ENSTA IP PARIS



Comptage de véhicules sur TRANCOS

KASETSART UNIVERSITY

FISE27

Robotique Autonome

30 septembre 2026

Florian MILIN
florian.milin@ensta.fr

Remerciements

Je tiens tout d'abord à remercier Juliette Lourdault et Clément Sartre, étudiants à l'ENSTA, campus de Brest, qui ont, comme moi, effectué leur stage à l'université de Kasetsart, sous la tutelle de Piya Limcharoen. Nous partagions le même bureau et, bien que nos sujets de stage soient différents, nos trois projets portaient sur le Machine Learning (ML). Cette proximité nous a permis de nous entraider tout au long du stage.

Je tiens également à remercier Piya Limcharoen pour son implication dans notre stage et pour le temps qu'il nous a consacré, notamment lors de nos réunions hebdomadaires. Je le remercie également pour ses nombreux conseils et suggestions, aussi bien concernant nos travaux que notre intégration en Thaïlande. Il s'est toujours montré disponible et ouvert à nos questions, ce qui nous a beaucoup aidés, aussi bien dans notre travail que dans la découverte de la culture thaïlandaise. Je souhaite également souligner son implication dans mon projet plus particulièrement. À certains moments, j'ai pu perdre confiance dans mon travail et dans l'avancement de mon stage ; son soutien, ses conseils et son travail m'ont alors beaucoup aidé à reprendre confiance et à poursuivre mon projet, comme nous le verrons par la suite.

Je souhaite enfin remercier l'ensemble des membres du département d'informatique de l'université de Kasetsart pour leur accueil et les efforts qu'ils ont faits pour nous intégrer. Dans notre bureau, nous faisions de notre mieux pour communiquer en anglais afin que chacun puisse se comprendre. Je remercie également les personnes qui ne travaillaient pas directement dans notre bureau, mais qui prenaient malgré tout le temps de venir nous parler et de nous adresser la parole lorsqu'elles passaient dans notre bureau. Ces petites attentions ont contribué à rendre notre intégration beaucoup plus naturelle et agréable.

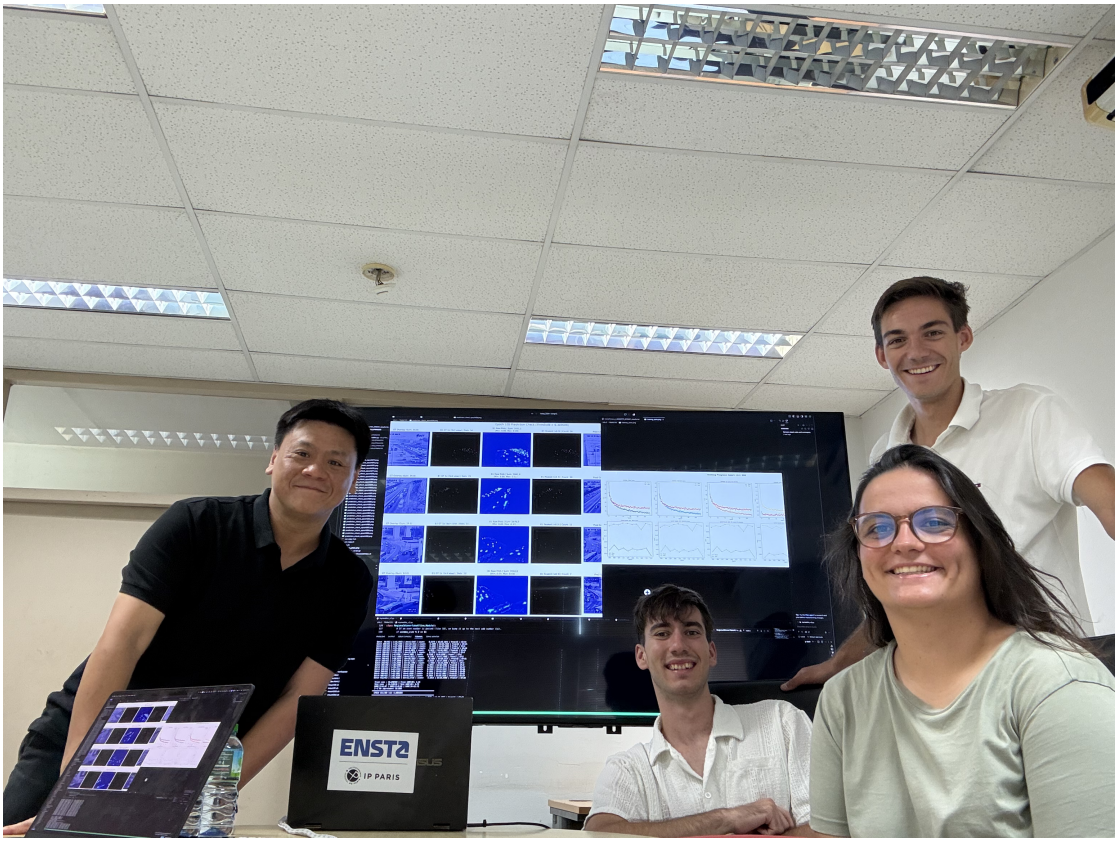


FIGURE 1 – Photo de mon tuteur, Piya et mes collègues Juliette et Clément.

Résumé

Ce rapport présente mon stage de deuxième année à l'université de Kasetsart (Bangkok), consacré au comptage de véhicules à partir d'images sur le jeu de données TRANCOS. Après une étude des méthodes par régression de carte de densité, nous avons réimplémenté CCNN, Hydra-3s et CSRNet, dont les résultats s'écartent de ceux publiés. Nous avons ensuite testé plusieurs améliorations : des cartes de densité à noyau adaptatif et des patchs centrés sur les véhicules pour CCNN, puis un back-end multi-échelle et une loss combinée pour CSRNet. Enfin, TinyNet, une architecture légère de 0.19 million de paramètres prédisant une carte de points, obtient un GAME(0) de 3.45 sur le jeu de test, proche de la valeur publiée pour CSRNet (3.56), avec 85 fois moins de paramètres. Le travail a donné lieu à un article scientifique rédigé en anglais.

Abstract

This report describes my second-year internship at Kasetsart University (Bangkok) on vehicle counting from images with the TRANCOS dataset. After reviewing density-map regression methods, we re-implemented CCNN, Hydra-3s and CSRNet, whose results differ from the published ones. We then tested several improvements : adaptive-kernel density maps and vehicle-centred patches for CCNN, and a multi-scale back-end and a combined loss for CSRNet. Finally, TinyNet, a lightweight architecture of 0.19 million parameters that predicts a point map, reaches a GAME(0) of 3.45 on the test set, close to the value published for CSRNet (3.56) with 85 times fewer parameters. This work led to a scientific paper written in English.

Table des matières

1 Introduction **4**

2 Contexte du stage **4**
2.1 Department of Computer Engineering 4
2.2 Piya Limcharoen 4

3 Problématisation **4**

4 État de l'art **5**
4.1 Détection d'objets 6
4.2 Détection point par point 7
4.3 Comptage par carte de densité 7
4.4 Le dataset : TRANCOS 8
4.5 Conclusion de l'état de l'art 9

5 Implémentation des méthodes existantes **9**
5.1 Architecture des méthodes 9
5.1.1 Une architecture commune 9
5.1.2 Mode de traitement : patchs vs image entière 10
5.1.3 CCNN 10
5.1.4 Hydra 11
5.1.5 CSRNet 12
5.1.6 Comparaison des architectures 12
5.2 Expérimentation 13
5.2.1 Experimentation Setup 13
5.2.2 Métrique d'évaluation : GAME 13
5.3 Résultats 14
5.3.1 Analyse des écarts 14
5.3.2 Résultats de CCNN 14
5.3.3 Résultats de Hydra-3s 15
5.3.4 Résultats de CSRNet 15
5.3.5 Facteurs limitants 15
5.3.6 Conclusion sur les implémentations 15

6 Développement d'une nouvelle méthode **15**
6.1 Les idées 15
6.1.1 La méthode K-Nearest Neighbors (KNN) 16
6.1.2 La méthode *centred-vehicules* 17
6.1.3 La méthode Multi-scale 18
6.1.4 La méthode Combined Loss 19
6.1.5 Les résultats de mes méthodes 20
6.2 TinyNet 21
6.2.1 Architecture 21
6.2.2 Sélection des pics et changement de la loss 21
6.2.3 Counting : de la point-map au comptage 21

7 Synthèse du stage **22**
7.1 Rédaction d'un article scientifique 22
7.2 Résultats sur TRANCOS 23
7.3 Complexité des modèles 23

8 Conclusion **23**

Glossaire **24**
Bibliographie 24

1 Introduction

Estimer le nombre de véhicules présents dans une image est une brique de base pour l'analyse du trafic routier et la perception des systèmes d'aide à la conduite. Cette tâche reste difficile lorsque les véhicules sont nombreux, proches les uns des autres ou partiellement occultés, et que le point de vue de la caméra varie d'une scène à l'autre.

Ce rapport présente le travail réalisé lors de mon stage de deuxième année au département Computer Engineering de l'université de Kasetsart, à Bangkok, sur le comptage de véhicules avec le jeu de données TRANCOS. Nous nous limitons aux méthodes par régression de carte de densité (CCNN, Hydra-3s et CSRNet), que nous avons réimplémentées et comparées aux résultats publiés. Nous avons ensuite testé plusieurs pistes d'amélioration : des cartes de densité à noyau adaptatif (KNN), l'extraction de patchs centrés sur les véhicules, un back-end multi-échelle et une loss combinée pour CSRNet. Enfin, nous présentons TinyNet, une architecture légère prédisant une carte de points, et l'article scientifique rédigé à l'issue du stage.

Le rapport est organisé comme suit. La section 2 présente le contexte du stage et la section 3 la problématique. La section 4 dresse l'état de l'art et présente le jeu de données, et la section 5 détaille l'implémentation des méthodes existantes. La section 6 décrit les méthodes testées et TinyNet, et la section 7 fait la synthèse du stage, dont la rédaction de l'article.

2 Contexte du stage

J'ai effectué mon stage de deuxième année dans un laboratoire de recherche du département Computer Engineering de la Kasetsart University (KU) à Bangkok, en Thaïlande.

2.1 Department of Computer Engineering

Le département fait partie de la faculté d'ingénieurie et concentre les activités d'ingénieurie informatique de l'université. Ce département a deux activités principales, la formation et la recherche. Pour ma part, j'étais dans un laboratoire recherche, entouré de doctorants et de chercheurs. Même si mon tuteur était aussi enseignant en parrallèle de ses recherches, je n'ai pas été en lien avec ce domaine d'activité.

2.2 Piya Limcharoen

Mon tuteur de stage est Piya Licharoen, il est docteur en ingénieurie informatique et travaille sur plusieurs domaines dont, l'Intelligence artificielle (IA), le ML, le Deep Learning (DL). Actuellement, il travaille sur le traitement de données brut sur des capteurs pour la robotique à base d'IA. L'objectif est d'améliorer la précision de ces capteurs en prédisant le compor.

3 Problématisation

L'objectif de ce stage est d'améliorer les performances du comptage de véhicules à partir d'images, en s'appuyant sur des techniques d'IA. Ce sujet s'inscrit dans un contexte où les caméras sont de plus en plus présentes, aussi bien dans les rues, sur les bâtiments que dans les véhicules. Les données qu'elles fournissent constituent une source d'information riche pour analyser l'environnement routier et, notamment, pour estimer l'état du trafic en temps réel.

L'estimation du nombre de véhicules présents dans une scène possède de nombreuses applications. Elle peut notamment contribuer à une meilleure perception de l'environnement pour les systèmes d'aide à la conduite, à l'analyse et à la prévision de la congestion du trafic, ou encore à l'optimisation de la gestion des flux de circulation. Cependant, obtenir un comptage précis reste un défi technique important. Les variations d'éclairage, les conditions

météorologiques, les occultations entre véhicules ou encore la diversité des points de vue peuvent fortement dégrader les performances des algorithmes.

Dans ce contexte, de nombreux travaux de recherche cherchent à améliorer la robustesse et la précision des méthodes de comptage automatique. Le principal enjeu de ce stage est donc de réduire les erreurs de prédiction du nombre de véhicules présents sur une image, tout en conservant des performances compatibles avec des applications réelles. La problématique étudiée peut ainsi être formulée de la manière suivante : *comment améliorer la précision du comptage de véhicules dans des scènes complexes, notamment lorsque les véhicules sont proches ou partiellement occultés, tout en conservant une méthode efficace ?*

Ce travail présente un intérêt pour le département de recherche, qui cherche à développer des solutions plus fiables pour l'exploitation de différents types de capteurs. Mon rôle a été d'apporter un regard neuf sur cette problématique, de proposer et d'évaluer de nouvelles pistes d'amélioration, en mobilisant les compétences acquises au cours de ma formation d'ingénieur.

Le travail qui m'a été confié consistait à étudier et améliorer des méthodes existantes de comptage de véhicules, puis à développer et évaluer une nouvelle approche. Le travail de recherche s'est déroulé en plusieurs étapes. Une première phase a été consacrée à l'étude de la littérature scientifique afin d'identifier les méthodes existantes. Les méthodes sélectionnées ont ensuite été implémentées et évaluées afin de vérifier leur fonctionnement et de disposer de références pour la comparaison avec les approches développées par la suite. Cette étape a également permis d'identifier leurs limites et d'explorer des pistes d'amélioration. Une seconde phase a ainsi été consacrée au développement et à l'évaluation d'une nouvelle méthode de comptage. Enfin, les résultats obtenus ont été synthétisés et formalisés dans un article scientifique. Le rapport suit cette même démarche :

— **État de l'art** : étude de la littérature scientifique afin d'identifier les travaux déjà réalisés sur le sujet, de comprendre les approches existantes et de définir un point de départ pour la recherche.
— **Implémentation des méthodes existantes** : reproduction et analyse des méthodes proposées dans les travaux précédents afin de s'assurer de leur bonne compréhension, d'en évaluer les performances et d'identifier leurs éventuelles limites.
— **Développement d'une nouvelle méthode** : phase de recherche et d'expérimentation visant à proposer une amélioration des approches existantes et à obtenir de meilleures performances.
— **Rédaction d'un article scientifique** : formalisation des travaux et des résultats obtenus sous la forme d'un article scientifique, afin de communiquer les résultats et de permettre la poursuite de ces travaux.

4 État de l'art

Le comptage de véhicules à partir d'images peut être abordé selon différentes approches, en fonction du niveau de représentation choisi pour décrire les objets. Les méthodes peuvent notamment reposer sur la détection d'objets, la détection point par point ou la prédiction d'une carte de densité. Ces différentes approches sont présentées dans cette section.

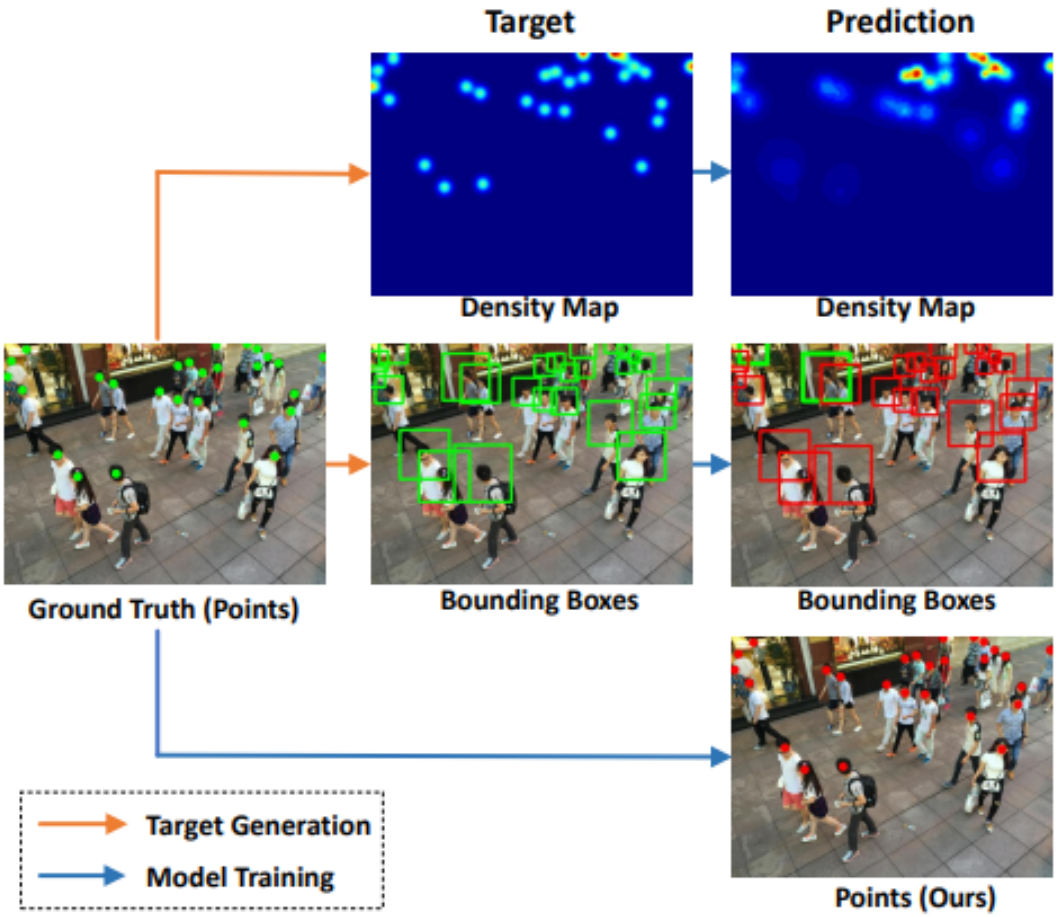


FIGURE 2 – Transformation d'une même image selon les différentes approches de comptage. Adapté de [1].

4.1 Détection d'objets

Le comptage de véhicules à partir d'images peut tout d'abord être abordé comme un problème de détection d'objets. Des méthodes telles que YOLO (*You Only Look Once*) [2] détectent individuellement les objets présents dans une image en prédisant notamment une *bounding box*, une classe et un score de confiance pour chaque détection.

Cette approche fournit une localisation précise de chaque véhicule et permet donc d'obtenir directement leur nombre en comptant les détections. Cependant, elle nécessite de prédire et de traiter une boîte englobante pour chaque objet. Cette information supplémentaire n'est pas nécessaire lorsque seule la position et le nombre des véhicules sont recherchés. Le coût computationnel associé à la détection complète peut ainsi constituer une limitation pour une tâche de comptage.

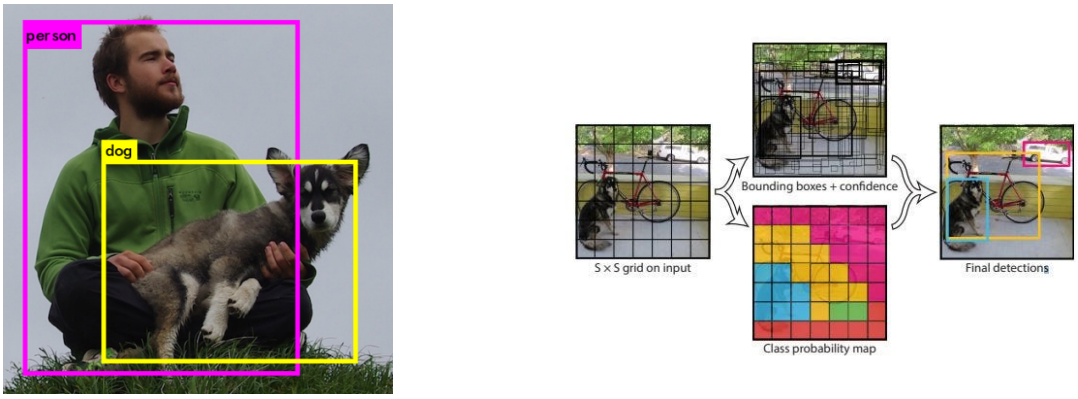


FIGURE 3 – Exemple de détection d'objets avec YOLO. Source : Redmon et al. [2].

D'autres approches cherchent donc à simplifier la représentation des objets afin de se concentrer directement sur leur comptage.

4.2 Détection point par point

Une autre approche est la détection point par point, notamment utilisée pour le comptage de personnes dans les scènes de foule. Cette méthode consiste à représenter chaque individu présent dans une image par un unique point, généralement situé au centre de sa tête. C'est notamment le principe proposé par Song et al. avec P2PNet [1]. Le réseau prédit directement un ensemble de points ainsi qu'un score de confiance associé à chacun d'entre eux. Le nombre de points détectés permet alors d'obtenir directement le nombre d'individus présents dans l'image.

Cette représentation est particulièrement intéressante dans les scènes fortement encombrées, car elle évite la prédiction de *bounding boxes*, qui peuvent devenir difficiles à définir lorsque les objets sont très proches ou se chevauchent.

P2PNet a cependant été conçu pour le comptage et la localisation de personnes dans des scènes de foule, et non spécifiquement pour les véhicules. Bien que cette approche puisse être adaptée à d'autres types d'objets, comme les véhicules dans notre cas, il n'est pas possible de comparer directement ses performances avec celles des méthodes étudiées dans la suite de ce travail. Elle ne constitue donc pas directement une des méthodes évaluées dans ce travail.

4.3 Comptage par carte de densité

Une autre famille de méthodes consiste à prédire une carte de densité représentant la répartition spatiale des objets dans l'image. Cette approche a notamment été utilisée pour le comptage de véhicules et d'objets dans des scènes fortement encombrées [3, 4].

Lors de l'entraînement, les annotations ponctuelles indiquant la position des véhicules sont transformées en une carte de densité. Chaque point est associé à une gaussienne 2D normalisée, de manière à produire une représentation continue de la densité des véhicules. Le réseau de neurones apprend alors à prédire cette carte à partir de l'image.

Le nombre total de véhicules peut ensuite être obtenu en intégrant les valeurs de la carte de densité. Cette représentation présente notamment l'avantage de rester pertinente lorsque plusieurs véhicules sont proches ou partiellement occultés. Elle permet ainsi de représenter la présence et la répartition des véhicules sans nécessiter une localisation individuelle précise de chacun d'entre eux.

Parmi les différentes approches de comptage présentées, ce travail se concentre sur les méthodes fondées sur la prédiction de cartes de densité. Plus précisément, trois architectures sont étudiées expérimentalement : CCNN, Hydra et CSRNet [3, 4]. Ces méthodes reposent toutes sur le principe de génération d'une carte de densité à partir des annotations ponctuelles, mais diffèrent notamment dans leur architecture et leur manière d'exploiter les caractéristiques spatiales de l'image.

4.4 Le dataset : TRANCOS

Les différentes méthodes étudiées dans ce travail sont évaluées sur *TRANCOS* [5], un jeu de données public dédié au comptage de véhicules dans des scènes de trafic dense. Il contient 1 244 images de résolution 640×480 pixels, issues de différentes caméras de vidéosurveillance, pour un total de 46 796 véhicules annotés par des points.

Les images présentent une grande diversité de scènes et de points de vue, sans carte de perspective fournie. Chaque image est également associée à une *Region Of Interest* (ROI), qui définit la zone prise en compte pour l'apprentissage et l'évaluation. Seuls les véhicules situés dans cette région sont donc pris en compte lors de l'évaluation.

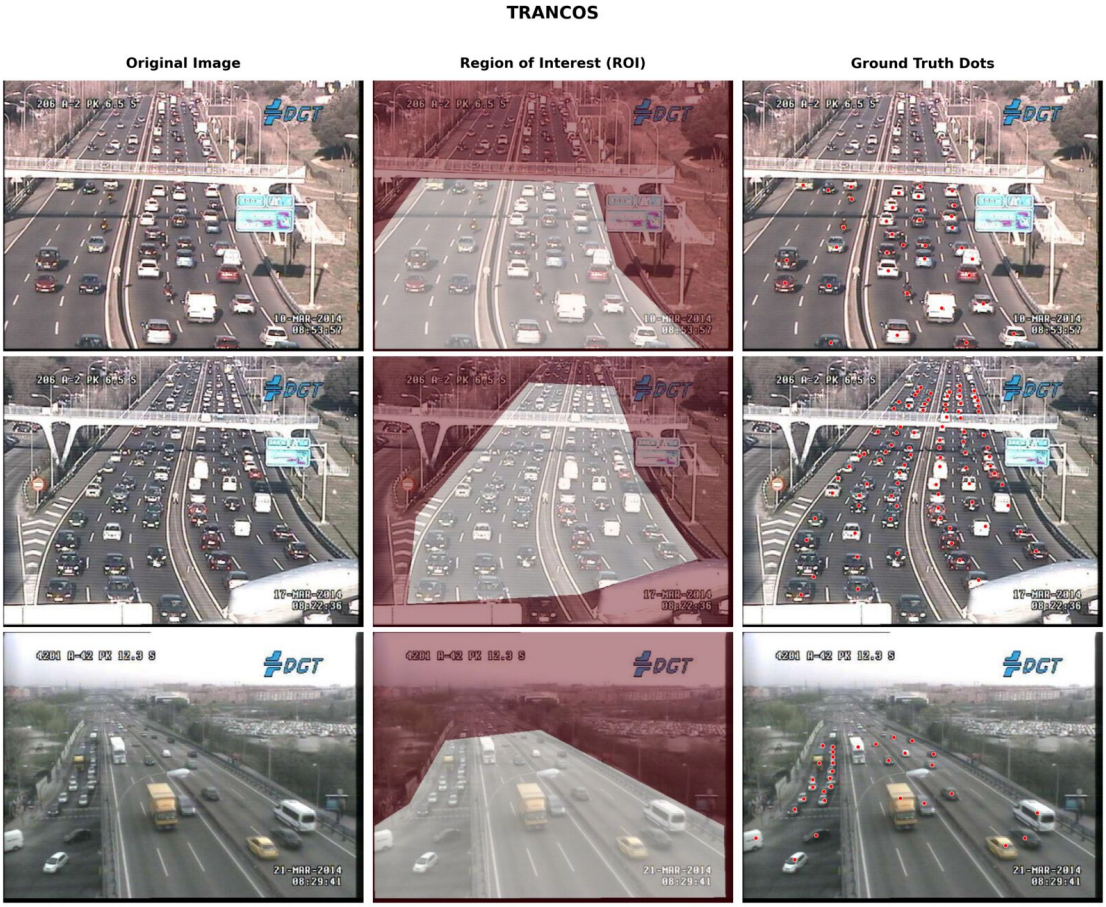


FIGURE 4 – Exemples d'images du dataset TRANCOS et de leurs régions d'intérêt (ROI).

Le dataset est officiellement divisé en 403 images d'entraînement, 420 images de validation et 421 images de test. Conformément aux protocoles utilisés pour CCNN, Hydra [3] et CSRNet [4], les ensembles d'entraînement et de validation sont regroupés pour former l'ensemble d'apprentissage, soit 823 images. Les 421 images du jeu de test sont conservées séparément pour l'évaluation finale.

Le choix de TRANCOS est donc particulièrement adapté à notre problème, puisque ses annotations ponctuelles permettent d'appliquer la méthodes de comptage par densité présentée précédemment. Il permet également de comparer les différentes méthodes dans des conditions similaire aux travaux précédents.

4.5 Conclusion de l'état de l'art

Ainsi, le choix des méthodes et du jeu de données est étroitement lié. Les méthodes de comptage par carte de densité étudiées dans ce travail reposent sur des annotations ponctuelles, ce qui rend le dataset TRANCOS particulièrement adapté à leur évaluation. Réciproquement, le choix de ces méthodes est motivé par leur utilisation dans les travaux précédents portant sur ce dataset, permettant ainsi de conserver un cadre expérimental comparable.

Nous retenons donc trois approches issues de la littérature : CCNN et Hydra-3S, toutes deux présentées dans [3], ainsi que CSRNet [4]. Ces trois architectures reposent sur la prédiction d'une carte de densité, tout en proposant des stratégies différentes pour exploiter les caractéristiques de l'image. Leur fonctionnement et leurs principales différences sont présentés dans la section suivante.

5 Implémentation des méthodes existantes

Trois méthodes existantes de comptage par carte de densité ont été retenues pour cette étude comparative. Les deux premières, CCNN et Hydra-3s, sont issues du même article d'Onoro-Rubio et López-Sastre [3] : CCNN correspond à l'architecture la plus simple, servant de base à l'approche de comptage par régression de densité, tandis que Hydra-3s en constitue l'extension multi-échelle, proposée comme meilleur résultat de cet article. La troisième méthode, CSRNet [4], provient d'un article plus récent de Li et al. et introduit une architecture entièrement convolutive à une seule colonne, exploitant des convolutions dilatées. Ces trois réseaux ont été choisis car ils partagent le même principe de régression de carte de densité tout en illustrant une progression de complexité architecturale, ce qui permet de mettre en évidence l'apport de chaque amélioration successive.

5.1 Architecture des méthodes

5.1.1 Une architecture commune

Les trois réseaux partagent le même pipeline général de comptage basé sur la régression de cartes de densité. La vérité terrain, initialement fournie sous forme d'annotations ponctuelles indiquant la position de chaque véhicule, est convertie en une carte de densité continue.

Chaque réseau est entraîné à régresser cette carte de densité à l'aide d'une fonction de perte euclidienne (erreur quadratique moyenne, MSE), pixel par pixel. Le nombre de véhicules prédit est obtenu en sommant les valeurs de la carte de densité prédite sur la région d'intérêt (ROI).

L'entraînement minimise la perte euclidienne suivante :

$$\mathcal{L}(\theta) = \frac{1}{2N} \sum_{i=1}^{N} |F(X_i; \theta) - Y_i|_2^2 \tag{1}$$

où N est le nombre d'échantillons dans le batch, X_i est le i-ème patch d'entrée, $F(X_i; \theta)$ est la carte de densité prédite, Y_i est la carte de densité de référence, θ est l'ensemble des paramètres appris du réseau, et $|\cdot|_2^2$ est la norme euclidienne au carré. Le facteur $\frac{1}{2}$ est une convention de normalisation facilitant l'optimisation du gradient.

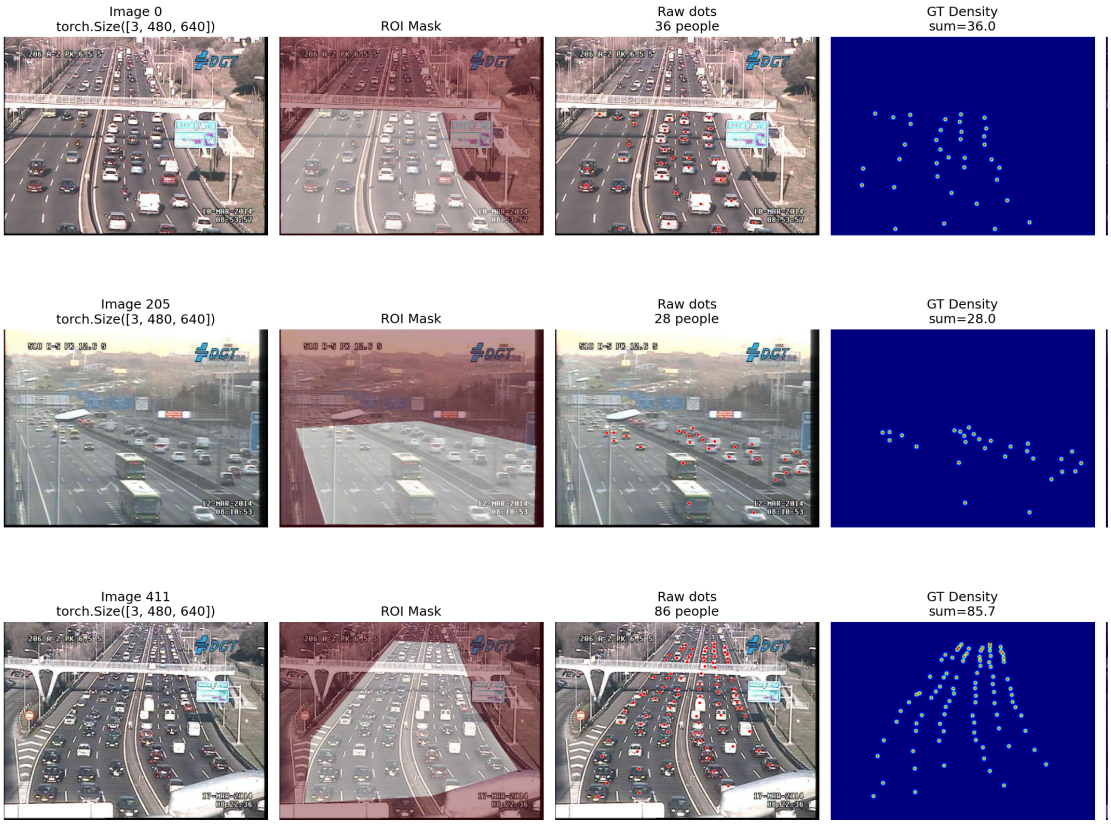


FIGURE 5 – Exemple d'images mises sous forme de carte de densité

5.1.2 Mode de traitement : patchs vs image entière

CCNN et Hydra-3s partagent un mode de traitement par patchs. Durant l'entraînement, des patchs de 115×115 pixels sont extraits aléatoirement dans chaque image, puis redimensionnés à 72×72 pixels avant d'être passés au réseau. À l'inférence, l'image complète est parcourue par une fenêtre glissante (pas de 10 pixels) de même taille, et les cartes de densité prédites pour chaque patch sont assemblées pour reconstituer la carte de densité complète de l'image.

CSRNet adopte une approche différente : entièrement convolutif, il traite l'image entière directement sans découpage en patchs. Cela permet une inférence plus efficace mais requiert davantage de mémoire GPU lors de l'entraînement.

5.1.3 CCNN

CCNN est un petit réseau entièrement convolutif composé de six couches de convolution, alternant convolutions, activations ReLU et opérations de max-pooling, qui transforme un patch d'entrée de 72×72 pixels en une carte de densité de sortie de 18×18 pixels. L'entraînement repose sur une unique perte euclidienne, sans utilisation d'information de perspective. Dans notre implémentation, les patchs d'entraînement sont extraits à des positions aléatoires dans l'image entière (sans contrainte de ROI), avec une augmentation par symétrie horizontale. La ROI n'intervient qu'à l'évaluation, pour restreindre le comptage prédit et la vérité terrain à la zone d'intérêt. Le réseau est optimisé par descente de gradient stochastique (SGD). CCNN constitue le modèle le plus léger des trois architectures étudiées et sert

de référence à échelle unique (*single-scale*).

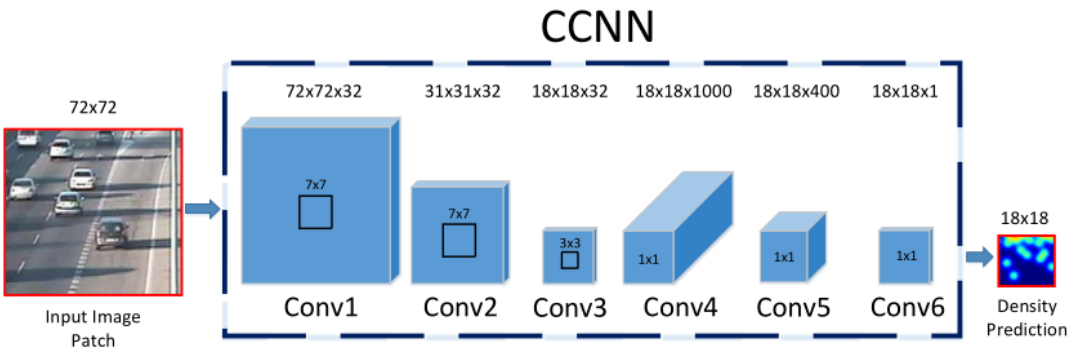


FIGURE 6 – Architecture de CCNN d'après [3].

5.1.4 Hydra

Hydra-3s rend le modèle sensible à l'échelle des objets. À partir de chaque patch d'entraînement, une pyramide à trois niveaux est construite : le patch complet, ainsi que deux recadrages centrés correspondant respectivement à 66% et 33% de sa taille, chacun redimensionné à 72×72 pixels. Chaque niveau de la pyramide est traité par une tête indépendante, construite sur la même architecture que CCNN mais sans sa dernière couche, afin de produire une carte de caractéristiques de taille $18 \times 18 \times 400$. Les trois cartes de caractéristiques issues des trois têtes sont ensuite concaténées puis transmises à un corps entièrement connecté, composé de deux couches cachées suivies d'une couche de sortie, redimensionnée pour former la carte de densité finale de 18×18 pixels. C'est ce corps entièrement connecté qui fait de Hydra-3s, de loin, le modèle le plus volumineux des trois architectures étudiées. L'extraction des patchs, l'augmentation de données et l'inférence par fenêtre glissante restent identiques à celles de CCNN.

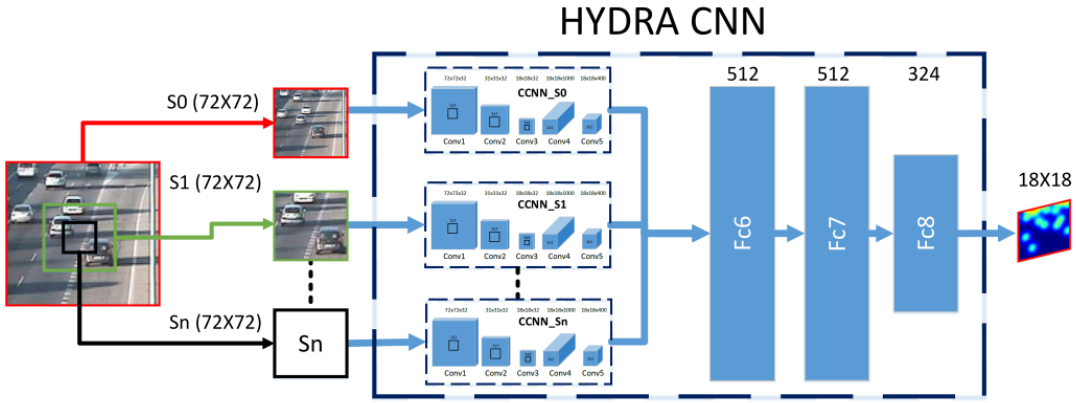


FIGURE 7 – Architecture Hydra-3s d'après [3].

5.1.5 CSRNet

CSRNet est un réseau à une seule colonne (*single-column*), traitant l'image dans son intégralité. Il repose sur un *front-end* constitué des dix premières couches convolutives de VGG-16, suivi d'un *back-end* composé de convolutions dilatées (taux de dilatation de 2), qui permettent d'élargir le champ récepteur du réseau tout en conservant une résolution égale à 1/8 de la résolution d'entrée. CSRNet compte environ 16 millions de paramètres, ce qui reste très inférieur à Hydra-3s. Dans notre implémentation, le front-end est initialisé à partir de poids pré-entraînés sur ImageNet, puis le réseau complet est pré-entraîné sur le jeu de données de comptage de foule ShanghaiTech, avant d'être affiné (*fine-tuning*) sur TRANCOS. Les patchs d'entraînement suivent le protocole propre à CSRNet (neuf recadrages par image à demi-résolution, chacun accompagné de sa version symétrisée), et le réseau est optimisé par SGD. Cette différence d'initialisation par rapport à CCNN et Hydra-3s doit être prise en compte dans l'interprétation de la comparaison.

Configurations of CSRNet			
A	B	C	D
input(unfixed-resolution color image)			
front-end (fine-tuned from VGG-16)			
conv3-64-1 conv3-64-1			
max-pooling			
conv3-128-1 conv3-128-1			
max-pooling			
conv3-256-1 conv3-256-1 conv3-256-1			
max-pooling			
conv3-512-1 conv3-512-1 conv3-512-1			
back-end (four different configurations)			
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-256-1	conv3-256-2	conv3-256-4	conv3-256-4
conv3-128-1	conv3-128-2	conv3-128-4	conv3-128-4
conv3-64-1	conv3-64-2	conv3-64-4	conv3-64-4
conv1-1-1			

FIGURE 8 – Architecture CSRNet d'après [3].

5.1.6 Comparaison des architectures

Le tableau 1 synthétise les caractéristiques architecturales clés et les défcis rencontrés lors de l'implémentation de chaque méthode.

TABLE 1 – Comparaison des architectures et des protocoles de traitement.

Méthode	**Architecture**	**Échelle**	**Entraînement**	**Sortie**	**Inférence**
CCNN	6 couches conv.	Simple	Patchs 72×72	18×18	Fenêtre glissante
Hydra-3s	3 branches conv. + FC	3 niveaux	Patchs 72×72	18×18	Fenêtre glissante
CSRNet	VGG-16 + conv. dilatées	Simple	Recadrages	1/8	Image entière

Cette comparaison met en évidence trois stratégies architecturales de complexité crois-

sante ou différente : CCNN repose sur une architecture simple à échelle unique, tandis que Hydra-3s introduit explicitement une représentation multi-échelle au prix d'un corps entièrement connecté beaucoup plus volumineux. CSRNet adopte quant à lui une architecture entièrement convolutive, combinant un front-end basé sur VGG-16 et des convolutions dilatées afin d'élargir le champ récepteur tout en maintenant une résolution spatiale relativement élevée et un nombre de paramètres inférieur à celui de Hydra-3s.

5.2 Expérimentation

5.2.1 Experimentation Setup

Les réglages d'entraînement suivent, dans la mesure du possible, les protocoles décrits dans les articles originaux.

CCNN et Hydra-3s sont entraînés par SGD, à partir de 800 patchs aléatoires de 115×115 pixels extraits par image d'entraînement. Ces patchs sont redimensionnés à 72×72 pixels puis symétrisés, ce qui porte le nombre total de patchs d'entraînement à 1 600 par image. Les cartes de vérité terrain sont générées à l'aide d'un noyau gaussien d'écart-type $\sigma = 15$, et l'inférence s'effectue par balayage de l'image de test avec un pas de 10 pixels.

CSRNet est entraîné par SGD avec un taux d'apprentissage fixe de 10^{-5}. L'entraînement utilise neuf recadrages par image, chacun de taille égale à la moitié de la largeur et de la hauteur de l'image, chacun étant également symétrisé. CSRNet est entrainé avec $\sigma = 10$.

Le tableau 2 résume les principaux réglages expérimentaux retenus pour chacune des trois méthodes.

TABLE 2 – Configuration d'entraînement des trois méthodes.

	CCNN	**Hydra 3S**	**CSRNet**
Optimiseur	SGD	SGD	SGD
Learning rate	10^{-4}	10^{-4}	10^{-5}
σ	15	15	10
Patience	5	5	5
Sigma	15	15	10
Pré-entraînement	–	–	ShanghaiTech

5.2.2 Métrique d'évaluation : GAME

L'évaluation des trois méthodes repose sur le *Grid Average Mean Absolute Error* (GAME), la métrique introduite conjointement au jeu de données TRANCOS [5]. Le GAME affine l'erreur absolue moyenne classique en pénalisant également une mauvaise localisation spatiale de la densité prédite. Pour un niveau L, l'image est divisée en une grille régulière de 4^L régions non chevauchantes couvrant la ROI, et l'erreur correspond à la somme des erreurs absolues de comptage sur ces régions, moyennée sur l'ensemble du jeu de test :

$$\mathrm{GAME}(L) = \frac{1}{N} \sum_{n=1}^{N} \sum_{l=1}^{4^L} \left| \hat{c}_n^l - c_n^l \right|, \tag{2}$$

où N est le nombre d'images de test, et $\hat{c}_n^l$ et c_n^l désignent respectivement le comptage prédit et le comptage réel dans la région l de l'image n. Un niveau L plus élevé correspond à une grille plus fine et donc à un critère plus exigeant spatialement. En particulier, GAME(0) se réduit à une unique région couvrant l'image entière et coïncide donc avec l'erreur absolue moyenne (MAE) classique sur le comptage global. Conformément à l'usage courant sur TRANCOS, les résultats sont reportés de GAME(0) à GAME(3), correspondant respectivement à des grilles de 1, 4, 16 et 64 régions.

5.3 Résultats

Afin de vérifier la bonne implémentation des différentes architectures, leurs résultats sont d'abord comparés à ceux rapportés dans les travaux de référence. Le tableau 12 présente les résultats publiés pour CCNN, Hydra-3s et CSRNet sur TRANCOS.

TABLE 3 – Résultats de référence en GAME sur TRANCOS.

Méthode	**GAME(0)**	**GAME(1)**	**GAME(2)**	**GAME(3)**
CCNN [3]	12.49	16.58	20.02	22.41
Hydra-3s [3]	10.99	13.75	16.69	19.32
CSRNet [4]	3.56	5.49	8.57	15.04

Le tableau 10 présente les résultats obtenus avec les implémentations réalisées durant le stage.

TABLE 4 – Résultats obtenus en GAME sur TRANCOS avec les implémentations réalisées.

Méthode	**GAME(0)**	**GAME(1)**	**GAME(2)**	**GAME(3)**
CCNN	9.23	18.52	26.82	34.65
Hydra-3s	12.17	22.71	31.10	38.79
CSRNet	6.01	9.21	11.69	14.86

5.3.1 Analyse des écarts

Les résultats obtenus diffèrent de ceux publiés dans les articles de référence. En ML, cette variabilité peut être normale : chaque entraînement produit des poids différents selon l'initialisation, l'ordre des données, le nombre d'epochs réalisés et tout un tas d'autres paramètres. Cependant, les écarts observés ici sont importants et méritent une analyse approfondie.

Pour évaluer quantitativement ces écarts, nous rapportons l'erreur relative de nos implémentations par rapport aux valeurs publiées via la formule :

$$\text{écart relatif} = \frac{\text{GAME}_{\text{notre}} - \text{GAME}_{\text{papier}}}{\text{GAME}_{\text{papier}}} \times 100\% \tag{3}$$

TABLE 5 – Écart relatif de nos implémentations par rapport aux valeurs publiées (%).

Méthode	**GAME(0)**	**GAME(1)**	**GAME(2)**	**GAME(3)**
CCNN	−26.1%	+11.7%	+34.0%	+54.6%
Hydra-3s	+10.7%	+65.2%	+86.4%	+100.9%
CSRNet	+68.8%	+67.6%	+36.4%	−1.2%

5.3.2 Résultats de CCNN

Le résultat de GAME(0) est meilleur que le papier (−26.1%), ce qui suggère une meilleure performance au comptage global. Cependant, cette amélioration disparaît et s'inverse pour les niveaux supérieurs de GAME (GAME(1) à GAME(3)), indiquant une moins bonne précision dans la localisation spatiale des véhicules. Cet écart peut provenir d'une différence dans la procédure de post-processing ou des hyperparamètres de fine-tuning.

5.3.3 Résultats de Hydra-3s

Les écarts pour Hydra-3s sont parmi les plus importants, particulièrement pour GAME(2) et GAME(3) (+86% et +101%). Cette dégradation progressive suggère que la pyramide multi-échelle ne converge pas correctement. Plusieurs hypothèses peuvent expliquer cela : le nombre d'epochs réalisé est probablement insuffisant pour une convergence complète (les papiers ne précisent pas le nombre exact), les hyperparamètres d'entraînement peuvent ne pas correspondre exactement à ceux du papier. D'ailleurs ce qui a été le cas, pour CCNN et hydra j'ai dû changé le learning rate de 1e-5 à 1e-4 sinon le modèle convergeait vers la moyenne des voitures sur chaque patch. La fusion des trois branches de la pyramide peut aussi avoir un bug subtil dans l'implémentation.

5.3.4 Résultats de CSRNet

CSRNet est également éloigné des résultats publiés, avec un GAME(0) de +68.8% (6.01 vs 3.56). Cet écart est particulièrement surprenant car le pré-entraînement sur ShanghaiTech Part A a été réalisé comme spécifié (20 epochs, meilleur val loss=2.65), et le fine-tuning sur TRANCOS a suivi le protocole standard (lr=10^{-5}, patience=5). CSRNet est réputée pour être une architecture robuste et bien documentée, ce qui suggère qu'une étape cruciale du protocole a été manquée ou mal implémentée.

5.3.5 Facteurs limitants

Plusieurs facteurs ont probablement impacté la qualité des résultats. L'entraînement sur CPU est considérablement plus lent qu'en GPU, ce qui a limité le nombre d'epochs viables dans les délais du stage. Bien que les papiers ne spécifient pas le nombre exact d'epochs, il est probable que les résultats publiés proviennent d'entraînements sur plusieurs centaines d'epochs ce qui est impraticable sur notre configuration.

Les articles de référence manquent également de certains détails cruciaux (nombre exact d'epochs, procédure exacte de génération des cartes de densité, normalisation des données, détails du post-processing), ce qui a obligé à faire des hypothèses qui peuvent être incorrectes.

Malgré ces limitations, une erreur subtile d'implémentation reste une hypothèse plausible pour expliquer les écarts observés.

5.3.6 Conclusion sur les implémentations

Cette partie ayant pris plus de temps que prévu, il était nécessaire de poursuivre l'étude avec les étapes suivantes du projet. Malgré les écarts observés par rapport aux résultats de référence, et avec l'accord de mon tuteur, les implémentations réalisées sont considérées comme suffisamment fonctionnelles pour poursuivre l'étude comparative hormis hydra.

Les méthodes présentent des comportements cohérents lors de l'entraînement et de l'évaluation. En particulier, CCNN et CSRNet montrent des tendances globalement conformes à celles attendues. Le classement relatif entre CCNN et CSRNet est également préservé.

Ces implémentations seront ainsi utilisées comme baselines pour la suite de l'étude et serviront de point de comparaison pour l'évaluation des méthodes développées dans le cadre de ce projet.

6 Développement d'une nouvelle méthode

6.1 Les idées

Les différentes méthodes étudiées nécessitent des temps d'entraînement importants. À titre d'exemple, l'entraînement de CCNN prend environ une journée, tandis que celui de CSRNet peut nécessiter jusqu'à trois jours.

Par ailleurs, mon tuteur m'a orienté vers l'idée que l'augmentation de la complexité du réseau n'est pas nécessairement la meilleure solution pour améliorer les performances. Des améliorations apportées au traitement des données et à la manière dont elles sont présentées

au réseau peuvent également permettre d'obtenir de meilleurs résultats, tout en conservant une architecture simple.

J'ai ainsi choisi de concentrer dans un premier temps mes recherches sur l'amélioration de CCNN, avec l'objectif de déterminer si une modification du traitement des données pouvait permettre à cette architecture plus légère d'atteindre, voire de dépasser, les performances de CSRNet.

6.1.1 La méthode KNN

Les différents travaux étudiés soulignent de nombreuses erreurs de comptage étaient fréquemment liées aux variations d'échelle des véhicules dans l'image. De plus, je trouvais étrange de fixer sigma à 15, donc j'ai voulu m'intéresser à ce coeficient, et pourquoi pas le changer ? Rappel : lors de la génération des cartes de densité, chaque véhicule est représenté par une gaussienne de même écart-type σ, alors que les véhicules peuvent présenter des tailles apparentes très différentes en fonction de leur position dans l'image. Cette représentation peut donc être inadaptée lorsque la perspective entraîne de fortes variations d'échelle. Le problème est également accentué par les occlusions : un véhicule partiellement masqué apparaît plus petit et peut ainsi présenter une apparence similaire à celle d'un véhicule situé plus loin dans la scène.

J'ai donc cherché à rendre l'écart-type σ des gaussiennes dépendant de l'échelle locale dans l'image, afin que leur taille soit mieux adaptée à la taille apparente des véhicules. Cependant, cette dernière n'est pas disponible dans les annotations de TRANCOS, qui sont uniquement constituées de points. Il est donc nécessaire d'utiliser une information indirecte pour estimer l'échelle locale.

L'idée repose sur le fait que la distance entre les véhicules varie elle aussi avec la perspective. Dans une même scène, les véhicules situés au premier plan sont généralement représentés par des points plus espacés que ceux situés au fond de l'image. J'ai ainsi introduit une méthode basée sur les KNN afin d'estimer, pour chaque véhicule, une distance caractéristique à partir de ses plus proches voisins. Cette distance est ensuite utilisée pour adapter l'écart-type σ de la gaussienne associée au véhicule.

Pour chaque véhicule i, on calcule d'abord la distance moyenne $\bar{d}_i$ à ses k_{eff} plus proches voisins :

$$\bar{d}_i = \frac{1}{k_{\text{eff}}} \sum_{j=1}^{k_{\text{eff}}} d_{ij},$$

où d_{ij} représente la distance entre le véhicule i et son j-ième plus proche voisin. L'écart-type associé à la gaussienne est ensuite obtenu en multipliant cette distance par un coefficient $\beta = 0.33$ ici, et en bornant le résultat entre $\sigma_{\min} = 2$ et $\sigma_{\max} = 20$ pixels :

$$\sigma_i = \begin{cases} 2 & \text{si } \beta\bar{d}_i < 2, \\ \beta\bar{d}_i & \text{si } 2 \leq \beta\bar{d}_i \leq 20, \\ 20 & \text{si } \beta\bar{d}_i > 20. \end{cases}$$

Cette approche permet ainsi d'adapter localement la taille des gaussiennes à l'échelle estimée des véhicules, tout en conservant des valeurs de σ comprises dans une plage définie. Voici le résultat :

Table 6 – Mes résultats sur TRANCOS v3.

Méthode	**GAME(0)**	**GAME(1)**	**GAME(2)**	**GAME(3)**
CCNN	9.23	18.52	26.82	34.65
KNN	9.71	20.04	28.46	36.60

Mon idée n'était donc pas bonne.

6.1.2 La méthode *centred-vehicules*

Suite à l'échec de l'approche basée sur le KNN, j'ai cherché à modifier la stratégie d'extraction des patchs utilisée pour l'entraînement de CCNN.

Dans la méthode CCNN, les patchs sont extraits aléatoirement dans les images. Or, certaines de ces extractions peuvent se trouver entièrement, ou en grande partie, en dehors de la région d'intérêt (ROI). Dans ce cas, le patch contient peu, voire aucun, véhicule pris en compte lors de l'évaluation finale. Le réseau apprend néanmoins à produire une carte de densité proche de zéro sur ces régions. J'ai alors fait l'hypothèse qu'une partie de ces exemples pouvait être moins informative pour l'objectif de comptage que des patchs contenant effectivement des véhicules.

L'idée de *centred-vehicules* est donc de concentrer l'extraction des patchs sur les véhicules présents dans l'image, afin de fournir au réseau une plus grande proportion d'exemples contenant des objets à compter. Avec cette stratégie, chaque patch est centré sur un véhicule, ce qui force davantage le réseau à apprendre à déterminer non seulement si des véhicules sont présents, mais également combien sont présents et où ils se situent dans le patch, car on a toujorus ce soucis de localisation avec CCNN et KNN.

Cette modification a également pour objectif d'améliorer l'efficacité de l'apprentissage. En présentant davantage de situations contenant des véhicules, chaque époque devrait théoriquement fournir une quantité plus importante d'informations directement pertinentes pour la tâche de comptage. Cette propriété peut être particulièrement intéressante dans le contexte de mon expérimentation, puisque le temps d'entraînement de CCNN est important et limite le nombre d'époques pouvant être réalisées. L'idée était donc qu'un apprentissage davantage ciblé permet au réseau de progresser plus rapidement.

Les résultats obtenus sont présentés dans le tableau 7.

Table 7 – Résultats de l'extraction de patchs centrés sur les véhicules sur TRANCOS v3.

Méthode	**GAME(0)**	**GAME(1)**	**GAME(2)**	**GAME(3)**
CCNN	9.23	18.52	26.82	34.65
KNN	9.71	20.04	28.46	36.60
Centrage sur les véhicules	10.92	19.48	26.84	34.18

Les résultats sont cependant loin de ceux attendus. Le GAME(0) se dégrade nettement, passant de 9.23 pour CCNN à 10.92 avec l'extraction centrée sur les véhicules. L'objectif d'améliorer le comptage global n'est donc pas atteint. Les métriques spatiales présentent un comportement légèrement différent : GAME(1) passe de 18.52 à 19.48 et reste donc moins bon, tandis que GAME(2) reste pratiquement identique (26.82 contre 26.84) et que GAME(3) s'améliore légèrement, passant de 34.65 à 34.18.

Ces résultats ne permettent donc pas de valider l'hypothèse initiale. Plusieurs explications peuvent être envisagées. En particulier, centrer systématiquement les patchs sur les véhicules modifie fortement la distribution des données présentées au réseau. Les exemples contenant peu ou aucun véhicule deviennent beaucoup moins fréquents, alors qu'ils peuvent malgré tout être utiles pour apprendre à distinguer les véhicules de l'arrière-plan, même au sein de la région d'intérêt. De plus, le fait de placer systématiquement un véhicule au centre du patch peut introduire un biais spatial : pendant l'entraînement, le réseau rencontre principalement des véhicules proches du centre, alors que les véhicules peuvent apparaître à n'importe quelle position dans la fenêtre glissante.

Au vu de ces résultats, il n'est donc pas possible d'affirmer que les patchs extraits en dehors de la ROI sont inutiles pour l'apprentissage. Au contraire, la dégradation du GAME(0) observée lorsque l'on réduit fortement leur présence suggère qu'ils peuvent apporter une information utile au réseau, notamment en lui permettant d'apprendre à distinguer les véhicules de l'arrière-plan et de rencontrer une plus grande diversité de configurations.

L'hypothèse selon laquelle concentrer l'apprentissage sur des patchs contenant systématiquement des véhicules permettrait de rendre chaque époque plus efficace n'est donc pas

confirmée par cette expérience. Cette dernière m'a au contraire amené à remettre en question mon interprétation initiale des patchs hors ROI : leur présence ne doit pas nécessairement être considérée comme un défaut de la stratégie d'apprentissage. Des expériences supplémentaires, par exemple en faisant varier progressivement la proportion de patchs centrés sur les véhicules et de patchs extraits aléatoirement, seraient nécessaires pour déterminer plus précisément leur rôle dans l'apprentissage de CCNN.

6.1.3 La méthode Multi-scale

Pour comprendre les problèmes de localisation observés, j'ai examiné les cartes de densité produites par CCNN sur une image test du dataset. La Figure 9 montre que la densité est très mal estimée par CCNN, la carte prédite ne localise pas correctement les véhicules. Cela explique les mauvaises performances observées pour GAME(1) à GAME(3), qui évaluent, entre autres, la qualité de la localisation spatiale.

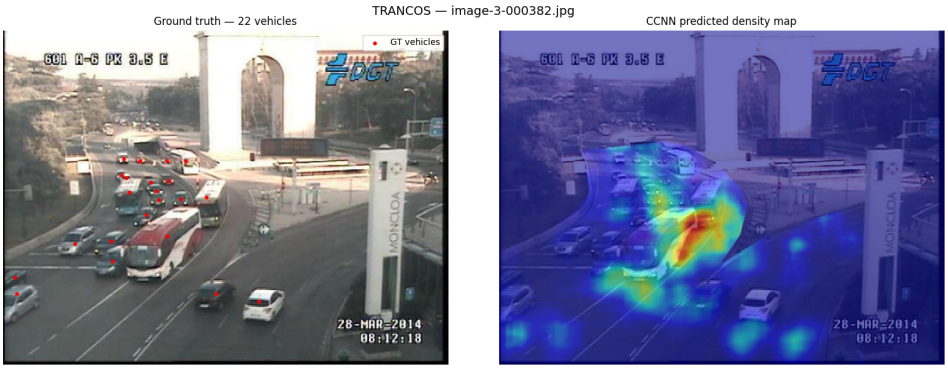


FIGURE 9 – À gauche, l'image avec les annotations ponctuelles du dataset. À droite, la carte de densité prédite par CCNN.

En comparaison, la Figure 10 présente la carte de densité prédite par CSRNet sur la même image. CSRNet produit une carte bien plus précise, avec une localisation claire des véhicules et peu d'artefacts spatiaux, ce qui corrèle avec ses meilleures performances sur les GAME(1) à GAME(3).

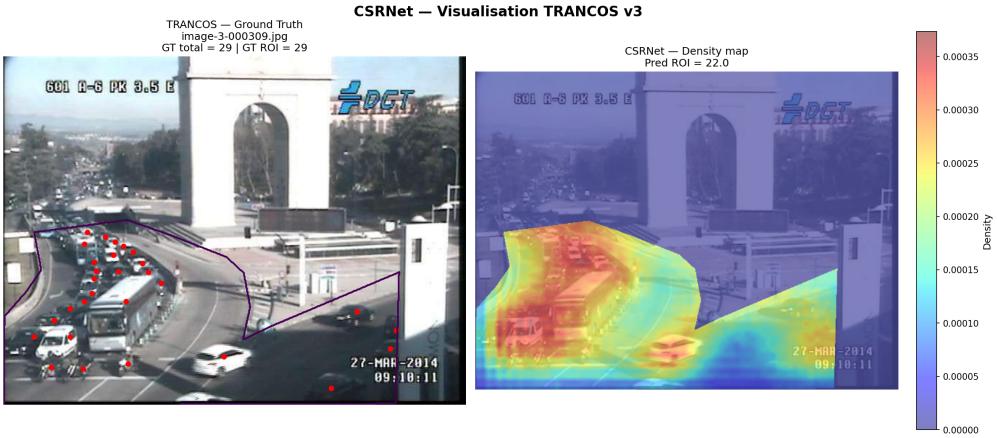


FIGURE 10 – À gauche, l'image avec les annotations ponctuelles du dataset. À droite, la carte de densité prédite par CSRNet.

Au vu de la différence significative de précision entre CCNN et CSRNet (visible sur les deux figures et confirmée par les métriques GAME), et considérant que l'amélioration de CCNN seul n'était pas garantie de surpasser CSRNet, mon tuteur m'a conseillé de chercher directement à améliorer CSRNet plutôt que de poursuivre sur CCNN.

L'amélioration "multi-scale" proposée s'inspire de l'architecture Hydra-3s en utilisant plusieurs branches parallèles pour capturer les véhicules à différentes échelles. L'idée est, comme on l'a vu précédement, que les véhicules apparaissent à différentes tailles dans l'image en fonction de leur distance à la caméra, les véhicules proches sont volumineux tandis que les véhicules lointains apparaissent petits. En combinant plusieurs champs réceptifs, on peut capturer simultanément ces deux extrêmes.

Contrairement à Hydra-3s qui traite plusieurs résolutions de l'image d'entrée, notre implémentation opère sur une seule image d'entrée mais utilise des champs réceptifs différents au niveau du back-end. Spécifiquement, le back-end de CSRNet est remplacé par trois branches parallèles de convolutions dilatées avec des taux de dilatation différents (1, 2 et 4). Chaque branche opère sur les mêmes caractéristiques extraites par le front-end VGG-16 mais explore un champ réceptif différent : une dilatation de 1 capture des détails fins (véhicules proches), tandis que les dilatations 2 et 4 capturent des patterns plus larges (véhicules lointains). Les cartes de densité produites par chaque branche sont ensuite combinées pour générer la prédiction finale.

Cette approche, bien que conceptuellement inspirée par Hydra-3s, ne reproduit pas exactement son architecture pour une bonne raison, exécuter trois branches comme Hydra (avec trois passages indépendants dans le front-end VGG-16) serait beaucoup plus coûteux que trois branches de dilatations partageant le même front-end. Avec les ressources limitées disponibles durant le stage, cette optimisation permet de bénéficier des avantages de traiter différentes échelles spatiales sans surcharge computationnelle. Les résultats de cette approche Multi-scale sont présentés dans le tableau 8 :

TABLE 8 – Résultats GAME : CSRNet baseline vs Multi-scale sur TRANCOS v3.

Méthode	**GAME(0)**	**GAME(1)**	**GAME(2)**	**GAME(3)**
CSRNet	6.01	9.21	11.69	14.86
Multi-scale	5.83	7.53	9.29	11.60

L'approche Multi-scale améliore les performances par rapport à notre implémentation de CSRNet. Le GAME(0) passe de 6.01 à 5.83, soit une amélioration de 3%. Cependant, ce qui est vraiment remarquable est la progression aux niveaux supérieurs de GAME : GAME(1) diminue de 18.3%, GAME(2) de 20.6%, et GAME(3) de 21.9%.

Cette amélioration progressive et croissante avec les niveaux de GAME révèle un insight important : les branches multi-échelle ne se contentent pas d'améliorer le comptage global, elles améliorent drastiquement la localisation spatiale des véhicules. Les erreurs localisées dans les régions de fines grilles (GAME(2) et GAME(3)) diminuent de plus de 20%, ce qui démontre que l'approche capture effectivement les véhicules de différentes tailles de façon bien plus efficace que le baseline CSRNet. Cette amélioration de localisation est précisément ce que recherchait l'approche inspirée par Hydra-3s.

Cependant, bien que Multi-scale soit plus efficace que notre implémentation CSRNet, les résultats restent éloignés de ceux rapportés dans les papiers de référence. Cette limitation persiste probablement pour les mêmes raisons identifiées précédemment : ressources computationnelles limitées, nombre d'epochs potentiellement insuffisant, ou détails d'implémentation manquants dans les articles.

6.1.4 La méthode Combined Loss

L'idée ici est de changer la formule de la loss d'entraînement en ajoutant un second terme calculé uniquement dans la zone ROI :

$$\mathcal{L} = \mathcal{L}_{\text{eucl}}(\hat{D}, D) + \mathcal{L}_{\text{eucl}}(\hat{D} \times M, D \times M)$$

où les éléments se définissent comme suit :

— $\mathcal{L}$: la loss.
— $\hat{D}$: la carte de densité prédite par le réseau.
— D : la carte de densité de référence.
— M : le masque ROI binaire.

Le premier terme $\mathcal{L}_{\text{eucl}}(\hat{D}, D)$ est la loss euclidienne classique que nous avons toujours utilisée. Il pénalise les erreurs sur l'image entière et permet au réseau d'apprendre une prédiction globale, réduisant l'erreur de comptage globale.

Le deuxième terme $\mathcal{L}_{\text{eucl}}(\hat{D} \times M, D \times M)$ pénalise spécifiquement les erreurs localisées uniquement dans la région de route. Par addition de ce terme, on accorde davantage d'importance à corriger les erreurs dans la ROI, forçant le réseau à optimiser sa performance précisément là où elle sera évaluée.

En combinant ces deux termes, chaque époque d'entraînement guide le réseau à optimiser sa prédiction à la fois globalement et localement dans les zones pertinentes.

Table 9 – Résultats GAME : CSRNet baseline, Multi-scale, et Combined Loss sur TRANCOS v3.

Méthode	**GAME(0)**	**GAME(1)**	**GAME(2)**	**GAME(3)**
CSRNet baseline	6.01	9.21	11.69	14.86
Multi-scale	5.83	7.53	9.29	11.60
Combined Loss	6.06	7.02	8.13	9.83

Combined Loss se dégrade légèrement sur GAME(0) (6.06 vs 5.83 pour Multi-scale) mais améliore drastiquement la localisation spatiale. GAME(1) atteint 7.02, GAME(2) 8.13, et GAME(3) 9.83, soit une amélioration de 15 à 30% comparé au baseline CSRNet. Cette approche démontre que les contraintes spatiales explicites peuvent considérablement améliorer la précision de localisation des véhicules au détriment d'une légère dégradation du comptage global.

6.1.5 Les résultats de mes méthodes

Les résultats globaux des implémentations réalisées sont présentés dans le tableau 10 :

Table 10 – Résultats obtenus en GAME sur TRANCOS avec les implémentations réalisées.

Méthode	**GAME(0)**	**GAME(1)**	**GAME(2)**	**GAME(3)**
CCNN	9.23	18.52	26.82	34.65
KNN	9.71	20.04	28.46	36.60
Centrage sur les véhicules	10.92	19.48	26.84	34.18
CSRNet	6.01	9.21	11.69	14.86
Multi-Scale	5.83	7.53	9.29	11.60
Combined Loss	6.06	7.02	8.13	9.83

Sur CCNN, ni la méthode KNN ni le centrage sur les véhicules n'améliorent le GAME(0) (9.71 et 10.92 contre 9.23) ; seul le centrage gagne légèrement sur GAME(3) (34.18 contre 34.65). CSRNet réduit l'erreur de CCNN de 35 % sur GAME(0) et de 50 à 57 % sur GAME(1) à GAME(3). Par rapport à CSRNet, Multi-scale obtient le meilleur GAME(0) (5.83, soit −3 %) et réduit GAME(1) à GAME(3) de 18 à 22 %. Combined Loss est équivalent sur GAME(0) (6.06) mais donne les meilleurs GAME(1) à GAME(3) (7.02, 8.13 et 9.83), soit 24 à 34 % de moins que CSRNet. Les gains portent donc surtout sur la localisation spatiale.

6.2 TinyNet

On arrive sur les dernières semaines de stage et mje suis perdu, je ne sais plus quoi faire pour améliorer les techniques existantes. J'en parle à mon tuteur qui décide de m'aider. En partant de mon travail sur ce problème, mon tuteur s'y est intéressé et voyant la fin du stage approcher, a écrit un programme avec bien moins de paramètres. D'où son nom, TinyNet, qu'on va présenter ici.

6.2.1 Architecture

TinyNet est une architecture légère composée d'une succession de couches convolutives (filtres 3×3, peu de canaux) suivie de deux branches de sortie distinctes.

La première branche, appelée *branche point-map*, prédit une carte de probabilités dans $[0, 1]$ où chaque pixel indique la présence probable d'un véhicule. Cette prédiction permet une supervision directe et précise, contrairement aux cartes de densité gaussienne qui introduisent une ambiguïté liée au choix de l'écart-type σ.

La seconde branche, appelée *branche count*, prédit directement le nombre total de véhicules en sommant les valeurs de la point-map prédite. Cette branche offre une régularisation globale du comptage.

Les deux sorties sont entraînées conjointement via une perte mixte combinant une focal loss (pour la point-map) et une MSE (pour le count) :

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{focal}}(\hat{P}, P_{\text{GT}}) + \lambda \cdot \mathcal{L}_{\text{MSE}}(\hat{C}, C_{\text{GT}}) \tag{4}$$

où $\hat{P}$ est la point-map prédite, $\hat{C} = \sum \hat{P}$ est le count prédit, et $\lambda = 0.1$ est le coefficient de pondération.

Contrairement à CCNN et Hydra qui traitent des patches de 72×72 pixels, TinyNet utilise des patches plus larges de 320×240 pixels pour mieux capturer le contexte spatial et les interactions entre véhicules proches.

6.2.2 Sélection des pics et changement de la loss

Le choix de la focal loss s'avère crucial pour améliorer la précision du comptage. La focal loss pénalise fortement les faux positifs (pixels prédits comme véhicules alors qu'ils ne le sont pas), ce qui améliore significativement la qualité de la point-map dans les régions d'arrière-plan peu denses.

La focal loss est définie comme :

$$\mathcal{L}_{\text{focal}} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{5}$$

avec $\alpha_t = 0.25$ pour les pixels négatifs, $\alpha_t = 0.75$ pour les pixels positifs, et $\gamma = 2$ (facteur de focalisation). Ce paramétrage force le modèle à se concentrer sur les pixels difficiles à classifier (surtout les faux positifs en région vide).

La supervision par point-map binaire (1 pixel unique par véhicule) est plus directe et stable que les cartes de densité gaussienne. Elle évite les ambiguïtés liées au choix de σ et permet une optimisation plus fine de la localisation.

L'intégration d'une data augmentation agressive (scaling 0.8–1.2×, flipping, colour jitter, cutout) renforce la robustesse du modèle face aux variations de perspective et d'illumination.

6.2.3 Counting : de la point-map au comptage

Le comptage dans TinyNet s'effectue en deux étapes. Tout d'abord, la point-map prédite (valeurs continues dans $[0, 1]$) est binarisée à un seuil de 0.5 pour obtenir une carte binaire. Chaque pixel à 1 représente un véhicule potentiel.

Le nombre total de véhicules est ensuite obtenu en sommant directement les valeurs de la point-map avant binarisation :

$$C_{\text{pred}} = \sum_{i,j} \hat{P}_{i,j} \tag{6}$$

Cette approche de sommation directe offre plusieurs avantages : (1) elle évite les erreurs d'arrondi et de discrétisation, (2) elle permet une optimisation en continu de la loss MSE pendant l'entraînement, et (3) elle génère naturellement un nombre de véhicules qui peut être fractionnaire pendant l'entraînement, offrant une meilleure régularisation que les approches de comptage discret.

Expérimentation et résultats

TinyNet est entraîné sur TRANCOS v3 avec les hyperparamètres suivants : optimiseur Adam avec un taux d'apprentissage de 5×10^{-4} et un facteur de décroissance exponentielle $\gamma = 0.95$ par époque, une taille de batch de 16 et une patience de 5 époques pour l'*early stopping*.

Sur l'ensemble de test, TinyNet obtient un GAME(0) de 3.45. Les valeurs obtenues pour les différents niveaux de GAME sont présentées dans le tableau **??**. L'erreur augmente progressivement avec la finesse de la grille, passant de 3.45 pour GAME(0) à 10.99 pour GAME(3). Cette progression indique que la localisation spatiale des véhicules constitue une difficulté supplémentaire par rapport au simple comptage global.

Malgré une architecture nettement plus légère que les modèles étudiés précédemment et l'absence de pré-entraînement, TinyNet obtient ainsi des performances comparables aux meilleurs résultats de référence présentés précédemment. Les résultats qualitatifs montrent également une bonne localisation des véhicules et peu de faux positifs dans les régions vides. Les principales difficultés apparaissent dans les scènes très denses, où les véhicules sont fortement imbriqués ou partiellement occultés.

Table 11 – TinyNet results on the TRANCOS test set.

Method	GAME(0)	GAME(1)	GAME(2)	GAME(3)
TinyNet (this work)	**3.45**	**5.85**	**8.12**	**10.99**

7 Synthèse du stage

7.1 Rédaction d'un article scientifique

L'idée du stage étant également de me faire découvrir ce qu'est la recherche, il était temps de passer à l'étape suivante, la rédaction d'un article scientifique.

Étant mon premier article, je me suis insipiré de la structure des autres articles, comme pour ce rapport on présente :

— Related work
— Présentation de notre travail
— Expérimentation
— Résultats

L'idée initiale était de faire une étude comaprative de toutes les méthodes sur un setup d'expérimentation commun. Cependant, au vu de la différence non négligeable dans l'implémentation des méthodes existantes, j'ai trouvé absurde d'en parler et de comparer ma méthode à celà. J'ai donc fait comme les autres papiers scientifique, j'ai comparé mes résultats aux résultats des auteurs dans leurs papiers.

7.2 Résultats sur TRANCOS

TABLE 12 – Résultats de référence en GAME sur TRANCOS.

Méthode	GAME(0)	GAME(1)	GAME(2)	GAME(3)
CCNN [3]	12.49	16.58	20.02	22.41
Hydra-3s [3]	10.99	13.75	16.69	19.32
CSRNet [4]	3.56	5.49	8.57	15.04
TinyNet (our)	**3.45**	**5.85**	**8.12**	**10.99**

TinyNet obtient des résultats du même ordre de grandeur que les méthodes de référence, avec un GAME(0) de 3.45, proche de celui rapporté pour CSRNet (3.56). L'écart avec CSRNet reste également faible pour GAME(1), avec respectivement 5.85 et 5.49. Pour les niveaux GAME(2) et GAME(3), TinyNet obtient en revanche des valeurs inférieures à celles rapportées pour CSRNet, avec respectivement 8.12 et 10.99. Ces résultats sont encourageants, mais ne permettent pas pour autant d'affirmer que TinyNet est plus performant que les méthodes existantes. En effet, les valeurs de référence proviennent des expérimentations originales des différents articles et les conditions d'entraînement, les paramètres et certains choix d'implémentation peuvent différer de ceux utilisés pour TinyNet. Cette comparaison permet donc principalement de situer les performances obtenues par TinyNet par rapport aux résultats publiés, plutôt que d'établir une comparaison directe entre les méthodes.

7.3 Complexité des modèles

TABLE 13 – Comparaison du nombre de paramètres des différentes architectures.

Méthode	Paramètres (M)	Facteur vs. TinyNet
TinyNet	0.19	1.00×
CCNN	0.57	3.00×
Hydra-3s	201.21	1059×
CSRNet	16.26	85.58×

Le tableau 13 met en évidence la faible complexité de TinyNet par rapport aux architectures étudiées. Avec seulement 0.19 million de paramètres, TinyNet est environ trois fois plus petit que CCNN et nettement plus léger que CSRNet et Hydra-3s, qui possèdent respectivement environ 86 et 1059 fois plus de paramètres. Même si la réduction du nombre de paramètres ne s'accompagne pas nécessairement de meilleures performances pour tous les niveaux de GAME, elle permet d'obtenir une architecture beaucoup plus compacte. Cette différence est particulièrement intéressante dans le cas de CCNN : TinyNet possède moins de paramètres tout en obtenant des performances nettement meilleures sur TRANCOS. Ces résultats montrent ainsi que la réduction de la complexité du modèle ne se fait pas nécessairement au détriment des performances, ce qui constitue un intérêt important de TinyNet pour des applications où les ressources de calcul sont limitées.

8 Conclusion

Ce stage portait sur l'amélioration du comptage de véhicules par régression de carte de densité sur TRANCOS. Nos réimplémentations de CCNN, Hydra-3s et CSRNet s'écartent des résultats publiés. Elles ont cependant servi de baselines, sauf Hydra-3s, dont l'implémentation ne nous a pas paru exploitable.

Les modifications testées sur CCNN (noyau adaptatif KNN, patchs centrés sur les véhicules) n'ont pas amélioré le comptage global. Sur CSRNet, le back-end multi-échelle et la loss combinée réduisent nettement les erreurs de localisation (GAME(1) à GAME(3)), sans gain notable sur GAME(0). Ces résultats proviennent d'un seul entraînement par configuration. Ils devraient être confirmés par plusieurs répétitions et des ablations.

TinyNet, architecture de 0.19 million de paramètres proposée par mon tuteur, obtient sur TRANCOS des erreurs du même ordre que les valeurs publiées pour CSRNet, avec 85 fois moins de paramètres. Cette comparaison reste indicative, car elle s'appuie sur des résultats publiés obtenus dans des conditions d'entraînement différentes. Il faudrait aussi mesurer le temps d'inférence et isoler l'apport de chaque composant (focal loss, taille des patchs, augmentation de données).

J'ai remis à l'université de Kasetsart un article en anglais, en annexe, ainsi qu'un sujet de recherche que mon tuteur poursuivra pour son propre article, sur lequel j'interviendrai comme co-auteur.

Glossaire

DL Deep Learning, sous-domaine du Machine Learning reposant principalement sur l'utilisation de réseaux de neurones artificiels comportant plusieurs couches.. 4

IA Intelligence Artificielle, ensemble de méthodes permettant à une machine d'effectuer des tâches nécessitant habituellement des capacités cognitives humaines.. 4

KNN K-Nearest Neighbors. 3, 16

KU Kasetsart University, université thaïlandaise située à Bangkok dans laquelle le stage a été réalisé.. 4

ML Machine Learning, ensemble de méthodes permettant à un système informatique d'apprendre à partir de données afin d'effectuer une tâche sans être explicitement programmé pour chaque situation.. 1, 4, 14

Bibliographie

[1] Qingyu Song et al. "Rethinking Counting and Localization in Crowds : A Purely Point-Based Framework". In : *arXiv preprint arXiv :2107.12746* (2021). DOI : 10.48550/arXiv.2107.12746.

[2] Joseph Redmon et al. "You Only Look Once : Unified, Real-Time Object Detection". In : *arXiv preprint arXiv :1506.02640* (2016). DOI : 10.48550/arXiv.1506.02640.

[3] Daniel Oñoro-Rubio et Roberto J. López-Sastre. "Towards perspective-free object counting with deep learning". In : *European Conference on Computer Vision (ECCV)*. 2016, p. 615-629. DOI : 10.1007/978-3-319-46478-7_38.

[4] Yuhong Li, Xiaofan Zhang et Deming Chen. "CSRNet : Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes". In : *arXiv preprint arXiv :1802.10062* (2018). Submitted April 11, 2018.

[5] Ricardo Guerrero-Gómez-Olmedo et al. "Extremely Overlapping Vehicle Counting". In : *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*. Springer, 2015, p. 423-431.

Annexes

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
☎ 00.33 (0) 2.98.34.87.70 / stages.brest@ensta.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name*_____ Kasetsart University_____

Adresse / *Address*_____Department of Computer Engineering, Kasetsart University 50 Ngamwongwan Rd, Chatuchak Bangkok 10900 Thailand_____

Tél / *Phone (including country and area code)*_____Tel. (+66) 0-2797-0999 ext 1403-04 1450___

Nom du superviseur / *Name of internship supervisor*
_____Piya Limcharoen_____
Fonction / *Function*_____Lecture_____

Adresse e-mail / *E-mail address*_____piya.lim@ku.th_____

Nom du stagiaire accueilli / *Name of intern* Florian Milin

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please attribute a mark from **A (excellent)** to **F (very weak).***

.1 MISSION / *TASK*

❖ La mission de départ a-t-elle été remplie ? A **B** C D E F
Was the initial contract carried out to your satisfaction?

❖ Manquait-il au stagiaire des connaissances ? oui/*yes* **non/*no***
Was the intern lacking skills?

Si oui, lesquelles ? / *If so, which skills?* _____

.2 ESPRIT D'EQUIPE / *TEAM SPIRIT*

❖ Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)*
A **B** C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here*_____
_____

Version du 05/04/2019

.3 COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?
Did the intern live up to expectations? (*Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?*

A **B** C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____
_____

.4 INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? A B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? A **B** C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____
_____

.5 CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? A **B** C D E F
Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____
_____

.6 OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : A **B** C D E F
Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? **oui/*yes*** non/*no*

Fait à _____, le _____
In _Kasetsart University, Bangkok__, on__Starting from April (same period as this year)__

Signature Entreprise_____Signature stagiaire
Company stamp _____*Intern's signature*

Merci pour votre coopération
We thank you very much for your cooperation

A Comparative Study of Density and Point-Based CNNs for Vehicle Counting on TRANCOS

Florian Milin, Piya Limcharoen

August 22, 2026

Abstract

Vehicle counting in congested traffic is challenging because vehicles frequently overlap, vary in size, and are difficult to detect individually. This work compares three density-based CNNs, CCNN, Hydra-3s and CSRNet, with TinyNet, a lightweight point-based model, on the TRANCOS benchmark. All approaches are evaluated using the GAME metric, with a particular focus on the trade-off between counting accuracy and model complexity. TinyNet achieves a GAME(0) of 3.45, close to the published CSRNet result of 3.56, while using only 0.19M parameters compared with 16.26M for CSRNet and 201.21M for Hydra-3s. These results suggest that point-based counting can achieve competitive accuracy while significantly reducing model complexity. However, differences between the original experimental protocols prevent a strict comparison of the reported results.

Contents

1 Introduction **3**

2 Related Work **4**
- 2.1 Counting Approaches . . . 4
  - 2.1.1 Counting by detection . . . 4
  - 2.1.2 Counting by density estimation . . . 4
- 2.2 Compared Architectures . . . 4
  - 2.2.1 Common counting pipeline . . . 4
  - 2.2.2 CCNN . . . 5
  - 2.2.3 Hydra-3s . . . 5
  - 2.2.4 CSRNet . . . 5
  - 2.2.5 Summary of the Compared Architectures . . . 6

3 Method **6**
- 3.1 Architecture : TinyNet . . . 6
- 3.2 Peak selection and loss . . . 7
- 3.3 Counting . . . 9
- 3.4 Summary of the Methods Architectures . . . 9

4 Experiments **10**
- 4.1 Dataset: TRANCOS . . . 10
- 4.2 Evaluation metric: GAME . . . 11
- 4.3 Experiment setup . . . 11
  - 4.3.1 CCNN . . . 11
  - 4.3.2 Hydra-3s . . . 12
  - 4.3.3 CSRNet . . . 12
  - 4.3.4 TinyNet . . . 12
  - 4.3.5 Summary of Experimental Settings . . . 13
- 4.4 Model Complexity . . . 13

5 Results on TRANCOS **13**
- 5.1 Published Reference Results . . . 14
- 5.2 TinyNet Results . . . 14
- 5.3 Accuracy and Model Complexity . . . 15

6 Conclusion **15**

1 Introduction

Counting vehicles in traffic images is useful for many transport applications, such as measuring congestion or estimating travel time. In dense traffic this is difficult because vehicles overlap, partially hide each other, and appear at very different sizes due to the camera perspective. These conditions make it difficult to reliably identify and count every vehicle individually.

A common approach to dense counting is to predict a *density map* instead of detecting each vehicle individually. A density map represents the spatial distribution of the vehicles in an image. The total number of vehicles can then be obtained by summing the values of the map. This representation is particularly useful in crowded scenes because several nearby vehicles can contribute to the same local region of the map.

Several network designs have been proposed for density-based counting, from simple single-scale networks to multi-scale architectures and fully convolutional networks using dilated convolutions. Comparing these methods directly from their original papers is difficult because the experiments often use different data splits, density-map generation procedures, and training settings. On TRANCOS, the reported results can therefore reflect both the network architecture and the experimental protocol.

In this paper, we study four representative approaches on the same dataset and use a common evaluation framework for our experiments. When available, results reported in the original papers are also included as reference points. We consider CCNN [1], a small single-scale network, Hydra-3s [1], its multi-scale extension, and CSRNet [2], which uses a VGG-16 [3] front-end followed by dilated convolutions. These three networks predict a continuous density map and do not require a perspective map. We also introduce TinyNet, our own point-based network, which instead predicts a probability map with localised peaks and counts vehicles from these detected points rather than by summing a continuous density map.

The main goal of this study is to characterise the trade-offs between density-based and point-based approaches for vehicle counting on TRANCOS, with particular emphasis on counting accuracy, model complexity to increasing vehicle density. Our contributions are the following:

- a comparative study of density-based and point-based approaches for vehicle counting on the TRANCOS benchmark
- re-implementations of CCNN, Hydra-3s and CSRNet used as reference implementations for our experimental study
- TinyNet, a lightweight point-detection alternative to density regression
- an analysis of the trade-off between counting accuracy and model complexity

The paper is organised as follows. Section 2 reviews related work. Section 2.2 presents CCNN, Hydra-3s and CSRNet. Section 3 presents TinyNet. Section 4 describes the experimental setup and reports the results on TRANCOS. Section 6 concludes the paper.

2 Related Work

2.1 Counting Approaches

2.1.1 Counting by detection

One family of methods counts by detecting each object and summing the detections. More recent approaches use convolutional object detectors, such as Faster R-CNN [4], YOLO [5] or SSD [6]. They work well when the vehicles are well separated, but in congested traffic the heavy overlap causes missed detections, and each missed vehicle is a counting error. For this reason, density-based approaches provide an attractive alternative for highly congested scenes.

2.1.2 Counting by density estimation

Density-estimation methods avoid detecting objects one by one. Instead of localising each vehicle, they learn to turn an image into a density map, an image where each pixel tells how many objects are nearby, and obtain the count by summing this map. Since nearby objects simply add up, the count stays correct even when they overlap. Convolutional networks made this idea much more accurate, although early multi-column designs were later criticised for learning redundant features [2], which motivated the simpler single-column, dilated-convolution design of CSRNet. For vehicles, some methods also use the temporal information of video, such as FCN-rLSTM [7]. From this family we compare three representative and historically linked networks, CCNN [1] (single-scale), its scale-aware extension Hydra-3s [1] (multi-scale), and CSRNet [2] (dilated), which we describe in detail in Section 2.2. Unlike these density-based approaches, our TinyNet model uses a lightweight point-based baseline which is described in Section 3.

2.2 Compared Architectures

2.2.1 Common counting pipeline

The three networks share the same counting pipeline. The ground-truth density map of an image is built by placing a normalised Gaussian kernel at each dot annotation, so that the sum of the map equals the number of vehicles. Each network is trained to regress this map with a pixel-wise Euclidean (mean-squared-error) loss, and the predicted count is obtained by

summing the predicted map over the region of interest. CCNN and Hydra are patch-based: they are trained on fixed-size 72×72 patches, and at test time the image is scanned with a sliding window (stride 10) and the patch predictions are assembled into a full-image density map. CSRNet is fully convolutional and processes the whole image directly. The dataset split, the Gaussian kernel width and the other training settings are given in Section 4.

2.2.2 CCNN

CCNN [1] is a small fully-convolutional network of six convolutional layers that maps a 72×72 patch to an 18×18 density map, with a single Euclidean loss and no perspective information. In our implementation, training patches are extracted at random positions inside the ROI (with horizontal flipping for augmentation), and the network is optimised with SGD. CCNN is the lightest of the three models and serves as the single-scale baseline.

2.2.3 Hydra-3s

Hydra-3s [1] makes the model scale-aware. From each training patch it builds a three-level pyramid (the full patch and two centred crops at 66% and 33%, each resized to 72×72) and feeds each level to one of three heads; a head is a CCNN without its last layer, so it outputs an $18 \times 18 \times 400$ feature map. The three feature maps are concatenated and passed to a fully-connected body of two hidden layers followed by an output layer reshaped into the 18×18 density map. This fully-connected body makes Hydra-3s by far the largest model of the three. Patch extraction, augmentation and sliding-window inference are the same as for CCNN.

2.2.4 CSRNet

CSRNet [2] is a whole-image, single-column network. It uses the first ten convolutional layers of VGG-16 [3] as a front-end and a back-end of dilated convolutions (dilation rate 2), which enlarges the receptive field while keeping the resolution at 1/8 of the input. It has about 16M parameters, far fewer than Hydra-3s. In our implementation, the front-end is initialised from ImageNet and the whole network is pre-trained on the ShanghaiTech crowd-counting dataset before being fine-tuned on TRANCOS. Training patches follow the CSRNet protocol (nine crops per image at half resolution, each mirrored) and the network is optimised with SGD. This difference in initialization should be considered when interpreting the controlled comparison.

2.2.5 Summary of the Compared Architectures

Table 1: Architectural comparison of the approaches considered in this study

Method	Representation	Scale handling	Inference	Parameters
CCNN	Density map	Single-scale	Sliding window	0.57M
Hydra-3s	Density map	Multi-scale	Sliding window	201.21M
CSRNet	Density map	Dilated convolutions	Whole image	16.26M

3 Method

The three networks above all regress a continuous density map. In contrast, our TinyNet implementation uses a point-based formulation and treats vehicle counting as a peak-detection problem. Rather than predicting a continuous density map, it produces a probability map in which each vehicle should correspond to a localised peak. A vehicle is counted when the network produces a sufficiently confident peak at its location.

3.1 Architecture : TinyNet

TinyNet uses a MobileNetV2 [8] backbone, kept only up to a stride of 8. We denote the three intermediate feature maps as f_2, f_4, and f_8, corresponding to spatial resolutions of 1/2, 1/4, and 1/8 of the input resolution, respectively. These maps provide complementary spatial information: f_2 preserves finer details, while f_4 and f_8 provide increasingly broader contextual information. Each map is reduced to 32 channels with a 1×1 convolution, and the two coarser maps, f_4 and f_8, are bilinearly upsampled to the resolution of f_2. A fourth branch processes the raw RGB image directly with a strided convolution to produce a 32-channel feature map at the same resolution. The four 32-channel maps are then concatenated and merged by two convolutional blocks with width 64, group normalisation, and a SiLU activation. Strong spatial dropout with $p = 0.5$ is applied to each branch before merging. A small decoder then upsamples the merged features twice by a factor of 2 using bilinear upsampling followed by a convolution, so the network predicts at twice the input resolution. This extra resolution makes it easier to separate two vehicles that are close together before they are counted. A final 1×1 convolution and a sigmoid turn this into a probability map, where each pixel represents the probability that a vehicle is centred at that location.

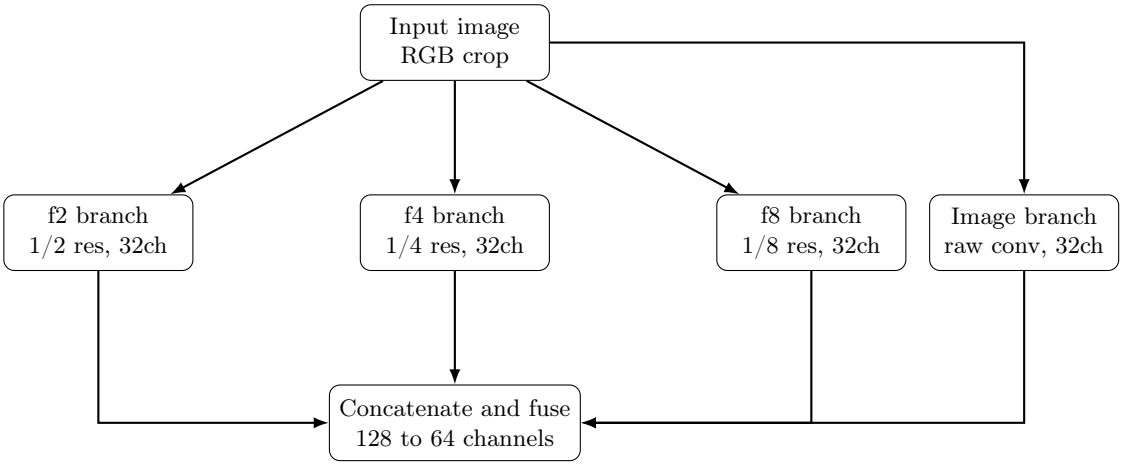


Figure 1: TinyNet encoder and multi-scale fusion. The input image feeds four parallel branches (three MobileNetV2 feature scales and a raw-image branch), which are concatenated and fused into a single feature map.

3.2 Peak selection and loss

To turn the probability map into discrete points, we add a non-maximum-suppression module: in every local 11×11 window, only the highest value is kept and every other value in the window is set to zero. This is applied at the double-resolution output, which is then max-pooled by 2 to obtain a peaked map at the original resolution. We use max-pooling rather than average-pooling so that a single sharp peak is not smoothed away when the resolution is halved.

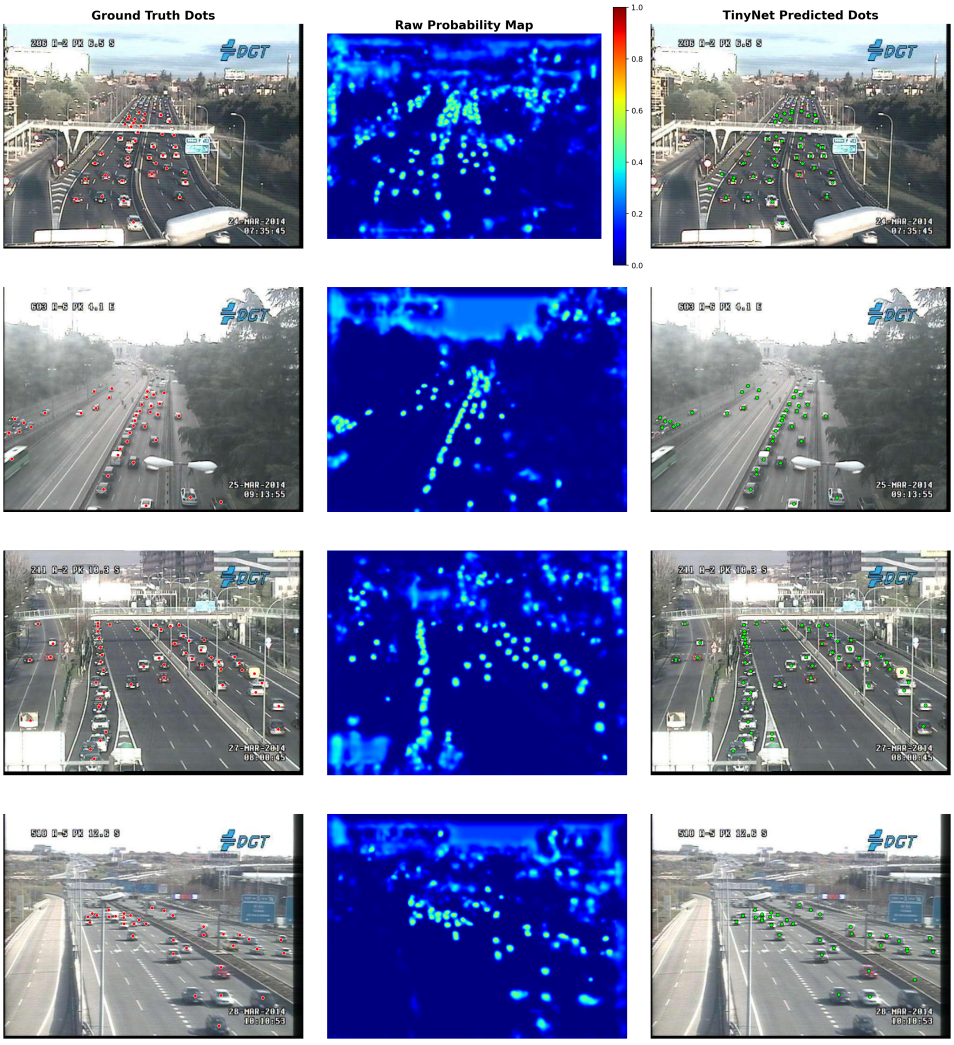


Figure 2: Qualitative TinyNet predictions on TRANCOS.

The ground truth is a binary point map, with a single pixel set to 1 at each vehicle annotation. The network is trained with a focal loss, computed on this point map at both the original and the double resolution and then summed. Positive and negative pixels are re-weighted so that the very large number of empty pixels does not dominate the loss. A second term penalises the squared error between the predicted count (the sum of the peaked map inside the ROI) and the true count. The two terms are combined into a single loss and optimised with AdamW. A Gaussian-blurred density map is also computed for each training image but is not used as a supervision target in the final loss. Only the point map and the count are supervised. Training patches are random 320×240 crops (half the image size), obtained after random scaling, cropping, flipping, colour jitter and cutout.

3.3 Counting

At test time, the peaked map is thresholded at 0.5 and the resulting points are summed inside the ROI to obtain the predicted count. This count is used for the GAME evaluation in the same grid-based way as for the other three networks (Section 4.2). Unlike CCNN, Hydra-3s and CSRNet, TinyNet never integrates a continuous density map: its final output is already a set of discrete, countable points.

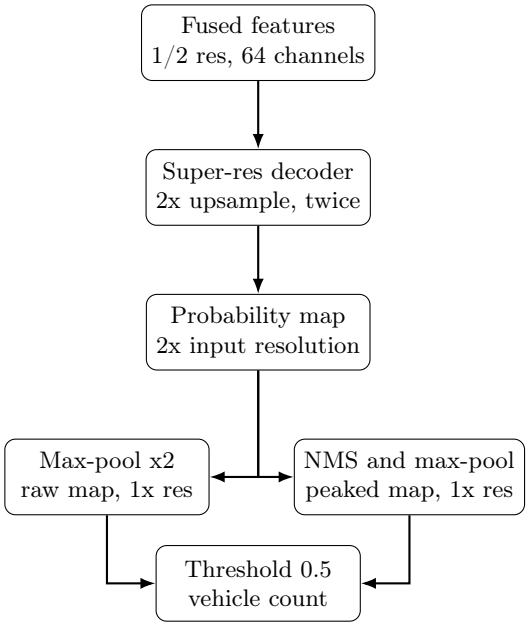


Figure 3: TinyNet decoder and peak-based counting. The fused features are decoded to twice the input resolution to produce a probability map, processed by a raw max-pooling path and a non-maximum-suppression path, and combined by thresholding to give the final vehicle count.

3.4 Summary of the Methods Architectures

Table 2: Architectural comparison of the approaches considered in this study

Method	Representation	Scale handling	Inference	Parameters
CCNN	Density map	Single-scale	Sliding window	0.57M
Hydra-3s	Density map	Multi-scale	Sliding window	201.21M
CSRNet	Density map	Dilated convolutions	Whole image	16.26M
TinyNet	Point map	Multi-scale	Whole image	0.19M

4 Experiments

4.1 Dataset: TRANCOS

TRANCOS [9] is a public benchmark for vehicle counting in congested traffic. It contains 1,244 images of 640 × 480 pixels captured by real video-surveillance cameras across many different scenes, for a total of 46,796 vehicles annotated by dots. The images are not taken from a fixed viewpoint and no perspective maps are provided, so a method must generalise across scenes and viewpoints rather than exploit any scene-specific calibration. Each image is accompanied by a region of interest (ROI) that delimits the area used for both supervision and evaluation.

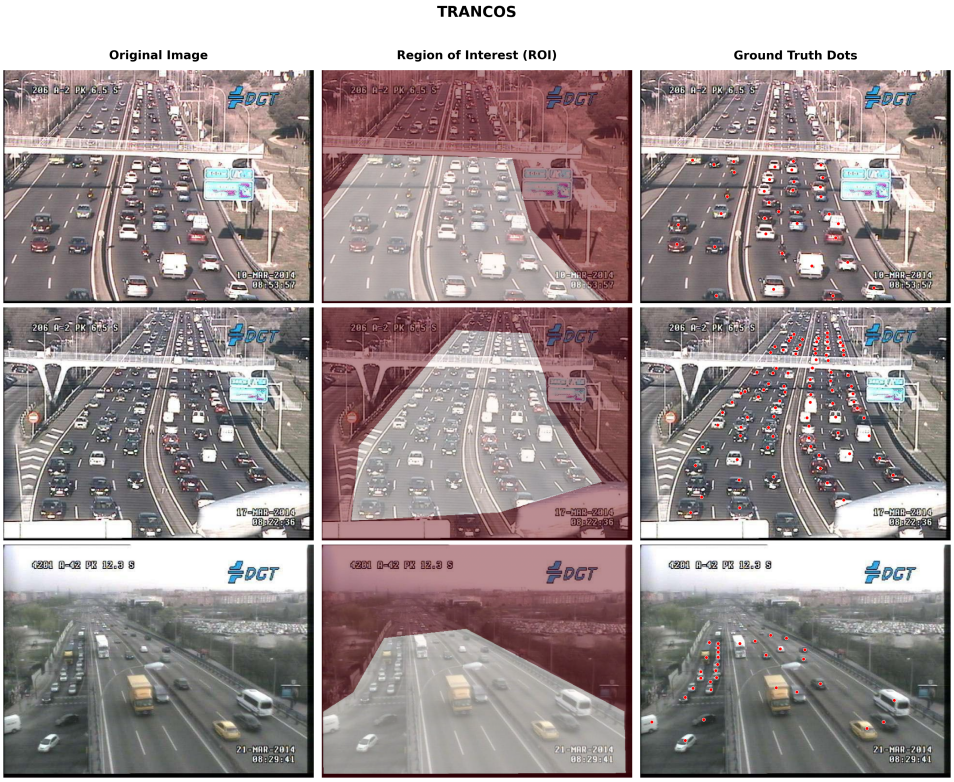


Figure 4: Examples from the TRANCOS dataset showing the image, ROI, and ground-truth annotations.

We follow the official partition into a training set (403 images), a validation set (420 images) and a test set (421 images). As in the protocol of the CCNN and Hydra models [1], all three networks are trained on the training and validation sets combined (823 images) and evaluated only once on the held-out test set; the test set is never used for model selection, so that the reported error is free of any test-set leakage.

Ground-truth density maps are generated by placing an isotropic Gaussian kernel of fixed standard deviation $\sigma = 15$ at each dot annotation, using the toolkit released with the dataset, and are masked by the ROI. The same kernel and the same generation procedure are applied to the three architectures, so that any difference in accuracy reflects the network and not the supervision.

4.2 Evaluation metric: GAME

We evaluate all models with the Grid Average Mean Absolute Error (GAME), the metric introduced together with TRANCOS [9]. GAME refines the mean absolute error by also rewarding predictions that place the density in the right location. For a level L, the image is divided into a regular grid of 4^L non-overlapping regions covering the ROI, and the error is the sum of the absolute count errors over these regions, averaged over the test set:

$$\mathrm{GAME}(L) = \frac{1}{N} \sum_{n=1}^{N} \sum_{l=1}^{4^L} \left| \hat{c} * n^l - c * n^l \right|, \tag{1}$$

where N is the number of test images, and $\hat{c}*n^l$ and $c*n^l$ are respectively the predicted and ground-truth counts in region l of image n. For density-based models, the regional predicted counts are obtained by integrating the predicted density map over each region. For TinyNet, the regional counts are obtained by counting the detected points inside each region.

A higher level L produces a finer grid and thus a stricter, more spatially demanding criterion. A model that predicts the correct global count but distributes the density poorly is penalised at high L but not at low L. In particular, GAME(0) uses a single region equal to the whole image and therefore coincides with the mean absolute error (MAE) on the count. Following common practice on TRANCOS [1, 2], we report GAME(0) through GAME(3), corresponding to grids of 1, 4, 16 and 64 regions.

4.3 Experiment setup

We follow the training settings reported in the original papers where they are available, while adapting the density-map generation and evaluation procedure where necessary to obtain a common protocol. The training procedure for each network is described below.

4.3.1 CCNN

CCNN is trained with SGD [1], using 800 random 115×115 patches per training image. The patches are resized to 72×72 and mirrored, resulting in 1,600 training patches per image. The ground-truth maps use a Gaussian

kernel with $\sigma = 15$, and inference scans the test image with a stride of 10 pixels [1].

4.3.2 Hydra-3s

Hydra-3s is trained with the same patch extraction and training procedure as CCNN. It uses SGD [1], with 800 random 115×115 patches per training image. The patches are resized to 72×72 and mirrored, resulting in 1,600 training patches per image. The ground-truth maps use a Gaussian kernel with $\sigma = 15$, and inference scans the test image with a stride of 10 pixels [1].

4.3.3 CSRNet

CSRNet is trained with SGD [2], with a fixed learning rate of 10^{-6}. Training uses nine crops per image at one quarter of the original resolution, consisting of four non-overlapping quadrants and five random crops. Each crop is mirrored. The original CSRNet paper uses $\sigma = 10$ for TRANCOS [2]. For our controlled comparison, we instead use $\sigma = 15$ for CSRNet so that CCNN, Hydra-3s and CSRNet are trained with the same density-map generation procedure.

4.3.4 TinyNet

TinyNet is trained with AdamW at a learning rate of 10^{-4}, using random 320×240 crops with scaling, flipping, colour jitter and cutout, as described in Section 3.2. TinyNet does not use a Gaussian density map as a supervision target. It is trained using the binary point map and the count loss described in Section 3.2.

Overall, CCNN, Hydra-3s and CSRNet use the same Gaussian kernel with $\sigma = 15$ in our controlled experiments, while TinyNet uses a point-map formulation and does not rely on density-map supervision. This common protocol allows differences in performance to be attributed more directly to the network architectures rather than to differences in the ground-truth representation.

4.3.5 Summary of Experimental Settings

Table 3: Summary of the experimental settings.

	CCNN	Hydra-3s	CSRNet	TinyNet
Input	72×72	72×72	Half-image	320×240
Representation	Density	Density	Density	Points
Loss	MSE	MSE	MSE	Focal + count
Optimizer	SGD	SGD	SGD	AdamW
Pre-training	–	–	ShanghaiTech	ImageNet

4.4 Model Complexity

Table 4 compares the parameter counts of the four architectures. The values reported for CCNN, Hydra-3s and CSRNet are taken from their corresponding implementations or publications, while the TinyNet value is measured from our implementation. Parameter count is independent of the hardware, whereas GFLOPs and inference speed depend on the input resolution, implementation and hardware.

Table 4: Model size comparison with TinyNet.

Method	Parameters (M)	× Proposed
TinyNet	0.19	1.0×
CCNN	0.57	3.0×
Hydra-3s	201.21	1059×
CSRNet	16.26	85.58×

5 Results on TRANCOS

We first report the results published for CCNN, Hydra-3s and CSRNet on TRANCOS as reference points. These values are taken from the corresponding papers and are not treated as results obtained under a common experimental protocol. The original publications do not provide all implementation and training details required for an exact reproduction, and some methodological choices therefore had to be made in our implementations. Our reproduced results can consequently differ from the published values. For this reason, we do not present our implementations as exact reproductions of the original experiments.

5.1 Published Reference Results

Table 5: Published GAME results on TRANCOS for the compared density-based architectures. These values are taken from the original publications and are reported only as reference results.

Method	GAME(0)	GAME(1)	GAME(2)	GAME(3)
CCNN [1]	12.49	16.58	20.02	22.41
Hydra-3s [1]	10.99	13.75	16.69	19.32
CSRNet [2]	3.56	5.49	8.57	15.04

The published results show a clear progression from the single-scale CCNN architecture to the multi-scale Hydra-3s model and then to CSRNet. CCNN obtains a GAME(0) of 12.49, while Hydra-3s reduces the global counting error to 10.99. CSRNet achieves a substantially lower GAME(0) of 3.56 in its original experimental setting. The same general progression is observed for GAME(1) and GAME(2). At GAME(3), however, the difference between CSRNet and the other architectures becomes smaller, with CSRNet obtaining a value of 15.04.

These results illustrate the potential benefit of increasingly expressive architectures for dense vehicle counting. However, they should not be interpreted as a controlled comparison because the original experiments use different training procedures, density-map settings and implementation details. The values are therefore used here to provide historical and methodological context rather than to establish a direct ranking under identical conditions.

5.2 TinyNet Results

We next evaluate TinyNet on the held-out TRANCOS test set containing 421 images. Unlike the results reported above, these values are obtained from our implementation using the training procedure described in Section 4.3.4. TinyNet is evaluated using the same GAME metric as the density-based approaches, with the predicted points counted inside each evaluation region.

Table 6: TinyNet results on the TRANCOS test set.

Method	GAME(0)	GAME(1)	GAME(2)	GAME(3)
TinyNet (this work)	**3.45**	**5.85**	**8.12**	**10.99**

TinyNet obtains a GAME(0) of 3.45, corresponding to a mean absolute counting error of 3.45 vehicles per image. The error increases to 5.85 at GAME(1), 8.12 at GAME(2) and 10.99 at GAME(3). This progression indicates that TinyNet can obtain a relatively accurate global count while

making more localisation errors when the evaluation becomes increasingly spatially demanding.

The GAME(0) obtained by TinyNet is close to the published CSRNet value of 3.56. TinyNet also obtains a lower published-reference GAME(3) value than CSRNet, with 10.99 compared with 15.04. However, these values were obtained under different experimental conditions and should therefore not be interpreted as evidence that TinyNet systematically outperforms CSRNet. The comparison instead indicates that the lightweight point-based formulation achieves competitive counting accuracy on TRANCOS despite its substantially smaller number of parameters.

5.3 Accuracy and Model Complexity

The accuracy results can be considered together with the model complexity reported in Section 4.4. TinyNet contains approximately 0.19 million parameters, compared with 0.57 million for CCNN, 16.26 million for CSRNet and 201.21 million for Hydra-3s. TinyNet is therefore substantially smaller than all three density-based architectures considered in this study.

In particular, TinyNet uses approximately 85 times fewer parameters than CSRNet and more than 1000 times fewer parameters than Hydra-3s. Despite this difference in model size, its GAME(0) of 3.45 is close to the published CSRNet value of 3.56. This result suggests that point-based counting can provide a favourable accuracy to model-size trade-off for vehicle counting on TRANCOS.

6 Conclusion

In this study, we compared three density-based approaches, CCNN, Hydra-3s and CSRNet, with our lightweight point-based TinyNet for vehicle counting on TRANCOS. The results highlight the evolution from simple single-scale density estimation to more expressive architectures, while showing that a point-based formulation can provide a different and lightweight alternative.

TinyNet achieves a GAME(0) of 3.45, which is close to the published CSRNet result of 3.56. More importantly, TinyNet uses only 0.19M parameters, making it substantially smaller than CCNN, Hydra-3s and CSRNet. This indicates that competitive counting accuracy can be achieved without relying on a large density-regression network.

However, the reported results were obtained under different experimental protocols, so they should not be considered a strict head-to-head ranking. Future work should evaluate all models under the same training and evaluation settings and investigate ways to improve TinyNet's localisation accuracy at higher GAME levels. Overall, these results suggest that lightweight point-based models are a promising direction for efficient vehicle counting in congested traffic scenes.

References

[1] D. Oñoro-Rubio and R. J. López-Sastre, "Towards perspective-free object counting with deep learning," in *European Conference on Computer Vision (ECCV)*, pp. 615–629, Springer, 2016.

[2] Y. Li, X. Zhang, and D. Chen, "CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1091–1100, 2018.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*, pp. 21–37, Springer, 2016.

[7] S. Zhang, G. Wu, J. P. Costeira, and J. M. F. Moura, "FCN-rLSTM: Deep spatio-temporal neural networks for vehicle counting in city cameras," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 3687–3696, 2017.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

[9] R. Guerrero-Gómez-Olmedo, B. Torre-Jiménez, R. López-Sastre, S. Maldonado-Bascón, and D. Oñoro-Rubio, "Extremely overlapping vehicle counting," in *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, pp. 423–431, Springer, 2015.