ordinateur. L'alternative pour s'en passer est d'utiliser un câble Ethernet, PoE préférablement, à la place d'un dongle Wifi qui est un plus gros consommateur d'énergie, et chauffe rapidement.

h. Interface réseau

Le nano-ordinateur comprend un contrôleur Ethernet pour brancher un câble réseau et se brancher sur Internet. Selon la configuration de la carte mère, le nano-ordinateur peut hériter de l'alimentation via Ethernet (PoE). Il comprend aussi un port PCIe interne qui permet de brancher une carte d'extension Wifi. L'autre alternative étant de passer par un dongle USB Wifi, ou un périphérique Wifi externe connecté au port USB. L'accès privilégié durant l'essai a été de brancher un câble Ethernet. Le PoE n'a pas été évalué.

b) Préparation de l'unité de stockage

Le nano-ordinateur est conçu pour fonctionner avec une microSD, et NVIDIA fournit uniquement de la documentation à cet effet. L'option d'utiliser un disque SSD interne est disponible sur Internet, mais n'est pas supporté officiellement par NVIDIA. Il existe néanmoins des articles à ce sujet dans le forum des développeurs²¹.

a. Carte microSD

NVIDIA fournit de la documentation claire et simple afin de préparer la carte microSD (formatage) et installer l'image du *JetPack*²².

²¹ <https://forums.developer.nvidia.com/t/how-to-connect-ssd-to-jetson-nano/74053>

²² <https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write>

b) Disque SSD

La procédure d’installation pour installer *JetPack* sur le SSD interne est disponible sur le site *jetsonhacks.com*²³. À noter qu’une carte microSD est tout de même nécessaire pour *bootstrapper*²⁴ le système d’exploitation. Il n’est pas nécessaire d’avoir une carte microSD performante puisqu’elle n’est utilisée que pour démarrer le système qui se trouve sur le SSD interne.

c) Configuration du système d’exploitation

La première fois que le système démarre, le système Ubuntu *Linux pour Tegra* (L4T) doit être configuré avec toutes les options personnalisées (langue, clavier, fuseau horaire, etc.).

d) Installation & compilation des librairies pour l’inférence

Les librairies pour la segmentation sémantique d’images et de vidéos via l’inférence de modèles déjà préparée sont mises à disposition par NVIDIA via un projet dans *GitHub*. La documentation pour l’installation et l’inférence est disponible directement dans la page *GitHub*.

e) Installation d’un matériel vidéo virtuel ‘v4l2loopback’

L’inférence fournie par NVIDIA est conçue pour utiliser la caméra du nano-ordinateur. Dans le cadre de cet essai, un matériel vidéo virtuel est utilisé pour simuler la caméra, et alimenter le modèle avec une vidéo enregistrée. La contrepartie de cette solution est à prendre en considération durant l’évaluation des performances : en effet la caméra demande plus de puissance au nano-ordinateur que le simulateur logiciel.

f) Tests

Afin de s’assurer que le nano-ordinateur est prêt pour être évalué, des tests matériels et logiciels sont effectués une fois le système monté et stabilisé. Les résultats des tests servent de référence pour évaluer l’état de santé du nano-ordinateur.

²³ <https://www.jetsonhacks.com/2019/09/17/jetson-nano-run-from-usb-drive/>

²⁴ Programme initial permettant de charger le noyau du système d’exploitation lors de la mise sous tension de l’ordinateur (Le Grand Dictionnaire Terminologique de l’Office Québécois de la Langue Française, consulté le 29 mai 2021)

3.6.2 Collecte des données

Le jeu d’images de *DeepScene* est celui qui semble le plus approprié, car il a été conçu pour détecter les chemins dans la forêt. De plus, il existe une version de l’architecture qui a été entraînée avec ce jeu. Comme un jeu d’images vérité terrain est disponible, cela procure un gain de temps non négligeable dans le cadre d’un essai. Le jeu d’images de *Cityscapes* est complet pour les scènes urbaines, mais comme il est moins spécialisé dans la détection de chemins ou de piste, son utilisation n’est pas priorisée. Il contient toutefois des images vérité terrain de routes, ce qui est avantageux dans notre contexte et le favorise par rapport aux deux derniers que nous avons à notre disposition. En effet le jeu d’images et de vidéos de l’APC-PJC, et celui que j’ai monté en prenant des vidéos de pistes cyclables de mon quartier, sont des jeux intéressants pour tester les résultats de la segmentation avec des images ou des vidéos qui viennent du site d’études, ou similaire, que celui de chemins forestiers. De plus ces images et vidéos sont loin des conditions parfaites (luminosité, qualité du sol, angle de vue, etc.).

3.6.3 Mise en place des solutions logicielles

a) *Jetson Nano*

Le nano-ordinateur est destiné à l’inférence. NVIDIA fournit tout un système d’installation, qui est nommé *JetPack*, et qui contient un système d’exploitation basée sur Ubuntu, *Linux pour Tegra* (L4T), la plateforme applicative et les librairies nécessaires pour l’inférence, telles que *Python*, *pytorch*, les modèles préentraînés au format ONNX, le compilateur *CUDA*, et le SDK *TensorRT*.

b) *Calcul Québec*

Le nano-ordinateur est destiné à l’inférence, et non l’entraînement d’architectures. Il n’est pas non plus destiné à être un environnement de développement. Un autre environnement de travail est donc nécessaire pour développer, et doit posséder les capacités matérielles (GPUs, mémoires, espace de stockage) et logicielles (librairies) pour entraîner une architecture. Le professeur Mickaël Germain, directeur de projet, m’a présenté l’environnement de Calcul Québec. Celui-ci fournit un espace de travail scientifique destiné aux chercheurs et aux universitaires, qui m’a permis de pouvoir travailler avec l’apprentissage profond, compiler un fork de *torchvision*, réentraîner des architectures, générer les versions ONNX, et ainsi contourner les limitations du

nano-ordinateur. Avoir accès à cet environnement de travail a été un élément clé dans le cadre de cet essai.

c) *Compte Calcul Québec*

Calcul Québec mets à disposition des ressources matérielles puissantes et l'accès a des libraires de haute technologie telle que pour l'apprentissage profond, permettant d'avoir un environnement de travail professionnel et performant rapidement. Les ressources matérielles à disposition sont des grappes de serveurs, de CPUs et GPUs de différents types, ainsi que de l'espace de stockage. Les librairies sont disponibles via un repository privé, et lorsque certaines étaient manquantes (ONNX et *onnxruntime*), j'ai fait une demande par courriel. L'administrateur a pu rendre disponible l'une des deux (ONNX), la seconde (*onnxruntime*) étant beaucoup plus complexe à installer, pour l'avoir tenté sur le nano-ordinateur.

L'autre avantage de l'environnement de Calcul Québec est la mise à disposition de *Jupyter Notebook*, afin de tester rapidement du code *Python*. Par contre il n'est pas conseillé d'exécuter du code nécessitant des délais, tels que l'entraînement d'une architecture.

L'un des irritants est de ne pas pouvoir exécuter un conteneur *Docker* tel quel. Il faut le convertir au format *Singularity*. Dans le cadre du projet cela m'aurait facilité la tâche, car NVIDIA fournit des conteneurs Docker prêts à l'utilisation pour le réentraînement. Je n'ai malheureusement pas pris le temps et la chance de convertir un conteneur Docker au format Singularity. Je ne sais pas si c'est une activité assez simple ou complexe, mais du peu que j'ai lu cela semble assez "rapide".

d) *Jupyter Notebook*

Le besoin de tester du code *Python* est toujours nécessaire. La console *Python* n'étant vraiment pas conviviale, un environnement *Jupyter Notebook* est un compromis incontournable. Heureusement Calcul Québec fournit un accès à des notebooks depuis Internet, permettant en plus d'hériter de leur environnement de travail. Il est à noter que les notebooks n'ont pas été utilisés pour entraîner une architecture ou générer les versions ONNX, mais de tester du code *Python* simple, comme visualiser des images, transformer des tensors, et évaluer la segmentation prédictive générée avec le véritable terrain (GT).

e) NVIDIA

a. Compte NVIDIA

NVIDIA met à disposition tout un écosystème éducatif permettant aux développeurs et aux chercheurs d'obtenir de l'aide au sujet de leur produit et librairie. Dans le cadre de l'essai, un compte NVIDIA a été créé, permettant d'accéder au forum de développeurs, et les conteneurs Docker par exemple. Il est aussi possible d'accéder à du matériel éducatif grâce à l'institut DeepLearning de NVIDIA, dont l'accès a été commandité par le professeur Mickaël Germain, directeur de projet. Le forum de développeurs a été un outil utile dans le cadre de ce projet, car le dépôt d'une question m'a permis de me débloquer. Je n'étais pas capable de régénérer la version ONNX à partir du code source et de la documentation fournie par NVIDIA pour une architecture FCN. Le développeur principal de l'application a répondu et m'a guidé dans la résolution du problème. Les autres ressources ont eu un impact limité dans le cadre de ce projet, puisque par exemple le conteneur Docker et DIGITS n'ont pas pu être utilisé. Le code source des architectures est disponible sans nécessiter de compte, de même que les SDKs Jetpack.

b. NVIDIA DIGITS

NVIDIA fournit aux développeurs un environnement visuel permettant de réentrainer les architectures FCN qu'ils fournissent avec leurs propres jeux de données. Cet environnement se nomme DIGITS. Malheureusement il est nécessaire d'avoir son propre matériel, le système d'exploitation Ubuntu 18.04 LTS, d'avoir au moins un GPU. DIGITS ne s'installe pas sur le nano-ordinateur, ni sous Windows, ni avec la version "WSL" (Sous-système Windows pour Linux). Cette option a donc été abandonnée rapidement.

c. Docker NVIDIA

NVIDIA fournit aux développeurs des conteneurs Docker, avec tout ce qui est nécessaire pour réentrainer une architecture et régénérer une version ONNX, par exemple. Malheureusement la capacité du nano-ordinateur ne permet pas de travailler efficacement avec un conteneur Docker, le nano-ordinateur devient sans réponse, nécessitant un redémarrage forcé. Cette option a donc été aussi abandonnée rapidement.

d. NVIDIA DeepStream

Durant le déroulement de l'essai, NVIDIA a mis à disposition un environnement d'apprentissage profond, nommé *DeepStream*, facilitant la conception et la génération de modèles, jusqu'à l'inférence. Cet outil n'a pas été évalué, mais pourrait être un outil alternatif pour réentraîner une architecture.

3.7 Évaluation

L'évaluation des performances se décompose de différents éléments suivants, et qui sont présentés dans le diagramme de la Figure 14.

3.7.1 Les indicateurs de performances.

Les performances matérielles durant l'inférence sont évaluées grâce aux indicateurs fournis par les utilitaires 'Tegrastats' de NVIDIA, 'free' et 'iostop'. Ces utilitaires sont brièvement décrits dans le tableau 2.

Les performances de la segmentation sont évaluées avec les indicateurs classiques²⁵: le "IoU" ("Intersection sur Union", ou "Jaccard index"); et le F1 score (ou "Dice coefficient").

3.7.2 Les résolutions

Les résolutions d'images et de vidéos sont utilisées pour déterminer lesquelles sont supportées par l'architecture évaluée, comme indiqué dans la section "3.7.3 Résolutions évaluées".

3.7.3 Médias

Les images et vidéos qui sont à notre disposition pour être évaluée proviennent de différentes sources de données, tel que décrit dans la section "3.6.2 Collecte des données".

3.7.4 Modèles FCN et Dataset

Enfin plusieurs modèles ont été sélectionnés comme candidats intéressants pour l'évaluation, et décrits dans la section "3.8.1 Choix de l'architecture FCN".

²⁵ <https://ilmonteux.github.io/2019/05/10/segmentation-metrics.html>

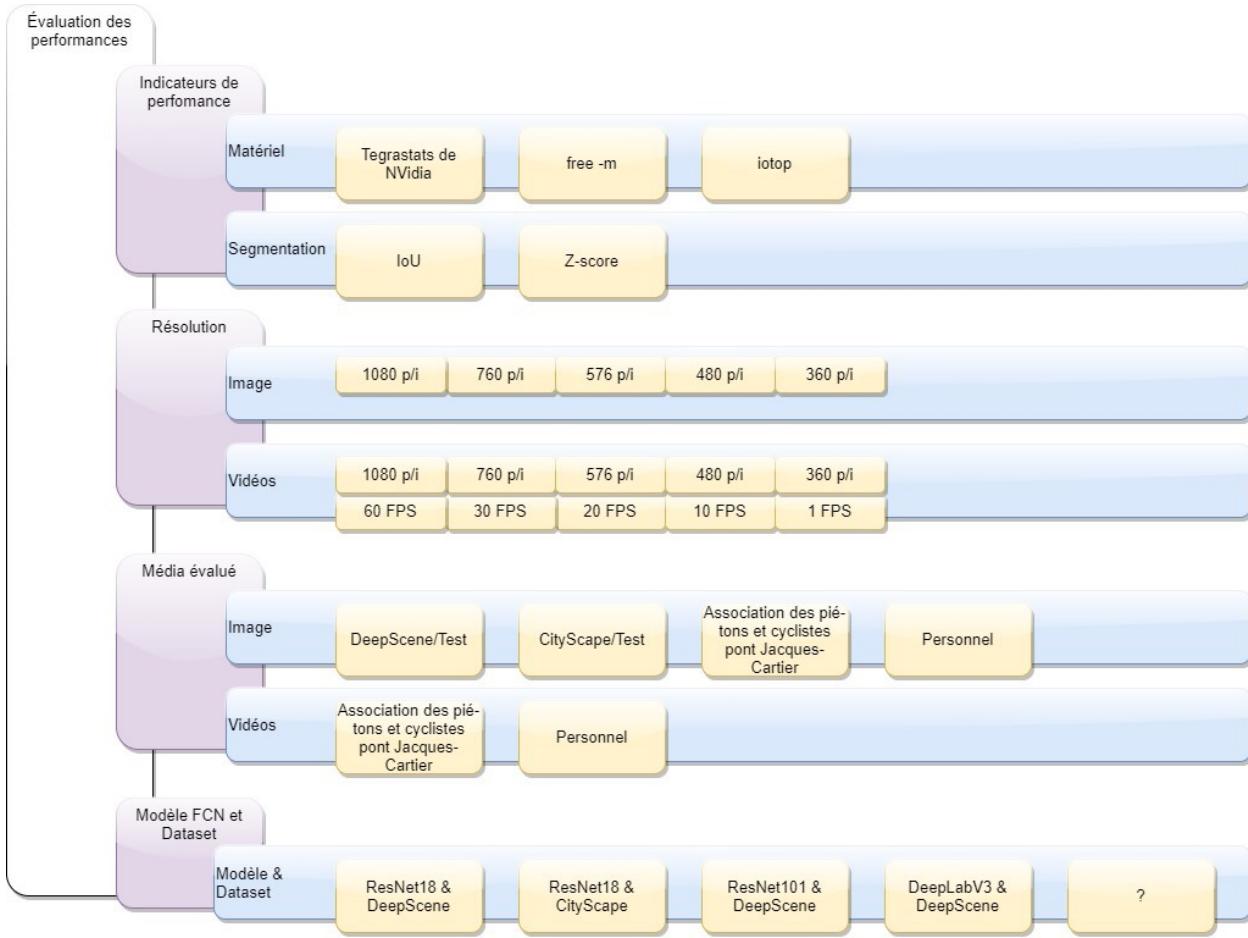


Figure 14: Éléments pour l'évaluation des performances

3.7.4 Stratégie de test de l'inférence

L'objectif principal de l'essai est de déterminer la capacité et les limites du nano-ordinateur d'inférer en temps réel des FCN pour la segmentation sémantique de vidéos. La stratégie qui est appliquée est de tester avec diverses architectures et divers niveaux de qualité vidéos, en espérant trouver le compromis qui répond le mieux à cet objectif.

1. Afin de s'assurer du bon fonctionnement du nano-ordinateur et d'avoir des résultats de référence propre à notre environnement, l'inférence est testée avec des modèles existants et préentraînés pour la segmentation sémantique, avec les images et les vidéos provenant des références, et dont les caractéristiques et les résultats sont disponibles.
2. En espérant que les tests de l'étape #1 donnent les résultats documentés dans les articles de références, ils sont repris avec les mêmes modèles, mais avec les images et les vidéos du site d'étude possédant la meilleure qualité acquise (1080p/i, 30FPS). Les données sources

(images et vidéos) devront subir certains prétraitements à cet effet, afin de répondre aux requis des architectures.

3. Selon les résultats de l'étape #2, les tests se concentreront sur l'inférence avec des vidéos, en réduisant progressivement la résolution (760p/i, 576p/i, 480p/i, 360p/i) et le nombre d'images par seconde (20FPS, 10FSP, 1FPS).
4. Les étapes intermédiaires de l'étape #3 sont de 1) valider les résultats de l'inférence avec des images avant de tester avec les vidéos, et 2) évaluer si les architectures FCNN doivent et/ou peuvent être adaptées facilement, en tenant compte de l'échéancier de l'essai, et ce afin de répondre à l'objectif principal.

3.7.5 Stratégie de collecte des indicateurs de performance matérielle

La méthodologie de la collecte des indicateurs est la suivante²⁶. La collecte est démarrée après un démarrage frais, manuellement, via un script Shell, qui exécute chaque utilitaire, et attend l'interruption du test. Chaque utilitaire qui est utilisé pour collecter les mesures possède son propre fichier. La date et l'heure de chaque indicateur collecté sont précisées. Afin de faciliter la documentation et l'analyse du test, des points d'intérêt sont ajoutés dans un fichier séparé pour marquer un moment particulier du test, avec la date, l'heure et un libellé. Ce point d'intérêt est fait grâce à une commande "Shell" qui vient ajouter une trace dans ce fichier. Chaque indicateur est collecté toutes les secondes. Une fois le test complété, la collecte est arrêtée manuellement. Chaque fichier est ensuite transformé en fichier CSV, via des commandes "Shell". À partir des fichiers CSV, un script *Python* génère les graphiques automatiquement. Chaque indicateur est une colonne du fichier CSV. Il existe le même nombre d'indicateurs à tout moment. La date et l'heure sont un champ. Avant tout début de tests, la collecte est démarrée sans activité autre que la collecte des indicateurs. Cela permet de prendre une base de référence sans aucune charge. Les indicateurs collectés permettent de créer des graphiques qui montrent la progression de chacun. Les performances matérielles du Jetson Nano sont évaluées grâce à différents utilitaires : "tegrastats" fournis par NVIDIA; "free"; et "iostop". Les performances de la segmentation sont évaluées grâce au IoU et au F1 score pour la classe du chemin / route. Les fonctions IoU et le F1 score utilisent l'image prédite (généré par l'architecture FCN) et l'image vérité terrain. Les images originales

²⁶ <https://vince7lf.github.io/2020/05/26/metrics.html>

sont pré sélectionnées selon leur intérêt et l'image vérité terrain créée. L'image prédite et vérité terrain doivent utiliser la même palette de couleurs et doivent être de la même résolution. Pour les images qui ne possèdent pas d'image vérité terrain, celle-ci est créée à la main avec l'éditeur "Gimp", en utilisant la même résolution de la segmentation de l'image produite par l'architecture du modèle de NVIDIA. Le besoin est d'évaluer et non d'entrainer, l'importance de la précision de la classification est moindre dans ce cas.

3.7.6 Résolutions évaluées

Les résolutions et images par seconde qui sont évaluées sont présentées dans le Tableau 5.

Tableau 5: Résolutions et images par seconde (FPS) qui sont évaluées

Résolutions
320x576, 480x640, 720x1280, 768x1024, 768x1152, 800x1152, 832x1024, 864x1024, 832x1120, 832x1152, 768x1280, 800x1280, 864x1152, 900x1152, 900x1280, 960x1600, 1080x1920, 1024x1024
Images par seconde (FPS)
60/1, 30/1, 15/1, 1/1

3.7.7 Segmentation avec des images

a) Préparation et post-traitement

Afin de pouvoir mesurer les performances de la segmentation (IoU, F1 score), les classes et la palette de couleur entre l'image vérité terrain et celles prédites doivent être les mêmes.

L'image vérité terrain du jeu de donnée original *DeepScene* ne possède pas la même palette de couleur ni exactement les mêmes classes que celle de l'architecture. Un travail d'uniformisation est nécessaire avant la segmentation, qui est résumée dans le Tableau 6.

Tableau 6: Classes et palettes de couleur

DeepScene		NVIDIA		Consolidée	
Classes	RGB	Classes	RGB	Classes	RGB
Road	170-170-170	Trail	200-155-75	Trail	170-170-170
Grass	0-255-0	Grass	85-210-100	Grass	0-255-0
Vegetation	102-102-51	Vegetation	15-100-20	Vegetation	102-102-51
Tree	0-60-0	-	-	Vegetation	102-102-51
Sky	0-120-255	Sky	0-120-255	Sky	0-120-255
Obstacle	0-0-0	Obstacle	255-185-0	Obstacle	0-0-0

De plus, l'image segmentée prédite par l'architecture ne possède pas précisément la même palette de couleur que celle qui est configurée, il y a quelques différences minimes dans les codes couleurs RGB (par exemple 0-119-255 au lieu de 0-120-255), mais qui doivent être arrangée afin de pouvoir être correctement évaluées.

Un travail de traitement de l'image segmentée prédite est nécessaire avant l'évaluation de la segmentation.

b) Segmentation et évaluation

Afin de tester la performance de la segmentation du modèle, deux images du jeu de données de *DeepScene* sont utilisées, car ce jeu contient déjà les images vérités terrain, un gain de temps non négligeable dans le cadre de l'essai. Uniquement la classe *Trail* est évaluée.

L'architecture fournit à l'utilitaire *segnet-console* est *fcn-resnet18-deepscene-576x320*²⁷. Un script *Python*²⁸ est utilisé afin de mesurer le *IoU* et le *F1 score* de la classe de l'image prédite par l'architecture.

²⁷ segnet-console-{}-network=fcn-resnet18-deepscene-{}-visualize=mask-{}-alpha=10000images/ city_0.jpgoutput.jp

²⁸ <https://gist.github.com/ilmonteux/8340df952722f3a1030a7d937e701b5a>

3.7.8 Segmentation avec des vidéos

a) Préparation et prétraitement

L'évaluation de la segmentation avec des vidéos va s'effectuer non pas avec la caméra, mais avec un matériel vidéo virtuel. En effet, il n'est pas réaliste de pouvoir travailler sur le terrain. La commande *segnet-camera* permet de fournir en option le matériel qui doit être utilisé, par exemple */dev/video0* pour la caméra. Le module *v4l2loopback*²⁹ permet de créer un matériel vidéo virtuel */dev/video1*. Ce matériel permet de recevoir un flux vidéo, qui pourra alors alimenter l'utilitaire *segnet-camera*, comme le ferait la caméra. Le flux vidéo est produit par l'utilitaire *gstreamer* avec comme données d'entrées le fichier de la vidéo et dirigé vers le matériel vidéo virtuel */dev/video1*.

La difficulté réside dans le fait que le matériel vidéo virtuel et le flux vidéo doivent être compatibles avec ce que l'utilitaire *segnet-camera* s'attend, et qui a été conçu pour être compatible avec une caméra. Le diagramme de la Figure 15 résume à haut niveau les relations entre ces éléments. Pour comparaison, le diagramme de la Figure 16 montre la segmentation avec la caméra.

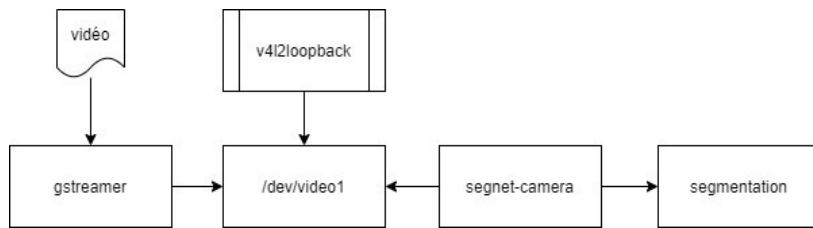


Figure 15: Diagramme d'architecture de la segmentation d'une vidéo

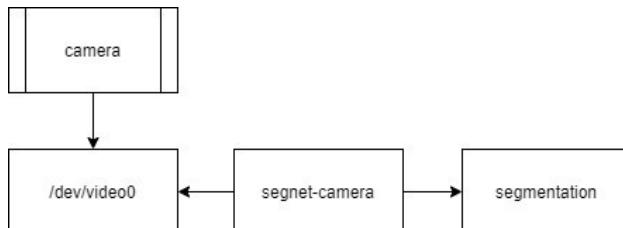


Figure 16: Diagramme d'architecture de la segmentation avec la caméra

b) Segmentation et évaluation

Les tests de performance de la segmentation de vidéos se déroulent de la manière précisée dans la section 3.7.1 *Stratégie de test de l'inférence*.

²⁹ <https://github.com/umlaeute/v4l2loopback>

L'un des avantages de l'utilitaire *gstreamer* est de pouvoir contrôler la résolution et le nombre d'images par seconde (FPS) de la vidéo qui doit être segmentée. Les différentes résolutions et FPS qui désirent être exécutées sont préparées dans un script *shell* écrit pour l'occasion. Le script s'occupe de démarrer *gstreamer* avec les bons paramètres, et en parallèle de démarrer la segmentation avec *segnet-camera*. Un jeu de résolution peut être testé unitairement³⁰, ou plusieurs en séquence³¹.

Les résolutions et images par seconde qui ont été testées sont résumées dans le tableau 4. Deux vidéos ont été utilisées pour tester la segmentation. La première vidéo est utilisée pour tester l'inférence avec une vidéo du site d'étude, et qui a été fournie gracieusement par l'APC-PJC. Cette première vidéo est intéressante, car elle est filmée en mouvement par un cycliste. Dans un intervalle de 30 secondes, l'angle de vue change rapidement. La piste cyclable est bordée d'un muret côté sud, et de la route avec les voitures qui circulent côté nord. Même si la journée est ensoleillée, la surface de la piste est aussi à un moment humide.

La seconde vidéo est utilisée pour tester la segmentation avec les différentes résolutions et images par seconde. C'est une vidéo d'une petite piste cyclable qui est dans mon quartier, et que j'ai prise en marchant avec mon téléphone intelligent. La vidéo est intéressante, car dans un intervalle de 30 secondes l'état de la piste passe d'une scène ensoleillée à ombragée, sèche à mi-sèche, avec un petit ou gros banc de neige en bordure, ou qui s'aventure un peu sur la piste, bordée d'herbe mouillée ou sèche.

3.8 Réentrainement

La phase de réentrainement sera initiée si le temps le permet, et s'il est jugé bon d'améliorer la qualité de la segmentation. Il y a deux types d'adaptation possible : matérielle et logicielle.

L'adaptation matérielle sera jugée nécessaire si le nano-ordinateur ne peut être utilisé tel quel pour répondre aux objectifs. Par exemple si le nano-ordinateur devient non utilisable (lent ou sans réponse) après un certain temps. La stratégie sera de diagnostiquer le comportement et d'évaluer un remède. L'une des pistes de solution privilégiée sera de lui apporter plus de puissance, comme

³⁰ https://github.com/vince7lf/gae724/blob/master/run_deepscene.sh

³¹ https://github.com/vince7lf/gae724/blob/master/run_deepscene_batch.sh

de l'ajout de mémoires ou un espace de stockage plus performant. Plus complexe, optimiser le processus, par exemple certains traitements, serait aussi une option.

Le réentraînement sera jugé nécessaire si la prédiction de la segmentation est en deçà des attentes, ou inutilisable. Le choix d'adapter une architecture avec les méthodes d'apprentissage par transfert (*Transfer Learning* en anglais) et d'adaptation de domaine (*Domain Adaptation* en anglais) sera la première option privilégiée, car cela procure un gain de temps non négligeable : il n'y a pas besoin de passer à travers tout le processus *essai-erreur*, couteux en temps, en énergie et en ressources matérielles, d'apprentissage et de paramétrisation de l'architecture. Des efforts conséquents sont nécessaires pour générer les images de la vérité terrain avec le jeu de données local, celui de l'APC-PJC de préférence, personnel ou autre sinon.

Le diagramme de la Figure 17 présente la méthodologie qu'il faut suivre pour réentraîner une architecture à un nouveau jeu de données et à une autre résolution. La première étape est de sélectionner une architecture déjà entraînée et qui semble pouvoir être la meilleure candidate pour aider à répondre à la problématique. C'est un travail de recherche et de test minutieux, qui est le plus important de toutes les étapes. La seconde activité est conséquente en efforts : préparer le jeu de données, incluant les images vérité terrain, les bonnes résolutions; et déterminer les classes et la palette de couleurs nécessaires. L'étape suivante est d'étudier l'architecture du modèle, afin de l'adapter au jeu de données, aux classes, et au besoin modifier les couches de l'architecture afin d'avoir une segmentation la plus précise et fine possible. Une fois ces étapes de préparation complétées, le réentraînement de l'architecture peut s'effectuer, et la segmentation évaluée. Cette phase de réentraînement est un processus itératif.

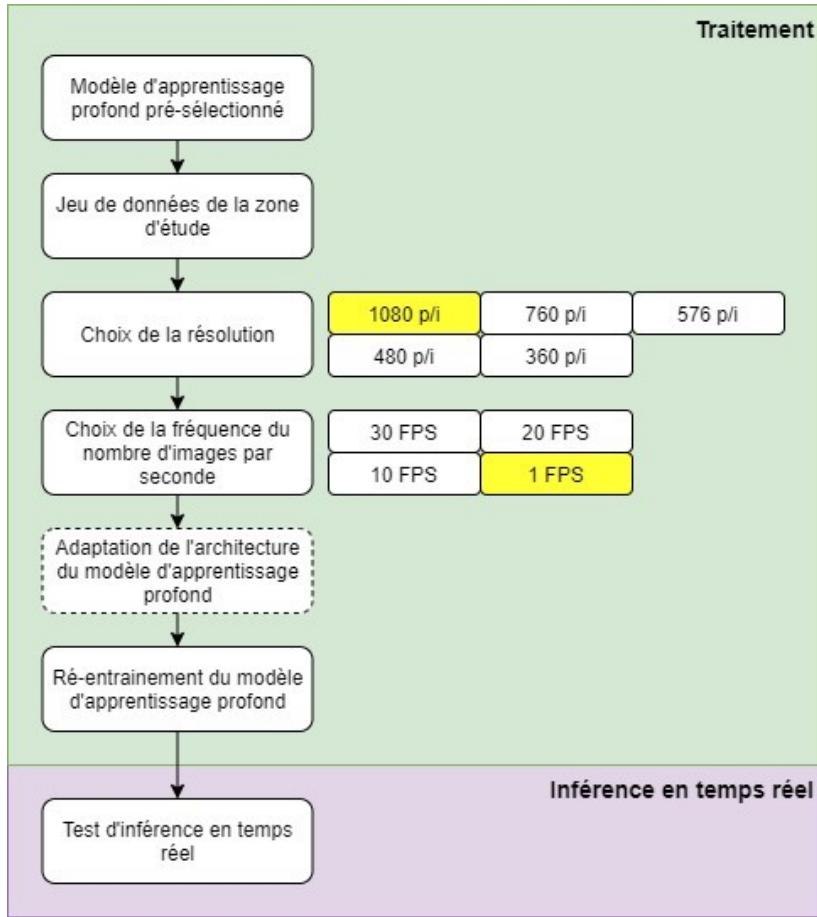


Figure 17: Méthodologie du réentraînement

3.8.1 Choix de l'architecture FCN

Le premier modèle qui est évalué est celui de l'architecture *SegNet18* entraînée avec le jeu de données *DeepSCene*, et fourni par NVIDIA. Le second de la liste, et qui est aussi déjà fourni par NVIDIA, est l'architecture de *SegNet18* entraînée avec le jeu de donnée *CityScapes*. Les deux autres architectures, *ResNet101 & DeepScene* et *DeepLabV3 & Deepscene*, ne sont pas disponibles et devront être préparées et entraînées, mais elles sont attrayantes du point de vue de leur réputation et de leur potentiel, et vouloir les adapter au contexte de l'essai semble logique. Une dernière boite vide est disponible, afin de laisser une porte ouverte à une potentielle opportunité d'entrainer une architecture tout à fait personnalisée, par exemple une adaptation de l'architecture de *DeepLabV3* avec le jeu de données de l'APC-PJC.

4 Résultats

4.1 Performances matérielles

4.1.1 Stockage de données

Pour tester les performances de la microSD et du disque SSD interne M.2 NVMe, l'utilitaire *hdparm* peut être facilement utilisé (Tableau 7).

Tableau 7: Comparaison des performances du "data read" entre un SSD M.2 NVMe et une microSD

Disk reads	MB	s	MB/s
Samsung 970 EVO Plus 250GB M.2 NVMe Internal Solid State Drive	1004	3	334.15
microSD Scan Disk Ultra 32Gb HC I Class 10	122	3.03	40.22
microSD Samsung EVO 64Gb Plus XC I Grade 3 Class 10	256	3.02	84.71
microSD Samsung EVO 64Gb Select XC I Grade 3 Class 10	92	3.01	30.54

4.1.2 Performances système

Les diagrammes suivants présentent l'état du nano-ordinateur avant la segmentation, pendant et après. Les indicateurs qui sont observés sont ceux de la mémoire, la fréquence, le I/O, la consommation, la température. Afin de montrer l'impact potentiel de l'application *Chromium*, elle est démarrée entre deux segmentations, et pendant la segmentation.

La carte microSD *Scan Disk Ultra 32Gb class 10 HC I* a été utilisée pour les tests de performance système. La carte microSD *Samsung EVO 64Gb Plus class 10 HC I* n'était malheureusement plus fonctionnelle au moment des tests, celle-ci ayant été réservée pour tenter d'adapter l'architecture aux images terrain locales.

Le test infère en temps réel la vidéo qui est capturée avec la caméra du nano-ordinateur. Le réseau FCN qui est utilisé est celui fournit par NVIDIA *fcn-resnet18-deepscene-576x320*. Ce modèle détecte automatiquement la résolution la plus appropriée avec cette caméra, c'est-à-dire 30 images par seconde (FPS) et une résolution de 1280x720. Le test dure 1400 secondes, un peu de plus de 23 minutes. Il peut se diviser en onze périodes d'observation, qui sont brièvement décrites ci-dessous et représenter dans la Figure 18:

1. La première période est celle entre la 1^{re} seconde et la 200^e seconde, et qui permet d'observer l'état du système au démarrage du nano-ordinateur sans opération mise à part celle de la collecte des statistiques.
2. La seconde période est entre la 200^e seconde et la 400^e, et qui correspond à la première segmentation avec la caméra. Elle permet d'observer le système lors du premier démarrage de la segmentation.
3. La troisième période est celle entre la 400^e seconde et le premier démarrage de *Chromium*. Elle permet d'observer la réaction du système après l'arrêt de la segmentation.
4. La quatrième période est celle entre le premier démarrage de *Chromium* et son arrêt. Elle permet d'observer le comportement du système lors de l'utilisation de *Chromium*, qui est suspecté de ralentir le système, lorsqu'actif (observations faites durant l'essai).
5. La cinquième période est celle entre l'arrêt de *Chromium* et le démarrage de la seconde segmentation avec la caméra. Cette période permet d'observer la réaction du système après l'arrêt de *Chromium*.
6. La sixième période est celle entre le démarrage de la seconde segmentation avec la caméra et son arrêt. Cette période permet d'observer la réaction du système pendant la seconde segmentation.
7. La septième période est celle entre l'arrêt de la seconde segmentation et le démarrage de la troisième segmentation avec la caméra. Elle permet d'observer la réaction du système après l'arrêt de la segmentation la seconde fois.
8. La huitième période est celle entre le démarrage de la troisième segmentation et le démarrage de *Chromium* la seconde fois. Cette période permet d'observer la réaction du système pendant le démarrage de la segmentation la troisième fois.
9. La neuvième période est celle entre le deuxième démarrage de *Chromium* et son arrêt. Elle permet d'observer le comportement du système lors de l'utilisation de *Chromium* pendant l'inférence.
10. La dixième période est celle entre l'arrêt *Chromium* la seconde fois et l'arrêt de la troisième segmentation. Cette période permet d'observer la réaction du système après l'arrêt de *Chromium* pendant l'inférence.

11. La onzième période est celle entre l'arrêt de la troisième segmentation et l'arrêt du test et de la collecte des statistiques. Elle permet d'observer la réaction du système après l'arrêt de la segmentation la troisième fois.

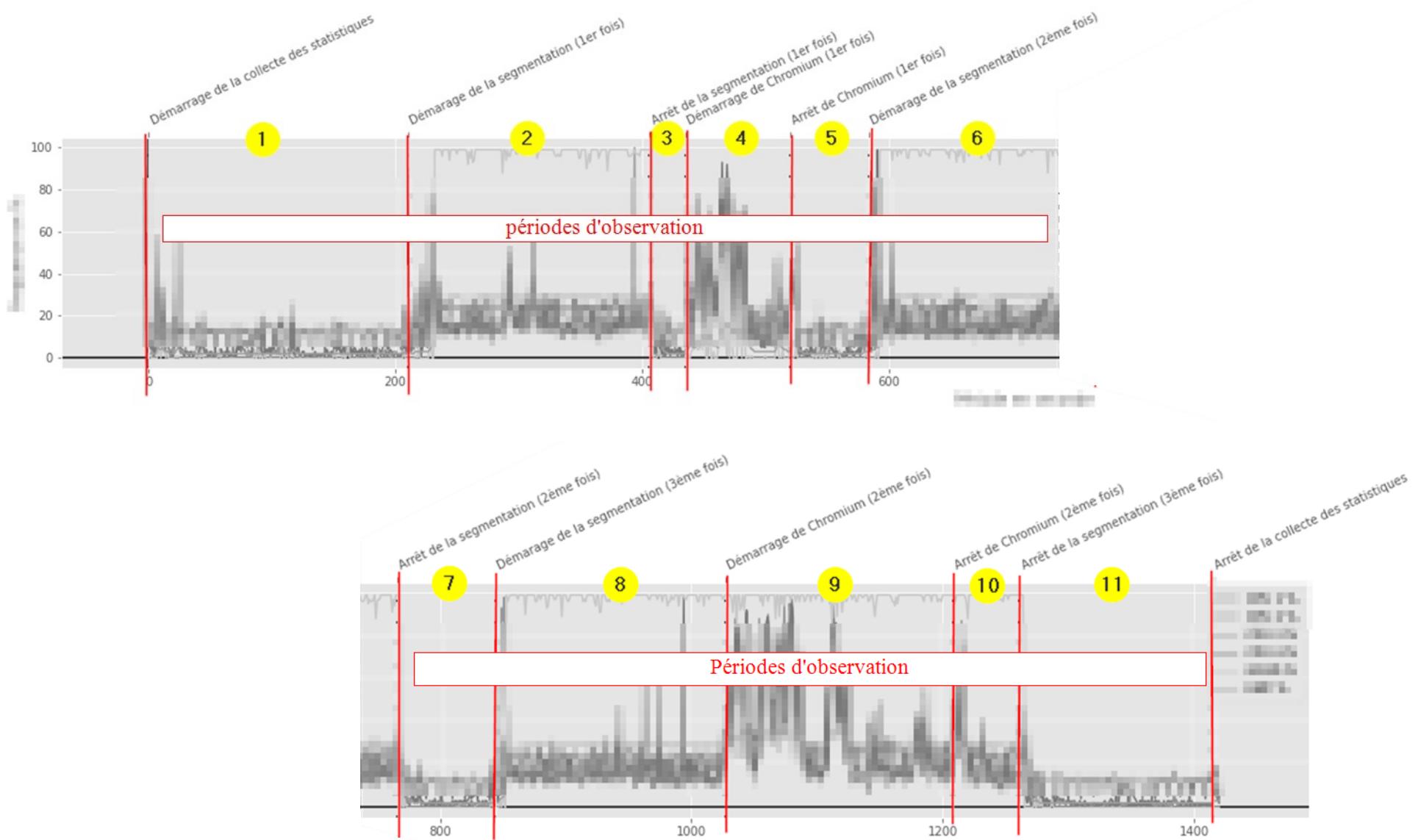


Figure 18: Présentation des périodes d'observation des performances système

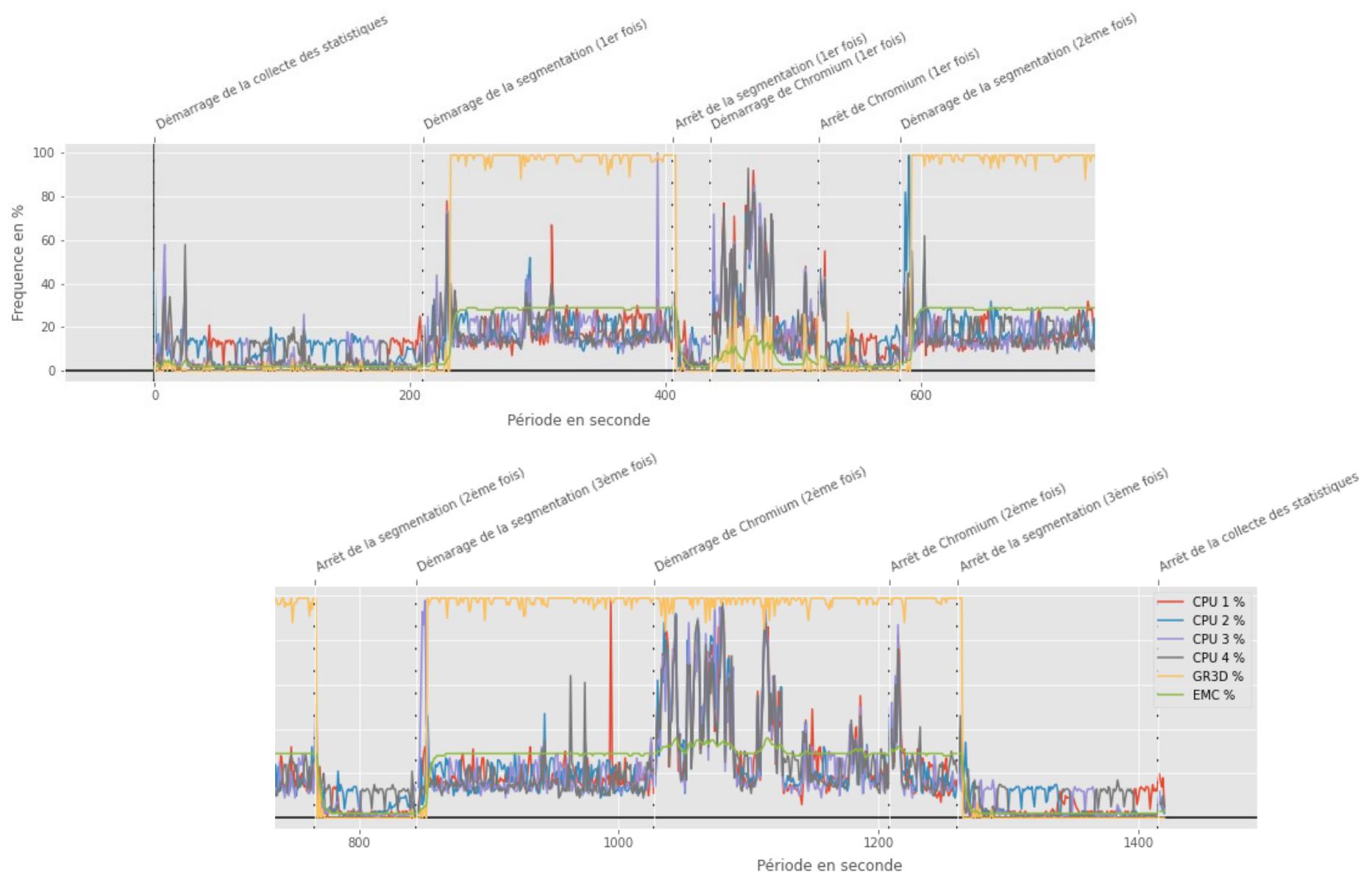


Figure 19 : Diagramme des performances système: la fréquence (haut et bas)

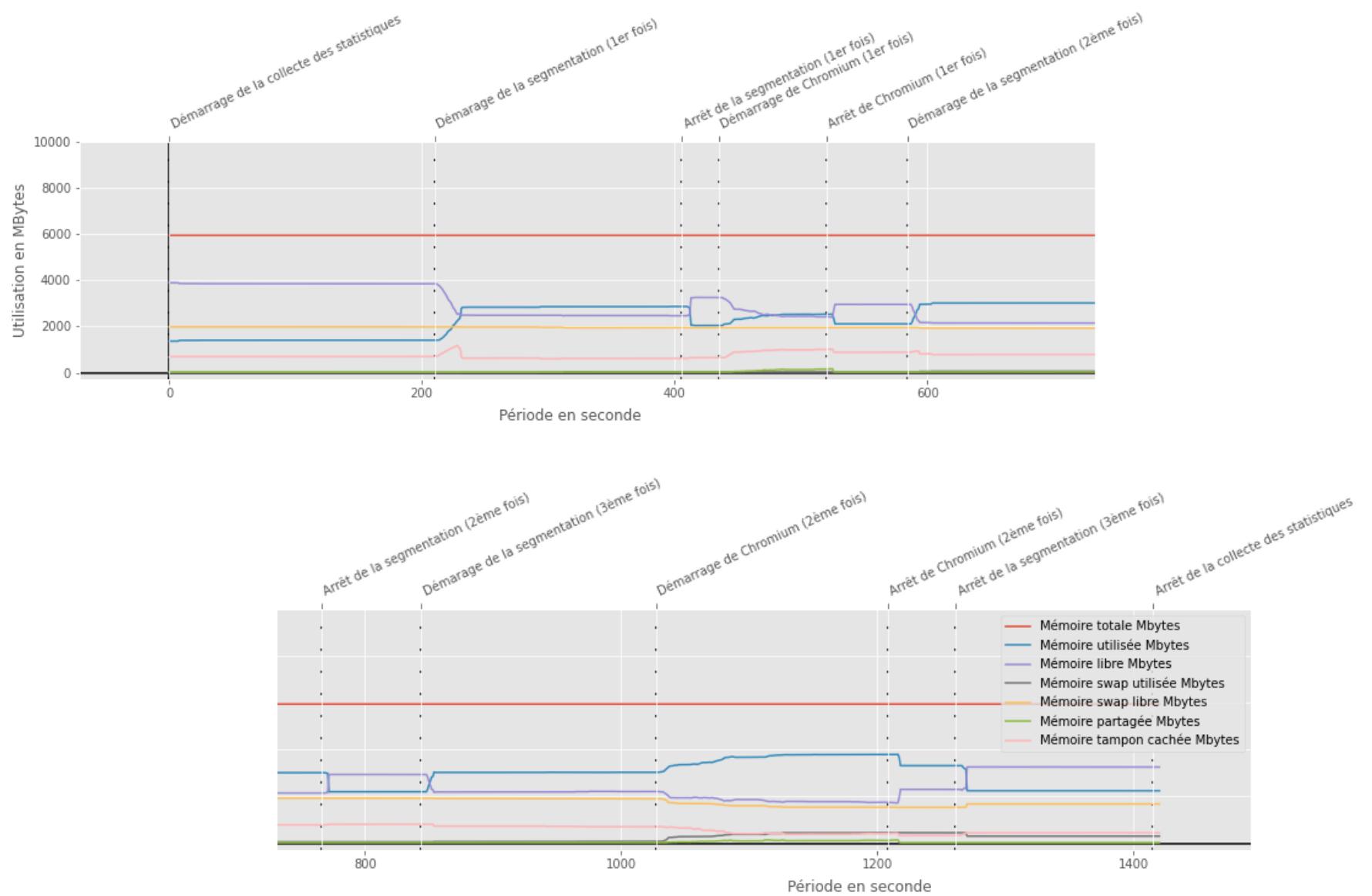


Figure 20 : Diagramme des performances système: la mémoire (haut et bas)

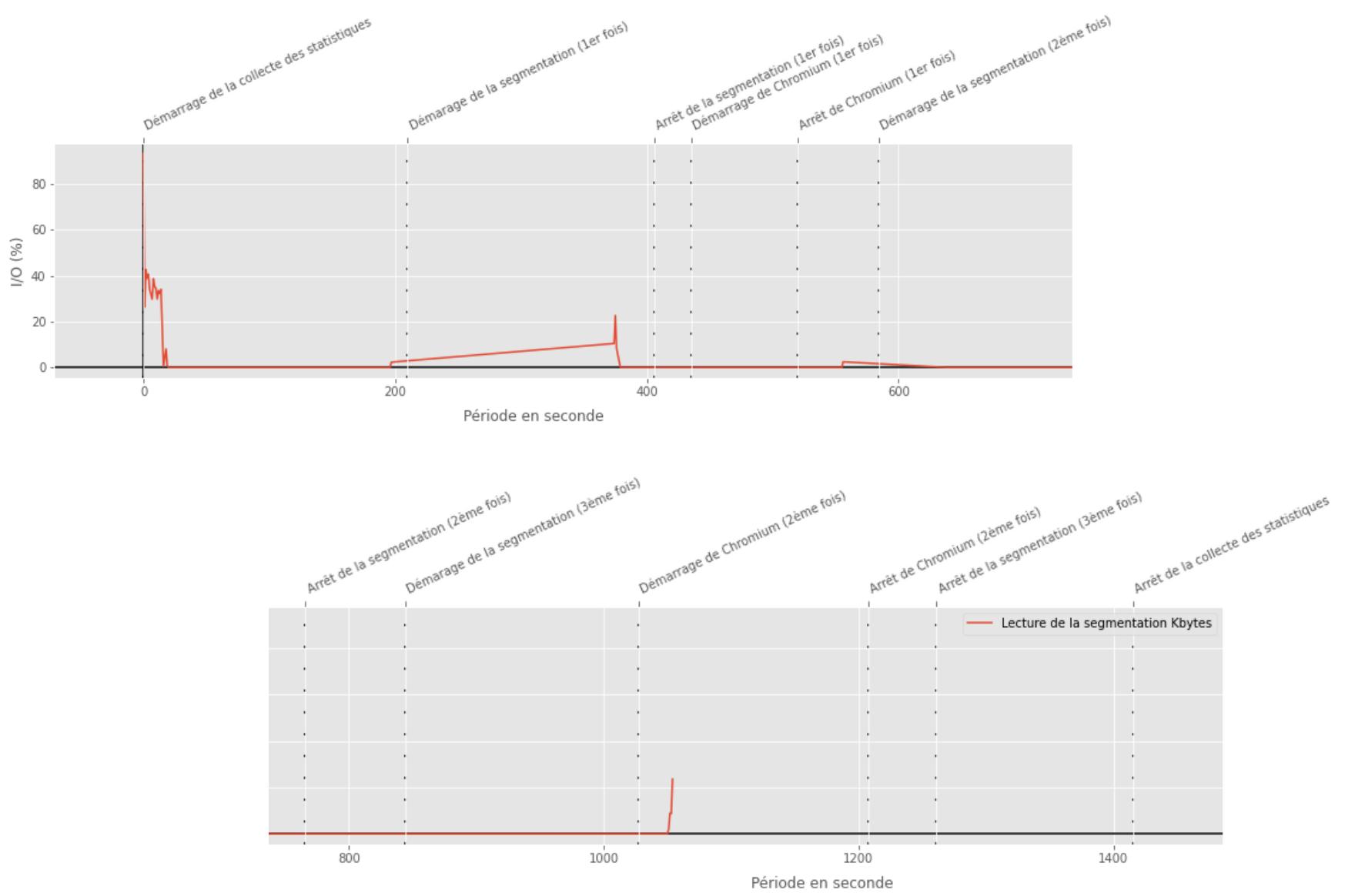


Figure 21 : Diagramme des performances système: le I/O total en % de la segmentation (haut et bas)

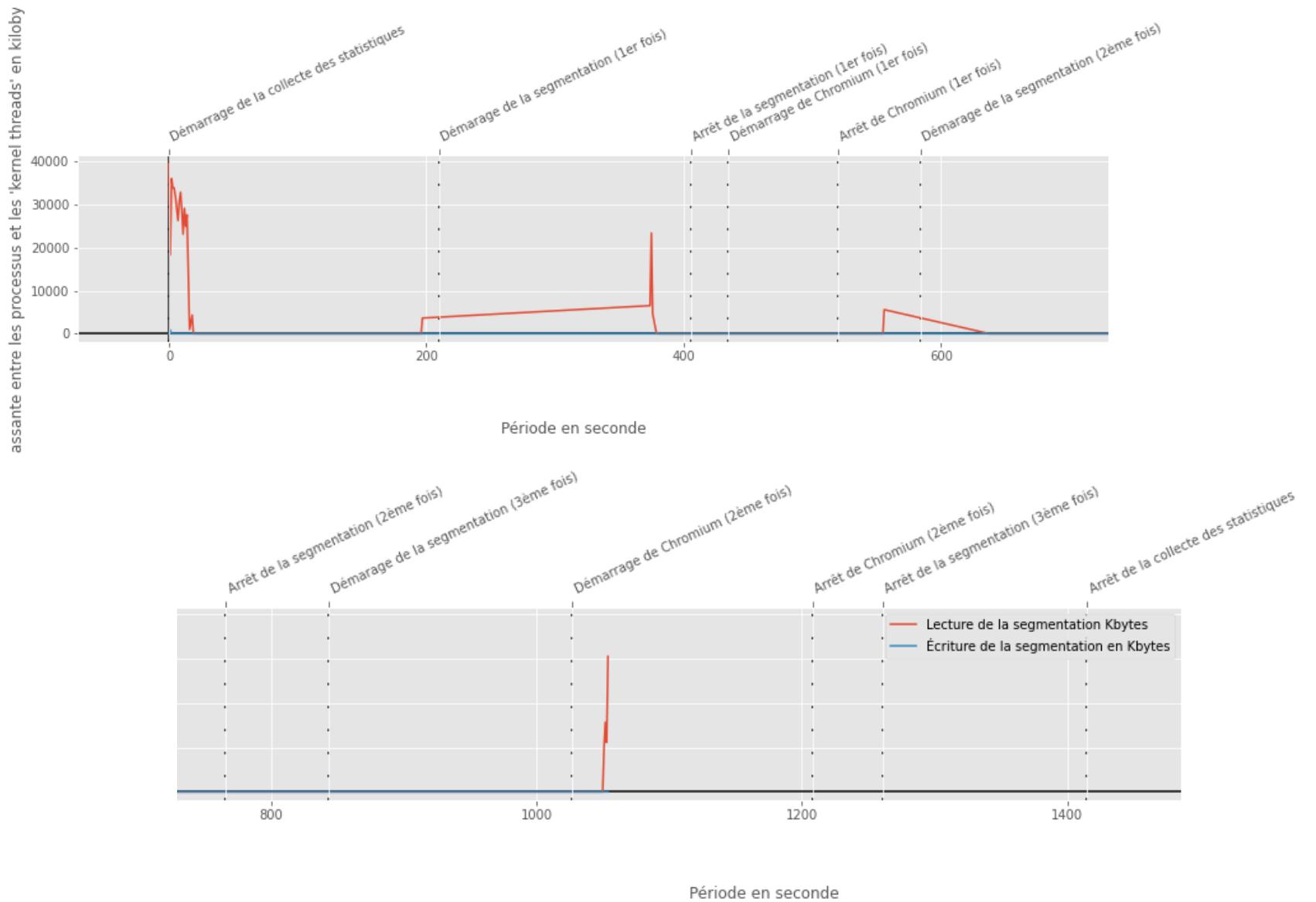


Figure 22: Diagramme des performances système: le I/O en KBytes de la segmentation (haut et bas)

assante entre les processus et les 'kernel threads' en kiloby

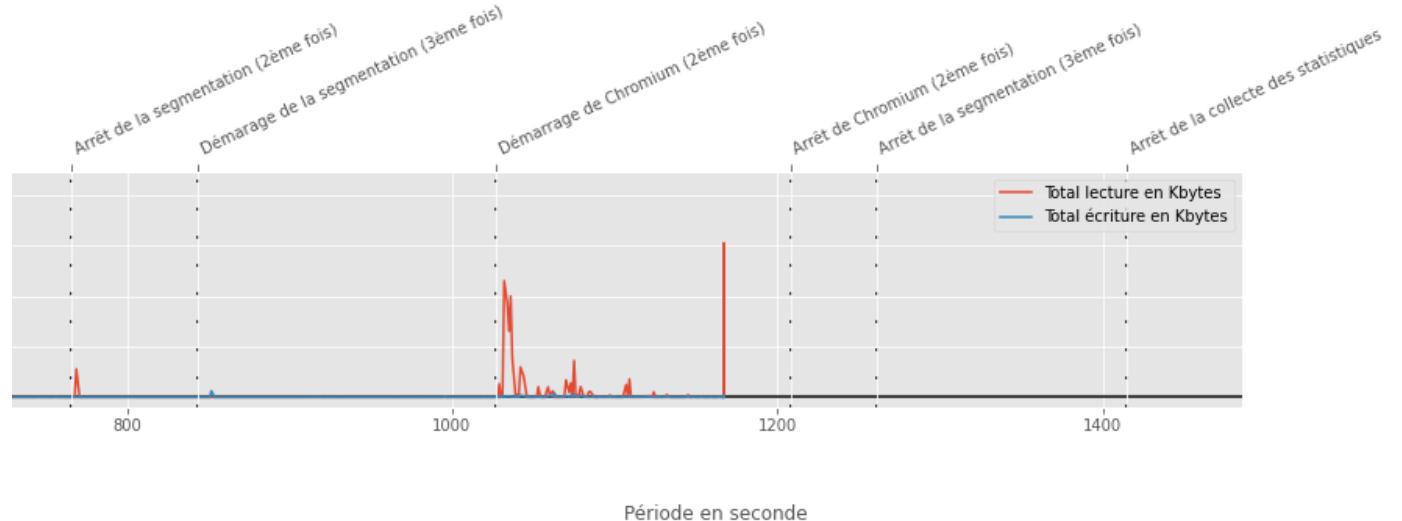
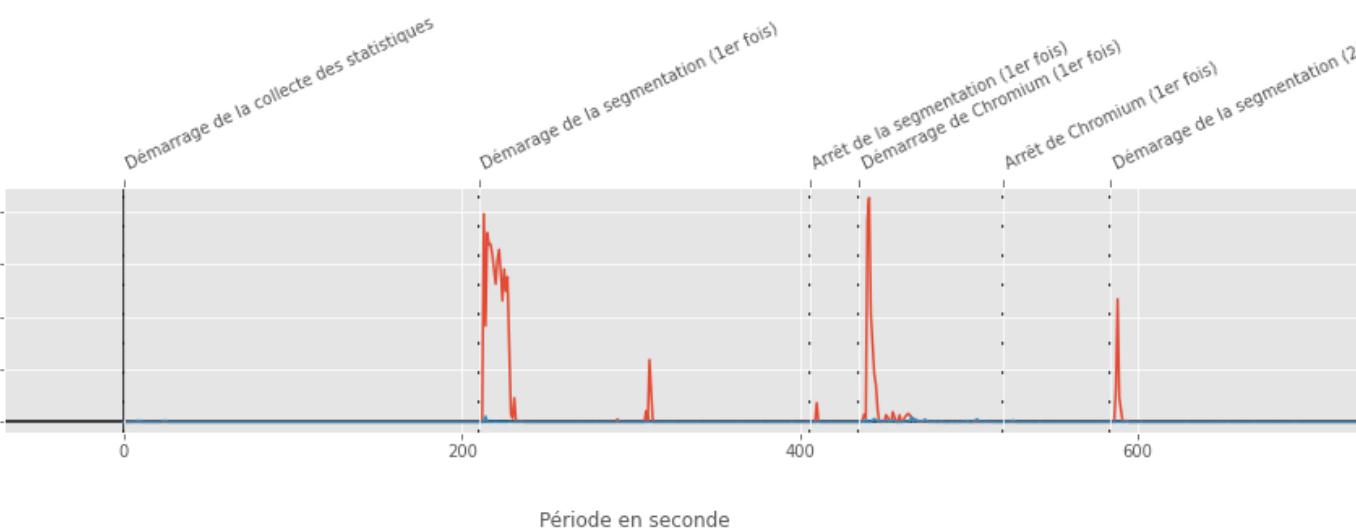


Figure 23: Diagramme des performances système : le I/O total du disque en KBytes (haut et bas)

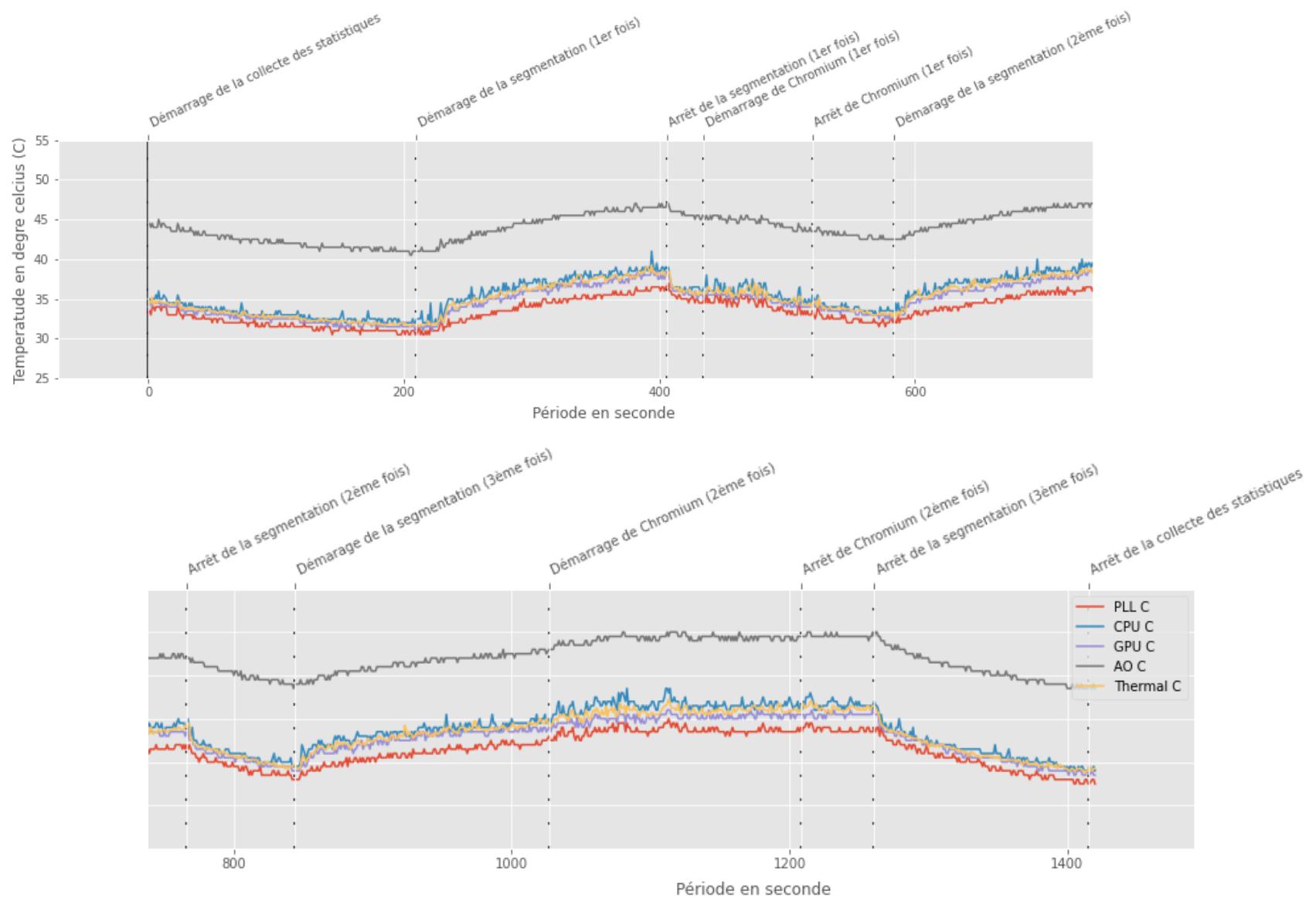


Figure 24 : Diagramme des performances système : les températures (haut et bas)

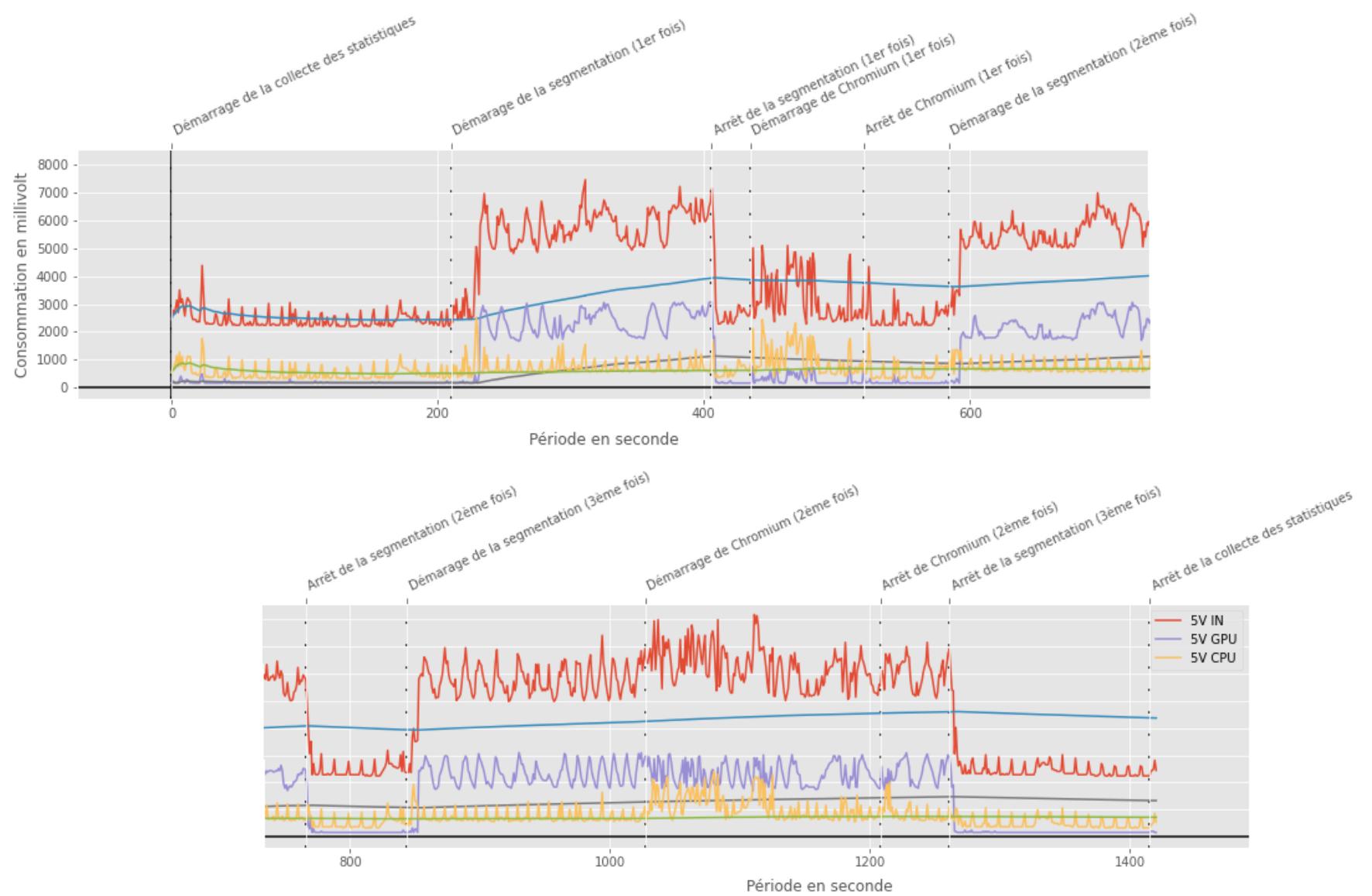


Figure 25: Diagramme des performances système: la consommation (*haut et bas*)

4.2 Performances de l'inférence

4.2.1 Images

Les tests ont été faits avec l'architecture *fcn-resnet18-deepscene-576x320* fournie par NVIDIA.

Lors de l'entraînement et l'inférence, le script montre un *IoU* moyen de 75 %. Mais l'objet d'intérêt de l'essai n'est pas la qualité de la segmentation de l'image complète, mais seulement de la piste cyclable. Certains efforts ont dû être dépensés³² afin de pouvoir observer le *IoU* et le *F1 score* de la segmentation sémantique de la piste cyclable uniquement.

Le résultat de la segmentation sémantique peut-être visualisé avec ces deux photos (Figure 26, Figure 27), prises du jeu de donnée de test de la forêt de Freiburg et utiliser comme jeu de données de test pour l'architecture. L'image utilisée possède une version vérité terrain. L'image générée est l'image prédictive et peut être comparée avec l'image vérité terrain, tant que la palette de couleur est identique à la version vérité terrain.

Il s'avère que le *IoU* et *F1 score* sont assez élevés pour les deux photos pour la classe *Chemin*.

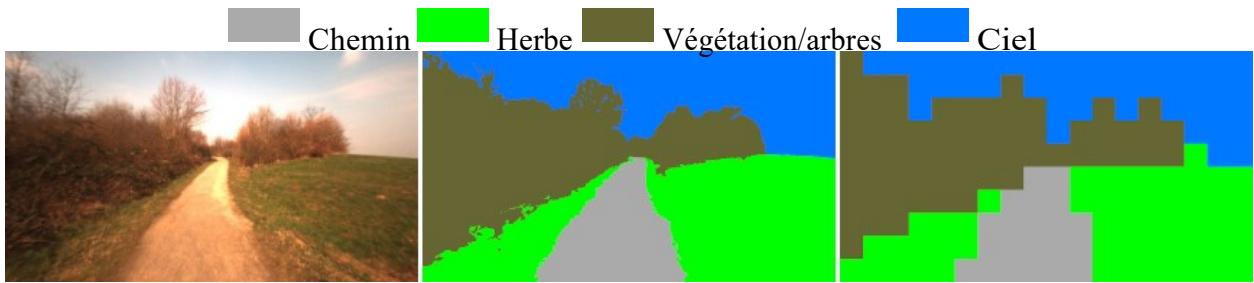


Figure 26: (gauche) Image originale (b1-09517); (centre) vérité terrain (GT); (droite) segmentation sémantique générée par l'architecture. Le *IoU* et le *F1 score* pour le chemin sont de +80 %.

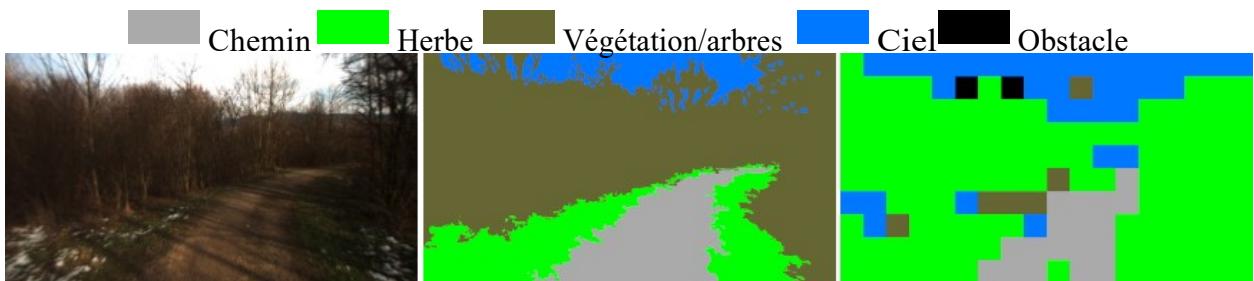


Figure 27: (gauche) Image originale (b378-61); (milieu) vérité terrain (GT); (droite) segmentation sémantique générée par l'architecture. Le *IoU* pour le chemin est +69 %.

³² https://github.com/vince7lf/vince7lf.github.io/blob/master/_notebooks/2020-06-21-image_pred_color.ipynb

4.2.2 Vidéos

Il y a deux captures vidéos qui ont été utilisées pour tester les performances de la segmentation d'une vidéo. Comme il n'est pas évident de montrer une vidéo dans un essai, des liens sont mis à disposition. Ces vidéos sont disponibles sur YouTube. Chaque vidéo a été créée en filmant avec un téléphone intelligent l'écran du nano-ordinateur pendant que la segmentation est exécutée. Cela produit une vidéo HD 1080p 30 FPS. Lors de la seconde vidéo, les performances du système et les statistiques *tigrastats* sont affichées en plus de la segmentation.

J'ai tenté de capturer le résultat (vidéos/images) de l'inférence directement depuis le nano-ordinateur, mais ce n'est pas une bonne idée, car trop intrusif, l'inférence est ralentie. Deux images sont produites par l'architecture : *overlay* et *mask*, qui sont directement rafraîchies dans une fenêtre *XWindow*.

- Lien ³³ vers une courte vidéo de 30 secondes démontrant l'inférence en temps réel de la segmentation sémantique d'une vidéo de la piste cyclable du pont Jacques-Cartier dans des conditions ensoleillées, mais avec un angle de vue qui change rapidement. Inférence effectuée en 30 FPS 1280x720 avec l'architecture *fcn-resnet18-deepscene-576x320*;
- Lien ³⁴ vers une vidéo longue de +8 minutes présentant l'inférence en temps réel de la segmentation sémantique d'une vidéo d'une piste cyclable dans des conditions ensoleillée, mais mouillée, avec présence de neige. Durée de plus de 8 minutes. La taille de la vidéo est de 800Mb. La même vidéo, d'une durée de 30 secondes, est utilisée successivement avec différentes images par seconde (60 / 30 / 15 / 1 FPS) et résolutions (720x1280 / 480x640 / 320x480 / 240x320). L'architecture est "fcn-resnet18-deepscene-576x320". Selon le titre de la fenêtre *XWindow* présentant la segmentation, le FPS est autour de 23-26 FPS.

³³ <https://youtu.be/hpb5JaShLtE>

³⁴ <https://youtu.be/yRVhMuqW1r4>

Le Tableau 8 montre les différentes résolutions et images par seconde (FPS) qui ont été testées avec l'architecture :

Tableau 8: Résolutions et images par seconde (FPS) testées

Résolutions qui fonctionnent
320x576, 480x640, 720x1280, 768x1024, 768x1152, 800x1152, 832x1024, 864x1024
Résolutions qui ne fonctionnent pas
832x1120, 832x1152, 768x1280, 800x1280, 864x1152, 900x1152, 900x1280, 960x1600, 1080x1920, 1024x1024
Images par seconde (FPS) supportées
60/1, 30/1, 15/1, 1/1

4.3 Réentraînement

Une tentative de réentraînement a été initiée. La première étape a été de vouloir régénérer le fichier ONNX tel que NVIDIA le fournit, sans autre effort de réentraînement. En effet les commandes fournies par NVIDIA pour segmenter une image ou une vidéo permettent de préciser une architecture personnalisée, tel que le fichier ONNX, les classes, les codes couleurs. L'idée est donc de bénéficier de cette possibilité. La seconde étape était de créer un jeu de données adapté au contexte, c'est-à-dire avec des photos de différentes sections de pistes cyclables avec une qualité de surface variable (sèche, mouillée, avec neige, ensoleillé, ombragé, etc.), avec les images fournies par l'APC-PJC et mon jeu personnel. Les difficultés attendues étaient d'uniformiser les résolutions des photos pour l'architecture *SegNet18*, mais surtout de créer des images vérités terrain, qui est beaucoup plus chronophage que difficile. Une fois le tout complété, la dernière étape est de re entraîné l'architecture *SegNet18* avec ce jeu, et régénérer le nouveau fichier ONNX.

Malheureusement la première étape, de régénérer le fichier ONNX, ne s'est pas déroulé aussi simplement qu'espéré, et a remis en question la suite du réentraînement. La génération du fichier ONNX a tout d'abord réussie assez facilement, mais une erreur à l'exécution a remis en question l'intégrité de ce fichier. Une longue période d'investigation a débuté³⁵³⁶, et finalement le fichier

³⁵ https://github.com/vince7lf/vince7lf.github.io/blob/master/_posts/2020-05-13-train-onnx-nano.md

³⁶ <https://forums.developer.nvidia.com/t/trying-to-regenerate-onnx-for-jetson-nano/125494?u=vincelf>

ONNX a pu être régénéré avec succès après la phase de réentraînement. Malheureusement le script fourni par NVIDIA qui adapte les photos du jeu de données de *DeepScene* pour le jeu d'entraînement et de test destiné au réentraînement de l'architecture *SegNet18*, génère des photos toutes noires, l'investigation s'est arrêtée à ce point.

5 Interprétation et discussion des résultats

5.1 Performances matérielles

5.1.1 Stockage de données

Les tests montrent que le SSD interne est de 4 à 11 fois plus efficaces qu'une carte microSD, pour l'opération de lecture de données.

5.1.2 Performances système

Le nano-ordinateur est capable d'exécuter l'inférence en temps réel pour une durée prolongée (23 minutes pendant nos tests), et rester réactif aux commandes. Il faut rester vigilant quant à l'utilisation des CPUs pendant l'inférence sur le long terme (Figure 19). La segmentation consomme de la mémoire qui semble ne plus être disponible pour les autres ressources du système par la suite (Figure 20). Le I/O de la segmentation est raisonnable (Figure 21, Figure 22, Figure 23), de même que celle du système. La température augmente lors de l'inférence mais revient à son point d'origine une fois complété (Figure 24). La consommation d'énergie est plus élevée pendant la segmentation, ce qui peut avoir une importance sur le budget en mode opérationnel continu (Figure 25).

Il est donc à noter que l'opérationnalisation constante de la segmentation aurait un impact non négligeable sur la durée de vie du Jetson Nano. Selon la documentation de NVIDIA (NVIDIA, 2020), une carte *Jetson Xavier TX2i* qui opère 24/7, selon certaines conditions, a une durée de vie théorique de 4,4 années.

5.2 Performances de la segmentation

5.2.1 Images

La segmentation prédite pour la classe *Trail* est assez surprenante. Le *IoU* est de 89 % et de 69

% respectivement dans le cas des deux images évaluées, ce qui est encourageant. Par contre les délimitations de la segmentation pour le chemin sont décevantes et questionnables, car l'architecture retourne une image de résolution faible, de 19 x 10 pixels. Le résultat visuel à l'écran est faussé grâce à un travail d'interpolation qui augmente la résolution et affine la segmentation de chaque classe, mais l'image brute peut difficilement être utilisée par un traitement informatique subséquent, par exemple pour réaliser des analyses précises de la surface de la piste cyclable.

5.2.2 Vidéos

La segmentation des vidéos n'a pas pu être évaluée avec des indicateurs de performances. C'est donc subjectivement que l'on peut donner une appréciation. Comme les images se succèdent rapidement, ce n'est pas non plus évident. Les vidéos proviennent d'un contexte autre que celui du jeu de données d'entraînement (la forêt), la surface de la piste est bien différente.

Je me questionne sur l'utilisation de la segmentation de vidéos en temps réel, car il n'y a pas moyen d'évaluer la qualité de la segmentation si la vérité terrain n'est pas disponible.

6 Conclusion et recommandations

6.1 Objectif principal

L'objectif principal de l'essai était d'évaluer la capacité du nano-ordinateur NVIDIA Jetson Nano à exécuter, en temps réel, une architecture FCNN permettant la segmentation sémantique d'une vidéo d'une piste multifonctionnelle. Il faut découper en plusieurs faits cet objectif afin de bien pouvoir l'évaluer :

- Le nano-ordinateur est capable de segmenter sémantiquement une vidéo représentant une piste cyclable grâce à une architecture FCN.
- La segmentation sémantique découlant de l'inférence n'a pas pu être mesurée, il n'y a aucun moyen qui m'est connu afin de récupérer un indicateur, un coefficient ou un score me permettant de juger si la segmentation d'une vidéo est bonne ou non, comme pour une image ou la mesure du *IoU* ou du *F1 score* est possible si l'image de la vérité terrain est disponible.
- L'image générée par l'architecture FCN *SegNet18* a une résolution faible, de l'ordre de 19 x 10 pixels. La délimitation de la segmentation, entre chaque classe, est donc grossière.

- Le temps réel a été simulé, et n'est donc pas celui qui sera utilisé sur le terrain.
- Le nano-ordinateur et l'architecture FCN supportent l'inférence d'une vidéo HD (résolution de $720 \times 1280 = 720p$) avec un nombre d'images par seconde de 60/1 FPS.

D'un point de vue de la performance matérielle et logicielle, le nano-ordinateur est capable de segmenter une vidéo avec une architecture FCN. Par contre, d'un point de vue qualitatif, 1) la qualité de la segmentation ne peut pas être mesurée. De plus, 2) la segmentation prédite est imprécise.

La première limitation qualitative semble être un défaut majeur. Mais si on replace l'objectif dans le contexte de la détection de la délimitation d'une piste cyclable, à partir d'un point de vue fixe, on peut s'interroger sur le besoin de faire de la télédétection en temps réel avec une vidéo en haute résolution.

La seconde limitation pourrait théoriquement être améliorée en utilisant un modèle dont l'architecture est plus performante, mais implicitement plus complexe, telle que l'architecture *SegNet101* ou *DeepLabV3*, mais qui risque d'être aussi plus demandant en ressources matérielles, GPU, CPU et mémoire. Ce qui risque de remettre en question les performances matérielles et logicielles du nano-ordinateur. C'est ainsi probablement la raison pour laquelle NVIDIA procure uniquement des jeux de modèles préentraînés de segmentation sémantique avec *SegNet18* pour le nano-ordinateur.

6.2 Limites

6.2.1 Limites matérielles

Au sujet des limites matérielles, durant l'inférence, il n'y a aucune limite qui est ressortie lors des tests de performance. Selon la documentation de NVIDIA (NVIDIA, 2020), un mode opérationnel 24/7 offre une durée de vie de 4.4 années au nano-ordinateur.

6.2.2 Limites applicatives

Au sujet des limites applicatives, durant l'inférence, il n'y a aucune limite qui est ressortie lors des tests de performance. Par contre, il a été observé durant l'essai que le nano-ordinateur ne devrait pas être utilisé comme machine de développement, pour par exemple pour réentraîner une architecture. L'entraînement de l'architecture *SegNet18* n'a pas fonctionné dans un *environnement virtuel Python*, ni dans un conteneur *Docker* sur le nano-ordinateur, celui-ci arrête de fonctionner.

Il n'y a pas eu d'investigation, mais il semble que le nano-ordinateur atteint une limite mémoire qui le ralentit jusqu'à un arrêt de fonctionnement. DIGITS ne peut pas non plus être utilisé, car il n'est pas compatible avec l'architecture ARM du nano-ordinateur. Si l'objectif est d'améliorer l'architecture en réentraînant à la demande en mode opérationnel, l'entraînement et l'inférence ne peuvent cohabiter simultanément, cela me semble donc impossible aujourd'hui, à moins d'investiguer et de trouver un moyen d'optimiser les ressources.

Durant l'essai, il a aussi été observé que l'utilisation prolongée de *Chromium* peut impacter les performances du nano-ordinateur en le ralentissant grandement.

6.3 Optimisation

6.3.1 Optimisation matérielle

Plusieurs initiatives ont été tentées afin d'optimiser le matériel. L'optimisation requise est celle d'utiliser un adaptateur 5 V 4 A, recommandé et fiable, afin de fournir assez de puissance au nano-ordinateur lorsque d'autres périphériques viennent s'y raccorder, comme une caméra et un ventilateur. Profiter du PoE de l'interface réseau n'a pas été testé, mais cela semble aussi être une option rapide et simple à mettre en place pour assister l'adaptateur. Enfin, forcer le démarrage du ventilateur dès le démarrage du nano-ordinateur est une autre optimisation simple, mais efficace à appliquer. Par contre, je ne recommande pas l'utilisation d'un dongle ou adaptateur Wifi, celui-ci étant énergivore, peu efficace, non fiable, ni stable. Il prendrait de plus un pourcentage d'utilisation non négligeable du Hub USB 3.0.

La seconde optimisation qui a été tentée est celle d'utiliser un SSD à la place d'une microSD, car il y aurait beaucoup d'avantages. Pour des raisons de performances d'abord, le gain peut-être d'au moins 4 fois plus grand en opération de lecture I/O. Ensuite, en durée de vie, une carte microSD est fragile et ne peut être considérée comme un système fiable sur le long terme. D'un point de vue capacité de stockage, un SSD peut offrir beaucoup mieux. Enfin, un SSD est plus adapté à la gestion d'un système opérationnelle et la manipulation de petits fichiers. En contrepartie, un SSD va demander plus de puissance au nano-ordinateur, et générer plus de chaleur. Ma recommandation serait de trouver un disque SSD interne au format NVMe, connecteur de type M.2, qui peut être facilement branché au port PCIe du nano-ordinateur.

Une autre optimisation matérielle qui n'est pas à négliger est le boîtier. Vu que le système a été conçu pour être en opération continue sur le terrain, le boîtier doit permettre de le protéger sur le

long terme. Il doit être bien adapté à ses périphériques, que sont la caméra et le ventilateur, et optionnellement un SSD interne.

6.3.2 Optimisation logicielle

La version de l'architecture *SegNet18* fournie par NVIDIA s'exécute avec fluidité, sans que l'on sente que le nano-ordinateur puisse devenir non réactif. Au démarrage de l'inférence, il y a une brève période de 2-3 secondes où le nano-ordinateur ne répond plus. Mais sinon, il est tout à fait possible d'utiliser le nano-ordinateur pendant l'inférence d'une vidéo ou avec la caméra, et même avec 5-6 onglets d'ouverts dans *Chromium*. Lorsque le nombre d'onglets, ou d'instances de *Chromium*, devient trop grand, il a été observé que le nano-ordinateur devenait lent, limite non fonctionnel, jusqu'à la fermeture des onglets. Ceci est probablement dû à une limitation mémoire.

Autrement, certaines corrections au code C++ ont dû être apportées au code source original fourni par NVIDIA : l'image de la caméra est à l'envers (et je ne pouvais monter la caméra dans le sens opposé dans le boîtier); le pipeline *gstreamer* interne de l'application est trop spécifique pour supporter un flux vidéo autre que celui provenant de la caméra; et la taille de la fenêtre *XWindow* qui s'ouvre pour afficher la segmentation de la vidéo est programmée pour prendre tout l'écran, nous faisant perdre ainsi l'accessibilité et visibilité aux autres fenêtres.

a) Segmentation

Comme observé durant les tests, la résolution de la segmentation avec l'architecture *SegNet18* est faible, 19 x 10 pixels. Le désavantage majeur dans le contexte de cet essai est que les délimitations des classes sont approximatives, incluant celle du chemin. Même si le *IoU* et le *F1 score* sont pourtant acceptable pour cette classe. Il semble que ce serait l'élément prioritaire à améliorer.

b) Réentraînement

Même si la phase de réentraînement a pu être initiée durant l'essai, elle n'a pas duré longtemps : régénérer le même fichier interopérable ONNX avec le code source original a été laborieux. Il est vrai que NVIDIA propose, avec DIGITS, un environnement de re entraînement des architectures qu'ils offrent. Mais dans le contexte de cet essai, je n'avais à ma disposition que l'environnement de Calcul Québec, qui n'est pas compatible avec DIGITS. Néanmoins je pense qu'il est important de pouvoir le faire tout en gardant le contrôle de son environnement, par exemple pour permettre d'adapter l'architecture de notre choix, plus performante, telle que *SegNet101* ou *DeepLabV3*,

entraîné avec le jeu de données *DeepScene*, et l'adapter à un jeu de données personnalisé. Le questionnement est de savoir comment le nano-ordinateur réagit avec une architecture beaucoup plus grosse et complexe que *SegNet18*. Dans une autre perspective, il serait bon de considérer un modèle de nano-ordinateur plus performant, tel que le *Jetson Xavier AGX*.

6.4 Documentation

La documentation des activités, des procédures, des scripts, des modifications, des erreurs, des références est disponible publiquement dans le blogue sur *GitHub*³⁷.

³⁷ <https://github.com/vince7lf/vince7lf.github.io>

Références

- Abouzahir, S., Sadik, M. & Sabir, E. (2017). IoT-Empowered Smart Agriculture: A Real-Time Light-Weight Embedded Segmentation System. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 319-332. https://doi.org/10.1007/978-3-319-68179-5_28
- Arduino (2021) <https://www.arduino.cc/>
- Alom, Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P. & Nasrin, M. S. (2018). The History Began from AlexNet : A Comprehensive Survey on Deep Learning Approaches, 39.
- Association des piétons et cyclistes du pont Jacques-Cartier. (2020). PontJacques-Cartier365.com. <http://pontjacquescartier365.com>
- Association des piétons et cyclistes pont Jacques-Cartier (2020). Flickr <https://www.flickr.com/photos/pontjacquescartier>
- Beam, A. (2017). *Deep Learning 101 - Part 1 : History and Background*. https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html
- Bernas, M., Placzek, B. & Sapek, A. (2017). Edge Real-Time Medical Data Segmentation for IoT Devices with Computational and Memory Constrains. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 119-128. https://doi.org/10.1007/978-3-319-67077-5_12
- Blanco-Filgueira, B., García-Lesta, D., Fernández-Sanjurjo, M., Brea, V. M. & López, M. (2019). Deep Learning-Based Multiple Object Visual Tracking on Embedded System for IoT and Mobile Edge Computing Applications. *IEEE Internet of Things Journal*, 5423-5431. <https://doi.org/10.1109/JIOT.2019.2902141>
- Catarinucci, L., R. Colella, S. I. Consalvo, L. Patrono, C. Rollo, and I. Sergi. (2020). IoT-Aware Waste Management System Based on Cloud Services and Ultra-Low-Power RFID Sensor-Tags. *IEEE Sensors Journal* 20 (24): 14873-14881. doi:10.1109/JSEN.2020.3010675
- Chollet, F. (2018). *Deep learning with Python* [OCLC : ocn982650571]. Manning Publications Co.

- Chong, C. P., Salama, C. A. T. & Smith, K. C. (1992). Real-Time Edge Detection and Image Segmentation. *Analog Integrated Circuits and Signal Processing*, 117-130. <https://doi.org/10.1007/BF00142412>
- Copel, M. (2016). *What's the Difference Between Deep Learning Training and Inference?* <https://blogs.nvidia.com/blog/2016/08/22/difference-deep-learning-traininginference-ai/>
- Cornioley, P. (2018). Intégration d'un module d'apprentissage profond dans l'architecture logicielle d'un SIG Web, 90.
- Dettmers, T. (2015). *Deep Learning in a Nutshell : History and Training*. <https://devblogs.nvidia.com/deep-learning-nutshell-history-training/>
- Dubey, A., & Shanmugasudaram, M. (2020). Agricultural plant disease detection and identification. *International Journal of Electrical Engineering and Technology*, 11(3), 354-363. doi:10.34218/IJEET.11.3.2020.038
- Dustin, F. (2019). *Realtime Semantic Segmentation on Jetson Nano in Python and C++*. <https://www.linkedin.com/pulse/realtime-semantic-segmentation-jetson-nanopython-c-dustin-franklin>
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, et Jian Sun. (2015). *Deep Residual Learning for Image Recognition*. *arXiv:1512.03385 [cs]*, décembre. <http://arxiv.org/abs/1512.03385>.
- Intel Corporation. (2021). Intel® Neural Compute Stick 2 (Intel® NCS2). <https://www.intel.com/content/www/us/en/developer/tools/neural-compute-stick/overview.html>
- Jiaconda. (2019). *A Concise History of Neural Networks*. <https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec>
- Kaggle (2020) <http://kaggle.com>
- Kilby, J. S. (2000). The Nobel Prize in Physics 2000. Récupérée 9 octobre 2021, à partir de <https://www.nobelprize.org/prizes/physics/2000/kilby/lecture/>
- Koh, J. Y. (2018). *Model Zoo - Deep Learning Code and Pretrained Models for Transfer Learning, Educational Purposes, and More*. <https://modelzoo.co/>
- Kurenkov, A. (2015). *A 'Brief' History of Neural Nets and Deep Learning*. <https://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deeplearning/>

- Long, J., Shelhamer, E. & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431-3440. <https://doi.org/10.1109/CVPR.2015.7298965>
- Mody, M., Kumar, D., Swami, P., Mathew, M. & Nagori, S. (2018). Low Cost and Power CNN/Deep Learning Solution for Automated Driving. *Proceedings - International Symposium on Quality Electronic Design, ISQED*, 432-436. <https://doi.org/10.1109/ISQED.2018.8357325>
- ModelZoo (2020) <http://modelzoo.co>
- Nguyen, T., Shivakumar, S. S., Miller, I. D., Keller, J., Lee, E. S., Zhou, A., Ozaslan, T., Loianno, G., Harwood, J. H., Wozencraft, J., Taylor, C. J. & Kumar, V. (2019). MAVNet : An Effective Semantic Segmentation Micro-Network for MAV-Based Tasks. *arXiv* :1904.01795[cs]. <http://arxiv.org/abs/1904.01795>
- NVIDIA. (2019a). *Jetson Nano*. <https://developer.nvidia.com/embedded/jetson-nano>
- NVIDIA. (2019b). *Jetson Nano: Deep Learning Inference Benchmarks*. <https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks>
- NVIDIA. (2020). NVIDIA Jetson Linux Developer Guide : Jetson Module Support | NVIDIA Docs. Récupérée 9 octobre 2021, à partir de https://docs.nvidia.com/jetson/archives/l4t-3242/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/jetson_module_support.html
- NVIDIA. (2021a). *Jetson Nano*. <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/>
- NVIDIA. (2021b). *Jetson TX2*. <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/>
- NVIDIA. (2021c). *Jetson Xavier*. <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/>
- Pathak, D. & El-Sharkawy, M. (2019). Architecturally Compressed CNN : An Embedded Realtime Classifier (NXP Bluebox2.0 with RTMaps). *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 0331-0336. <https://doi.org/10.1109/CCWC.2019.8666495>

PJCCI. (2018a). Fiche de la piste multifonctionnelle du pont Jacques-Cartier.

https://jacquescartierchamplain.ca/wp-content/uploads/2018/10/IMG_Fiche_piste-multi_pont_JC_FR_vfinale_web_2018-10-10.pdf

PJCCI. (2018b). Rapport post-mortem sur le projet pilote d'entretien hivernal de la piste multifonctionnelle du pont Jacques-Cartier. https://jacquescartierchamplain.ca/wp-content/uploads/2018/10/RPP_piste_PJC_2018-10-10-1.pdf

Raspberry Pi Foundation. (2021). <https://www.raspberrypi.org/>

Rodriguez-Conde, I., C. Campos, et F. Fdez-Riverola. (2021). On-Device Object Detection for More Efficient and Privacy-Compliant Visual Perception in Context-Aware Systems. *Applied Sciences (Switzerland)* 11 (19). <https://doi.org/10.3390/app11199173>

Valada, A., Oliveira, G., Brox, T. Burgard, W. (2016). Deep Multispectral Semantic Scene Understanding of Forested Environments using Multimodal Fusion. *International Symposium on Experimental Robotics (ISER)*. <http://deepscene.cs.uni-freiburg.de/>

Sharma, N., Shamkuwar, M. & Singh, I. (2019). *The History, Present and Future with IoT*. Springer Science; Business Media Deutschland GmbH. https://doi.org/10.1007/978-3-030-04203-5_3

TensorFlow. (2020). *DeepLabV3*. <https://github.com/tensorflow/models/tree/master/research/deeplab>

Wu, X., Sahoo, D. & Hoi, S. C. H. (2019). Recent Advances in Deep Learning for Object Detection [arXiv : 1908.03673]. *arXiv :1908.03673 [cs]*. Récupérée 9 août 2020, à partir de <http://arxiv.org/abs/1908.03673>

Xie, Q. (2021). UI Design of Visual Communication of Coastal City Landscape Based on Embedded Network System and Remote Sensing Data. *Arabian Journal of Geosciences* 14 (11). doi:10.1007/s12517-021-07167-3.

Zheng, J., Li, J., Liu, Y. & Zhang, W. (2020). Real-Time Semantic Segmentation Network for Edge Deployment. In Y. Jia, J. Du & W. Zhang (Éd.), *Proceedings of 2019 Chinese Intelligent Systems Conference* (p. 243-249). Springer Singapore. https://doi.org/10.1007/978-981-32-9698-5_28

7 Annexes

7.1 Exemples de nano-ordinateurs qui supportent les SDK pour l'IA

Tableau 9: Comparaison des trois nano-ordinateurs supportant les SDK pour l'IA

NVIDIA Jetson Nano	NVIDIA Jetson Xavier AGX	Raspberry Pi 4B + Intel NCS2
99USD	599USD	134USD (55USD + 79USD)
45 x 69.6 mm, 250 gr, 5-10 W	100 x 87 mm, 630 gr, 10-15-30 W	56 x 85.60 mm + 27x72 mm, 45 gr + 18.1 gr, 15 W
128-core NVIDIA Maxwell GPU	512-core NVIDIA Volta GPU with 64 Tensor Cores	Intel Movidius Myriad X VPU 16 SHAVE cores
Quad-Core ARM Cortex-A57 MPCore	8-core NVIDIA Carmel Arm v8.2 64-bit CPU 8MB L2 + 4MB L3	Quad-core ARM Cortex-A72 64-bit @ 1.5 GHz
4 GB 64-bit LPDDR4	32 GB 256-bit LPDDR4	4GB LPDDR4
0.47 TFLOPS@FP16	5.5-11.5 TFLOPS@FP16; 20-32 TOPS@INT8	4 FLOPS@FP16, 1 TOPS@INT8

7.2 Résumé des différents tests de configuration du nano-ordinateur avec les disques SSD

Concernant le disque SSD M.2 NVMe connecté à la carte d'extension M.2 via le Hub USB 3.0 interne, le système L4T de NVIDIA ne supporte pas les SSD M.2 NVMe connecté au port USB³⁸. Il n'est pas reconnu ou détecté automatiquement par le système d'exploitation et ne peut donc pas être utilisé rapidement ("plug & play" en anglais). Comme il serait risqué pour l'essai de se lancer dans la recompilation du noyau ("kernel") du L4T, une alternative trouvée sur le développeur forum de NVIDIA est de passer par un adaptateur M.2 NVMe connecté au port PCIe interne.

³⁸ À noter que la carte d'extension T100 est discontinued et remplacée par la T130

Malheureusement cette alternative a rapidement été abandonnée. Il a été possible de démarrer et installer le système d'opération sur le SSD M.2, et faire quelques tests, mais pour une raison inconnue, le système n'était pas stable et devenait non opérationnel assez rapidement, le système perdant la connexion au SSD. La durée la plus longue de stabilité observée a été de moins 30 minutes. Une hypothèse est une baisse d'énergie qui survient à un moment et qui impacte l'alimentation du SSD, chaque volt et milliampère étant important pour la stabilité du nano-ordinateur. De plus, le raccordement du câble de la carte d'extension M.2 NVMe PCIe avec le SSD M.2 NVMe est compliqué et risqué pour le câble lui-même. Une autre limitation importante est que cette solution ne permet pas d'utiliser le boitier, car le SSD M.2 ne rentre pas et ne peut même pas être fixé.

Différentes options pour optimiser l'alimentation ont été explorées : a) utiliser un HUB USB externe et auto alimenté; b) brancher un câble Ethernet au lieu d'utiliser un Dongle Wifi; c) allumer le ventilateur dès le démarrage du nano-ordinateur; d) et l'option de fournir 6 A directement supportée par la carte mère via les pins; e) explorer d'autres solutions sur les forums de discussion³⁹⁴⁰.

7.3 Communication avec l'Association des Piétons et Cyclistes du Pont Jacques-Cartier

L'Association des Piétons et Cyclistes du Pont Jacques-Cartier (APC-PJC) a été contacté afin de leur demander la permission d'utiliser leurs fichiers multimédias de la piste cyclable du pontJacques-Cartier, tel que leurs images et leurs vidéos. Voici les détails de la communication et les conditions d'utilisation.

7/31/2020

Courriel - Vincent Le Falher - Outlook

RE: Bonjour !

Mickaël Germain <Mickael.Germain@USherbrooke.ca>

Mer 2020-02-19 23:30

À : Vincent Le Falher <Vincent.Le.Falher@USherbrooke.ca>; Piétons-cyclistes pont Jacques-Cartier <apc.pontjc@gmail.com>

Bonjour,

Merci pour les informations. Nous n'avons pas d'entente avec PJCCI pour le rapport de maîtrise. Nous ne partagerons pas vos informations sans votre accord.

Cordialement,
Mickaël

De : Vincent Le Falher <Vincent.Le.Falher@USherbrooke.ca>
Envoyé : 18 février 2020 17:55

³⁹ <https://www.kingston.com/en/community/articledetail/articleid/48543>

⁴⁰ <https://geekworm.com/products/nvidia-jetson-nano-nvme-m-2-ssd-shield-t100-v1-1>

À : Piétons-cyclistes pont Jacques-Car er Cc : Mickaël Germain Objet : Re: Bonjour !

Bonjour M. Démontagne,

Je copie mon directeur de projet pour le no fier des condions entourant l'usage des fichiers médias (photos et vidéos) que vous nous permettez de lire aux fins de mon essai de recherche pour étude.

Je comprends vos demandes et je les appliquerai en bonne et due forme, cela me fera plaisir.

@Mickaël stp noter qu'il est important de ne fier l'association si les fichiers médias sont utilisés par d'autres étudiants, *surtout* dans le contexte du projet avec PJCCI.
+ De plus sais-tu s'il existe une entente ou convention avec PJCCI par rapport à l'utilisation du rapport de maîtrise ?

@M. Démontagne je vous remercie encore pour votre aide, et, si vous le désirez, je pourrais vous tenir au courant de l'évolution de mon projet. Mon objectif est de compléter mon essai avant la mi-août 2020.

Au plaisir de discuter de nouveau avec vous, au besoin.

Vincent

--

Vincent Le Falher
514-229-3863

De : Piétons-cyclistes pont Jacques-Car er <apc.pontjc@gmail.com>

Envoyé : 18 février 2020 17:11

À : Vincent Le Falher <Vincent.Le.Falher@Usherbrooke.ca> Objet : Re: Bonjour !

Bonjour Vincent,

Suite à notre conversation de ce mardi midi, je vous autorise à utiliser les photos et vidéos disponibles sur le [compte Flickr](#) de l'Association ainsi qu'aux vidéos disponibles sur notre [compte YouTube](#).

Le dossier complet (photos, vidéos) contient environ 250 Go de données (classement par années). Celui qui correspond sensiblement aux données du compte Flickr environ 30 Go (classé par thèmes).

Nous vous demanderons simplement de **mettre l'Association des piétons et cyclistes du pont Jacques-Car er** lors de la répartition des crédits des médias. Nous vous demandons également que les médias utilisés dans le cadre de votre projet de maîtrise servent à alimenter uniquement votre projet. **S'ils devaient être transférés à PJCCI** pour être exploités dans le cadre des opérations de la Société (entre en, études, etc.) **nous vous demanderons de nous aviser**.

D'autre part, si vous avez une **entente ou convention avec PJCCI** par rapport à l'utilisation de votre rapport de maîtrise, nous vous demanderons de nous faire parvenir pour que nous puissions prendre connaissance de son contenu.

<https://www.flickr.com/photos/151964858@N02/> <https://www.youtube.com/channel/UCD2cxmKjEP88LmZ21chTKHw>

Cordialement,

François Démontagne,
Président de l'Association des piétons et cyclistes du pont Jacques-Car er

Le sam. 15 févr. 2020, à 13 h 32, Vincent Le Falher <Vincent.Le.Falher@usherbrooke.ca> a écrit :
Bonjour M. Démontagne. Merci pour votre intérêt. Je vais tenter de vous contacter Mardi sur l'heure du déjeuner. À Mardi, Vincent.

De : Piétons-cyclistes pont Jacques-Car er <apc.pontjc@gmail.com>

Envoyé : 12 février 2020 16:02

À : Vincent Le Falher <Vincent.Le.Falher@usherbrooke.ca>

Cc : Mickaël Germain <Mickael.Germain@Usherbrooke.ca>

Objet : Re: Bonjour !

<https://outlook.office.com/mail/search/id/AAQkAGRkMzY2ZjNlTJjZTMtNGI2My1hNTc2LTQ0ZjIxMzExNzYxMAAQAGWvRXbRHIOhn9Ro85udEw%3D> 1/2 7/31/2020
Courriel - Vincent Le Falher - Outlook

Bonjour M. Le Falher,

Nous disposons effectivement d'un certain nombre de photos et vidéos de la piste dans différentes conditions d'utilisation.

Je vous propose de me contacter en soirée ou entre 12 h et 13 h au 514-927-6366 pour discuter de votre projet. Au plaisir,

François Démontagne
Président de l'Association des piétons et cyclistes du pont Jacques-Car er

Le mer. 12 févr. 2020, à 13 h 00, Vincent Le Falher <Vincent.Le.Falher@usherbrooke.ca> a écrit : Bonjour cher responsable de l'Association des piétons et cyclistes du PJC.

Mon nom est Vincent Le Falher. Je suis étudiant à l'université de Sherbrooke et je suis en plein essai de recherche en géométrie appliquée dans le cadre de la Maîtrise en géographie, cheminement géodéveloppement durable. Et mon sujet de recherche a pour site d'étude la piste multi fonctionnelle du pont Jacques-Car er. Bébé oui ☺!

Voici une page d'introduction et de mise en contexte de mon projet de recherche, si cela vous intéresse : <https://vince7lf.github.io/about.html>

En gros, je travaille sur un système qui permet de détecter automatiquement les délimitations de la piste cyclable, peu importe l'angle de vue et les conditions de la surface (mouillée, neige, etc.).

Et j'aurais besoin de "données", c'est à dire des images et des vidéos de la piste multi fonctionnelle. J'en ai besoin pour tester et adapter des modèles de reconnaissance d'images. Et c'est super car vous en avez ... pas mal quand même (<https://www.flickr.com/photos/pontjacquescar.er/>). J'ai trouvé le tout grâce à votre site (<http://pontjacquescar.er365.com/contact/>), que j'ai trouvé en faisant des recherches d'images via Google.

Le premier objectif de mon email est de vous demander la permission d'utiliser ces images et ces vidéos. À des fins de recherche pour ma maîtrise. Le second objectif, si vous me donnez la permission, est de pouvoir récupérer ces images et vidéos, incluant toutes leurs différentes résolutions.

Si vous voulez en discuter, cela me fera plaisir, je suis disponible au :

Ou via email aussi.

J'a ends de vos nouvelles avec impa ence.
En a endant, je vous souhaite une excellente journée ! Merci
Vincent Le Falher

-- --
Associa on des piétons et cyclistes du pont Jacques-Car er
<https://www.facebook.com/associa on.pietons.cyclistes.pont.jacques.car.er/> <https://twitter.com/APCPontJCar.er>

-- --
Associa on des piétons et cyclistes du pont Jacques-Car er
<https://www.facebook.com/associa on.pietons.cyclistes.pont.jacques.car.er/> <https://twitter.com/APCPontJCar.er>

<https://outlook.office.com/mail/search/id/AAQkAGRkMzY2ZjNlTjZTMtNGI2My1hNTc2LTQ0ZjIxMzExNzYxMAAQAGWvRXbRHROhn9Ro85udEw%3D> 2/2

i

<https://docs.nvidia.com/jetson/l4t/index.html#page/Tegra%2520Linux%2520Driver%2520Package%2520Development%2520Guide%2Foverview.html%23>