Estimation de profondeur à partir d'images monoculaires par apprentissage profond

Michel Moukari

▶ **To cite this version:**

Michel Moukari. Estimation de profondeur à partir d'images monoculaires par apprentissage profond. Traitement des images [eess.IV]. Normandie Université, 2019. Français. NNT : 2019NORMC211 . tel-02426260

HAL Id: tel-02426260
https://theses.hal.science/tel-02426260v1

Submitted on 2 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Estimation de profondeur à partir d'images monoculaires par apprentissage profond

Présentée et soutenue par
Michel MOUKARI

Thèse soutenue publiquement le 01/07/2019 devant le jury composé de		
M. VINCENT FREMONT	Professeur des universités, EC PR SCIENCES INFORMAT NANTES	Rapporteur du jury
M. PHILIPPE MARTINET	Professeur des universités, INRIA Sophia Antipolis	Rapporteur du jury
M. MICHEL DHOME	Directeur de recherche, Institut Pascal	Président du jury
Mme SYLVIANE PICARD	Responsable industriel, Safran	Membre du jury
M. LOIC SIMON	Maître de conférences, Université Caen Normandie	Membre du jury
M. FREDERIC JURIE	Professeur des universités, Université Caen Normandie	Directeur de thèse

Thèse dirigée par FREDERIC JURIE, Groupe de recherche en informatique, image, automatique et instrumentation

UNIVERSITÉ CAEN NORMANDIE

GREYC

Table des matières

<

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Figure 1.1 – Exemple d'un nuage de points 3D représentant une chaise

De nos jours, l'intelligence artificielle est devenue omniprésente. On la rencontre aussi bien à la maison qu'au travail ou dans les rues au travers d'une multitude d'outils tels que les smartphones, la domotique, les nouveaux dispositifs médicaux ou encore la publicité. Une étape importante pour doter une machine d'intelligence est de lui donner la capacité d'interagir avec son environnement afin qu'elle puisse prendre en compte des signaux extérieurs et y répondre en conséquence, comme nous le permettent nos sens. Différents capteurs permettent de donner ce type d'aptitude aux machines, il n'est par exemple pas rare que l'on soit entouré de caméras destinées à la sécurité, aux transports, au marketing ou tout simplement à notre confort.

La vision par ordinateur est la branche de l'intelligence artificielle dont le but est de permettre à une machine d'analyser, de traiter et de comprendre le contenu d'images ou de vidéos numériques. La compréhension de scène en particulier est un enjeu majeur en vision par ordinateur. Elle passe par une caractérisation à la fois sémantique et structurelle de l'image, permettant d'une part d'en décrire le contenu et, d'autre part, d'en comprendre la géométrie. Tandis que l'espace réel est de nature tridimensionnelle, l'image qui le représente, elle, est bidimensionnelle. Une partie de l'information 3D est donc perdue lors du processus de formation de l'image et il est d'autant plus complexe de décrire la géométrie d'une scène à partir d'images 2D de celle-ci.

1.1 Qu'est ce que l'inférence de cartes de profondeurs ? Pourquoi est-ce important ?

Il existe plusieurs manières de retrouver l'information de profondeur perdue lors de la formation de l'image. Dans cette thèse nous ne nous intéressons pas directement à la reconstruction 3D de scène, pour laquelle il est nécessaire de recouvrer les coordonnées 3D des points dans le repère monde (représentés sous la forme d'un nuage de points 3D, voir Figure 1.1) parce que ce type de représentation est relativement lourd et ne permet pas directement une correspondance point à point des pixels de l'image avec leurs coordonnées 3D (représentation parcimonieuse). Nous nous intéressons à la production d'une carte de profondeur (image 2D) étant donné une seule image de la scène. Dans ce cas, l'information de profondeur correspond, pour chaque pixel, à la distance entre la caméra et l'objet représenté en ce pixel.

FIGURE 1.2 – Exemple d'une paire d'images $\mathcal{RGB}$-$\mathcal{D}$. À gauche l'image $\mathcal{RGB}$ et à droite la carte de profondeur $\mathcal{D}$ correspondante. On peut la considérer comme une image en niveau de gris ou les valeurs des pixels correspondent à leur distance au capteur. Ici plus la distance est grande plus l'intensité lumineuse du pixel est élevée.

En d'autres termes, nous cherchons à estimer la carte de profondeur, dénotée $\mathcal{D}$, correspondant à une image couleur issue d'une caméra monoculaire classique, dénotée $\mathcal{RGB}$ de la scène. Un couple $\mathcal{RGB}$-$\mathcal{D}$ est illustré en Figure 1.2. Les chercheurs en vision par ordinateur ont depuis des années proposé différentes approches pour aborder le problème de l'estimation de la profondeur dans les images tant les applications sont nombreuses (Wei, 2005).

L'estimation automatique d'une carte de distances de la scène à partir d'une image est en effet une brique algorithmique critique dans de très nombreux domaines, avec une incidence considérable dans notre quotidien. Le secteur de l'automobile, par exemple, peut parfaitement en tirer parti, que ce soit pour l'autonomisation des systèmes ou simplement pour l'amélioration des véhicules personnels. Il suffirait d'ajouter une seule caméra sur le véhicule (ou d'en exploiter une déjà présente) pour aider à sa navigation et aux différentes manœuvres, incluant celles de sécurité. La carte de profondeur peut effectivement être utilisée pour localiser spatialement des obstacles tels que des piétons, d'autres voitures ou même des animaux traversant la route. Grâce à cette détection, la trajectoire du véhicule peut être adaptée pour éviter les collisions et ainsi réduire le nombre d'accidents.

Mais l'automobile n'est pas le seul secteur qui pourrait bénéficier de ce type d'algorithme. Toutes les industries dans lesquelles des systèmes autonomes sont amenés à évoluer dans un environnement non contrôlé (robotique, aéronautique, défense, spatial, etc.) sont concernées. En effet, l'estimation automatique de profondeur permet plus généralement de faciliter les interactions d'un agent avec son environnement par la compréhension globale de la scène et de sa structure. Il est par exemple envisageable d'utiliser l'estimation de profondeur pour l'aide au guidage de personnes malvoyantes. Il existe d'ailleurs déjà des travaux allant en ce sens, comme le projet ANR MOBI-DEEP. Son objectif principal est de développer à la fois des technologies permettant la navigation autonome de robots mobiles en milieux ouverts et inconnus, mais aussi l'aide au guidage des personnes déficientes visuelles au moyen de capteurs bas coût tels que des caméras numériques. Les contraintes principales sont des contraintes de légèreté et le fonctionnement en temps réel.

D'autre part, l'estimation de profondeur permettant de recouvrer la géométrie 3D de la scène, elle permet également d'aider à la détection et la reconnaissance d'objets compte tenu de leurs positions spatiales. Cela s'avère particulièrement utile dans le domaine de la vidéo-surveillance où la connaissance de la distance des personnes à la caméra a un impact direct sur les performances de détection, de reconnaissance et de suivi. C'est aussi le même principe en marketing pour proposer dans la rue de la publicité ciblée aux personnes passant devant des panneaux équipés de caméras. La reconstruction 3D à partir d'images quant à elle est beaucoup utilisée dans le domaine médical qui pourrait bénéficier de l'estimation monoculaire dans certains cas (utilisation d'une sonde équipée d'une caméra pendant une opération par exemple). Enfin, les applications impliquant la réalité augmentée ou la réalité virtuelle sont aussi tributaires de l'information 3D et de la géométrie de scène pour le calcul du placement d'objets virtuels relativement à l'utilisateur. Ces technologies sont notamment utilisées dans le secteur du jeu vidéo, mais tendent à se démocratiser dans d'autres domaines.

1.2 Vers l'inférence de cartes de profondeur par apprentissage statistique

Bien que le problème de l'estimation de profondeur à partir d'une seule image soit un problème difficile et intrinsèquement mal posé – nous le verrons en détail par la suite (chapitre 3) – nous savons qu'il est toujours possible pour l'Homme d'apprécier les distances avec un seul œil. Cette capacité n'est pas innée, mais elle est acquise et est en grande partie grace à l'identification d'indices reflétant de la connaissance a priori de ce qui nous entoure. Par ailleurs, nous savons que des algorithmes d'apprentissage automatique peuvent extraire ces indices directement depuis les images $\mathcal{RGB}$. Bien que des méthodes d'apprentissage statistique basées sur les réseaux de neurones existent depuis des décennies, ce n'est que récemment que les avancées matérielles et logicielles ainsi que la disponibilité croissante de données ont permis d'exploiter pleinement leur potentiel. Ces avancées ont contribué au développement rapide de réseaux de neurones de plus en plus complexes qui établissent une nouvelle littérature appelée apprentissage profond ou *deep learning*. Le *deep learning* ayant permis des percées majeures dans de nombreux domaines comme notamment celui de la vision par ordinateur, avec des performances surpassant parfois celles de l'Homme (Kokkinos, 2015; Lu et Tang, 2015), nous nous intéresserons en particulier à ce type de méthodes dans le cas de l'estimation de profondeur monoculaire.

Dans le cadre du *deep learning*, différentes problématiques se posent à nous. D'abord dans un contexte supervisé, il est nécessaire d'avoir à disposition de grandes quantités de données annotées pour la tâche à apprendre. De plus, il est nécessaire que ces données couvrent correctement la distribution des données qui seront effectivement utilisées en pratique lors du test de l'algorithme. C'est en particulier vrai pour l'estimation de profondeur où le type de scène

vu pendant l'apprentissage influencera grandement le comportement du réseau en situation de test. En effet, on peut savoir à l'avance qu'une caméra ne fonctionnera qu'en extérieur comme dans le cas de la conduite autonome, mais des changements de conditions météorologiques (pluie, neige, brouillard) peuvent grandement modifier les sorties produites par l'algorithme si la distribution des données d'entraînement ne couvrait pas ces cas de figure. À l'inverse, un réseau appris avec uniquement des données d'extérieur de tout type et avec un taux d'erreur très faible produira des sorties inutilisables en intérieur, les indices des profondeurs de ce genre de scène étant très différents. Or, il est ardu d'avoir au préalable des données de tout type, prévoyant à l'avance tous les cas de figure probables auxquels le réseau sera soumis.

Il existe justement des travaux de recherche spécifiques pour pallier ces difficultés comme l'apprentissage par transfert (*transfer learning*) qui consiste à transférer les connaissances d'un réseau entraîné à une tâche particulière sur une autre tâche partageant des similitudes (Pan et Yang, 2010). L'apprentissage avec très peu de données, voire un seul ou même aucun échantillon (*few-shot learning*, *one-shot learning* et *zero-shot learning*) adresse également ce problème (Wang et al., 2019). D'autres s'intéressent au développement d'algorithmes non supervisés pour apprendre des espaces de représentation des données sans annotations cibles (Godard et al., 2017). Enfin, des recherches se portent sur le développement de bases de données d'images de synthèse. La synthèse d'images a l'avantage de permettre de générer des données rapidement avec des environnements contrôlés et en très grande quantité. Il faut ensuite apprendre à utiliser correctement ces données de telle sorte que les réseaux en tirent parti dans des environnements réels, c'est l'adaptation de domaine (*domain adaptation*) (Ganin et Lempitsky, 2014).

Nous nous intéresserons plutôt dans cette thèse à tout ce qui pourra nous aider à produire les meilleures estimations de profondeur possible. Tout d'abord il faut préciser que cette notion est dépendante de l'application visée. Par exemple en navigation il est primordial que la distance des objets au premier plan soit correctement évaluée pour pouvoir éviter les obstacles potentiels, mais la précision n'a pas besoin d'être au pixel près et on peut accepter des erreurs sur les plans les plus lointains. D'un autre côté, pour des applications utilisant la réalité augmentée, une estimation de profondeur erronée de quelques pixels peut donner lieu à des projections 3D entièrement déformées. La qualité de la carte de profondeur est donc à juger quantitativement en mesurant des taux d'erreur moyens sur un jeu de données test, comme cela se fait usuellement, mais également qualitativement en fonction de la tâche finale ciblée. En outre, un des aspects importants inhérents aux algorithmes de régression est leur capacité à auto évaluer leurs résultats en affichant une mesure d'incertitude assortie à leurs prédictions. La production d'une telle sortie apprise par un réseau de neurones ainsi que l'évaluation de la qualité de cette incertitude sont autant de points qui ne sont encore que trop légèrement abordés dans la littérature et qui représentent pourtant le complément manquant à une prédiction complète. Pour finir, nous savons que le problème de l'estimation à partir d'une seule image est difficile et

il nécessite de pouvoir extraire d'une représentation en 2D des attributs à la fois géométriques et sémantiques de l'image. L'architecture de réseau adoptée joue un rôle important puisqu'elle permet de guider l'apprentissage dans une direction favorable. Il faut alors comprendre et choisir un réseau adapté pour l'application ciblée, celui qui permettra de trouver le meilleur compromis entre bonne prédiction, avec tout ce que cela implique, et complexité algorithmique.

1.3 Plan du mansucrit

Dans la suite de ce manuscrit, nous commencerons dans un premier temps par présenter un état de l'art général sur les méthodes d'estimation de profondeur, en particulier sur les méthodes issues de la littérature du *deep learning* (2).

Dans le deuxième chapitre, nous nous intéresserons aux nombreux hyperparamètres qui interviennent dans l'entraînement d'un réseau de neurones (architecture et optimisation) ainsi qu'à leur influence sur les capacités de généralisation d'un CNN dans le cas de l'estimation de profondeur monoculaire. Nous comparerons nos résultats entre eux ainsi qu'avec ceux de l'état de l'art pour comprendre quels sont les facteurs les plus importants pour atteindre les meilleures performances pour cette tâche (3).

Forts des conclusions tirées au chapitre précédent, nous tenterons d'obtenir des performances d'estimation de profondeur monoculaire qui surpassent l'état de l'art. Pour ce faire, nous nous intéresserons à l'apprentissage de caractéristiques multi échelles de l'image en utilisant des architectures particulières tirées d'autres domaines comme celui de la segmentation sémantique, ou que nous proposerons spécifiquement pour répondre à notre problématique (4).

Dans le chapitre suivant, nous nous placerons dans un contexte applicatif où l'on cherche à estimer non pas une carte de profondeur uniquement, mais également les incertitudes liées à cette prédiction. Nous présenterons une étude des méthodes et métriques d'évaluation existantes dans différents domaines de la littérature et proposerons également une métrique spécifique, plus adaptée au cas particulier de l'évaluation d'approches de *deep learning* (5).

Enfin dans le sixième chapitre, nous traiterons de la complétion de profondeur à partir de données LiDAR. En effet, nous nous intéresserons aux applications de conduite autonome où il est commun d'utiliser différents capteurs hétérogènes. Nous étudierons en particulier la possibilité de fusion de données issues de caméras et de LiDAR dans le but d'obtenir des estimations de profondeur précises et fiables. Nous proposerons alors une méthode pour prédire à la fois une carte de profondeur ainsi qu'une incertitude de prédiction associée (6).

Nous finirons enfin ce mémoire par une conclusion globale sur les méthodes d'estimation de profondeur monoculaire par *deep learning* et leur utilisation dans des contextes applicatifs comme celui de la conduite autonome (7).

Chapitre 2

État de l'art

Capteur	Technologie	Coût	Champ de vision	Portée	Précision	Environnement	Carte de profondeur
LiDAR	Laser	+++	$360° \times 25°$	km	mm	Intérieur, Extérieur	Parcimonieuse
Caméra TOF	Temps de vol	++	$60° \times 40°$	m	mm	Intérieur	Dense
Kinect	Lumière structurée	+	$57° \times 43°$	m	cm	Intérieur	Dense

TABLE 2.1 – Tableau récapitulatif des propriétés de chaque capteur de profondeur actif présenté dans cette section

Dans ce chapitre nous présentons les différents moyens utilisés dans l'état de l'art pour produire une carte de profondeur. Nous verrons d'abord les méthodes actives, utilisant directement un capteur d'acquisition spécifique, puis les méthodes passives, basées sur le traitement d'une ou plusieurs images. Nous appellerons prédictions ou estimations de profondeur les cartes de profondeur $\mathcal{D}$ obtenues par ces méthodes de traitement.

2.1 L'acquisition de profondeur

Tout d'abord, il est possible d'obtenir directement une information de profondeur au moment de l'acquisition des données en utilisant certains types de capteurs actifs. Il n'est donc pas rare que certains d'entre eux soient utilisés pour adresser des tâches de vision par ordinateur. Les principales propriétés des capteurs que nous présentons ici sont résumées dans la table 2.1.

2.1.1 La télémétrie laser

Le télémètre laser est un appareil permettant de mesurer la distance entre le dispositif de mesure et un point unique de l'espace. Un rayon laser est envoyé sur la cible et la distance est calculée en mesurant le temps entre l'émission du laser et la réflexion de la lumière sur la cible. C'est un capteur très précis qui a une très grande portée (jusqu'à plus de 20km pour certains appareils à applications militaires). Sa résolution est d'une dizaine de millimètres et ne décroit que peu avec la distance. Il est utilisable de jour comme de nuit, en intérieur comme en extérieur. Ses mesures peuvent cependant s'avérer erronées lorsque le faisceau traverse certaines surfaces entraînant la réfraction de la lumière (Hrabar, 2012). Ce capteur perd donc en fiabilité pour des mesures sur certains matériaux (verre, plastique, surfaces vitrées) ou dans certaines conditions météorologiques (brouillard).

Le télémètre laser ne permet de mesurer la distance qu'en un seul point de l'espace à la fois. En pratique, on utilise souvent plusieurs télémètres lasers motorisés pour effectuer des mesures de plusieurs points d'une scène. Ce montage est appelé *Light Detection and Ranging* (LiDAR). Le LiDAR scanne incrémentalement chaque point d'une scène à 360°en horizontal par environ 25°en vertical (la résolution verticale dépend du nombre de télémètres utilisés et elle varie selon les capteurs). Certains LiDAR peuvent enregistrer jusqu'à 300 000 points par seconde (Zhu et al., 2012; Disa, 2011; Vaze et Teng, 2007). Toutefois, étant donné que les points

FIGURE 2.1 – Exemple de scan LiDAR sur une scène d'extérieur (illustrée à gauche par une image en niveaux de gris). Les points 3D mesurés sont projetés en 2D pour obtenir une carte de profondeur de la scène (droite). Le résultat est parcimonieux : les pixels colorés correspondent aux points dont la distance a été mesurée par le LiDAR (le code couleur est fonction de la profondeur) tandis qu'on n'a pas d'information sur les pixels noirs

sont enregistrés séquentiellement, il est moins précis sur des objets en mouvement.

Par ailleurs, le principe de fonctionnement du LiDAR implique que le nuage de points 3D qu'il calcule produit une image de profondeur parcimonieuse en sortie. Cela signifie que, dans le repère 2D de l'image, tous les pixels de la carte de profondeur obtenue par un LiDAR n'ont pas forcément de mesure associée (voir figure 2.1). On parlera de pixels valides pour les pixels de $\mathcal{D}$ dont la profondeur est connue.

2.1.2 La mesure du temps de vol

D'autres technologies permettent d'obtenir directement les coordonnées 3D de plusieurs points de l'espace en même temps. Les caméras *Time of Flight* (TOF), par exemple, fonctionnent sur le principe du temps de vol et utilisent également la vitesse de la lumière pour calculer les distances. Ces caméras envoient des éclairs de lumière sur la scène et calculent le temps de retour du signal lumineux sur le capteur. Cette mesure est effectuée indépendamment pour chaque pixel de la caméra, permettant ainsi d'obtenir une image complète en 3D de l'objet mesuré. L'acquisition est plus rapide que sur les systèmes LiDAR (jusqu'à 160 images par seconde) avec une très grande précision (au dixième de millimètre) mais la portée et le champ de vision sont plus limités (moins d'une dizaine de mètres de portée pour environ 60°x 40°de champ de vision) et, bien qu'utilisable en extérieur, les performances se dégradent lorsque la scène présente une forte luminosité (Li, 2014; Hansard et al., 2012; Kolb et al., 2010).

2.1.3 La lumière structurée

Les capteurs dédiés que nous venons de passer en revue sont souvent plutôt onéreux. La caméra Kinect de Microsoft permet également de calculer la carte de profondeur d'une scène mais à moindre coût. Pour ce faire, elle utilise le principe de la lumière structurée : un émetteur infrarouge projette sur la scène un motif de points prédéfinis et connus à l'avance. L'analyse de la distorsion des points sur chacun des éléments de la scène à l'aide d'une caméra infrarouge permet alors de calculer la distance à l'émetteur et ainsi de construire une carte de profondeur

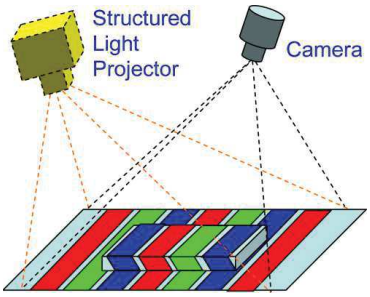


FIGURE 2.2 – Principe de la lumière structurée : un motif prédéfini et connu est projeté sur un objet. Une caméra filme la scène et l'analyse de la distorsion du motif permet de calculer les distances (Geng, 2011)

de la scène (voir figure 2.2).

La Kinect est relativement précise (de quelques millimètres à environ 4cm, en fonction de la distance) et, contrairement aux autres méthodes, elle permet d'obtenir une carte de profondeur dont les valeurs sont directement associées aux pixels de l'image $\mathcal{RGB}$ correspondante. En effet, cette association n'est pas triviale et doit se faire séparément dans les cas du LiDAR et de la caméra TOF qui ne calculent que les distances. Il est donc nécessaire de leur associer un second capteur, généralement une caméra classique, pour obtenir une paire $\mathcal{RGB}$-$\mathcal{D}$ de la scène. Il faut ensuite ajouter à cela une dernière étape où les images résultantes sont projetées et alignées dans un repère commun. La Kinect permet, elle, d'obtenir directement une paire $\mathcal{RGB}$-$\mathcal{D}$ alignée (correspondance pixel à pixel). Cependant elle a une portée très limitée (environ 4m), son champ de vision est assez réduit (57˚x 43˚) et elle ne peut pas fonctionner en extérieur puisque la lumière du soleil affecte l'efficacité de la caméra infrarouge (Andersen et al., 2012; Han et al., 2013; Zeng, 2012; Khoshelham et Elberink, 2012).

Bien que chacune de ces technologies fonctionne différemment, elles permettent toutes d'obtenir la distance entre l'objet visé et la caméra dès l'acquisition. C'est pourquoi on retrouve ces capteurs à la base de nombreuses méthodes de modélisation de scènes en 3D (Premebida et al., 2007; Ganapathi et al., 2010; Izadi et al., 2011). Dans ce cas, obtenir une carte de profondeur implique donc bien sûr d'avoir effectué l'acquisition directement avec le capteur adéquat.

Cependant le coût, le poids, l'encombrement et la consommation supplémentaires ajoutés par l'utilisation d'un capteur spécifique dédié pour le calcul de la profondeur peuvent être problématiques pour certaines applications (par exemple la conduite autonome). De plus il existe un grand nombre de données déjà acquises pour lesquelles il serait intéressant de recouvrer l'information 3D (médical, vidéo-surveillance, cartographie, etc.). Des lors se pose la question de l'estimation de profondeur de scènes dont les représentations n'incluent pas explicitement d'information de distance, comme par exemple à partir d'images $\mathcal{RGB}$. C'est justement pour profiter de la grande quantité de données déjà existantes et des avantages que peuvent offrir les caméras classiques qu'un autre pan de la recherche s'intéresse à des techniques de traitement

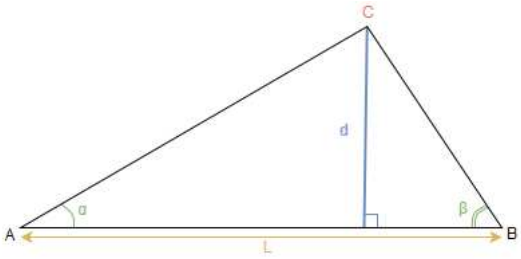


FIGURE 2.3 – Principe de triangulation. Pour connaitre la distance d du point C observé, il suffit de connaitre la distance L entre deux points A et B connus ainsi que les angles α et β qu'ils forment avec C. On calcule par trigonométrie que $L = \frac{d}{\tan\alpha} + \frac{d}{\tan\beta}$ et il est alors facile de déduire d

d'images dans le domaine de la vision par ordinateur.

2.2 L'estimation de profondeur

La perception de la profondeur à partir d'images se base sur différents types d'indices visuels qu'il est possible de classer en trois catégories : en premier lieu ceux qui requièrent une vision binoculaire, ensuite ceux qui dépendent du mouvement et enfin, ceux perçus avec un seul œil. On peut alors discerner deux grandes familles de méthodes :

1. Les méthodes multi-oculaires (nécessitant deux images ou plus) qui utilisent des indices binoculaires ou des indices liés au mouvement
2. Les méthodes monoculaires ne nécessitant qu'une seule image et qui se basent sur des indices monoculaires

2.2.1 Les méthodes multi images

Dans le cas des méthodes multi-oculaires, on peut distinguer plusieurs cas. En effet, les indices visuels nécessaires à l'estimation de profondeur peuvent être obtenus soit à partir d'images issues de plusieurs caméras fixes et fournissant plusieurs angles de vues de la scène ; soit en utilisant une seule caméra mobile ; soit encore avec une caméra fixe filmant des objets en déplacement.

Avec plusieurs caméras

Les méthodes de vision multi-oculaires s'inspirent généralement du modèle biologique. Nos deux yeux étant légèrement espacés, nous voyons toujours deux images de deux points de vue différents de la scène. C'est ce qu'on appelle la disparité binoculaire. De plus, lorsque l'on regarde un objet, nos muscles oculaires se contractent pour faire converger nos yeux dans la bonne direction. C'est ce qu'on appelle la convergence binoculaire. Elle correspond à l'angle formé par les yeux fixant un point de l'espace et varie donc selon la distance de ce point. Étant

donné l'espacement et les angles formés par nos yeux, il est possible de retrouver la distance du point observé grâce au principe de triangulation (illustré en figure 2.3). Notre cerveau utilise ces indices binoculaires pour fusionner les deux images en deux dimensions qu'il reçoit et ainsi former une troisième image, en trois dimensions, de la scène observée.

C'est ce même principe qui est exploité en stéréovision : on utilise deux angles de vues différents (ou plus) d'une même scène pour pouvoir reconstruire cette dernière en trois dimensions. Si on considère M le point de coordonnées homogènes $(X, Y, Z, 1)$ dans le repère monde (l'espace 3D) et m, son image dans le plan (l'image 2D), de coordonnées $(x, y, 1)$, on peut calculer la projection de M dans le plan image par la relation :

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = kP \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.1}$$

où k est un scalaire représentant un facteur d'échelle et P est la matrice de projection associée à la caméra. P regroupe deux types de paramètres : les paramètres intrinsèques de la caméra, qui sont ses caractéristiques internes, indépendantes de sa position (distance focale et coordonnées du centre optique dans le plan image) et ses paramètres extrinsèques qui représentent la position et l'orientation de la caméra par rapport au repère monde (rotation et translation).

P se décompose alors de la manière suivante :

$$P = \begin{pmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} & R_{3\times3} & & t_x \\ & & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

avec f_x et f_y la distance focale dans le repère image, exprimée en nombre de pixels par axe, x_0 et y_0 les coordonnées du centre optique dans le repère image, R la matrice de rotation de la caméra et t_x, t_y, t_z les composantes du vecteur de translation T.

Dans le cas stéréoscopique les images proviennent de différentes caméras calibrées, la résolution de ces équations permet alors de retrouver les coordonnées 3D d'un point de manière déterministe dès lors qu'il est visible et reconnu dans les deux vues de la scène. Ainsi, le problème de stéréovision peut se résumer à un problème d'appariement de points et il a été très largement étudié dans la littérature (Seitz et al., 2006; Scharstein et Szeliski, 2002; Hartley et Zisserman, 2003; Scharstein et al., 2001).

La plus grande avancée de la dernière décennie dans ce domaine est l'accélération des algorithmes existants permettant de travailler en temps réel (Banz et al., 2010; Domanski

FIGURE 2.4 – Un exemple d'estimation de profondeur utilisant la stéréovision, par Hadfield et Bowden (2015). L'image d'entrée à gauche, leur estimation de profondeur au milieu et la vérité terrain à droite.

et al., 2015; Ramos-Diaz et al., 2015). Les performances des algorithmes de stéréovision sont souvent mesurées en pourcentage d'erreur sur les disparités calculées. Les méthodes récentes, par exemple Hadfield et Bowden (2015) pour n'en citer qu'une, présentent de très bonnes performances quantitatives (environ 10% d'erreur relative moyenne) ainsi que qualitatives, avec des cartes de profondeur détaillées et précises (voir figure 2.4).

Cependant, bien que la stéréovision permette d'obtenir de très bons résultats, elle pose plusieurs problèmes majeurs. D'abord, il est nécessaire de calibrer les caméras pour pouvoir effectuer une reconstruction 3D déterministe de la scène. Ensuite, le problème d'appariement des points des différentes vues est d'autant plus complexe que les zones de l'images sont homogènes : il est très difficile de mettre en correspondance des zones de l'image peu texturées. De plus les prédictions ne sont pas parfaitement denses : il est impossible de reconstruire les zones occultées, n'apparaissant que sur l'une des images (voir la vérité terrain figure 2.4 qui comporte quelques valeurs manquantes de profondeur). Enfin, comme nous l'avons vu, il est nécessaire d'utiliser plusieurs caméras pour la stéréovision ce qui peut être contraignant suivant l'application visée. Il existe cependant des méthodes multi-oculaires ne nécessitant qu'une caméra.

Avec une seule caméra

Il existe par ailleurs d'autres méthodes basées sur plusieurs images mais ne nécessitant qu'une unique caméra. Les techniques de *depth from focus* ou *depth from defocus*, par exemple, se basent sur des indices de profondeur relatifs à la distance focale de la caméra (Xiong et Shafer, 1993). Grâce à deux images de la scène prises depuis le même angle de vue mais en changeant les paramètres de la caméra (typiquement le réglage focal) on peut trouver les objets les plus éloignés du plan focal en trouvant les zones les plus floues de l'image.

L'information temporelle peut également être prise en compte : en effet les différentes frames d'une séquence peuvent fournir différentes vues de la même scène et le problème devient plus ou moins semblable au problème de stéréovision où un appariement des points est nécessaire (Zhang et al., 2009). De plus, des indices de profondeur relatifs au déplacement peuvent être utilisé : par exemple les objets proches de la caméra ont l'air de bouger plus

rapidement que ceux en fond (Kowdle et al., 2012). Ce phénomène est appelé le parallaxe. Par exemple, Wedel et al. (2006) déterminent la structure de la scène à partir de son mouvement : ils étudient les changements d'échelles des régions d'une image en suivant l'évolution de ces régions dans le temps et peuvent ainsi estimer la profondeur de la scène.

Cet aspect temporel permet de s'affranchir de la nécessité d'avoir plusieurs caméras puisque les objets en déplacement, voire la caméra elle même, fournissent des points de vues différents de la scène. Cependant l'utilisation de vidéos n'est pas sans contrainte : le principe de reconstruction à partir de différentes vues présentent les mêmes limites que nous avons vues dans le cas de la stéréovision, qui peuvent même être accentuées par la dynamique de la scène ou de la caméra elle même. Il faut en effet être capable de détecter et reconnaître les mêmes objets d'une image à l'autre pour l'étape d'appariement sans savoir à l'avance s'ils seront sortis du champ de vision entre les temps t et $t + 1$. Les régions peu texturées demeurent difficiles voire impossible à reconstruire.

2.2.2 Les méthodes mono image

Bien que l'estimation de profondeur à partir d'une seule image soit un problème fondamentalement mal posé, et intrinsèquement ambigu (dans ce cas il n'est pas possible d'appliquer le principe de triangulation et il existe alors une infinité de solutions 3D pour une image donnée), les enjeux de l'estimation monoculaire sont importants. Par ailleurs, il existe des indices de profondeur monoculaires que l'on peut extraire à partir d'une seule image. Ces indices de perception sont multiples (Kardas, 2005) :

- la perspective : les lignes parallèles semblent converger vers un même point à mesure que la distance augmente
- l'élévation : plus un objet apparait haut dans le champ de vision, et proche de la ligne d'horizon, plus il est éloigné de l'observateur
- le flou : en extérieur pour de grands distances, les objets lointains paraissent plus flous, avec des contours moins marqués, c'est la perspective atmosphérique. La mise au point de la caméra produit également des images avec des régions plus ou moins floues, c'est le defocus.
- les ombrages : les ombrages donnent des indices sur le niveau de relief des objets et permettent de distinguer un enfoncement d'un monticule
- les textures : le niveau de détails que l'on voit sur un objet est corrélé avec sa distance, plus l'objet est proche et plus les textures sont riches
- la taille relative : les objets sont perçus plus grands lorsqu'il sont proches et de plus en plus petits à mesure qu'ils s'éloignent
- les occultations : lorsqu'un objet en masque partiellement un second alors on peut en déduire leur distance relative, l'objet occulté étant le plus éloigné

FIGURE 2.5 – En rouge un exemple du *layout* d'une pièce, issu de la base de données LSUN (Yu et al., 2015)

Parmi ces indices, certains sont uniquement géométriques (perspective, élévation, flou, ombrages) tandis que d'autres se basent sur la connaissance a priori des objets qui composent la scène (textures, taille relative, occultations). Les méthodes d'estimation monoculaire que l'on trouve dans l'état de l'art peuvent elles aussi se classer en deux catégories : les modèles basés sur des a priori géométriques de la scène et ceux directement basés sur un apprentissage des caractéristiques monoculaires (géométriques ou non).

A priori géométriques

On trouve dans la littérature plusieurs travaux qui exploitent des indices géométriques tels que les points de fuite de l'image, la ligne d'horizon, le plan du sol, etc. ainsi que des propriétés connues a priori pour estimer la structure 3D d'une scène. Un terme récurrent dans la communauté est le *shape-from-X* où X peut désigner les ombrages (*shading*) (Barron et Malik, 2012), la texture (*texture*) (Payet et Todorovic, 2011) ou encore les jonctions de lignes (*line junctions*) (Yu et al., 2008). Yu et al. (2008) par exemple regroupent des ensembles de contours calculés à partir de l'image RGB selon des propriétés géométriques comme leur parallélisme/orthogonalité ou la perspective pour former des quadrilatères correspondants à des plans homogènes. Ils déterminent finalement une structure 3D où ces plans sont ordonnés selon leur distance aux points de fuite.

Pour des images d'intérieur, une hypothèse classiquement utilisée est de considérer la pièce comme un cube vu de l'intérieur. Ainsi on s'attend à observer différents plans structurant la scène, correspondants aux faces du modèle cubique. Fouhey et al. (2015) se basent uniquement sur cette hypothèse qu'ils exploitent à l'aide d'un estimateur de points de fuite et d'un détecteur de textures. L'ensemble des lignes démarquant la séparation entre ces plans, à savoir les différents murs, le sol et le plafond est appelé le *layout* de la pièce (illustré figure 2.5).

Certains travaux se concentrent uniquement sur l'estimation du *layout* comme Lee et al. (2010) qui proposent pour ce faire de prendre en considération les contraintes volumiques des objets (interactions 3D entre les objets présents sur l'image et le *layout* estimé). Schwing et Urtasun (2012) utilisent également des contraintes géométriques telles que l'orientation des contours ou les lignes qui convergent vers les points de fuite pour définir chacune des faces du *layout*. Hedau et al. (2010) calculent un modèle probabiliste qui prend en compte la position

Figure 2.6 – Un exemple d'estimation de profondeur monoculaire à partir d'indices de flou résultant du defocus de la caméra (Zhuo et Sim, 2009)

géométrique relative des objets et du *layout* de la pièce pour former un détecteur d'objets 3D, qui propose un volume cubique englobant l'objet recherché.

Des a priori géométriques sont aussi utilisés pour des scènes d'extérieur, non structurées. L'hypothèse de Manhattan est fréquemment utilisée. Elle suppose que le monde est composé de grandes surfaces planes avec seulement 3 directions orthogonales dominantes (Gupta et al., 2010; Ramalingam et Brand, 2013; Concha et al., 2010). Par exemple, le but de Hoiem et al. (2005) est de retrouver la structure géométrique 3D inhérente à une scène d'extérieur en apprenant des modèles d'apparences de surfaces à différentes orientations. Ils se servent fortement de l'hypothèse que la scène est constituée de plans horizontaux comme le sol et de murs verticaux (les façades des différents bâtiments), sans compter le ciel.

Bien qu'utiles pour reconstruire une structure 3D de la scène ou pour hiérarchiser les différents plans sur une image, ces approches géométriques ne donnent pas directement l'information de profondeur en chaque point et ne sont souvent qu'une approximation grossière de la structure géométrique de la scène. D'autres indices monoculaires comme l'utilisation du flou naturellement présent dans les images n'ont pas ce défaut.

Wang et al. (2012) se basent sur le degré de flou dans différentes régions de l'image pour estimer la profondeur. Ils distinguent deux sources à l'origine des zones floues : le defocus provenant de la caméra, qu'ils modélisent comme un flou Gaussien et la réflectance atmosphérique, qu'ils modélisent suivant un modèle de l'état de l'art (He et al., 2011). Aslantas (2007), eux, n'utilisent que le defocus de la caméra pour calculer la profondeur à laquelle se trouve les régions floues grâce a un filtre qui leur permet d'affiner l'image. De même, Zhuo et Sim (2009) estiment d'abord la quantité de flou aux contours de l'image (le passage d'un plan à un autre induit du flou). L'image d'origine est ensuite elle même floutée et ils montrent que le ratio entre les deux images ne dépend que du defocus de la caméra, ce qui leur permet d'estimer une carte de profondeur à niveaux (voir figure 2.6).

Cependant les indices provenant des zones de flous dans une image sont peu fiables à eux seuls. En effet, un objet flou dans une image peut se trouver des deux cotés du plan focal de la caméra et n'en est pas forcément plus éloigné. De même, il se peut que l'on trouve des zones nettes à l'extérieur du plan focal comme des zones floues à l'intérieur. Ces ambiguïtés ne

permettent pas d'avoir des résultats très performants et se traduisent également par des cartes de profondeurs souvent irrégulières ou présentant des artefacts.

Apprentissage statistique

Enfin les méthodes d'apprentissage automatique permettent, par l'analyse statistique de données $\mathcal{RGB}$-$\mathcal{D}$, d'apprendre les caractéristiques monoculaires présentes dans les images et leur lien avec les cartes de profondeur associées.

Dans cette littérature, on trouve une grande variété d'approches permettant l'estimation de profondeur monoculaire. Certaines méthodes visent à comparer directement les données d'entrée avec une référence. Lopez et al. (2014) choisissent par exemple d'exploiter le contenu de l'image mais aussi le savoir de l'utilisateur qui interagit avec le système. L'utilisateur doit désigner au moins un point à l'arrière plan et un point au premier plan de la scène. Supposant que les régions avec des gradients faibles ont la même profondeur, l'algorithme permet ensuite de déterminer, par propagation, une carte de profondeur. Karsch et al. (2014) ou encore Konrad et al. (2012) estiment, eux, la structure 3D de la scène en comparant l'image donnée en entrée à un dictionnaire d'images où l'information de profondeur est connue a priori. Ils cherchent ensuite les images dont le contenu est le plus similaire puis fusionnent et alignent leurs cartes de profondeur avec l'image d'entrée.

Ces approches nécessitent cependant l'intervention d'un utilisateur extérieur ou l'existence d'une base de donnée exhaustive pour représenter tous les types de scènes qu'il est possible de rencontrer et elles ne sont donc utilisables que dans des conditions particulières. D'un autre côté, la disponibilité croissante de jeux de données de plus en plus importants permettent de développer des approches statistiques fiables, basées sur l'apprentissage supervisé.

Jusqu'à récemment une des approches les plus étudiée est celle utilisant les champs aléatoire de Markov (*Markov Random Fields* ou MRF). Il s'agit d'un graphe non orienté permettant de modéliser des interactions en 2D, par exemple dans une image. Le MRF est un modèle génératif. Il a souvent été utilisé comme outil en vision par ordinateur pour des applications comme la reconstruction 3D (Saxena et al., 2007), la segmentation d'image (Wang et Wang, 2004; Yonetani et al., 2012), ou encore la labellisation d'objets (Lai et al., 2012) et depuis quelques années aussi pour l'estimation de profondeur dans une image (Saxena et al., 2005, 2007, 2008).

En effet, Saxena et al. (2005) modélisent le problème d'estimation de profondeur monoculaire à l'aide d'un MRF multi-échelles et utilisent des caractéristiques locales et globales de l'image pour l'apprentissage (Saxena et al., 2007). Ils se servent également de la segmentation d'image par superpixels pour créer des petits patchs de structure homogène (Saxena et al., 2008). Ils utilisent un MRF pour inférer un jeu de paramètres pour chaque superpixel, capturant ainsi sa localisation et son orientation dans l'espace. Les données qu'ils utilisent pour l'évaluation ont été collectées avec un LiDAR, spécialement pour leurs recherches, et les performances qu'ils

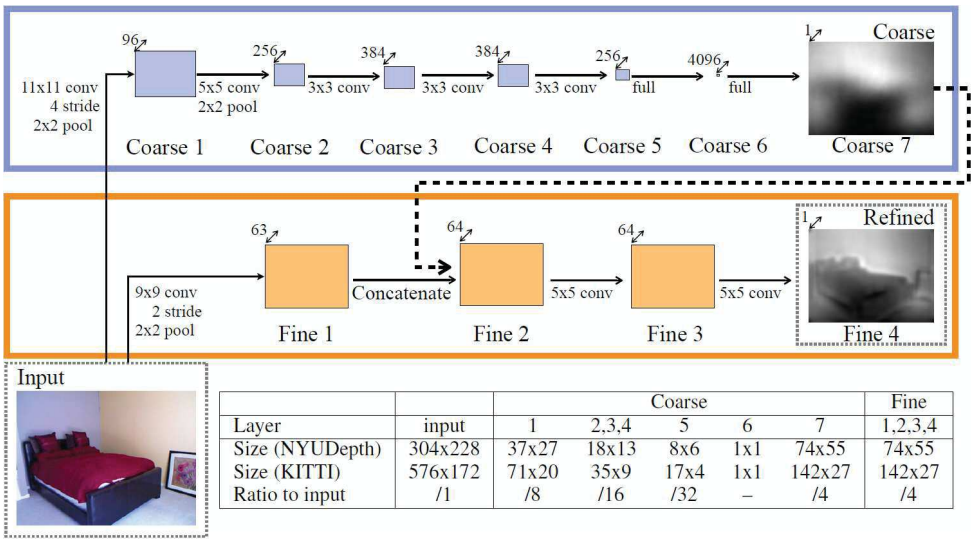
obtiennent dans leur article le plus récent sont inférieures à 40% d'erreur relative moyenne (Saxena et al., 2008).

Cependant l'avènement de méthodes d'apprentissage supervisé basées sur des gros volumes de données, telle que le *deep learning*, a permis un incroyable essor en vision par ordinateur. Formalisés pour la première fois dans les années 40 par les travaux de McCulloch et Pitts (1943), les réseaux de neurones ne connaissent un véritable succès que dans les années 90 suite aux travaux de LeCun et al. (1989) qui proposent l'algorithme de rétropropagation pour entrainer le réseau en utilisant le gradient de l'erreur. Plus récemment, les travaux de Krizhevsky et al. (2012a) en reconnaissance d'images popularisent l'utilisation de réseaux de neurones convolutionnels (*Convolutional Neural Network* ou CNN) dont le but est d'entrainer des filtres spécifiques pour répondre à une tâche donnée (ici la classification d'images). Les CNN ont depuis prouvé leur efficacité sur nombre d'applications de vision par ordinateur (Razavian et al., 2014).

L'estimation de profondeur en particulier bénéficie des avantages qu'offrent le deep learning puisque les réseaux convolutionnels permettent de régresser directement depuis l'entrée $\mathcal{RGB}$ vers la sortie $\mathcal{D}$ voulue en apprenant des filtres capables de détecter dans l'image des indices optimaux pour la prédiction ; à condition d'avoir suffisamment de données d'entrainement. Ce type d'algorithme permet d'apprendre des espaces de représentation très performants et il constitue désormais l'approche dominante pour l'estimation de profondeur monoculaire. En effet la disponibilité d'une grande quantité de paires $\mathcal{RGB}$-$\mathcal{D}$ collectées avec des capteurs de profondeur dédiés permet aux CNN d'afficher aujourd'hui des résultats impressionnants dans ce domaine quand bien même le problème est complexe et mal posé. Mohan (2014) et Eigen et al. (2014) ont été parmi les premiers à proposer des algorithmes d'estimation de profondeur utilisant des CNN profonds. De nombreuses publications ont depuis étendu ce type d'approches dans différentes directions.

Nous pouvons grossièrement classer en deux catégories les méthodes de deep learning adressant l'estimation de profondeur monoculaire : les méthodes globales qui estiment une carte de profondeur à partir de toute l'image d'entrée et les méthodes locales qui travaillent localement sur des patchs de l'image. Par exemple, Eigen et al. (2014); Eigen et Fergus (2015); Laina et al. (2016); Cao et al. (2017) sont des méthodes globales qui traitent l'intégralité de l'image par un ou plusieurs CNN en feed-forward. D'un autre côté, des approches comme Liu et al. (2016); Li et al. (2015); Roy et Todorovic (2016) sont basées sur des prédictions locales. Bien que certaines d'entre elles renforcent ensuite la cohérence globale entre les patchs (Liu et al., 2016; Li et al., 2015), elles restent fondamentalement locales. Certaines approches hybrides ont également été spécifiquement conçues pour bénéficier des avantages de chacune des catégories en fusionnant les prédictions locales et globales (Wang et al., 2015b; Chakrabarti et al., 2016).

Le travail séminal de Eigen et al. (2014) a introduit une approche efficace basée sur un CNN



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

FIGURE 2.7 – Schéma de l'architecture du réseau de Eigen et al. (2014)

profond permettant d'estimer la profondeur d'une image RGB monoculaire de manière directe en *feed-forward*. L'architecture qu'ils proposent est constituée de deux réseaux en cascade opérants à différentes échelles. Elle est illustrée en figure 2.7. Le premier réseau, en bleu sur la figure, vise à prédire une première carte de profondeur grossière de la scène. En effet, il comporte des couches entièrement connectées (non convolutionnelles) qui relient l'ensemble des neurones entre eux. Ainsi, les spécificités locales sont perdues mais une structure globale peut être apprise. Le second réseau, en orange, réutilise cette première sortie grossière ainsi que l'image RGB d'entrée et affine localement la prédiction en appliquant des convolutions successives avec des petits noyaux.

Ils introduisent également une fonction de coût invariante au changement d'échelle permettant, dans une certaine mesure, de mesurer les relations de profondeur entre les pixels plutôt que de calculer l'erreur absolue commise. Les performances obtenues par ce réseau ont instauré un nouvel état de l'art avec environ 20% d'erreur relative moyenne pour des bases de données d'images d'intérieur et d'extérieur. Cette approche a constitué la première contribution majeure pour l'estimation de profondeur monoculaire par *deep learning*.

Liu et al. (2016, 2015) proposent également d'utiliser le *deep learning* pour l'estimation de profondeur monoculaire mais avec une approche plus locale. En effet, ils choisissent d'utiliser conjointement les avantages d'un CNN avec ceux d'un *Conditional Random Field* (CRF). Comme les MRF, les CRF sont des graphes non orientés. Cependant contrairement au MRF, ce sont des modèles discriminatifs et ils permettent de considérer les interactions entre des variables voisines. Liu et al. (2016) proposent alors un framework unifié dans lequel ils apprennent à un réseau à produire à la fois les sorties globales et les interactions locales utiles à l'optimisation d'un CRF continu. Ils sur-segmentent d'abord l'image et travaillent au niveau des superpixels

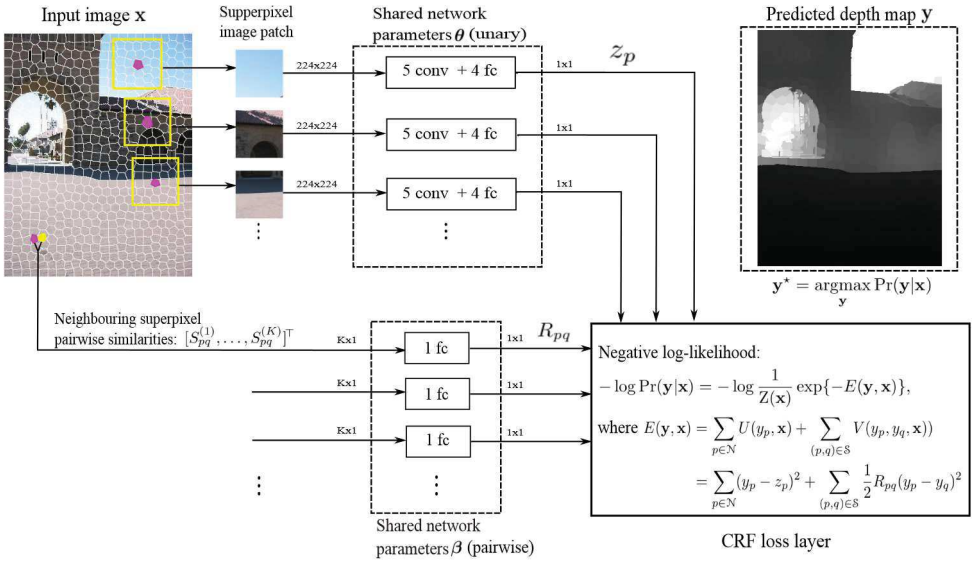


FIGURE 2.8 – Schéma du framework DCNF proposé par Liu et al. (2016)

pour modéliser les relations entre voisins. Leur algorithme est appelé *Deep Convolutional Neural Fields* (DCNF), il est illustré en figure 2.8.

D'autres types d'approches locales ont été développées. On trouve des approches similaires, comme Li et al. (2015) qui combinent également un CNN profond avec un CRF, respectivement aux niveaux de super-pixels et des pixels, pour obtenir des cartes de profondeur plus fines. D'abord, ils travaillent au niveau super-pixels et font une régression multi-échelles avec un CNN profond pour extraire des caractéristiques utiles à partir de patchs de l'image. Ils estiment donc une carte de profondeur des super-pixels, qu'ils raffinent par la suite au niveau pixellique en utilisant un modèle de CRF hiérarchique pour renforcer la cohérence locale.

On trouve également des approches différentes, comme Roy et Todorovic (2016) qui eux combinent des CNN avec des forêts d'arbres décisionnels. La forêt d'arbre décisionnel prend des patchs locaux de l'image et, pour chaque nœud, l'échantillon est filtré par un petit CNN associé à ce nœud. Puisque le patch va parcourir la forêt, il sera progressivement filtré par plusieurs réseaux. Cette cascade de petits CNN leur permet d'entrainer des CNN profonds avec des coûts computationnels plus faibles. Les prédictions obtenues à la sortie de la forêt d'arbres décisionnels sont donc locales et il fusionnent ensuite les résultats pour obtenir la carte de profondeur finale.

Des méthodes hybrides ont aussi été considérées. Wang et al. (2015b) utilisent par exemples 3 réseaux de neurones pour guider leur apprentissage : un premier réseau produit une sortie globale ainsi qu'une estimation du *layout* de la pièce ; un second réseau produit une sortie locale ainsi qu'une carte des contours, classés selon leurs orientations, et le dernier fusionne toutes ces prédictions avec l'image $\mathcal{RGB}$ et les points de fuite de l'image pour produire la sortie

finale. D'un autre côté, Chakrabarti et al. (2016) entrainent d'abord un réseau à produire une représentation intermédiaire, caractérisant la structure locale de la profondeur en prédisant les dérivées de différents ordres, orientations et échelles d'un patch de l'image originale. Pour chaque dérivée, le réseau produit des distributions de probabilité qui sont ensuite combinées en utilisant une procédure globale pour estimer la carte de profondeur finale.

Enfin il faut aussi mentionner certaines méthodes qui traitent le sujet différemment comme Cao et al. (2017) qui choisissent de formuler le problème d'estimation de profondeur comme un problème de classification pour chaque pixel. Ils suggèrent que cette manière d'apprendre est plus facile et permet d'obtenir de meilleurs résultats. Cela leur permet en plus d'appliquer un CRF a posteriori pour renforcer la cohérence locale des interactions sur les cartes de probabilités qu'ils obtiennent, ce qui améliore encore un peu leurs résultats.

Mais l'estimation de profondeur monoculaire n'a pas été uniquement appréhendée comme un objectif isolé. En effet, on trouve plusieurs approches multi-tâches dont le but est d'optimiser conjointement l'estimation de profondeur avec une autre tâche complémentaire. Par exemple comme nous l'avons vu, de nombreux indices monoculaires reposent sur la connaissance a priori des objets de la scène. Il y a donc un aspect sémantique non négligeable dans la prédiction des distances. L'impact positif de la segmentation sémantique sur les performances de l'estimation de profondeur (et vice-versa) a été étudié (Ladický et al., 2014) et il est également utilisé dans de nombreux travaux.

Par exemple, Liu et al. (2010) effectuent d'abord une segmentation sémantique de l'image pour aider l'estimation de profondeur. En effet, les contraintes géométriques peuvent être renforcées : on sait par exemple que le label "ciel" doit se situer très loin à l'horizon et que le label "sol" se trouve à l'horizontal. L'estimateur de profondeur est appris séparément pour chaque classe et il est inclus dans un MRF qui prend en compte la sémantique pour la reconstruction du modèle 3D.

Plus récemment, des approches *deep learning* ont également émergé. Wang et al. (2015a) qui formulent par exemple le problème joint de l'estimation de profondeur et de la segmentation sémantique pour bénéficier de leur complémentarité. Ils proposent un framework unifié où un CNN est entrainé à prédire conjointement et globalement la profondeur et la segmentation sémantique. Ensuite, ils affinent leurs prédictions en travaillant sur des segments locaux de l'image. Pour finir, ils renforcent leurs sorties globales et locales en utilisant un CRF hiérarchique pour produire leurs cartes de profondeur et de segmentation sémantique finales.

Les travaux de Eigen et al. (2014) ont également été poursuivis en ajoutant une troisième échelle à leur architecture et en combinant l'apprentissage simultané de plusieurs tâches complémentaires. Ainsi, ils prédisent non seulement la profondeur mais également les normales aux surfaces ainsi que les labels sémantiques de l'image RGB d'entrée (Eigen et Fergus, 2015). Ils proposent aussi de rajouter un terme à leur fonction de coût pour aligner les gradients

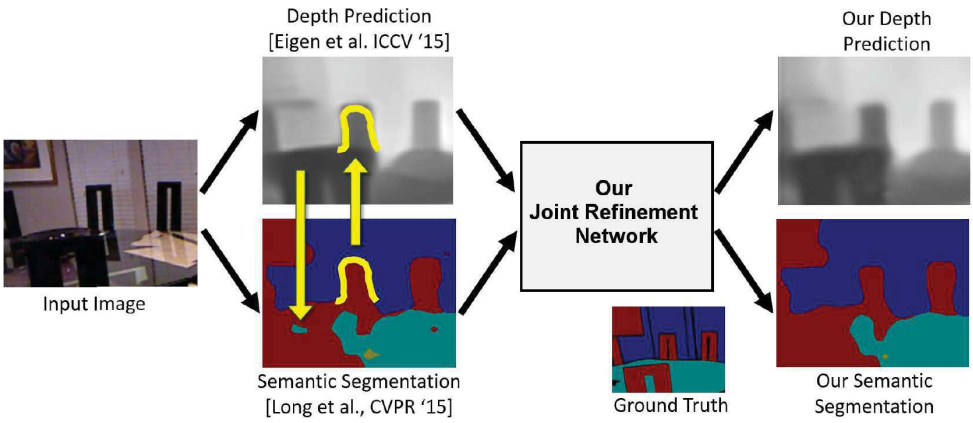


FIGURE 2.9 – Schéma de l'algortithme de fusion proposé par Jafari et al. (2017) pour améliorer conjointement l'estimation de profondeur et la segmentation sémantique d'une image

de la prédiction avec ceux de la vérité terrain de profondeur. Ils montrent que chacune des tâches bénéficie des autres puisqu'ils se positionnent au-dessus de l'état de l'art de la segmentation sémantique sur plusieurs bases de données et améliorent également leurs performances précédentes d'estimation de profondeur à environ 16% d'erreur relative moyenne.

Enfin certains travaux se portent sur l'amélioration a posteriori de la sémantique et de la profondeur. Jafari et al. (2017) proposent à cet égard un réseau qui fusionne la segmentation sémantique et la cartes de profondeur d'une scène pour affiner chacune des deux (voir figure 2.9). Ils n'apprennent pas comment produire ces sorties qu'ils considèrent a priori. En revanche, ils apprennent comment les fusionner par un CNN qui les raffine conjointement en faisant une analyse multi-échelle, leur permettant d'utiliser leurs affinités respectives pour les améliorer. Ils proposent également un protocole expérimental pour mesurer l'influence de la modalité croisée sur ces tâches et montrent qu'elles sont en effet corrélées et complémentaires.

Malgré le gain et leur interdépendance notable, l'apprentissage joint de différentes modalités comme la sémantique et la profondeur dans le cadre du *deep learning* implique d'avoir accès à une grande quantité d'images doublement annotées pour ces tâches. Cela reste relativement couteux, avec des apprentissages souvent plus lourds et nécessite la disponibilité de bases de données spécifiques.

Nous reviendrons par la suite plus en détails sur certaines méthodes de l'état de l'art et en présenterons également d'autres, issues de la littérature de l'estimation de profondeur ou spécifiques à d'autres domaines relatifs aux chapitres étudiés.

2.3 Les critères d'évaluation

Nous présentons ici plusieurs métriques d'évaluation parmi les plus utilisées dans la littérature de l'estimation de profondeur. Ce sont ces mesures que nous utiliserons également tout le long de ce mémoire pour évaluer nos algorithmes et nous comparer à l'état de l'art puisqu'elles permettent de mesurer la qualité globale des prédictions. $\hat{y}_i$ et y_i dénotent respectivement la prédiction et la vérité terrain de profondeur au pixel i. N représente le nombre total de pixels.

1. La *Mean Absolute Error* (mae) : $\frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$

   La *Mean Absolute Error* mesure l'amplitude moyenne des erreurs commises, en valeur absolue. Elle a l'avantage d'être exprimée dans la même unité que la variable d'intérêt. Ces deux propriétés la rendent très facilement interprétable. Elle est orientée négativement et est minorée par 0.

2. La *Root Mean Squared Error* (rms ou rmse) : $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$

   La *Root Mean Squared Error* mesure l'ordre de grandeur moyen des erreurs et, comme la MAE, elle est exprimée dans la même unité que la variable d'intérêt. Puisqu'elle est quadratique et que les erreurs sont mises au carré avant d'être moyennées, elle pénalise plus fortement les grandes erreurs que les autres métriques. Elle est orientée négativement et minorée par 0.

3. La *Mean Relative Error* (rel) : $\frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|/y_i$

   La *Mean Relative Error*, aussi connue sous le nom de *Mean Absolute Percentage Error* (MAPE), exprime le pourcentage moyens des erreurs absolues commises. Elle est orientée négativement et est minorée par 0. Cependant, elle n'a pas de borne supérieure (lorsque $y > 2\hat{y}$, la valeur excède 1) et elle ne peut pas se calculer pour une vérité terrain nulle (division par 0).

4. La *Mean* log_{10} *Error* (log) : $\frac{1}{N}\sum_{i=1}^{N}|log_{10}\,\hat{y}_i - log_{10}\,y_i|$

   La *Mean* log_{10} *Error* est la version logarithmique de la *Mean Absolute Error*. Elle permet de mesurer le ratio moyen absolu entre les prédictions et les vérités terrain. L'échelle logarithmique permet de pénaliser moins fortement les erreurs lorsque les distances augmentent. C'est pertinent pour la tâche d'estimation de profondeur où l'on peut considérer qu'il est moins grave de se tromper à des distances très éloignées. De plus il y a moins d'informations visuelles à mesure que la distance augmente. Elle est orientée négativement et minorée par 0.

5. *Threshold* : % de y_i tels que $max\left(\frac{\hat{y}_i}{y_i}, \frac{y_i}{\hat{y}_i}\right) = \delta < thr$. Dans nos évaluations, $\forall k \in \{1,2,3\}$, δ_k représente la métrique de *threshold* $\delta < 1.25^k$.

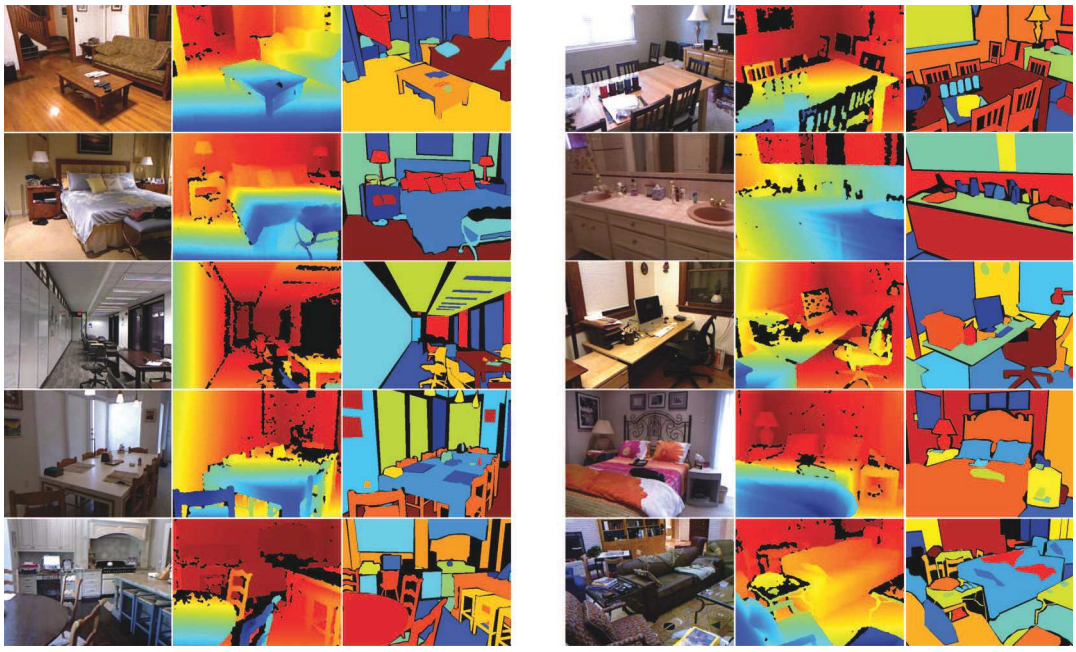


FIGURE 2.10 – Exemples d'image RGB, de carte de profondeur brutes et de labels sémantiques issus du dataset NYU Depth v2.

Les mesures de *Threshold* permettent de calculer le pourcentage des ratios d'erreurs inférieurs à un seuil. Ainsi, plus la valeur k est élevée, plus on relâche la contrainte et donc plus élevé sera le pourcentage calculé. Par exemple, pour $k = 1$, on compte le nombre de prédictions différentes de la vérité terrain d'au plus 25%, pour $k = 3$ on autorise jusqu'à 95% d'écart. Ces métriques sont orientées positivement et bornées entre 0% et 100%.

2.4 Les bases de données

Nous présentons ici les principales bases de données que nous mentionnerons dans la suite de ce manuscrit. Ces jeux de données proposent tous des paires d'images $\mathcal{RGB}$-$\mathcal{D}$ et sont par conséquent parmi les plus utilisés dans la littérature de l'estimation de profondeur par *deep learning*.

2.4.1 NYU Depth v2

Commençons par le populaire et non moins difficile benchmark NYU Depth v2 (Nathan Silberman et Fergus, 2012). Ce jeu de données est composé de 464 séquences vidéo de paires d'images $\mathcal{RGB}$-$\mathcal{D}$, acquises avec une caméra Kinect de Microsoft. Il comprend une grande variété de scènes d'intérieur dont des bureaux, chambres, bibliothèques, salons, cuisines, classes, etc. On y trouve des paires d'images $\mathcal{RGB}$ et $\mathcal{D}$ alignées dans un espace 2D commun. La base de données est divisée en 249 séquences pour l'apprentissage et 215 séquences pour le test,

parmi lesquelles seulement 795 images de train et 654 images de test sont densément annotées avec des valeur de profondeur en tout point ainsi que des annotations de labels sémantiques pour plus de 1000 classes d'objets, voir figure 2.10 . La résolution spatiale des images est de 640x480 pixels et la plage des profondeurs enregistrées par la Kinect va de 0 à 10 mètres.

Cette base de données propose en tout plus de 400K paires d'images $\mathcal{RGB}$ et $\mathcal{D}$ pour lesquelles il n'y a pas d'annotation sémantique et où les cartes de profondeur ne sont pas denses. En effet comme nous l'avons vu, la Kinect permet d'obtenir des cartes de profondeur en utilisant le principe de la lumière structurée. Cependant, étant basé sur une projection lumineuse, le capteur de profondeur de la Kinect est notamment sensible à la lumière et à la réfraction des matériaux. C'est pourquoi les cartes de profondeur brutes obtenues sont souvent incomplètes, avec des données manquantes sur les régions surexposées ou les surfaces vitrées par exemple. La kit de développement de NYU Depth inclut une routine de colorisation pour densifier les cartes de profondeur. Initialement proposée par Levin et al. (2004), elle permet d'étendre des annotations de couleur sur des images en niveaux de gris. C'est cette procédure qui a été utilisée pour générer les vérités terrains des 654 images de test officielles.

Or dans la littérature, ce sont toujours ces 654 images officielles qui sont utilisées pour évaluer les différents algorithmes proposés. Les cartes de profondeurs denses de ces images sont imparfaites et forcément biaisés par la routine de colorisation choisie. Dans l'optique de battre les performances de l'état-de-l'art et en suivant ce benchmark, il est donc favorable d'utiliser une méthode d'apprentissage qui apprendrait cette façon de coloriser plutôt que de laisser l'algorithme apprendre à densifier lui même les cartes de profondeur brutes. En d'autres termes, nous constatons que pour se comparer à l'état de l'art dans les meilleures conditions, il vaut mieux apprendre sur les vérités terrains colorisées suivant Levin et al. (2004) plutôt qu'uniquement sur les points valides des cartes de profondeur brutes.

2.4.2 KITTI

La base de données KITTI (Geiger et al., 2013), tout aussi populaire, comporte des séquences vidéo stéréo enregistrées en extérieur et acquises par un véhicule en circulation équipé de nombreux capteurs, incluant un LiDAR. Les scènes sont variées et on compte un peu plus de 40K paires d'images de 56 environnements différents de type urbains, résidentiels, de routes, de campus ou encore de séquences avec principalement des piétons. Le jeu de données propose donc des cartes de profondeur parcimonieuses (acquisition LiDAR) mais également d'autres types d'annotations comme le flot optique ou encore des labels sémantiques (Geiger et al., 2012; Menze et Geiger, 2015) (voir figure 2.11).

La résolution des images est de 1392×512 pixels et, bien que le LiDAR ayant servi pour l'acquisition ait une portée de 120m, on plafonne en pratique souvent les distances à 80m au maximum pour que cela corresponde à une portion raisonnable de pixels de l'image $\mathcal{RGB}$.

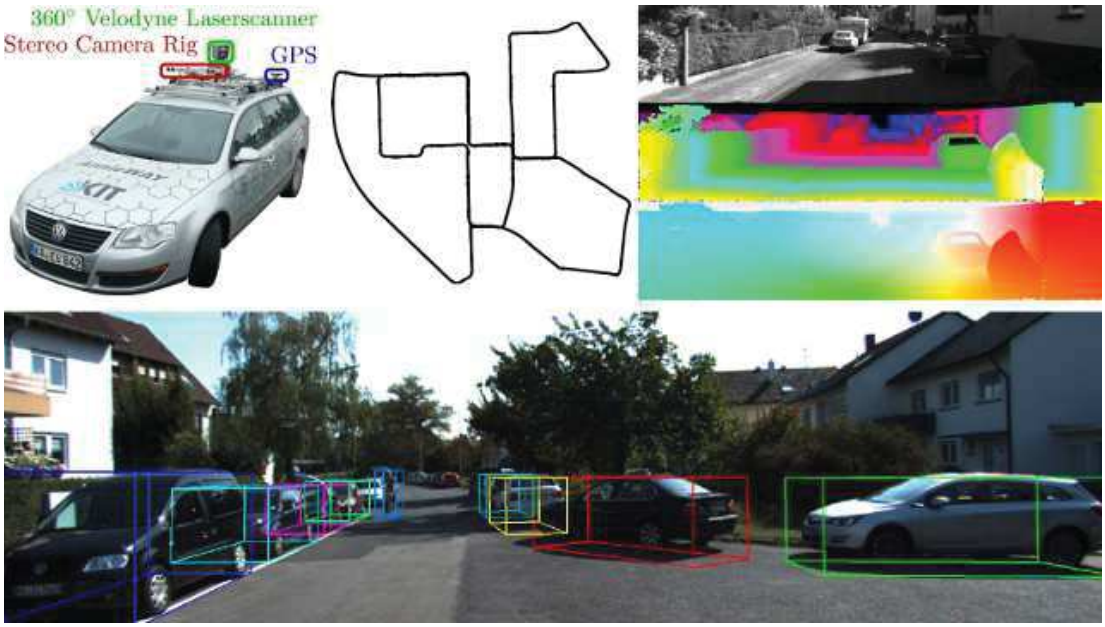


FIGURE 2.11 – Illustration des données présentes dans KITTI

FIGURE 2.12 – A gauche des images du jeu de données KITTI et à droite leur équivalent dans Virtual KITTI

Notons également que les cartes de profondeur de KITTI ne présentent de pixels valides que sur la partie basse de l'image, correspondant au champ d'action du LiDAR, tandis que les images $\mathcal{RGB}$ qui leur sont associées capturent une vue plus large de la scène. Suivant les travaux de Eigen et al. (2014), on trouve généralement 28 scènes utilisées pour l'entrainement et les 28 autres pour le test.

2.4.3 Virtual KITTI

Enfin, la base de données Virtual KITTI est une base de vidéos synthétiques, photo-réaliste, présentant des environnements similaires à ceux rencontrés dans KITTI (Gaidon et al., 2016).

En effet, les scènes modélisées dans Virtual KITTI sont conçues pour être semblables à celles que l'on trouve dans le jeu de données KITTI (voir figure 2.12). Virtual KITTI offre plusieurs types d'annotations comme les labels sémantiques, le flot optique et les cartes de profondeur des images. On trouve 50 vidéos monoculaires comportant plus de 20K frames générées à partir de 5 types d'environnements différents (urbains) et avec différentes conditions météorologiques.

L'avantage d'une base synthétique est la disponibilité d'une grande quantité de données avec des annotations parfaites, au pixel près, et entièrement denses. Cependant, même pour des images photo-réalistes, il existe un gap de domaine entre des données synthétiques et des données réelles. Cet écart s'explique par des textures ou des configurations de scène souvent moins riches dans les jeux de données synthétiques. Un apprentissage sur Virtual KITTI n'est donc pas directement transférable sur une autre base de données, même similaire comme KITTI : cela nécessite une étape dite d'adaptation de domaine pour obtenir des résultats probants.

Chapitre 3

Influence des hyperparamètres

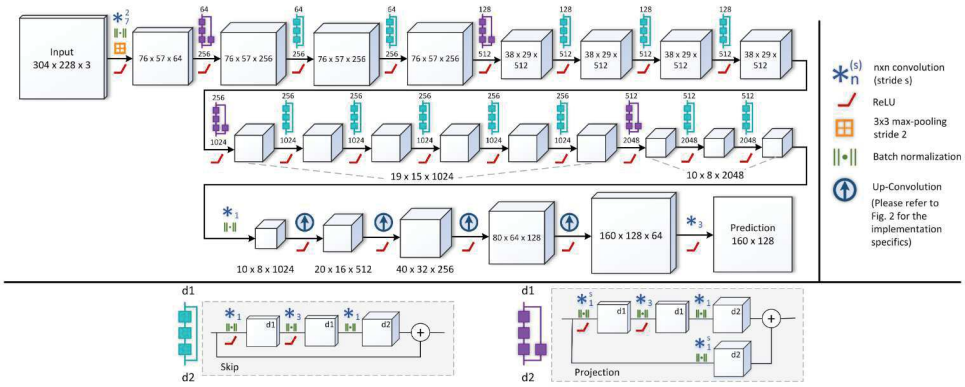


FIGURE 3.1 – Schéma de l'architecture du réseau de Laina et al. (2016)

L'optimisation des algorithmes d'apprentissage statistiques repose généralement sur le choix d'un certain nombre de paramètres liés à l'entraînement du modèle. Ces paramètres, appelés *hyperparamètres*, peuvent être relatifs à la structure du modèle en lui même ou à la manière d'optimiser ses paramètres pour une tâche donnée. Dans le cas des réseaux de neurones, un grand nombre d'hyperparamètres peuvent être mis en jeu et influent considérablement sur les performances obtenues. Dans ce chapitre, nous cherchons à étudier l'influence des principaux hyperparamètres sur l'apprentissage d'un réseau de neurones profond entraîné pour l'estimation de profondeur monoculaire.

Nous commençons par inventorier et préciser le rôle de ces hyperparamètres régissant le processus d'apprentissage. Ensuite, nous proposons un plan d'expériences 'à étages', en partant d'un réseau standard et en modifiant au fur et à mesure chaque hyperparamètre dans le but de comprendre comment chacun d'eux influence l'apprentissage du réseau, à la fois en termes de performance quantitative pure des estimations de profondeur, mais aussi en termes de qualité visuelle des cartes prédites. Pour ce faire, nous travaillons sur la base de données NYU Depth v2 (Silberman et al., 2012) où les jeux d'entraînement et de test seront fixés.

3.1 Introduction

Au moment où nous écrivons ces lignes, les architectures de réseaux de neurones qui constituent l'état de l'art pour l'estimation de profondeur ont une structure en sablier (avec un encodeur et un décodeur connectés). L'encodeur encodeur est typiquement préentraîné sur des tâches de classification sémantiques. Par exemple, Laina et al. (2016) utilisent comme extracteur de features (encodeur) un réseau 'ResNet' dont ils retirent les dernières couches, celles initialement prévues pour la classification, et les remplacent par un décodeur. Cette architecture est illustrée Figure 3.1. Laina et al. (2016) instaurent un nouvel état de l'art en termes de performances sur l'estimation de profondeur à partir d'une seule image monoculaire,

avec une erreur relative d'environ 13% sur le jeu de données NYU Depth v2 (Nathan Silberman et Fergus, 2012). C'est à ce réseau et son architecture que nous nous intéressons principalement dans ce chapitre.

Dans un premier temps, nous nous intéressons au rôle que joue chaque composant (encodeur/décodeur) pour ce type d'architecture. Nous regardons en particulier l'influence de l'encodeur en comparant les performances obtenues par les principaux encodeurs de la littérature. De la même manière, nous nous intéressons à l'architecture du décodeur en comparant un décodeur très simple à celui plus élaboré proposé par Laina et al. (2016). Enfin, nous regarderons quelle influence a la régularisation par *dropout* sur les capacités de généralisation du réseau.

Pour pouvoir tirer des conclusions pertinentes, nous fixons dans nos expériences la méthode d'optimisation des réseaux, pour ne comparer que l'élément d'architecture étudié. Nous choisissons d'utiliser une optimisation de type 'Adam' avec un pas d'apprentissage de 10^{-4} pendant 100 époques, puis nous appliquons une minimisation de type 'SGD' pour 20 époques à un pas d'apprentissage de 10^{-3}. Le paramètre de 'moment' (inertie du gradient) est égal à 0.9 et la régularisation de type *weight decay* égale à 5.10^{-4}. Nous apprenons sur les 795 images d'entraînement officielles du jeu de données NYU Depth v2, avec une fonction de coût $\mathbb{L}_2$ classique et comparons nos résultats sur la base de test officielle de 654 images. Enfin, nous appliquons également de l'augmentation de données en ligne lors de l'entraînement, en suivant la procédure décrite dans (Eigen et al., 2014), qui comprend :

- Mise à l'échelle : les entrées et les sorties sont suréchantillonnées par un facteur d'échelle $s \in [1, 1.5]$ pour simuler un rapprochement de la caméra ; les profondeurs sont donc divisées par s pour conserver la géométrie de la scène.
- Rotation : les entrées et les sorties subissent une rotation de $r \in [-5, 5]$ degrés.
- Translation : les entrées et les sorties, après suréchantillonnage, sont aléatoirement recadrées à la taille désirée pour simuler une translation de la caméra.
- Couleur : les valeurs RGB des entrées sont multipliées par une valeur aléatoire comprise entre $c \in [0.8, 1.2]^3$.
- Flip : les entrées et les sorties ont une chance de subir un retournement horizontal avec une probabilité de 0.5.

Dans un second temps, c'est à architecture fixée que nous nous pencherons sur l'étude d'un certain nombre de paramètres liés à l'optimisation. En effet, ces derniers s'avèrent très importants pour garantir la convergence vers un bon minimum et sont très influents sur la capacité de généralisation de nos réseaux. Nous verrons en particulier l'impact du choix du type d'optimiseur dans la descente de gradient, l'importance de l'initialisation des paramètres ainsi que celle de la fonction de coût. Pour finir nous observerons l'influence de la quantité de données d'entraînement utilisées pour l'apprentissage.

FIGURE 3.2 – Une image de test de NYU Depth v2. À gauche l'image $\mathcal{RGB}$ (entrée) et à droite la carte de profondeur $\mathcal{D}$ associée (sortie attendue)

Afin de pouvoir comparer visuellement les résultats qualitatifs obtenus, nous choisissons de regarder les estimations que nous obtiendrons dans ce chapitre sur une même image de la base de test de NYU Depth v2. L'image d'entrée $\mathcal{RGB}$ en question ainsi que la vérité terrain de profondeur $\mathcal{D}$ associée sont données Figure 3.2. Nous recommandons de regarder les figures en couleur.

3.2 Architecture

Dans cette partie, nous nous intéresserons aux aspects architecturaux du réseau. Comme le nombre d'hyperparamètres liés à l'architecture est très grand (nombre de convolutions, taille de leurs noyaux, nombre de canaux, non linéarités, connexions spatiales, manières de sous-échantillonner et de sur échantillonner, ordonnancement des couches, etc.), nous choisissons de restreindre cette étude aux critères que nous pensons être les plus influents.

3.2.1 Encodeur

Comme nous l'avons dit ci-dessous, les architectures types encodeur-décodeur (sablier) sont très populaires dans la littérature pour les tâches de régression de profondeur. C'est en particulier parce qu'elles permettent de recouvrer la résolution spatiale des données fournies en entrée. C'est un avantage majeur quand les données à inférer sont structurées spatialement, comme c'est le cas pour nous pour l'estimation de cartes de profondeur. La partie encodeur permet d'extraire des caractéristiques de l'image $\mathcal{RGB}$ qui seront utiles à son décodage en carte de profondeur. Ainsi, il faut que les descripteurs produits par l'encodeur représentent les indices monoculaires contenus dans l'image utiles au passage de la modalité couleur vers la modalité de profondeur. Nous allons évaluer 3 types d'encodeurs différents au sein de notre architecture, à savoir les réseaux :

— AlexNet Krizhevsky et al. (2012b)

— VGG (Simonyan et Zisserman, 2014)

— ResNet He et al. (2016)

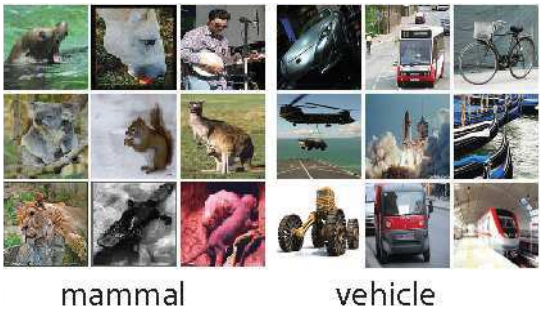


FIGURE 3.3 – Illustration d'images de deux classes issues de la base de données *ImageNet* (Russakovsky et al., 2015)

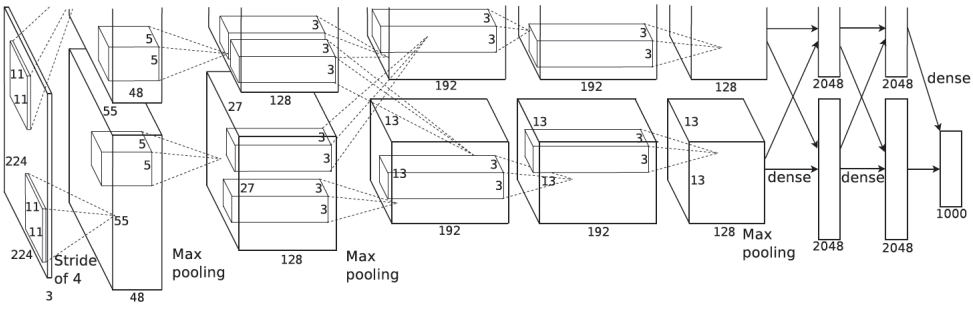


FIGURE 3.4 – Architecture du réseau AlexNet proposé par Krizhevsky et al. (2012b). Les deux flux de données schématisés correspondent à la parallélisation du réseau sur 2 GPU

AlexNet

AlexNet est un réseau de neurones convolutionnel conçu par Krizhevsky et al. (2012b) et connu pour avoir donné des résultats impressionnants en 2012 à la compétition *ImageNet Large Scale Visual Recognition Challenge* (Russakovsky et al., 2015). Pour cette compétition est mise à disposition la base de données *ImageNet* qui comporte plusieurs millions d'images annotées avec la liste des objets présents dans chaque scène (voir Figure 3.3). Le but de la compétition est de prédire la classe des images (présence d'objet). AlexNet atteint des performances affichant un écart de plus de 10 points avec la deuxième meilleure méthode.

Le réseau AlexNet est composé de 8 couches : les 5 premières sont des couches de convolutions dont le rôle est de calculer des descripteurs de l'image tandis que les 3 dernières sont des couches entièrement connectées et permettent d'effectuer la tâche de classification (illustré Figure 3.4). Les auteurs mettent en avant le fait que la profondeur du modèle est essentielle pour obtenir ces hautes performances et, bien que le coût de calcul et surtout l'occupation mémoire soient plus élevés, l'entraînement du réseau est rendu possible grâce à la parallélisation des calculs sur plusieurs cartes graphiques. Notons que c'est le même réseau qui est utilisé pour apprendre des descripteurs de l'image $\mathcal{RGB}$ et estimer une première carte de profondeur $\mathcal{D}$ grossière par Eigen et al. (2014).

FIGURE 3.5 – Schéma de l'architecture du réseau VGG16 (Simonyan et Zisserman, 2014) comportant 13 couches de convolution et 3 couches de classification

VGG

VGG (Simonyan et Zisserman, 2014) est une architecture de réseau de neurones convolutionnel profond développée par une équipe de recherche de l'université d'Oxford connue sous le nom de *Visual Geometry Group*. Ce réseau s'est également fait connaître grâce à la compétition *ImageNet Large Scale Visual Recognition Challenge* (Russakovsky et al., 2015). En effet en 2014, VGG se place en première position de la compétition pour la tâche de localisation d'objets et en deuxième position pour la tâche de classification d'images.

Le modèle comporte entre 16 et 19 couches dont 3 couches entièrement connectées à la fin pour la classification. Le reste est composé de couches de convolutions avec des successions de filtres dont les noyaux sont de petite taille uniquement (ici 3x3, voir Figure 3.5). Les auteurs montrent, là encore, que la profondeur du réseau joue un rôle important sur les résultats obtenus et que les réseaux les plus profonds atteignent de meilleures performances.

ResNet

Cependant, plus le réseau est profond, plus il est difficile de l'entraîner. En effet lors de la rétropropagation de l'erreur, les gradients sont calculés à l'aide du théorème de dérivation des fonctions composées. Le gradient des premières couches est donc le résultat d'une multiplication des gradients des couches suivantes. Lorsque les couches sont nombreuses et que les gradients impliqués sont inférieurs à 1, on se retrouve avec des valeurs très faibles pour les premières couches ; leur apprentissage s'en trouve fortement ralenti. Ce problème est connu sous le nom d'*évanouissement du gradient* (Bengio et al., 1994).

C'est à cause du problème de l'évanouissement du gradient que, pendant des années, il a été difficile d'entraîner des réseaux très profonds, avec notamment des temps de calcul beaucoup plus longs (Simonyan et Zisserman, 2014). De plus, on remarque qu'à un certain point, augmenter le nombre de couches n'améliore pas forcément les performances du réseau tandis que sa capacité de représentation est supposée accrue. C'est en partant de ce constat que He et al. (2016) proposent une nouvelle architecture appelée *Residual Network* ou 'ResNet'.

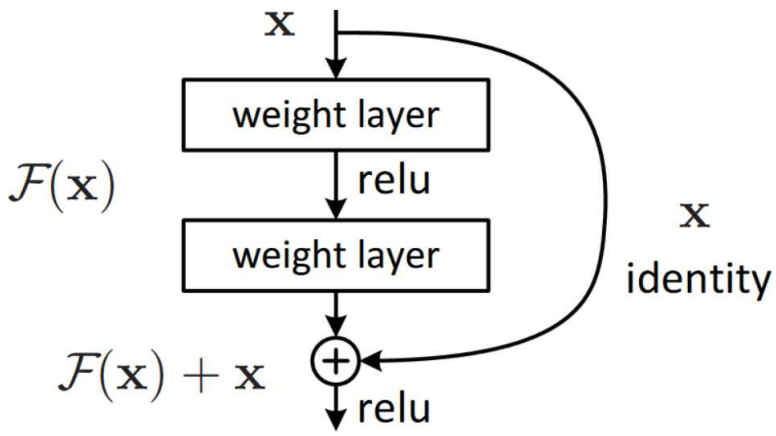


FIGURE 3.6 – Schéma du principe d'un bloc résiduel utilisé dans les ResNet (He et al., 2016)

Encodeur	RMSE sur profondeur	Gain relatif profondeur	Erreur top-5 ImageNet classification
AlexNet	1.001	-	20.91%
VGG16	0.899	+10.2%	9.62%
ResNet18	0.756	+24.5%	10.92%
ResNet50	0.726	+27.5%	7.13%
ResNet200	0.690	+31.1%	5.79%

TABLE 3.1 – Impact de l'encodeur sur l'inférence de la profondeur

Cette architecture est basée sur des blocs de convolution spécifiques dont le but est d'apprendre localement des fonctions résiduelles. Concrètement, cela se traduit par une connexion directe reliant deux endroits du réseau en sautant plusieurs convolutions. Ces connexions, appelée *skip connections*, opèrent une simple fonction identité. Leur principe est illustré Figure 3.6. Les ResNet adressent par ce biais le problème d'évanouissement du gradient puisque les *skip connections* offrent des chemins alternatifs, raccourcis entre l'entrée et la sortie du réseau pour les gradients. Ainsi, les couches proches de l'entrée se trouvent connectées à des couches plus proches de la sortie et les gradients ne sont pas atténués par toute la profondeur du réseau. Ce principe a permis d'entraîner des architectures très profondes (plus de 150 couches) et ainsi d'obtenir des performances impressionnantes dans des domaines tels que la classification et la détection d'objets (He et al., 2016). Le 'ResNet' existe en plusieurs versions standards selon le nombre de couches utilisées pour son implémentation (de 18 couches à 200 couches). On notera ResNet$L$ le ResNet avec L couches de convolutions.

Comparaison des 3 encodeurs

Nous comparons maintenant ces trois réseaux utilisés comme encodeur d'une architecture en sablier. Pour se faire, nous enlevons pour chacun les dernières couches entièrement connec-

tées, dédiées à l'origine à la tâche de classification, et nous les remplaçons par un décodeur. Ce décodeur est identique pour toutes les architectures, composé simplement de 5 déconvolutions permettant de recouvrer en sortie la même résolution spatiale que celle de l'image d'entrée de 240x320 pixels. Ces déconvolutions ont des filtres de taille 4x4 avec un pas de 2 pour doubler la dimension spatiale des descripteurs. Le nombre de canaux de sortie est fixe et égal à 256. Enfin à la suite se trouve une dernière convolution, de noyau 3x3, avec un seul canal de sortie et utilisée pour produire la prédiction. Entre chaque déconvolution se trouve une couche de régularisation de type *dropout* spatial avec une probabilité de masquer les canaux de 0.3. Nous reviendrons sur cela par la suite.

Chaque encodeur est initialisé avec ses poids appris pour la tâche de classification sur les données *ImageNet*. Les résultats sont résumés Table 3.1. La première ligne est prise en référence pour le calcul du gain relatif (3e colonne). Nous affichons également dans la table les performances de classification que chaque réseau obtenait dans sa configuration d'origine (avec les dernières couches entièrement connectées) sur la base de test d'*ImageNet*. L'erreur top-5 signifie que la classe correcte doit être dans le top 5 des classes de plus grande probabilité prédites par le réseau pour que la prédiction soit considérée comme valide. Le score est un pourcentage d'erreur et il est donc à minimiser.

Nous constatons tout d'abord que le choix de l'encodeur a un très fort impact sur les performances que l'on obtient sur la base de test. En effet, on voit des gains relatifs allant de 10 à plus de 30% d'amélioration. C'est d'autant plus vrai que les encodeurs que nous utilisons sont très différents et présentent eux-mêmes une grande disparité dans leurs résultats de classification sur *ImageNet*. En général, meilleur est l'encodeur sur la tâche de classification, meilleurs seront les performances pour l'estimation de profondeur. Ces résultats sont assez cohérents avec ce que nous avons vu au chapitre 2 : on trouve plusieurs indices monoculaires qui se basent sur la connaissance a priori des objets qui composent la scène comme les textures, la taille relative ou les occultations. Il n'est donc pas étonnant de constater qu'un encodeur qui présente de meilleures performances pour une tâche sémantique puisse extraire de meilleurs descripteurs sémantiques de l'image et que le réseau global arrive au final à produire des estimations de profondeur de meilleure qualité.

La seule exception que nous constatons concerne les résultats obtenus avec l'encodeur VGG16 et ceux avec le ResNet18. En effet, VGG16, qui affiche une erreur de classification sur *ImageNet* de 9.62%, obtient un moins bon score lorsqu'il est utilisé pour l'estimation de profondeur que le ResNet18 dont l'erreur de classification sur *ImageNet* est pourtant de 10.92%. Nous pensons que c'est dû à l'architecture du ResNet qui, comme nous l'avons vu, facilite son entraînement grâce aux *skip connections* qui empêchent l'évanouissement du gradient. Le réseau VGG avec ses 16 couches est presque aussi profond que le ResNet18 mais sa spécialisation pour la tâche d'estimation de profondeur est plus difficile car son architecture est justement sujette à l'évanouissement du gradient (Simonyan et Zisserman, 2014).

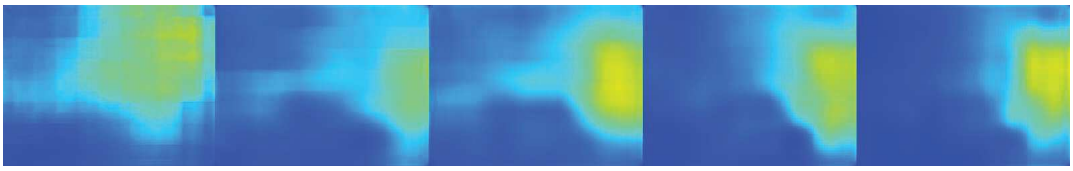


FIGURE 3.7 – Estimations de profondeur obtenues avec différents encodeurs. De gauche à droite : AlexNet, VGG16, ResNet18, ResNet50 et ResNet200

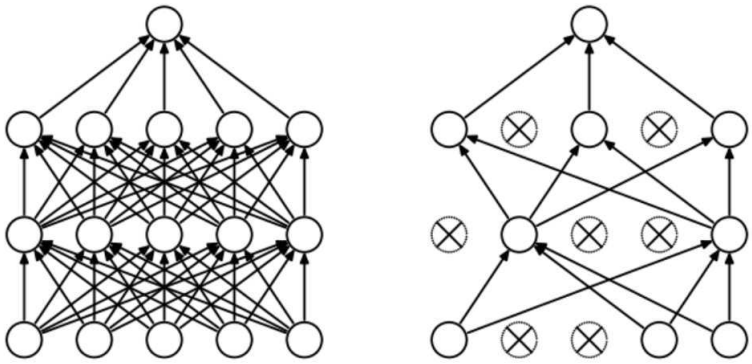


FIGURE 3.8 – Régulation par 'dropout' avec la méthode proposée par Srivastava et al. (2014)

La Figure 3.7 illustre les estimations obtenues avec ces différentes architectures sur l'image de test utilisée pour la comparaison visuelle. Les estimations de profondeur reflètent bien les résultats quantitatifs que nous venons de voir. On constate que les prédictions sont progressivement plus fines et plus détaillées, avec de moins en moins d'artefacts de reconstruction. On voit par exemple que l'utilisation de l'encodeur AlexNet produit une carte de profondeur très grossière comme c'est la cas pour le réseau global proposé par Eigen et al. (2014) (dont l'encodeur est le même) et que les cartes de profondeur deviennent plus précises à mesure que l'encodeur gagne en puissance de représentation sémantique (on commence à distinguer les formes du meuble de gauche pour les 2 ResNet les plus profonds). Pour la suite, nous fixons l'encodeur et utilisons le ResNet200 qui présente ici les meilleures performances.

3.2.2 Dropout

Le dropout est une technique de régularisation utilisée dans les réseaux de neurones pour réduire le surapprentissage en masquant aléatoirement une proportion choisie de neurones sur la couche où il est appliqué (voir Figure 3.8). Il a été proposé par Hinton et al. (2012) pour prévenir la redondance ainsi que la coadaptation des neurones lors de l'entraînement.

En effet, on souhaite qu'idéalement chacun des neurones du réseau fonctionne comme un détecteur de caractéristiques discriminant, indépendamment de ce qu'apprennent les autres neurones. Si plusieurs neurones détectent les mêmes caractéristiques de manière répétée (redondance) alors le réseau n'utilise pas pleinement sa capacité : il perd une partie de ses

Dropout	RMSE	Gain relatif
Dropout = 0.3	0.690	-
Dropout = 0	0.681	+1.3%
Dropout = 0.5	0.699	-1.0%

TABLE 3.2 – Impact du dropout sur l'estimation de profondeur.

ressources en calculant l'activation de neurones qui font la même chose.

D'autre part lors de l'apprentissage, certains neurones peuvent changer de sorte à corriger les erreurs commises par d'autres neurones (co-adaptation). La contribution de ces unités devient donc dépendante des sorties d'autres unités. Cela tend à conduire au surapprentissage, car ces coadaptations complexes ne généralisent pas pour de nouvelles données (Srivastava et al. (2014)). En mettant en œuvre la technique du dropout, le modèle est contraint d'avoir des neurones capables d'apprendre des caractéristiques discriminantes sans s'appuyer sur les autres neurones.

Le dropout, initialement conçu en 1D pour des réseaux de neurones entièrement connectés, existe en version 2D adaptée aux réseaux de neurones convolutionnels. On parle alors de *dropout spatial*. Dans le cas du dropout spatial, ce n'est pas un neurone isolé qui a une probabilité d'être masqué mais tout un canal du tenseur représentant le descripteur de la couche considérée. En effet le dropout standard ne fonctionnerait pas aussi bien sur des images car les pixels adjacents sont naturellement fortement corrélés. Le dropout de cartes de caractéristiques entières correspond donc plus à l'intention originale initiée par Hinton et al. (2012).

Nous nous intéressons ici à l'effet du *dropout spatial* dans notre architecture. Comme nous l'avons mentionné précédemment, nous avons dans notre décodeur 4 couches de *dropout spatial* situées entre les couches de déconvolution du décodeur. La probabilité de masquer les canaux était précédemment fixée à 0.3. Nous étudions l'influence qu'à cette probabilité dans l'apprentissage en l'enlevant complètement (probabilité de 0) et en l'augmentant (probabilité de 0.5, soit en moyenne un canal sur deux masqué lors de l'apprentissage). Les résultats obtenus pour ces apprentissages sont résumés dans la table 3.2. La première ligne correspond à la probabilité de 0.3 que nous avions précédemment et est prise en référence pour le calcul du gain relatif.

Nous constatons d'abord que les écarts de performances affichés pour ces différentes expériences ne sont pas très prononcés, avec environ 1% d'amélioration ou de détérioration. Par ailleurs, nous constatons que la désactivation du *dropout* entraîne une légère amélioration des performances tandis que nous sommes dans un cas typique sujet au surapprentissage : nous utilisons un réseau très profond (200 couches pour l'encodeur, 5 couches pour le décodeur et une dernière couche pour produire la prédiction), avec peu de données d'apprentissage (795 images).

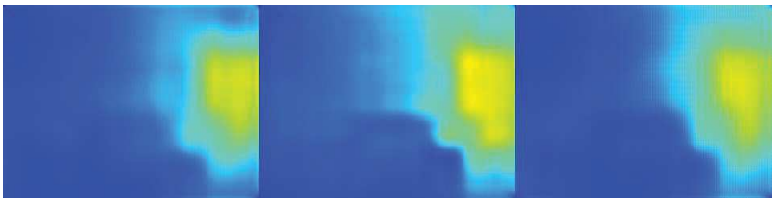


FIGURE 3.9 – Estimations de profondeur obtenues avec différents taux de *dropout*. De gauche à droite : *dropout*=0.3, *dropout*=0, *dropout*=0.5

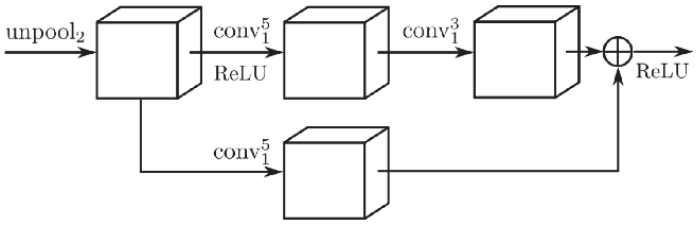


FIGURE 3.10 – Illustration du module de projection introduit par Laina et al. (2016). Le nombre de canaux est réduit à la sortie de ce module tandis que la résolution spatiale des descripteurs est multipliée par 2.

L'utilisation du *dropout* devrait donc aider à éviter le surapprentissage (s'il y en a effectivement un) et donc améliorer la capacité de généralisation du réseau. Cependant, les couches de *dropout* sont situées dans le décodeur uniquement, entre les couches de déconvolutions. Nous pensons que l'augmentation de la probabilité de *dropout* entraîne une trop grande réduction de la capacité du décodeur, qui est pourtant déjà très simple.

Cependant les écarts de performances constatés étant très faibles même dans ces conditions, nous pensons que le *dropout* spatial n'influence pas fortement les performances du réseau. Nous pouvons donc conserver une valeur relativement faible comme 0.3 pour éviter un possible surapprentissage, notamment si le décodeur utilisé est plus profond comme celui auquel nous allons nous intéresser par la suite.

La Figure 3.9 présente les cartes de profondeur obtenues pour ces différentes valeurs de *dropout*. Nous constatons que lorsque le *dropout* a une probabilité nulle, l'estimation produite est un peu plus détaillée (contours du meuble un peu plus marqués) tandis que les deux autres estimations se ressemblent et sont un peu plus grossières.

3.2.3 Décodeur

Pour leur partie décodeur, Laina et al. (2016) proposent des blocs de suréchantillonnage basés sur le principe de résidu des couches du ResNet : des *skip connections* intègrent des caractéristiques multi-échelles locales et permettent au gradient de passer par différentes branches.

Décodeur	RMSE	Gain relatif
5 déconvolutions + 1 convolution	0.690	-
5 modules de projection + 1 convolution	0.661	+4.2%

TABLE 3.3 – Impact du décodeur sur l'inférence de profondeur.

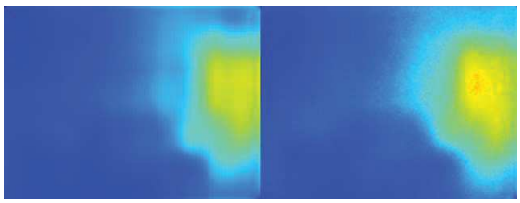


FIGURE 3.11 – Estimations de profondeur obtenues avec deux décodeurs différents. À gauche le décodeur classique et à droite le décodeur avec des modules de projection.

Une opération dite d'unpooling consiste à doubler la résolution spatiale des descripteurs reçus en entrée en ajoutant des zéros entre les valeurs. Une convolution est ensuite utilisée pour densifier les descripteurs suréchantillonnés. La Figure 3.10 illustre la structure de ce module, nommé *module de projection*.

Nous utilisons également les modules de projection proposés par Laina et al. (2016) pour construire un décodeur plus complexe que celui que nous utilisions jusqu'alors afin de comprendre son importance dans l'architecture. Nous entraînons donc un réseau avec le même encodeur (ResNet200) et 5 modules de projection à la place des 5 déconvolutions que nous utilisions. Nous conservons la dernière convolution 3x3 permettant de produire la prédiction. Les résultats sont reportés dans la table 3.3. Nous constatons sans surprise un gain de 4.2% à l'utilisation d'un décodeur plus élaboré, constitué de modules de projection. Cependant cet écart de performance n'est pas aussi important que celui constaté avec les changements d'encodeur. A titre de comparaison, le passage du ResNet50 au ResNet200, qui n'ont pourtant que 1.34% de différence pour résultats sur *ImageNet*, donne une amélioration des estimations de profondeur de 5%.

Les résultats qualitatifs sont présentés Figure 3.11. Nous constatons cette fois une nette différence visuelle. En effet, l'estimation obtenue avec des déconvolutions est très carrée et l'on peut distinguer un sorte de motif qui se répète. C'est certainement un artefact de décodage dû à l'utilisation des déconvolutions qui procèdent au suréchantillonnage des descripteurs par blocs. D'un autre côté, l'estimation de profondeur obtenue avec le décodeur basé sur des modules de projection est plus lisse et présente un grain particulier, différent de l'artefact précédent. Nous pensons que ce grain est dû à une forme de surapprentissage puisque des pixels voisins appartenant aux mêmes objets et donc avec des textures $\mathcal{RGB}$ similaires ne présentent pas forcément la même estimation de distance. Ce n'est pas surprenant puisque la capacité du réseau a grandement augmenté et donc le modèle présente une plus forte variance. Pour la

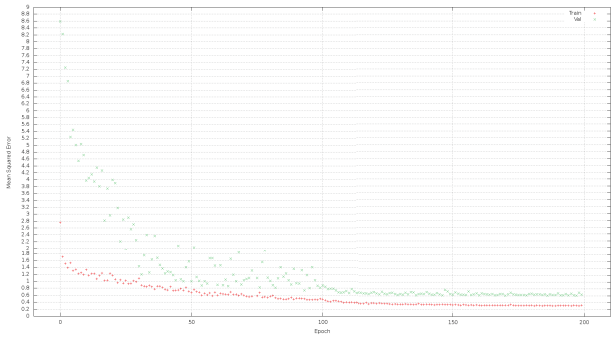


FIGURE 3.12 – Courbe d'évolution de l'erreur RMSE en fonction du nombre d'époques de l'apprentissage. En rouge l'erreur d'entraînement et en vert l'erreur de validation

suite, nous conserverons l'architecture de décodeur de Laina et al. (2016) basée sur les modules de projection puisque les résultats sont légèrement meilleurs quantitativement mais surtout pour que la qualité visuelle des estimations.

3.3 Étude des paramètres liés à l'optimisation

Dans cette section, nous conservons l'architecture du réseau précédent et étudions l'effet des hyperparamètres liés à l'optimisation sur les performances de généralisation. Nous choisissons de prendre un ResNet200 pour encodeur et le décodeur basé sur le module de projection de Laina et al. (2016) ainsi que de conserver des couches de *dropout* spatial de 0.3 comme nous l'avons vu précédemment. Nous étudierons ici plus spécifiquement l'importance de l'initialisation de l'encodeur (préentraîné ou non sur *ImageNet*) et du décodeur ; l'influence de l'optimiseur et de ses paramètres ; l'impact de la fonction de coût minimisée ; l'importance du nombre d'images d'entraînement et enfin l'impact de l'application d'un algorithme d'augmentation des données en ligne.

3.3.1 Arrêt de l'apprentissage

Tout d'abord, nous vérifions ici que notre choix d'arrêter nos apprentissages après 120 époques est bien justifié et qu'il n'altère pas la validité des comparaisons effectuées. En effet, nous avons choisi d'optimiser nos réseaux pour 120 époques, dont 100 époques d'abord effectuées avec l'optimiseur Adam puis 20 époques avec SGD. Nous reviendrons sur le choix de ces optimiseurs par la suite. Ce que nous cherchons à montrer ici est qu'un tel apprentissage nous amène bien à la convergence des réseaux, avec une erreur de validation stable, ce qui garantit une comparaison juste des différentes méthodes. Pour ce faire, nous utilisons un jeu de validation de 1000 images sélectionnées aléatoirement dans le jeu d'entraînement brut de NYU Depth v2. Nous traçons la courbe d'évolution de l'erreur RMSE en fonction de l'époque, voir figure 3.12.

Initialisation	RMSE	Gain relatif
Initialisation sur *ImageNet*	0.661	-
Initialisation aléatoire	0.949	-43.6%

Table 3.4 – Impact de l'initialisation de l'encodeur

D'abord, nous constatons que durant les 100 premières époques, l'erreur de validation décroît rapidement mais est également très bruitée. C'est dû à l'optimiseur Adam qui permet de converger plus rapidement vers un minimum mais qui est plus sujet aux oscillations, nous y reviendrons. Ensuite, nous passons à l'optimiseur SGD et nous constatons que l'erreur se stabilise. En outre, nous pouvons voir qu'à partir de là elle atteint très rapidement sa valeur minimum sur le jeu de validation et stagne alors pour le reste de l'apprentissage tandis que l'erreur d'entraînement ne décroît que très lentement. Avec 120 époques d'entraînement, nous nous assurons donc d'être déjà parvenu dans le bassin correspondant à un minimum local trouvé par le réseau : poursuivre l'apprentissage au delà de cette valeur n'a plus d'intérêt. Nous avons observé un comportement similaire pour les différentes architectures testées et c'est pourquoi nous choisissons de fixer à 120 époques nos apprentissages : c'est un bon compromis entre garantie de convergence et temps d'apprentissage.

3.3.2 Importance de l'initialisation

Encodeur

Comme nous l'avons vu, l'encodeur a été choisi pour ses qualités de représentation sémantique des données (hautes performances de classification sur *ImageNet*). Nous avons également observé la corrélation existante entre les performances de l'encodeur pour la classification et celles obtenues pour le réseau global en estimation de profondeur. Nous poussons ici l'analyse en regardant les performances que l'on obtient avec le meilleur encodeur testé, ici le ResNet200, mais avec une initialisation aléatoire plutôt qu'avec ses poids appris pour la classification sur *ImageNet*. Les résultats sont présentés en table 3.4. La différence est impressionnante, nous constatons une dégradation des performances de plus de 40%. Pour rappel, l'erreur RMSE que l'on obtenait avec le plus petit et le moins bon encodeur, AlexNet, était de 1. Ici lorsque le réseau avec le meilleur encodeur est initialisé aléatoirement on obtient une erreur très proche, égale à 0.95. Ces résultats renforcent nos conclusions et montrent une fois encore que l'aspect sémantique est primordial pour l'estimation de profondeur.

Nous présentons également les estimations de profondeur que nous obtenons Figure 3.13. Sans surprise, nous constatons aussi une différence dans la qualité de la prédiction lorsque le ResNet n'est pas préentraîné sur *ImageNet*. L'estimation obtenue est plus grossière et moins détaillée, tout comme l'étaient les prédictions que nous obtenions avec des encodeurs moins

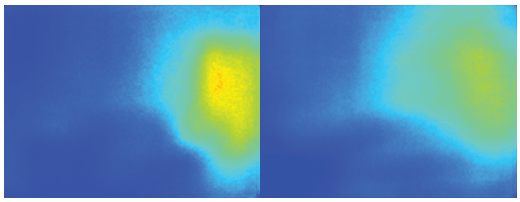


FIGURE 3.13 – Estimations de profondeur obtenues avec deux initialisations différentes de l'encodeur. À gauche l'encodeur est préentraîné sur *ImageNet* ; à droite il est initialisé aléatoirement.

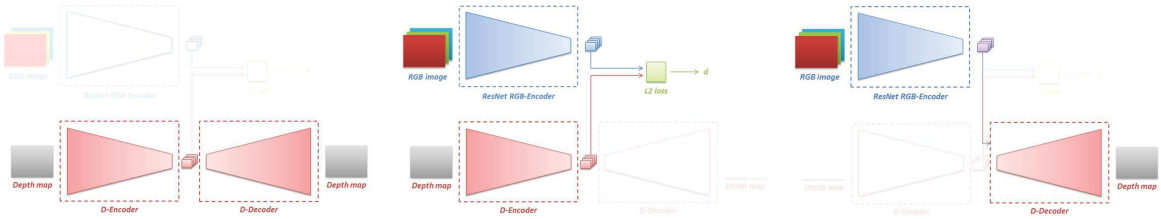


FIGURE 3.14 – Schéma illustrant les 3 étapes utilisées pour initialiser le décodeur de profondeur. De gauche à droite : i) apprentissage de l'auto-encodeur de profondeur, ii) alignement des descripteurs $\mathcal{RGB}$ et $\mathcal{D}$, iii) d'affinage de l'ensemble $\mathcal{RGB}$-$\mathcal{D}$

puissants dans la section précédente. Nous notons cependant toujours la présence du grain particulier dû à l'utilisation de modules de projection dans le décodeur.

Décodeur

L'architecture du décodeur a montré des améliorations limitées par rapport au gain constaté avec un puissant encodeur. Nous vérifions ici si une initialisation particulière du décodeur peut conduire à de meilleurs résultats. Pour ce faire, nous proposons une méthode composée de 3 étapes où la clé est d'apprendre un décodeur de profondeur de manière non supervisée et d'utiliser ses poids pour notre réseau d'estimation de profondeur. Cette méthode est décrite ci-après et illustrée Figure 3.14.

Étape 1 : Auto-encodeur de profondeur Tout d'abord, nous apprenons un auto-encodeur spécifique à la reconstruction de la profondeur. Cet auto-encodeur est appris de manière non supervisée et nous utilisons une fonction $\mathbb{L}_2$ classique pour l'entraînement, où la reconstruction est comparée à l'image d'entrée. Nous voulons par la suite réutiliser le décodeur appris de cette façon pour effectuer nos estimations de profondeur à partir d'images $\mathcal{RGB}$.

Étape 2 : Alignement Nous proposons un moyen d'aligner un encodeur $\mathcal{RGB}$ sur la représentation interne apprise par l'auto-encodeur de profondeur, que le décodeur utilise pour sa reconstruction. Pour ce faire, nous utilisons un ResNet200 pour l'encodage et nous lui apprenons à s'aligner sur la représentation interne de l'auto-encodeur de profondeur. Plus spécifiquement, pour chaque image, nous calculons d'abord sa représentation latente de profondeur en pas-

Architecture	RMSE	Gain relatif
Auto-encodeur de profondeur	0.209	-
ResNet200 pré entraîné sur *ImageNet* + décodeur initialisé aléatoirement	0.661	-
ResNet200 initialisé aléatoirement + décodeur initialisé aléatoirement	0.949	-43.6%
ResNet200 initialisé aléatoirement + décodeur initialisé par auto-encodeur	0.955	-44.5%
ResNet aligné + décodeur initialisés par auto-encodeur (fin étape 2)	0.710	-7.4%
ResNet + décodeur pré entraînés, alignés et affinés (fin étape 3)	0.663	-0.3%

TABLE 3.5 – Impact de l'initialisation du décodeur sur l'inférence de profondeur.

sant sa carte de profondeur dans l'auto-encodeur appris à l'étape 1. Nous entraînons ensuite notre ResNet200 à s'aligner sur ces représentations latentes, c'est-à-dire à produire les mêmes descripteurs. Cet alignement est réalisé en minimisant une distance $\mathbb{L}_2$ directement entre les descripteurs.

Étape 3 : Affinage La troisième étape consiste à affiner la carte de profondeur obtenue en optimisant la tâche d'estimation finale via une procédure d'apprentissage globale. Nous utilisons la fonction de coût $\mathbb{L}_2$.

Nous comparons les résultats obtenus Table 3.5 où nous rappelons également la ligne correspondant à une initialisation entièrement aléatoire. Nous voyons tout d'abord que l'auto-encodeur de profondeur présente des performances qui n'ont rien à voir avec les estimations de profondeur à partir d'images $\mathcal{RGB}$. En effet, la tâche de reconstruction apprise avec un auto-encodeur est beaucoup plus simple et cela se reflète dans les résultats. Ensuite, nous avons procédé à l'apprentissage de notre architecture avec un ResNet200 initialisé aléatoirement et le décodeur initialisé par l'auto-encodeur de profondeur. Nous constatons que les performances obtenues sont les mêmes que celles avec une initialisation entièrement aléatoire du réseau. Cette conclusion est très parlante puisqu'elle montre bien que, tandis qu'un bon encodeur peut arriver à de bons résultats quelque soit le décodeur (nous venons de le voir), un décodeur pré entraîné seul ne suffit pas à produire de bonnes estimations. Dans ce cas l'initialisation du décodeur n'apporte rien. Voyons si notre méthode en 3 étapes affiche de meilleures performances.

A la fin de l'étape 2, on a un ResNet200 pré-entraîné sur *ImageNet* et dont les descripteurs de sortie sont alignés sur les descripteurs d'entrée du décodeur de l'auto-encodeur de profondeur. Nous prenons cet encodeur aligné et lui accolons le décodeur de l'étape 1. On voit que l'alignement s'est relativement bien passé puisque les performances affichées sont meilleures que celles avec l'encodeur initialisé aléatoirement. Cependant après l'étape 3 d'affinage on constate que, comme dans les cas d'initialisation aléatoire de l'encodeur, l'initialisation de notre décodeur n'a pas été utile. En effet, on retombe sur la même performance que pour l'entraînement de référence avec un décodeur initialisé aléatoirement. Ces résultats nous amènent à conclure que l'initialisation du décodeur par une tâche de reconstruction n'a pas d'influence sur la capacité

de généralisation du réseau.

3.3.3 Optimiseur et paramètres

Comme nous l'avons vu dans la section précédente, nous avons choisi jusqu'à présent d'entraîner nos réseaux avec une modalité spécifique. En particulier, nous utilisions d'abord l'optimiseur 'Adam' pour 100 époques puis une SGD pour 20 époques. Nous voulons ici mesurer l'importance du choix de l'optimiseur et montrons d'abord les résultats en utilisant un seul des deux pendant 120 époques. Dans un second temps, nous nous intéresserons à l'influence de la régularisation de type *weight decay* dans l'apprentissage (pénalisation $\mathbb{L}_2$ sur les poids du réseaux). Commençons d'abord par expliquer plus en détail ces différents composants.

Descente de gradient stochastique (SGD)

L'algorithme de descente de gradient stochastique (*Stochastic Gradient Descent* ou SGD) est une méthode itérative permettant de minimiser une fonction objectif différentiable. C'est une approximation stochastique de l'optimisation par descente de gradient qui permet de trouver le minimum d'une fonction en calculant son gradient au point courant et en faisant un pas dans la direction opposée, le tout de manière répétée. Soit un modèle paramétrique de paramètres $\theta \in \mathbb{R}^d$ et la fonction objectif associée $J(\theta)$. On note $\nabla_\theta J(\theta)$ le gradient de $J(\theta)$ en fonction des paramètres θ et η le pas d'apprentissage de l'algorithme d'optimisation. Le calcul de la mise à jour des poids pour l'optimisation par descente de gradient est le suivant :

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{3.1}$$

On peut ajouter un terme dit d'inertie appelé 'moment' à notre optimisation pour prendre en compte les gradients précédemment calculés et ainsi diminuer les oscillations et accélérer la convergence. L'idée consiste à mettre à jour nos poids en fonction non seulement du gradient actuel mais également en fonction d'une moyenne glissante des gradients précédents. Le moment est pondéré par un paramètre α contrôlant l'importance à donner aux gradients précédents par rapport au gradient actuel. La formule de mise à jour des poids devient alors la suivante :

$$v_t = \alpha v_{t-1} + (1 - \alpha)\eta \nabla_\theta J(\theta) \tag{3.2}$$

$$\theta = \theta - v_t \tag{3.3}$$

L'optimiseur Adam

La méthode de l'estimation adaptative du moment (*Adaptative Moment Estimation* ou Adam) (Kingma et Ba, 2014) est une méthode où le pas d'apprentissage est calculé automatiquement et de manière spécifique pour un jeu donné de paramètres. Pour ce faire, Adam utilise des estimations des moments d'ordre 1 et 2 du gradient pour adapter le pas d'apprentissage à chaque

Optimiseur	RMSE	Gain relatif
Adam (100) puis SGD (20)	0.661	-
SGD seul (120)	0.721	-9.1%
Adam seul (120)	0.718	-8.2%

TABLE 3.6 – Impact de la méthode d'optimisation sur l'inférence de profondeur

paramètre du réseau de neurones. En effet, en plus de stocker une moyenne des gradients passés comme pour le calcul du moment, Adam conserve également une moyenne des gradients au carré qu'il utilise aussi pour la mise à jour des poids. Soient m_t et v_t les estimations respectives des moments d'ordre 1 et 2 du gradient, pondérées par les paramètres β_1 et β_2 et définies par :

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.4}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.5}$$

où g_t est le gradient au pas de temps t (il y a eu t actualisations des paramètres du modèle), fonction des paramètres courants θ_t, définit par $g_t = \nabla_{\theta_t} J(\theta_t)$. Cependant Kingma et Ba (2014) observent que ces estimateurs sont biaisés autour de zéro et proposent à la place les estimateurs $\hat{m}$ et $\hat{v}$ suivants :

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{3.6}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{3.7}$$

Le modèle est ensuite mis à jour en utilisant le moment d'ordre 1 (moyenne) et celui d'ordre 2 (variance) des gradients pour normaliser ses paramètres :

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon} \tag{3.8}$$

Les auteurs proposent les valeurs de 0.9 pour β_1, 0.999 pour β_2 et 10^{-8} pour ϵ qui sont celles que nous utilisons par défaut.

Comparaison des méthodes d'optimisation

L'optimiseur est une composante importante de l'apprentissage. Nous comparons ici les deux méthodes d'optimisation que sont SGD, avec un moment de 0.9, et Adam. Dans la table 3.6, nous reportons nos résultats lorsque nous utilisons d'abord Adam pour 100 époques puis SGD pour 20 époques comme nous l'avions fait jusqu'à présent. Nous comparons ces résultats à l'utilisation d'une seule de ces deux méthodes d'optimisation pendant 120 époques. D'abord,

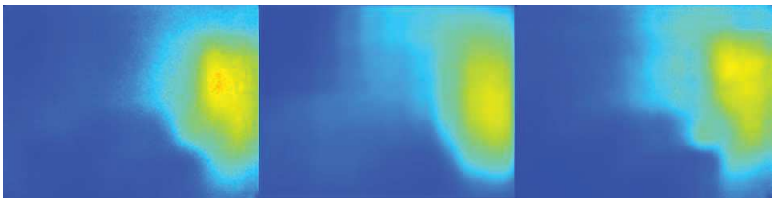


FIGURE 3.15 – Estimations de profondeur obtenues avec des stratégies d'optimisation différentes. De gauche à droite : utilisation des méthodes Adam puis SGD, utilisation de la méthode SGD seule, utilisation de la méthode Adam seule

nous constatons que les 2 méthodes d'optimisation appliquées seules affichent des performances similaires, avec un très léger avantage pour Adam. Nous voyons également qu'utiliser d'abord Adam puis appliquer une SGD est plus efficace puisque l'erreur obtenue est plus petite. C'est parce que souvent, par construction, Adam converge plus vite vers une solution que SGD. Il est donc utile de commencer les premières époques de son apprentissage avec cet optimiseur qui va plus rapidement trouver un bassin contenant un minimum local. Cependant, son côté adaptatif implique qu'il est plus instable qu'une descente de gradient avec moment et en pratique, pour un pas d'apprentissage fixe comme ici, on verra plus d'oscillations de l'erreur avec Adam. C'est pourquoi nous avons choisi de finaliser notre apprentissage avec une SGD pour les dernières époques, comme on le trouve parfois dans la littérature (Keskar et Socher, 2017).

Bien que les performances soient similaires, nous voyons sur Figure 3.15 que les estimations de profondeur sont bien différentes. En effet, on voit que l'estimation obtenue avec SGD uniquement est encore très grossière, tandis que celle obtenue avec Adam uniquement ressemble plus à la méthode de référence combinant les deux approches. Nous pensons que c'est parce que SGD descend plus lentement vers un minimum et que, malgré les performances similaires, on doit ici être beaucoup plus loin de la convergence que pour le cas où l'on utilise Adam. Les deux minimums trouvés par les deux optimiseurs sont donc bien différents et il est probable que, pour des durées d'entraînement beaucoup plus long, on trouve une meilleure solution avec SGD (Keskar et Socher, 2017). C'est pourquoi nous favorisons ici le compromis Adam puis SGD.

La régularisation de type *weight decay*

Le *weight decay* est une technique de régularisation utilisée pour éviter le surapprentissage dans les réseaux de neurones. Le principe est de contraindre la norme des poids du réseau à être petite en ajoutant un terme à la fonction de coût $\mathcal{L}$ à minimiser. En effet non contraints, les poids peuvent prendre de grandes valeurs et le modèle présente alors une forte variance ce qui rend les réseaux de neurones sujets au surapprentissage. L'ajout d'un terme de régularisation pour contraindre les poids à avoir une norme faible aide à diminuer la variance globale et améliore ainsi la capacité de généralisation du modèle (Geman et al., 1992). Lorsque ce terme est une régularisation $\mathbb{L}_2$ comme c'est le cas la plupart du temps, la fonction de coût totale

Weight decay	RMSE	Gain relatif
Weight decay = 5.10^{-4}	0.661	-
Weight decay = 5.10^{-3}	0.657	0.6%
Weight decay = 5.10^{-5}	0.667	-0.9%

TABLE 3.7 – Impact de la régularisation de type *weight decay* sur l'inférence de profondeur

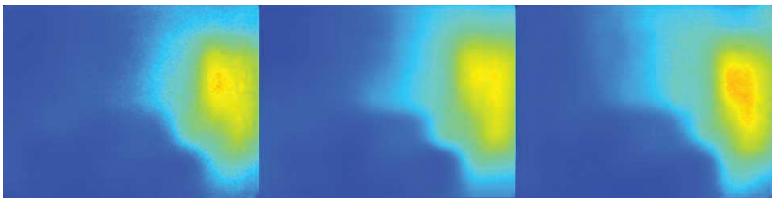


FIGURE 3.16 – Estimations de profondeur obtenues avec différentes valeurs de *weight decay*. De gauche à droite *weight decay*=5.10^{-4}, *weight decay*=5.10^{-3}, *weight decay*=5.10^{-5}

devient alors :

$$\mathcal{L}_{total} = \mathbb{L}_2 + \lambda \|W\|_2^2 \tag{3.9}$$

où W représente la matrice des poids du réseau et λ est le coefficient de pondération de la régularisation.

Nous évaluons ici 2 nouvelles valeurs de *weight decay* pour essayer d'observer son influence sur les performances. Notre *weight decay* était fixé à 5.10^{-4}, nous évaluons les valeurs de 5.10^{-3} et 5.10^{-5}. Les résultats sont présentés Table 3.7. Nous voyons que, en termes quantitatifs, la valeur du *weight decay* n'influence quasiment pas le score d'erreur que nous obtenons. D'un autre côté, nous constatons une grande différence visuelle. En effet, on voit Figure 3.16 que l'estimation de profondeur obtenue avec le *weight decay* le plus fort ne présente pas le grain d'image particulier que nous voyions jusqu'à présent. Cela confirme bien que l'ajout du décodeur basé sur des modules de projection augmente la capacité du réseau et donc sa variance globale, tandis qu'une valeur élevée de *weight decay* tend à contraindre davantage les poids et à diminuer la variance. Ainsi, l'estimation que nous observons présente une bien meilleure consistance spatiale. Cependant, puisque les performances quantitatives sont très similaires, nous conservons pour la suite un *weight decay* de 5.10^{-4} ce qui nous permettra de renforcer notre analyse visuelle, notamment en ce qui concerne le surapprentissage.

3.3.4 Influence de la fonction de coût

Laina et al. (2016) montrent qu'entraîner le réseau avec une fonction de coût différente peut améliorer davantage les résultats finaux. En particulier, ils proposent d'utiliser la fonction 'berHu' qui améliore un peu les performances en comparaison avec la $\mathbb{L}_2$ classique. La berHu (Owen, 2007; Zwald et Lambert-Lacroix, 2012) est définie par :

$$B(x) = \begin{cases} |x| & |x| \leq c, \\ \frac{x^2+c^2}{2c} & |x| > c. \end{cases} \tag{3.10}$$

Elle est égale à la norme L^1 lorsque $x \in [-c, c]$ et à la norme L^2 sinon. La variable c est calculée pour un lot d'images et définit la transition entre les normes.

Dans le même esprit, nous proposons de regarder ici l'influence de la fonction de coût choisie pour l'entraînement. Nous comparons la $\mathbb{L}_2$ standard avec une $\mathbb{L}_1$ et la berHu que nous venons de voir mais aussi avec l'entropie croisée utilisée pour la classification et avec une $\mathbb{L}_2$ pondérée que nous détaillerons par la suite.

Pour rappel, la $\mathbb{L}_2$ s'écrit comme suit :

$$\mathbb{L}_2 = \frac{1}{N} \sum_{i=1}^{N} (\hat{y_i} - y_i)^2 \tag{3.11}$$

La $\mathbb{L}_1$ est définie par :

$$\mathbb{L}_1 = \frac{1}{N} \sum_{i=1}^{N} |\hat{y_i} - y_i| \tag{3.12}$$

L'entropie croisée est une fonction utilisée pour la classification, et s'écrit :

$$- \sum_{i=1}^{C} y_i \log(p_i) \tag{3.13}$$

où C dénote le nombre de classes du problème, y_i est la probabilité de la vérité terrain et p_i est la probabilité prédite pour la classe i. Pour pouvoir appliquer l'entropie croisée, nous rendons discret notre problème de régression en définissant une classe tous les 5 centimètres de profondeur, soit 200 classes pour la base de données NYU Depth v2 dont les vérités terrain sont comprises entre 0 et 10m.

Enfin, nous proposons d'utiliser une MSE pondérée pour l'apprentissage. Partant du principe qu'une bonne carte de profondeur d'une scène implique une reconstruction 3D de la scène avec de bonnes propriétés géométriques, nous proposons d'exploiter la notion de changement de point de vue 3D d'une image et de vérifier si la reprojection dans l'espace 2D (image) se passe correctement. En effet, connaissant une image et sa carte de profondeur, nous sommes en mesure de projeter la scène comme un nuage de points et de simuler une nouvelle position de la caméra en 3D. Nous pouvons ensuite reconstruire une image $\mathcal{RGB}$ 2D de cette nouvelle vue. Nous appelons cette procédure la simulation de changement de point de vue (*Viewpoint Change Simulation* ou VCS). Nous pensons que si nous estimons une bonne carte de profondeur, le VCS doit être aussi proche que possible de celui calculé avec la vérité terrain de profondeur (contenant théoriquement des informations 3D parfaites).

Nous appelons $\tilde{I}$ une image $\mathcal{RGB}$ reconstruite en utilisant la procédure du VCS. Soit $\tilde{I}_gt$ l'image $\mathcal{RGB}$ reconstruite en utilisant la vérité terrain de profondeur et $\tilde{I}_pred$ l'image $\mathcal{RGB}$ reconstruite en utilisant l'estimation de profondeur. Pour obtenir la meilleure reconstruction $\mathcal{RGB}$ possible lorsque nous réalisons un VCS à partir d'une carte de profondeur estimée (donc imparfaite), nous souhaitons nous concentrer sur certaines parties de l'image critiques pour cette reconstruction, et nous souhaitons que l'estimation de profondeur soit aussi précise que possible sur ces parties. En particulier, nous pensons que prédire une profondeur précise sur des objets éloignés est en quelque sorte moins important que de se concentrer sur le premier plan, car un petit déplacement de la caméra aura un impact plus grand sur la position des objets au premier plan. De la même manière, nous pensons que nous serons généralement plus intéressés par la prédiction d'une profondeur précise sur les régions texturées, car les erreurs dans la profondeur des régions lisses n'auront aucune incidence sur la reconstruction $\mathcal{RGB}$. Nous proposons d'implanter ces hypothèses par les équations suivantes, dans le cas simplifié où le déplacement 3D est une translation.

Appelons $I(x, y)$ une image et $z(x, y)$ la carte de profondeur correspondante. Un petit changement de point de vue 3D donnera une image $\mathcal{RGB}$ reconstruite $\tilde{I}_gt$ en utilisant la vérité terrain de profondeur et $\tilde{I}_p$ en utilisant l' estimation de la profondeur (imparfaite).

$$\widetilde{I_{gt}}(x, y) = I(x + \varepsilon\frac{f}{z_{gt}(x, y)}, y)$$

où ε est le décalage effectué sur l'axe des x et f est la distance focale de la caméra considérée. Nous notons :

$$g(z) = \varepsilon\frac{f}{z_{gt}(x, y)}$$

et nous avons :

$$\widetilde{I_p}(x, y) = I(x + \varepsilon\frac{f}{z_p(x, y)}, y)$$

où $z_p(x, y) = z_{gt}(x, y) + \delta z$, δz étant la quantité d'erreur sur la profondeur prédite.

Pour une petite translation, nous pouvons calculer le développement limité d'ordre 1 autour $g(z)$ pour I_p. Nous avons donc

$$X = x + \varepsilon\frac{f}{z_p(x, y)}$$

et

$$X_0 = x + \varepsilon\frac{f}{z_{gt}(x, y)} = x + g(z)$$

Nous pouvons écrire :

$$\widetilde{I_p}(x, y) \approx I(X_0, y) + \frac{\delta}{\delta x}I(X_0, y).[X - X_0]$$

$$\widetilde{I}_p(x,y) \approx I(x+g(z),y) + \frac{\partial}{\partial x} I(x+g(z),y)[\frac{\varepsilon f}{z_p(x,y)} - \frac{\varepsilon f}{z_{gt}(x,y)}]$$

De plus, pour une petite erreur δz sur la carte de profondeur, nous pouvons calculer le développement limité d'ordre 1 de $\frac{1}{Z_p(x,y)}$ et nous pouvons réécrire :

$$\frac{1}{z_p(x,y)} = \frac{1}{z_{gt}(x,y) + \delta z}$$

$$\frac{1}{z_p(x,y)} = \frac{1}{z_{gt}(x,y)} . \frac{1}{1 + \frac{\delta z}{z_{gt}(x,y)}}$$

$$\frac{1}{z_p(x,y)} \approx \frac{1}{z_{gt}(x,y)} . (1 - \frac{\delta z}{z_{gt}(x,y)})$$

Finalement :

$$\widetilde{I}_p(x,y) \approx I(x+g(z),y) + \frac{\partial}{\partial x} I(x+g(z),y).[(\frac{\varepsilon f}{z_{gt}(x,y)}).(1 - \frac{\delta z}{z_{gt}(x,y)}) - \frac{\varepsilon f}{z_{gt}(x,y)}]$$

$$\widetilde{I}_p(x,y) \approx I(x+g(z),y) + \frac{\partial}{\partial x} I(x+g(z),y).[-\frac{\varepsilon f}{{z_{gt}}^2(x,y)} . \delta z]$$

Nous concluons que l'erreur de reconstruction résiduelle $R_y(x)$ étant donnée une petite translation le long de l'axe des x et une carte de profondeur imparfaite est, dans les coordonnées d'origine de l'image d'origine, donnée par :

$$R_X(x,y) = \frac{\partial I(x,y)}{\partial x} . \frac{\varepsilon f}{{z_{gt}}^2(x,y)}$$

Ceci est également valable pour une translation le long de l'axe des y, en utilisant le gradient de l'image sur y, on obtient :

$$R_Y(x,y) = \frac{\partial I(x,y)}{\partial y} . \frac{\varepsilon f}{{z_{gt}}^2(x,y)}$$

A partir de là, nous pouvons calculer une carte d'importance I_{wm} comme suit :

$$I_{wm}(x,y) = \log(1 + \|(R_X(x,y), R_Y(x,y))\|)$$

Cette carte d'importance capture les pixels critiques pour effectuer une bonne reconstruction $\mathcal{RGB}$ dans le cadre d'un VCS. Nous pouvons voir qu'il implique le gradient de l'image, le

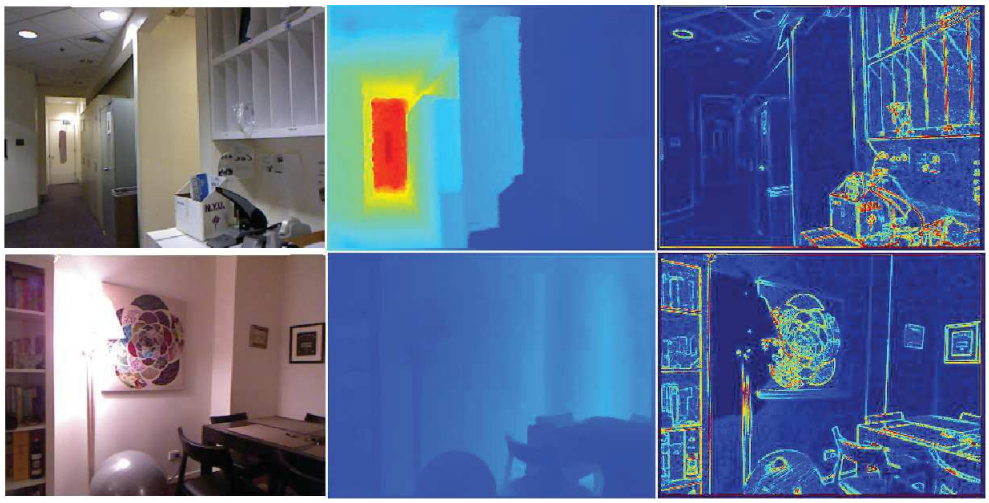


FIGURE 3.17 – Gauche : Image RGB originale. Milieu : Vérité terrain de profondeur. Droite : Carte d'importance calculée. Les valeurs, de faibles à élevées, vont du bleu au rouge.

Fonction de coût	RMSE	Gain relatif
$\mathbb{L}_2$	0.661	-
$\mathbb{L}_1$	0.654	+1.1%
berHu	0.656	+0.8%
Cross Entropy	0.733	-10.9%
WMSE	0.656	+0.8%

TABLE 3.8 – Impact de la fonction de coût sur l'inférence de la profondeur

décalage 3D considéré, l'erreur commise sur la carte de profondeur prédite ainsi que de la vérité terrain. Pour chaque carte, nous étirons l'histogramme et normalisons les valeurs pour qu'elles soient comprises entre 1 et 10 afin de mettre davantage en évidence les régions d'importance (voir Figure 3.17).

Notre objectif est d'utiliser ces cartes comme pondération par pixel afin de donner plus d'importance aux régions critiques pour la reconstruction de l'image, qui semblent effectivement être les gradients proches dans l'image et les objets fortement texturés. Nous calculons ces cartes pour chaque image d'entraînement et nous apprenons un réseau à l'aide d'une MSE pondérée, noté WMSE :

$$WMSE = \frac{1}{n}\sum_{i=1}^{n}(w_i * (x_i - yi))^2$$

où w_i est un scalaire entre 1 et 10, correspondant à l'importance du pixel i pour le VCS.

Nous utilisons donc ces 5 fonctions de coût différentes pour entraîner notre architecture à la tâche d'estimation de profondeur. Comme à chaque fois, nous évaluons les performances

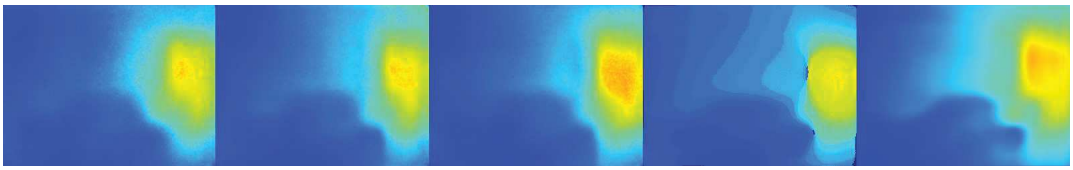


FIGURE 3.18 – Estimations de profondeur obtenues pour des apprentissages avec différentes fonctions de coût. De gauche à droite : $\mathbb{L}_2$, $\mathbb{L}_1$, berHu, entropie croisée, WMSE

de ces réseaux sur le jeu de test de NYU Depth v2 en calculant leurs RMSE. Les résultats sont reportés Table 3.8. D'abord, nous pouvons constater que, excepté l'entropie croisée, les différentes fonctions de coût utilisée n'ont qu'un impact mineur sur les performances quantitatives obtenues. Elles ont toutes tendance à améliorer les résultats mais l'écart n'est pas très significatif. Nous constatons en particulier que l'utilisation de la fonction berHu améliore effectivement les performances, comme le soutiennent Laina et al. (2016) mais cette amélioration est faible (moins de 1%) et elle est même un peu plus prononcée pour l'utilisation d'une $\mathbb{L}_1$ simple. D'un autre côté, notre procédure visant à pondérer l'apprentissage en insistant plus sur les régions importantes pour une bonne reconstruction 3D après un changement de point de vue (WMSE) ne donne pas non plus de résultats très concluants en terme d'erreur moyenne sur le jeu de test.

En ce qui concerne l'utilisation de l'entropie croisée, et donc le passage d'une tâche de régression à une tâche de classification, on remarque une importante baisse des performances. C'est dû au fait que la discrétisation du problème induit forcément des erreurs en fonction d'une métrique d'évaluation de régression telle que la RMSE. En effet, tandis que pour une tâche de régression on peut prédire une valeur infiniment proche de la vérité terrain (à la précision numérique du calcul près), la discrétisation du problème ne permet pas cette finesse. Soit on prédit la bonne classe de profondeur, soit on commet une erreur et on prédit au mieux la classe voisine, qui est ici à un écart de 5cm de la bonne classe. De plus, même une prédiction de la bonne classe produira une profondeur arrondie à 5cm près et donnera lieu à de petites erreurs lors de l'évaluation selon la RMSE. Cette accumulation d'erreurs dues aux arrondis induits par la discrétisation du problème fait qu'il est difficile de comparer les résultats qualitatifs obtenus pour un entraînement avec une entropie croisée. Cependant nous pouvons nous intéresser aux qualités des estimations de profondeur obtenues pour ces différents apprentissages.

La Figure 3.18 illustre les différentes prédictions que l'on obtient pour notre image de référence. On remarque que l'apprentissage avec la berHu produit une carte de profondeur qui a l'air très similaire à celle obtenue avec la $\mathbb{L}_2$, avec peut être certains contours un peu plus fins comme pour la démarcation entre le mur de gauche et l'arrière-plan. C'est probablement dû à la partie $\mathbb{L}_1$ de la berHu qui permet de produire des estimations plus fines. En effet, on voit que l'apprentissage avec une $\mathbb{L}_1$ est visuellement plus saillant. C'est dû au fait que les très petites erreurs sont pénalisées de la même manière en utilisant une $\mathbb{L}_1$ alors que leur impact est grandement atténué avec une $\mathbb{L}_2$. On constate que l'apprentissage avec l'entropie

Nombre d'images	RMSE	Gain relatif
795 images	0.661	-
2K images	0.661	+0%
4K images	0.638	+3.5%
8K images	0.625	+5.5%
16K images	0.617	+6.7%

TABLE 3.9 – Impact du nombre d'images sur l'inférence de la profondeur

croisée produit une carte de profondeur à niveaux, correspondant à des tranches de l'image que l'algorithme a apprises et qu'il suppose appartenir au même plan. Bien que les résultats quantitatifs soient moins bons pour cette méthode et que l'estimation obtenue soit particulière, on constate que cette dernière présente les mêmes qualités que les autres estimations : au global le meuble et le mur de gauche sont bien prédits, l'arrière-plan est bien démarqué et le dégradé de distance semble respecté. Enfin, l'estimation obtenue avec notre WMSE est la plus fine d'entre toutes, avec notamment des contours d'objets plus marqués (les angles du meuble en bois), ce qui correspond effectivement à un des types de régions d'importance accentué pendant l'apprentissage.

3.3.5 Nombre d'images d'entraînement

Nous nous intéressons ici à l'impact du nombre d'images utilisées pendant l'apprentissage pour les capacités de généralisation du réseau. Pour ce faire, nous utilisons le jeu de données brut de NYU Depth v2 au sein duquel nous avons aléatoirement sélectionné des images équiréparties. Ainsi, nous formons plusieurs jeux d'entraînement de différentes tailles. Nous choisissons d'extraire des jeux de 2000, 4000, 8000 et 16000 images et comparons les résultats obtenus avec un apprentissage de référence sur les 795 images d'entraînement officielles de NYU Depth v2. Nous reportons les résultats Table 3.9. Sans surprise, nous constatons que les performances obtenues s'améliorent au fur et à mesure que le nombre d'images utilisées pour l'apprentissage augmente. En effet dans le cas de l'apprentissage statistique, bien que le jeu d'entraînement ne représente qu'une petite fraction des possibles (des scènes existantes dans notre cas), nous comptons sur le fait qu'il soit assez varié pour couvrir convenablement l'espace des possibles et représenter aux mieux l'ensemble des données. De manière générale avec plus d'images d'entraînement, la distribution des données de l'espace est mieux représentée et on fournit au réseau des scènes présentant une plus grande variabilité ce qui lui permet de généraliser plus facilement. Nous constatons également qu'il n'y a pas de différence de résultat pour les entraînements à 800 et à 2K images, le gain n'est observé qu'à partir de 4K images. C'est certainement dû au fait que nous utilisons de l'augmentation de données en ligne durant l'apprentissage qui pallie, dans une certaine mesure, au manque de données. Nous y reviendrons plus en détail dans la section suivante.

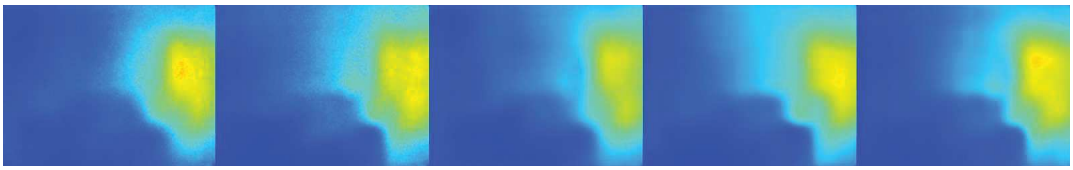


FIGURE 3.19 – Estimations de profondeur obtenues pour des apprentissages avec des jeux d'entraînement de tailles différentes. De gauche à droite : 795 images, 2K images, 4K images, 8K images et 16K images d'entraînement

Augmentation de données	RMSE	Gain relatif
795 images, avec augmentation	0.661	-
795 images, sans augmentation	0.676	-2.3%
16K images avec augmentation	0.617	-
16K images sans augmentation	0.623	-1.0%

TABLE 3.10 – Impact de l'augmentation de données sur l'inférence de profondeur

Les différentes cartes de profondeur prédites sont présentées en figure 3.19. Comme nous l'attendions aux vues des résultats quantitatifs obtenus, nous constatons que les prédictions sont plus fines à mesure que le nombre d'images d'entraînement augmente. De plus, on peut noter la disparition progressive du grain particulier d'image que nous avons identifié précédemment comme étant dû à une forme de surapprentissage : le modèle était trop complexe pour le nombre de données sur lesquelles il était entraîné. Ici, nous constatons que pour 4K images d'entraînement la carte de profondeur prédite ne présente quasiment plus de bruit et qu'il disparaît même complètement à partir de 8K images d'entraînement. Le modèle est donc moins sujet au surapprentissage puisque le nombre de données a augmenté.

3.3.6 Augmentation de données

Enfin, nous nous penchons sur l'impact qu'à l'augmentation de données en ligne pendant l'entraînement. Pour ce faire, nous proposons de comparer deux apprentissages différents, utilisant respectivement 795 et 16K images d'entraînement et nous observons les performances obtenues avec et sans application de l'augmentation de données. Les résultats sont présentés Table 3.10. On voit d'abord que l'utilisation de l'augmentation de données pendant l'apprentissage améliore dans les deux cas les performances. On constate également que cette amélioration est plus marquée pour un petit jeu d'entraînement (795 images) que pour un apprentissage utilisant beaucoup plus de données (16K images). C'est logique puisque l'augmentation de données est utilisée pour augmenter artificiellement le jeu d'entraînement à disposition en présentant au réseau des versions légèrement altérées de chaque image. Le réseau gagne alors en robustesse et a accès à des données plus variées (en fonction de l'augmentation appliquée). Son intérêt est donc amoindri lorsque l'on a à disposition un grand nombre d'images d'entraînement qui

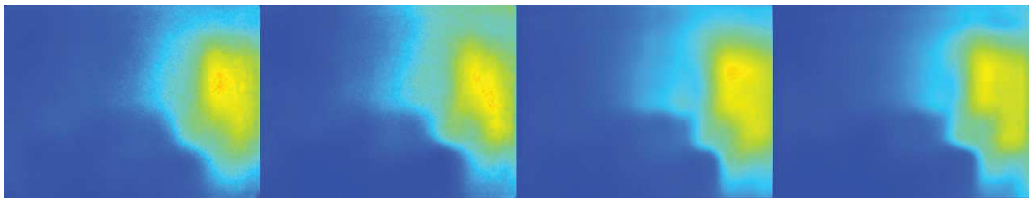


FIGURE 3.20 – Estimations de profondeur obtenues pour des apprentissages avec et sans augmentation de données. De gauche à droite : 795 images d'entraînement avec augmentation, 795 images d'entraînement sans augmentation, 16K images d'entraînement avec augmentation, 16K images d'entraînement sans augmentation

présente naturellement plus de diversité. Cela dit, nous pensons qu'il y a encore à gagner à utiliser de l'augmentation de données sur 16K images parce que les images sélectionnées pour l'apprentissage étant équiréparties, elles ne sont pas voisines dans le temps et ne présentent donc pas forcément les petits changements simulés par notre augmentation de données en ligne (notamment la rotation, la translation, la mise à l'échelle et la couleur).

Les résultats visuels sont présentés Figure 3.20. Ici, comme les résultats quantitatifs le laissaient penser en raison de faibles écarts de performances, nous constatons que les estimations de profondeur sont très semblables avec et sans l'utilisation de l'augmentation de données. Le réseau que nous utilisons étant nativement puissant, avec notamment un encodeur à l'état de l'art, les descripteurs extraits de l'image sont déjà très pertinents pour notre tâche. De plus, l'encodeur étant préentraîné pour des tâches sémantiques, il a déjà vu un grand nombre d'images $\mathcal{RGB}$. C'est pourquoi l'augmentation de données n'a qu'un faible impact pour notre configuration, même sur peu d'images d'entraînement.

3.4 Conclusions

Dans ce chapitre, nous avons passé en revue un certain nombre d'hyperparamètres impliqués dans l'apprentissage d'un réseau de neurones profond pour l'inférence de profondeur. Nous avons proposé un plan d'expérience comparatif visant à comprendre l'influence de ces hyperparamètres sur la capacité de généralisation d'un réseau entraîné à l'estimation de profondeur monoculaire. Bien que nous ayons observé que chacun des hyperparamètres étudiés avait une influence non nulle sur les performances obtenues, tant sur l'erreur moyenne que sur la qualité visuelle des estimations, nous avons pu en identifier certains comme étant critiques pour obtenir une bonne prédiction.

En effet, plusieurs expériences nous ont montré l'importance primordiale qu'avaient les descripteurs extraits par l'encodeur depuis les images $\mathcal{RGB}$. En particulier, nous avons vu que les meilleurs résultats étaient obtenus par des réseaux dont l'encodeur avait une architecture plus élaborée et plus profonde, avec donc un pouvoir de représentation plus grand. Par ailleurs, nos expériences ont clairement montré que ces encodeurs étaient d'autant plus utiles qu'ils

sont pré entraînés sur des tâches sémantiques, les caractéristiques sémantiques d'une image permettant de trouver une bonne partie des indices monoculaires qu'elle renferme. D'un autre côté étant donné un encodeur si puissant, nous avons observé que l'architecture et l'initialisation du décodeur, elles, n'étaient pas de première importance. D'autre part, nous avons vu que le nombre des données utilisées pour l'apprentissage revêtait aussi une importance particulière puisque les résultats constatés sont directement corrélés au nombre d'images d'entraînement et à l'augmentation de données. Enfin, nous avons observé que les fonctions de coût étudiées influençaient surtout la qualité visuelle des estimations avec un niveau de détails et de finesse variables, pour des erreurs moyennes globales relativement équivalentes.

Chapitre 4

Analyse multi-échelle

Ce chapitre étudie le rôle de l'utilisation d'indices multiéchelles pour l'estimation de profondeur à partir d'images monoculaires. À notre connaissance, au moment où nous avons fait cette étude, il n'existait pas de travaux dans la littérature qui se penchaient spécifiquement sur cette question.

Une des difficultés avec les réseaux de neurones utilisés pour l'inférence de profondeur est que la carte de profondeur inférée doit idéalement avoir la même résolution que l'image d'entrée. Or les réseaux profonds ne donnent de bons résultats que lorsque les architectures réduisent l'information spatiale pour obtenir une information de haut niveau sémantique (plus riche, mais moins résolue). Pour inférer la profondeur à la même résolution que l'image de départ, il faut alors augmenter la résolution de cette information de haut niveau (architecture en sablier). C'est dans ce contexte qu'il nous paraît pertinent de combiner les représentations à différentes échelles, d'où le nom d'approches multiéchelles, pour obtenir une inférence de profondeur spatialement plus précise.

Inspirés par le domaine de la segmentation sémantique, nous proposons dans ce chapitre 4 CNN d'architectures différentes permettant d'utiliser explicitement de telles caractéristiques multiéchelles. Nous introduisons le concept de *champ réceptif de réseaux de neurones* et expliquons en quoi les descripteurs des architectures proposées, calculés à des champs réceptifs variables, intègrent des informations contextuelles plus ou moins larges de l'image, et permettent ainsi une analyse multiéchelles de l'image.

La validation expérimentale et faite en deux temps, sur la base de données NYU Depth v2. Nous commençons dans un premier temps par utiliser un jeu d'entraînement limité et étudions l'apport de l'information multiéchelles dans ce cas. Nous comparons les différentes architectures à une approche monoéchelle classique, dont les performances sont au niveau de l'état de l'art. Nous montrons que toutes les méthodes multiéchelles obtiennent de meilleures performances que la méthode monoéchelle classique. Cette étude comparative nous permet également de déterminer quelle est la méthode multiéchelles la plus performante parmi les 4 proposées.

Dans un second temps, nous utilisons l'architecture choisie précédemment pour calculer les performances quantitatives et qualitatives avec un jeu d'entraînement plus conséquent. Pour la plupart des métriques d'évaluation usuelles, nous montrons que la méthode multiéchelle que nous avons sélectionnée présente des performances au-dessus de celles de l'état de l'art ainsi que des estimations de profondeur plus fines, de meilleures qualités visuelles.

4.1 Introduction

Comme nous l'avons vu, l'estimation de l'information de profondeur à partir d'images monoculaires a une très longue histoire dans la littérature de la vision par ordinateur. La piste

FIGURE 4.1 – Exemple d'inférence de profondeur monoculaire au niveau de l'état de l'art, entraînée avec un CNN multiéchelles.

la plus étudiée est celle de la stéréovision (Scharstein et al., 2001). Comme il est mentionné par Michels et al. (2005), de nombreux chercheurs ont également été inspirés par la manière dont les humains utilisent les indices monoculaires (par exemple la texture, la perspective, le defocus) pour estimer la profondeur (Wu et Ooi, 2004; Bulthoff et al., 1998).

Les CNN permettent d'apprendre des espaces de représentation très performants et constituent désormais une des approches dominantes dans ce domaine. Malgré cela, aucun travail n'avait encore explicitement examiné l'importance de l'information multiéchelle sur la précision de l'estimation de la profondeur. Une telle information multiéchelle peut être apportée en utilisant des descripteurs calculés à partir de contextes plus ou moins larges de l'image d'entrée.

En effet, dans le cas d'un neurone sensoriel, le champ réceptif correspond à la région de l'espace dans laquelle la présence d'un stimulus sera responsable de la modification de la réponse de ce neurone. C'est le même principe dans le cas du neurone formel où le champ réceptif d'un neurone correspond à la portion de l'entrée à laquelle ce neurone est sensible. Plus généralement dans les réseaux de neurones convolutifs, on parlera de champ réceptif d'un descripteur en particulier pour caractériser le nombre de pixels de l'image d'entrée qui affectent le calcul de ce descripteur. Le champ réceptif d'un CNN est une notion importante et on cherche en général à l'augmenter pour que le calcul des descripteurs de haut niveau sémantique soit effectué à partir d'un maximum de contexte de l'image. Pour ce faire, il existe plusieurs techniques comme l'augmentation du nombre de couches du réseau ou le sous-échantillonnage des descripteurs. Nous y reviendrons par la suite.

La suite de ce chapitre est structurée de la manière suivante : la section 4.2 présente les travaux de l'état de l'art issus de la segmentation sémantique ayant des liens avec la question de l'utilisation de descripteurs multiéchelles. La section 4.3 explicite les différentes architectures multiéchelles que nous avons proposées. Dans la section 4.4, nous comparons les méthodes proposées entre elles et avec une architecture monoéchelle dont les performances sont au niveau de l'état de l'art. Nous tirons enfin des conclusions concernant l'importance d'indices multiéchelles pour l'estimation de profondeur en fin de chapitre.

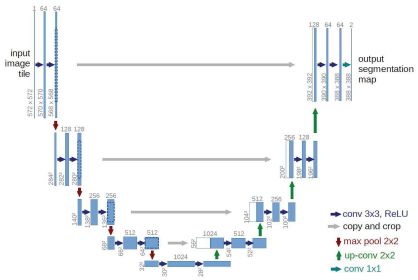


FIGURE 4.2 – L'architecture de type *U-Net* proposée par Ronneberger et al. (2015)

4.2 Littérature associée

Au moment où nous écrivons ces lignes, les architectures CNN donnant les meilleures performances pour l'estimation de profondeur sont de type *encodeur-décodeur*, et sont dotées d'un puissant encodeur préentraîné pour des tâches de classification ou de segmentation sémantiques. Par exemple, Laina et al. (2016) utilisent comme extracteur de caractéristiques un ResNet dont les dernières couches (initialement prévues pour la classification) sont remplacées par un décodeur. Ils entraînent le réseau résultant pour réaliser la tâche d'inférence de profondeur. Il proposent des blocs de suréchanntillonnage avec des *skip connections*, basés sur le principe du réseau ResNet (He et al., 2016) avec des *skip connections*, et montrent qu'entraîner le réseau avec une fonction de coût adaptée peut améliorer encore davantage les performances. En particulier, ils proposent d'utiliser une fonction de coût de type *berHu* Owen (2007); Zwald et Lambert-Lacroix (2012), définie par :

$$B(x) = \begin{cases} |x| & |x| \leq c, \\ \frac{x^2+c^2}{2c} & |x| > c. \end{cases} \tag{4.1}$$

Cette fonction de coût est égale à la norme L^1 lorsque $x \in [-c, c]$ et à la norme L^2 sinon. La variable c est calculée par batch et définit la transition entre les normes. Laina et al. (2016) instaurent ainsi un nouveau l'état de l'art en termes de précision de l'estimation de profondeur à partir d'images monoculaires. Cependant, bien que leurs travaux donnent des résultats impressionnants, ils n'intègrent pas d'analyse multiéchelle.

Les architectures de CNN utilisant des descripteurs multiéchelles ont été en revanche étudiées dans d'autres domaines tels que la détection et la reconnaissance d'objets ou encore la segmentation sémantique. Le U-net Ronneberger et al. (2015) en est un exemple. C'est une architecture encodeur-décodeur populaire où des *skip connections* permettent au décodeur de bénéficier des descripteurs miroir de l'encodeur, l'aidant à considérer des détails plus fins de l'image (voir figure 4.2). En effet, les descripteurs extraits à des endroits moins profonds du réseau résultent de convolutions avec un champ réceptif plus petit sur l'entrée. Elles mettent donc en valeur des indices plus locaux de l'image, par exemple des gradients et leurs orientations.

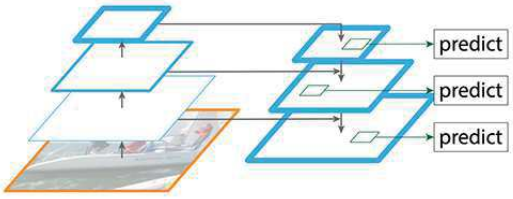


FIGURE 4.3 – L'architecture du réseau *Feature Pyramid Network* proposé par Lin et al. (2017a)

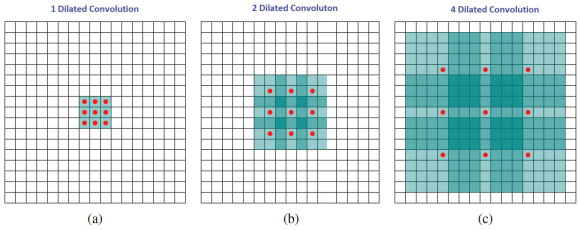


FIGURE 4.4 – Illustration de la convolution dilatée.

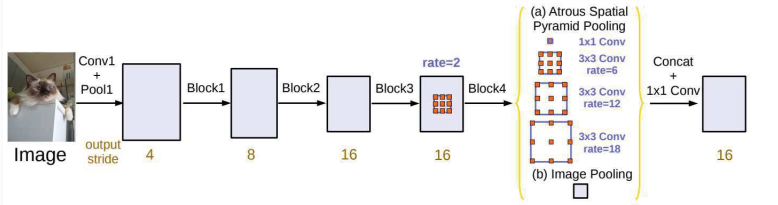


FIGURE 4.5 – L'architecture du réseau *DeepLab v3*, proposée par Chen et al. (2017) et utilisant un bloc ASPP.

De leur côté, Lin et al. (2017a) soutiennent qu'en plus des *skip connections*, il est utile d'avoir des sorties intermédiaires dans le décodeur à des résolutions plus basses, construisant ainsi des descripteurs avec de hauts niveaux d'abstraction sémantique. À chacune de ces sorties est associée une fonction de coût à minimiser ce qui permet d'apprendre à reconstruire les images à différentes échelles (voir figure 4.3). Ils montrent que cela permet d'améliorer à la fois les tâches de détection d'objets et de segmentation sémantique.

Par ailleurs, il existe un type de convolution, dont le noyau est dit *dilaté*, qui a été introduit pour éviter le sous-échantillonnage dans les réseaux, tout en préservant un champ réceptif de même taille. Ces convolutions, connues sous le nom de convolution à trous ou convolutions dilatées, sont des convolutions dont le noyau est plus grand, mais est rendu parcimonieux (voir figure 4.4) pour limiter le nombre de paramètres. Ainsi, pour un nombre de paramètres fixés, la convolution dilatée a un champ réceptif plus large Yu et Koltun (2015). On appelle coefficient de dilatation le taux de parcimonie induit dans ce cas. Une convolution dilatée avec un taux de dilatation de 1 est équivalente à une convolution classique, tandis qu'un taux de dilatation de 2 signifie que l'on ne considère qu'un pixel sur deux de l'entrée, et ainsi de suite.

Ces couches de convolution dilatée ont été exploitées pour la segmentation sémantique

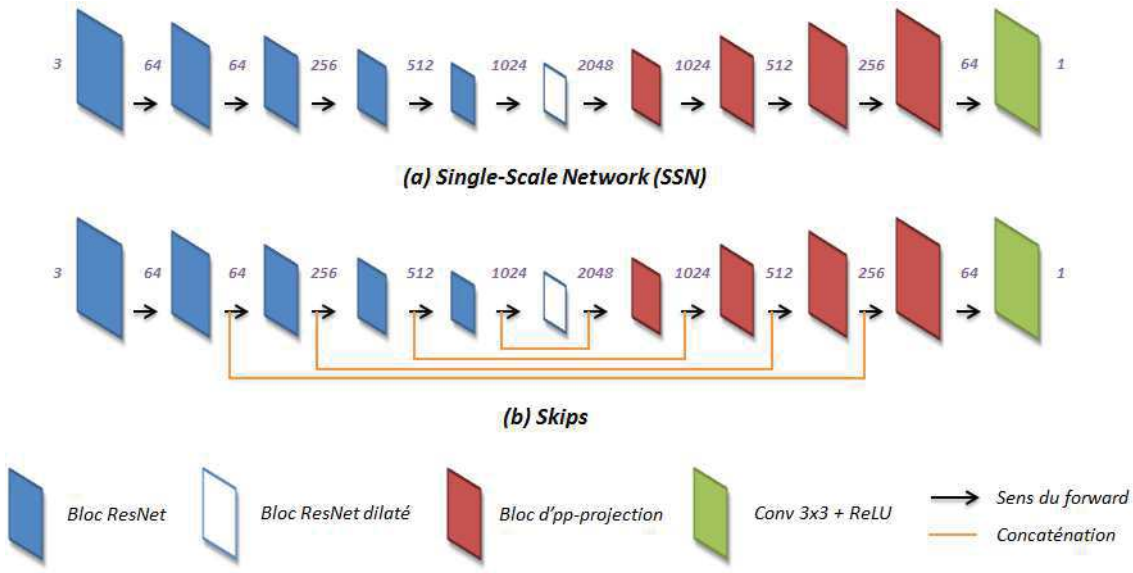


FIGURE 4.6 – Schéma des architectures SSN et Skips.

dans les travaux récents de Chen et al. (2017). Chen et al. (2017) proposent une extraction de caractéristiques sous la forme d'une pyramide spatiale de convolutions à différentes échelles, nommée *Atrous Spatial Pyramid Pooling (ASPP)* (voir figure 4.5). Leur modèle surpasse les meilleurs modèles de segmentation sémantique existant. Les auteurs notent que l'amélioration provient de la façon dont la pyramide de convolutions dilatées permet d'encoder le contexte multiéchelle dans l'image.

4.3 Architectures multiéchelles pour l'estimation de profondeur

Nous proposons ici 4 architectures inspirées de travaux récents de l'état de l'art mentionnés ci-dessus, notamment issues du domaine de la segmentation sémantique, et adapté pour l'inférence de profondeur. Chacune de ces 4 architectures est construite sur la base d'une architecture monoéchelle soigneusement optimisée pour l'estimation de profondeur, que nous présentons au préalable. Les architectures de ces 5 CNN sont illustrées et décrites ci-après.

4.3.1 Architecture dite *Single Scale Network (SSN)*

Ce réseau monoéchelle a une architecture similaire aux réseaux type "sablier", combinant un encodeur et un décodeur (voir figure 4.6 (a)). Nous utilisons le ResNet200 He et al. (2016) comme extracteur de caractéristiques, préentraîné sur une tâche de classification sur la base de données ImageNet. Les deux dernières couches, initialement dédiées à la classification, sont retirées. De plus, nous modifions les convolutions standards du dernier bloc du ResNet pour les remplacer par des convolutions dilatées avec un coefficient de dilatation de 2. Dans ce cas, les descripteurs

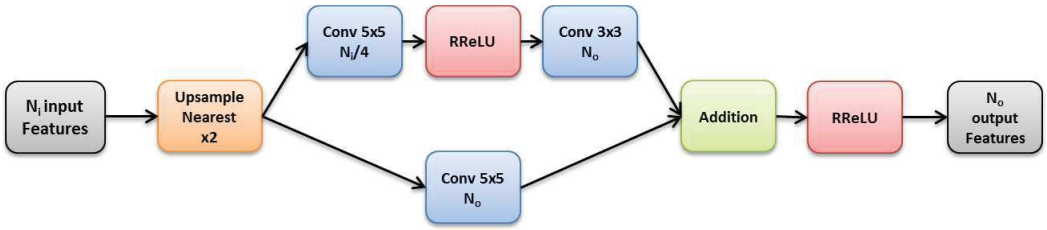


FIGURE 4.7 – Notre module suréchantilonnage modifié. Contrairement à l'original proposé dans (Laina et al., 2016), nous utilisons un suréchantilonnage de type "plus proche voisin" ainsi que des ReLU stochastiques.

sortant de l'encodeur sont de résolutions spatiales plus grandes, tout en maintenant le même champ réceptif du réseau. Ce type de stratégie permet d'alléger le décodeur qui aura besoin d'une couche de suréchantillonnage en moins pour recouvrer la résolution spatiale de l'entrée. Il est généralement désigné par le nom *dilated ResNet* (Yu et al., 2017). Ainsi, notre ResNet200 encode 2048 descripteurs à 1/16 de la résolution spatiale des images d'entrée.

Pour la partie décodeur, nous utilisons une version légèrement modifiée du module de suréchantilonnage initialement proposé par Laina et al. (2016). Dans notre version (voir figure 4.7), nous choisissons d'utiliser un suréchantillonnage par plus proche voisin à la place de la couche d'"unpooling" proposée parce que, en pratique, nous l'avons trouvée plus rapide et aussi performante. De plus, étant donné que nous comptons mener des expériences à la fois sur de grandes bases de données, mais également sur de plus petits jeux d'entraînement, nous utilisons des ReLU stochastiques (Xu et al., 2015) comme fonction d'activation à la place des ReLU originales. Nous avons constaté que cela permet de mieux généraliser lorsque nous apprenons sur des jeux d'entraînement de petite échelle (typiquement moins de 800 images). Des résultats similaires ont été observés dans (Xu et al., 2015), qui a noté l'efficacité des randomized ReLU pour lutter contre le surapprentissage sur des petits jeux d'entraînement. À la toute fin du réseau, nous utilisons une convolution 3x3 avec un seul canal de sortie, suivi par une activation de type ReLU. La sortie de cette dernière couche est notre estimation de profondeur finale.

4.3.2 Architecture à base de *Skip Connections (Skips)*

Le premier réseau multiéchelle que nous proposons consiste à considérer différents niveaux de contexte pour la reconstruction de la carte de profondeur. Elle est inspirée par le réseau *U-Net* (Ronneberger et al., 2015), qui est une architecture performante initialement conçue pour la segmentation d'images biomédicales. Comme notre réseau de référence, le U-Net est composé d'une réduction de dimension spatiale qui constitue la partie encodeur, puis d'une expansion des descripteurs pour recouvrer la résolution spatiale pour la partie décodeur. En plus, des *skip connections* sont ajoutées entre l'encodeur et le décodeur pour fusionner (par concaténation) les descripteurs miroirs de même résolution spatiale. Nous suivons également ce procédé et utilisons des *skip connections* entre les couches de même résolution spatiale dans l'encodeur et

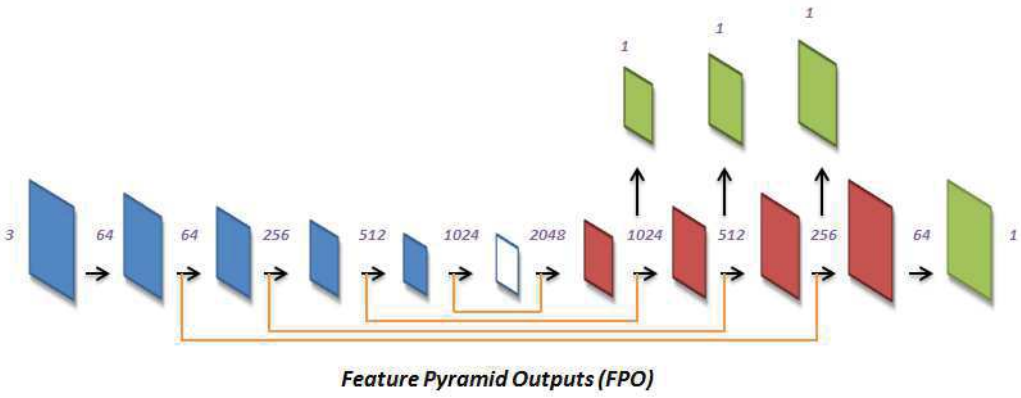


FIGURE 4.8 – Schéma de l'architecture FPO.

dans le décodeur. Ces *skip connections* nous permettent de fusionner (par concaténation) les descripteurs de l'encodeur avec leurs homologues de même taille dans le décodeur. Pendant la phase de décodage, on considère ainsi non seulement les descripteurs courants, de haut niveau sémantique, mais également des descripteurs de bas niveau de l'encodeur, proches de l'entrée et donc calculés avec un champ réceptif beaucoup plus restreint (voir Figure 4.6 (b)).

4.3.3 Architecture dite *Feature Pyramid Outputs (FPO)*

Cette architecture est inspirée des travaux de Lin et al. (2017a). Ils proposent pour améliorer la détection de petits objets d'apprendre une pyramide de caractéristiques à différentes échelles, tout en prédisant des sorties à différentes résolutions. C'est en somme la même architecture que celle utilisée dans *Skips*, mais avec des sorties intermédiaires supplémentaires après chacun des blocs d'up-projection. Ces sorties additionnelles sont conçues comme des branches indépendantes du réseau et chacune d'elle contient une convolution 3x3 suivie d'une ReLU pour permettre de produire des estimations de profondeur à différentes échelles (voir figure 4.8). Ainsi, à chaque sortie intermédiaire correspond un coût de régression dont la vérité terrain est la carte de profondeur désirée, sous échantillonnée à l'échelle correspondante. On peut alors entraîner le réseau soit de bout en bout, en minimisant la somme des différentes pertes, soit séquentiellement en considérant les résolutions intermédiaires séparément. En pratique, nous avons obtenu des résultats légèrement meilleurs en entraînant chaque résolution intermédiaire de manière séquentielle, en commençant par la résolution la plus faible et en fixant la partie précédente déjà entraînée du réseau pour apprendre la résolution suivante.

4.3.4 Architecture dite *Multi-Scale Middle Layer (MSML)*

Contrairement aux architectures précédentes, nous nous sommes interrogés sur l'extraction d'informations multiéchelles uniquement dans l'encodeur, avec un décodeur classique auquel on fournirait une seule fois un ensemble de descripteurs multiéchelles. Pour ce faire, nous

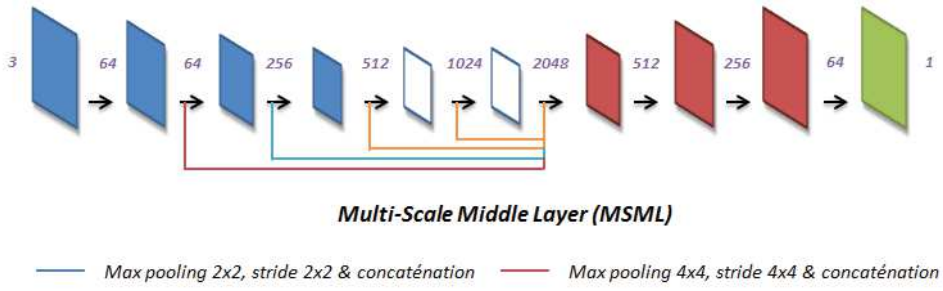


FIGURE 4.9 – Schéma de l'architecture MSML.

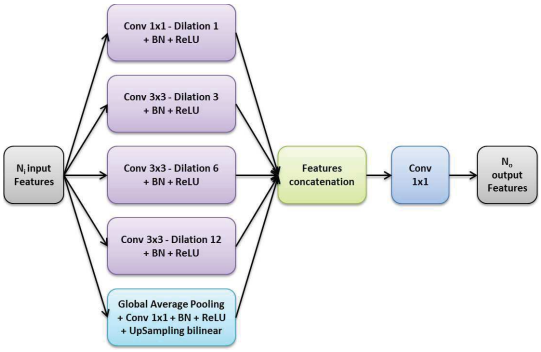


FIGURE 4.10 – Le bloc 'ASPP' (adapté de (Chen et al., 2017)), avec une pyramide de 1, 3, 6 et 12 convolutions dilatées. BN dénote la 'Batch Normalization'.

utilisons également un encodeur basé sur le *Dilated ResNet*. Cependant cette fois ce sont les deux derniers blocs de ResNet dont les convolutions sont remplacées par des convolutions dilatées avec des coefficients de dilatation de 2 et 4. Ainsi, nous conservons une résolution spatiale plus grande dans la couche centrale du réseau, dont la taille devient 1/8 de celle de l'image d'entrée. Puis, nous concaténons tous les descripteurs de sortie de chaque bloc de l'encodeur au niveau de la couche centrale, avant d'y accoler à la suite le décodeur. Pour cela, les premiers descripteurs qui ne sont pas à la même résolution spatiale sont d'abord passés par des couches de max pooling pour les sous-échantillonner avant la concaténation (voir figure 4.9). Cette architecture est donc composée d'un encodeur qui extrait des caractéristiques à l'aide d'informations multiéchelles provenant de différents endroits du réseau, suivi d'un décodeur classique considérant en entrée un ensemble de caractéristiques multiéchelles, mais sans réinjection d'information de l'encodeur à l'intérieur.

4.3.5 Architecture dite *Dilated Spatial Pyramid (DSP)*

De même que précédemment, cette architecture est composée d'un encodeur multiéchelle et d'un décodeur. Ce réseau s'appuie sur un bloc nommé *Atrous Spatial Pyramid Pooling (ASPP)*. Introduit par Chen et al. (2017), ce bloc permet de calculer des descripteurs à différents niveaux d'abstraction grâce à sa pyramide de convolutions dilatées à différents taux de dilatation. Cette pyramide permet d'obtenir des descripteurs calculés avec des champs réceptifs variables et

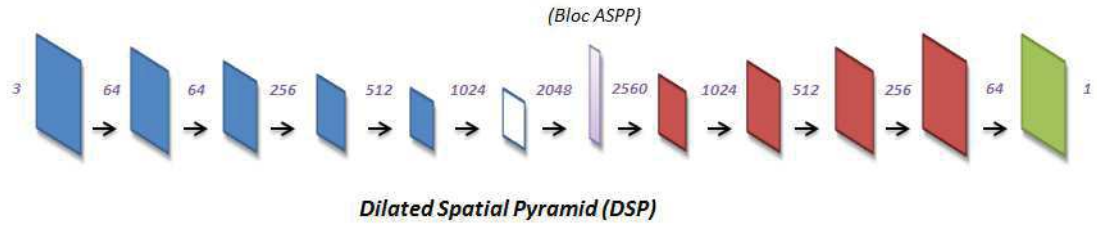


FIGURE 4.11 – Schéma de l'architecture dite 'DSP'.

Architectures d'up-projections (UpProj) des décodeurs				
Méthode	UpProj1	UpProj2	UpProj3	UpProj4
SSN	[2048-1024]	[1024-512]	[512-256]	[256-64]
Skips	[3072-1024]	[1536-512]	[768-256]	[320-64]
FPO	[3072-1024]	[1536-512]	[768-256]	[320-64]
MSML	[3904-512]	[512-256]	[256-64]	-
DSP	[2560-1024]	[1024-512]	[512-256]	[256-64]

TABLE 4.1 – Tableau récapitulatif des différentes architectures précisant les tailles des différents modules de suréchantillonnage utilisés dans nos expériences.

capture donc des caractéristiques à différentes échelles de l'image. Comme eux, nous utilisons ce bloc à la fin de notre encodeur pour extraire un ensemble de descripteurs multiéchelle avant de passer à la partie décodeur (voir Figure 4.11). Nous utilisons une pyramide de 1, 3, 6 et 12 convolutions dilatées pour construire le bloc ASPP au lieu des 1, 6, 12 et 18 dilatations originales, pour nous adapter à la taille de nos descripteurs (voir illustration Figure 4.10). Cette architecture est la meilleure pour la tâche de segmentation sémantique parce que le bloc ASPP, inséré entre l'encodeur et le décodeur, permet d'agréger du contexte multiéchelle grâce à sa pyramide de convolutions dilatées à champs réceptifs variables.

4.3.6 Architectures des décodeurs

Pour chacune des architectures présentées, nous cherchons à obtenir des sorties dont la résolution spatiale est la même que celle de l'entrée. C'est pourquoi entre les 5 architectures, le nombre de modules de suréchantillonnage impliqués dans la partie décodeur peut légèrement varier, de même que le nombre de descripteurs d'entrée et de sortie de chaque module. Nous précisions Table 4.1 l'architecture de chacun des décodeurs avec la syntaxe suivante permettant d'abréger les notations : $[N_i\text{-}N_o]$. Chaque crochet représente un module de suréchantillonnage avec N_i descripteurs en entrée et N_o en sortie.

	à minimiser			à maximiser		
Méthode	rel	log	rms	δ_1	δ_2	δ_3
SSN	0.167	0.066	0.603	77.9	94.9	98.9
Skips	0.163	0.065	0.599	79.2	95.1	98.9
MSML	0.160	0.066	0.601	78.1	**95.4**	**99.0**
FPO	0.161	0.065	0.599	78.5	95.3	98.9
DSP	**0.159**	**0.064**	**0.592**	**79.3**	**95.4**	98.9

TABLE 4.2 – Comparaison des erreurs d'inférence des méthodes multiéchelles sur le jeu de test de la base NYU Depth v2. Nous utilisons uniquement le sous jeu officiel de 795 images pour l'entraînement des réseaux.

4.4 Expériences

Cette section présente une validation expérimentale des différentes architectures proposées dans la section précédente. Ces architectures sont évaluées à la fois quantitativement et qualitativement, puis comparés aux résultats récents de l'état de l'art.

4.4.1 La base de données NYU Depth étendue

Toutes nos expériences ont été menées suivant le protocole du jeu de données *NYU Depth v2* (Nathan Silberman et Fergus, 2012) présenté dans le chapitre 2. Dans la perspective de pouvoir comparer nos approches aux résultats de l'état de l'art, nous avons étendu le jeu de données NYU officiel. Des jeux d'entraînement non officiels similaires ont en effet été utilisés dans de récentes publications comme (Laina et al., 2016). Pour ce faire, nous avons aléatoirement sélectionné des images equiréparties du jeu d'entraînement brut de NYU Depth. Nous nous retrouvons ainsi avec environ 16K images uniques, que nous traitons en utilisant la routine de complétion de la profondeur fournie dans la toolbox matlab de NYU Depth.

4.4.2 Détails sur la procédure d'apprentissage

Pour pouvoir effectuer les comparaisons de la manière la plus juste possible, nous entraînons toutes nos architectures avec les mêmes hyperparamètres et les mêmes procédures d'apprentissage. Nous utilisons à chaque fois un critère de coût L2 entre profondeur mesurée et profondeur estimée, et une descente de gradient stochastique (SGD) avec des batchs de 3 images (excepté pour MSML et pour les deux dernières échelles de FPO où, dus à des limitations de mémoire, nous avons utilisé des batchs de 2 images). Nous entraînons chaque réseau sur au moins 800 époques et arrêtons l'entraînement lorsque l'on observe du surapprentissage sur la base de validation, avec arrêt quand l'erreur augmente pendant plus de 50 époques. Le learning rate est fixé à 5e-3, le 'weight decay' à 5e-4 et le 'moment' à 0.9. Pour les données d'entraînement et de test, nous sous-échantillonnons les images par un facteur de 2 en utilisant l'algorithme du

	à minimiser			à maximiser		
Méthode	rel	log	rms	δ_1	δ_2	δ_3
Eigen et Fergus (2015)	0.158	-	0.641	76.9	95.0	98.8
Li Li et al. (2017)	0.143	0.063	0.635	78.8	95.8	99.1
Laina Laina et al. (2016)	0.138	0.060	0.592	78.5	95.2	98.7
Cao Cao et al. (2017)	0.141	0.060	**0.540**	81.9	96.5	99.2
LS-SSN (L2)	0.150	0.062	0.594	80.2	96.1	98.9
LS-DSP (L2)	**0.133**	**0.057**	0.569	**83.0**	**96.6**	**99.3**

TABLE 4.3 – Comparaison de notre réseau LS-DSP, entraîné sur le jeu de données étendu, avec les résultats de l'état de l'art sur NYU Depth.

plus proche voisin. Nous appliquons également de l'augmentation de données en ligne lors de l'entraînement, en suivant la procédure suivante décrite dans (Eigen et al., 2014) :

- Mise à l'échelle : les entrées et les sorties sont suréchantillonnées par un facteur d'échelle $s \in [1, 1.5]$ pour simuler un rapprochement de la caméra ; les profondeurs sont donc divisées par s pour conserver la géométrie de la scène.
- Rotation : les entrées et les sorties subissent une rotation de $r \in [-5, 5]$ degrés.
- Translation : les entrées et les sorties, après suréchantillonnage, sont aléatoirement recadrées à la taille désirée pour simuler une translation de la caméra.
- Couleur : les valeurs RGB des entrées sont multipliées par une valeur aléatoire $c \in [0.8, 1.2]^3$.
- Flip : les entrées et les sorties ont une chance de subir un retournement horizontal avec une probabilité de 0.5.

4.4.3 Critères d'évaluation

Nous utilisons pour l'évaluation quatre des critères de performance standards présentés dans le Chapitre 2. Comme nous l'avons vu, ces critères sont parmi les plus classiquement utilisés dans la littérature de l'inférence de profondeur. Ils permettent de mesurer la précision globale des prédictions en mesurant les écarts avec les vérités terrain. Il s'agit ici de la *Mean Relative Error* (rel), de la *Mean* log_{10} *Error* (log), de la *Root Mean Squared Error* (rms) et enfin des mesures de *Threshold* δ_1, δ_2 et δ_3.

4.4.4 Expériences sur le jeu de données NYU Depth officiel

Comme première étape de l'analyse, nous entraînons toutes nos architectures sur le petit jeu d'entraînement officiel de NYU Depth, comportant 795 images. Les résultats sont présentés dans le tableau 4.2. Nous constatons d'abord que tous les réseaux multiéchelles ont des performances

	Amélioration relative DSP/SSN					
Nombre d'images d'entraînement	rel	log	rms	δ_1	δ_2	δ_3
800 images	4.8%	3.0%	1.8%	1.8%	0.5%	0%
16K images	11.3%	8.1%	4.2%	3.5%	0.5%	0.4%

TABLE 4.4 – Amélioration relative de DSP par rapport à SSN pour chaque métrique avec des jeux d'entraînement de différentes tailles.

similaires ou meilleures que celles de SSN. Nous pouvons d'abord conclure que pour des jeux d'entraînement aussi petits, une architecture soigneusement conçue comme SSN est très compétitive. Ensuite, bien que l'écart ne soit pas très prononcé, il apparaît que l'information multiéchelles contribue à l'amélioration des résultats de manière consistante.

Si l'on ne considère que les méthodes multiéchelles, DSP surpasse systématiquement les autres architectures. Par conséquent, DSP se distingue comme étant la méthode la plus prometteuse pour obtenir les meilleures performances par la suite. Notons également que MSML, dont l'architecture est proche de celle de DSP (un encodeur qui extrait des caractéristiques multiéchelles et un décodeur sans *skip connections*), montre des résultats intéressants notamment sur les métriques de *Threshold*. Cependant en pratique, MSML est plus lourd à entraîner, avec des résolutions spatiales en sortie d'encodeur plus grande ainsi que des *skip connections* menant à la concaténation de nombreux descripteurs. DSP de son côté n'a besoin que d'une couche supplémentaire comportant seulement 5 convolutions. Nous pensons que c'est cette raison qui fait que MSML ne parvient pas aux mêmes résultats que DSP, avec une procédure d'entraînement comparable en temps d'apprentissage.

4.4.5 Expériences sur le jeu de données NYU Depth étendu

Sur la base de notre analyse précédente, nous avons sélectionné la méthode DSP pour l'entraînement à plus grande échelle sur le jeu de données NYU Depth étendu. La procédure d'apprentissage est la même, excepté que l'entraînement du réseau est limité à 60 époques. Nous entraînons de la même manière notre réseau monoéchelle SSN à titre comparatif. Ces entraînements portent le préfixe LS pour *Large Scale* (LS-SSN et LS-DSP). Les résultats sont présentés dans la Table 4.3 aux côtés d'autres résultats récents de l'état de l'art.

Nous voyons que la méthode LS-DSP que nous proposons atteint de meilleures performances pour pratiquement toutes les métriques usuelles. Notons que, dans le tableau, les résultats de Laina et al. (2016) sont délibérément restreints à l'apprentissage avec une L2, tandis que les auteurs préconisent la fonction de coût berHu. Ce choix intentionnel a été pris pour atténuer l'effet de la fonction de coût dans la comparaison des performances. Néanmoins, par souci de rigueur scientifique, il convient de mentionner que Laina et al. (2016) obtient des résultats légèrement meilleurs sur les métriques *rel* et *log*, tandis que nous obtenons de meilleures

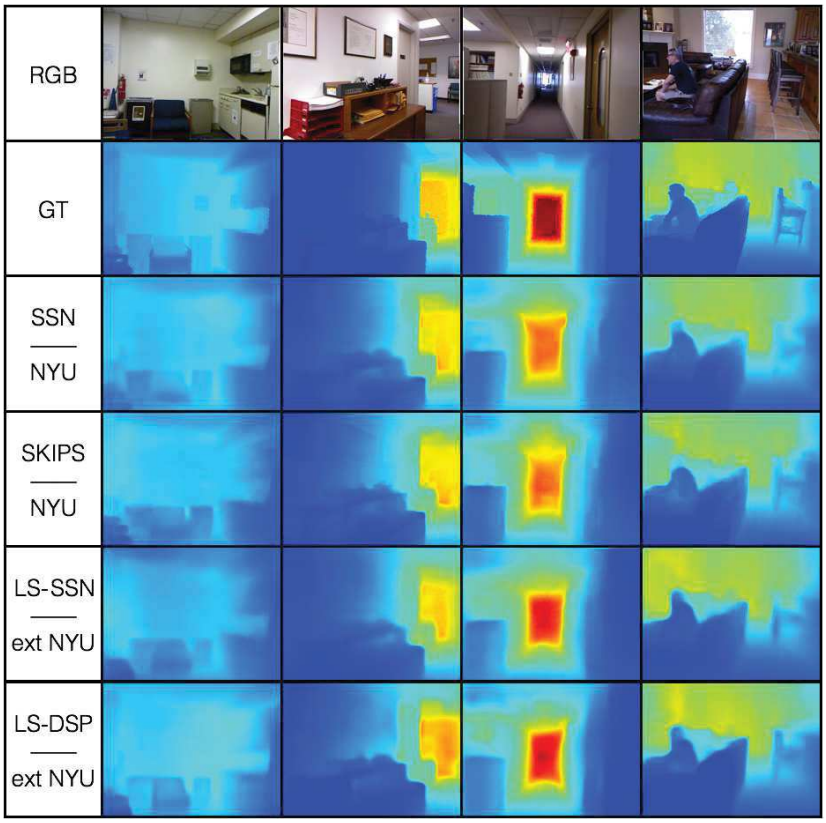


FIGURE 4.12 – De haut en bas : image RGB test, vérité terrain de profondeur, puis estimations par les réseaux SSN, Skips et LS-DSP.

performances sur les 4 métriques restantes.

De plus, nous résumons Table 4.4 les améliorations relatives obtenues avec l'architecture DSP par rapport à SSN dans les deux cas d'apprentissage différents que nous étudions, à savoir des entraînements sur 800 et 16K images. Nous constatons que l'amélioration est plus importante lorsqu'un jeu de données de plus grande échelle est impliqué dans l'apprentissage. Les améliorations constatées sur l'apprentissge de LS-DSP ne sont donc pas seulement dues au nombre de données d'apprentissage, mais bien à l'architecture permettant l'utilisation de caractéristiques multiéchelles dans le réseau.

4.4.6 Évaluation qualitative

La Figure 4.12 permet de comparer qualitativement les résultats du réseau monoéchelle SSN avec ceux de deux des réseaux multiéchelles que nous avons entraînés (Skips et DSP). En effet, les différentes méthodes multiéchelles proposées peuvent globalement être classées en deux catégories : les méthodes avec des skips connections reliant l'encodeur et le décodeur et les méthodes, dont les skips connections ne se font que dans l'encodeur, ou ne sont pas du tout utilisées.

Tout d'abord, nous observons que les méthodes comportant des skips connections dans le

décodeur produisent généralement des sorties plus fines que les autres, ce que l'on voit bien par exemple sur les contours des meubles de rangement en 2e et 3e colonne de la méthode Skips, sur la chaise de la 4e colonne ou encore sur les arrêtes du plafond en 1re colonne. C'est dû au fait que lorsque des *skip connections* sont situées proches de la fin du réseau, des descripteurs bas niveau (comme notamment les gradients de l'image RGB) sont directement transférés du début de l'encodeur à la fin du décodeur. Cela permet en l'occurrence de produire des estimations plus saillantes, mais également au prix de certains artefacts. En effet, on note que les contours des images RGB qui ne sont pas présents dans la vérité terrain de profondeur peuvent aussi être transférés à la carte de profondeur estimée. On peut le voir notamment sur la 2e image dans laquelle des tableaux sont accrochés au mur et dont les contours ne sont pas présents sur la vérité terrain de profondeur. Pourtant l'estimation effectuée avec la méthode Skips met en évidence ces gradients. Il apparaît donc qu'il ne reste pas assez de couches de convolution pour apprendre à lisser ces régions et les gradients sont alors visibles dans l'estimation finale. Nous avons constaté les mêmes types de comportements sur FPO dont l'architecture est proche de Skips.

D'un autre côté, les méthodes dont les skips connections s'arrêtent à l'encodeur sont un peu moins précises sur les bords d'objets, mais elles n'ont pas ce problème de transfert de gradients de l'image RGB. En l'occurrence, nous présentons nos meilleurs résultats obtenus en utilisant LS-DSP, entraîné sur le jeu de données NYU Depth étendu. Par rapport aux autres méthodes, nous constatons non seulement que le réseau généralise mieux, mais également qu'il produit des cartes de profondeur de meilleure qualité. La capacité de généralisation peut par exemple se voir sur l'estimation des distances lointaines. Lorsque SSN ou Skips ont du mal à mettre le fond du couloir de la 3e image à la distance maximum, on voit que LS-DSP a appris des distances plus éloignées. On peut voir que l'estimation issue de LS-SSN montre également cette propriété, mais dans une moindre mesure ; le nombre d'images d'entraînement n'explique donc pas à lui seul cette capacité accrue de généralisation et c'est bien l'utilisation de caractéristiques multiéchelles qui permet ce niveau de détails. On voit également que les formes estimées par LS-DSP sont mieux définies et plus précises, par exemple la tête de la personne sur la 4e image est plus ronde que celles obtenues par les autres estimations, tout comme la chaise en arrière-plan sur la même image dont la forme est précise et bien définie sur l'estimation faite par LS-DSP.

4.5 Conclusions

Dans ce chapitre, nous étudions l'utilisation de l'information multiéchelle dans l'estimation de profondeur, une question qui n'avait pas été analysée jusqu'à présent dans la littérature. Les expériences présentées montrent que, sur le jeu de données NYU Depth officiel comportant 795 images d'entraînement, les architectures multiéchelles proposées surpassent systématiquement

l'architecture équivalente monoéchelle, mais avec peu d'écart. Entraînée sur de plus larges jeux de données, de taille comparable à celles des publications récentes sur cette tâche, la méthode DSP proposée donne des résultats quantitatifs impressionnants. Elles surpassent non seulement la version monoéchelle de référence, mais également l'état de l'art. En outre, nous obtenons des estimations de profondeur d'une grande qualité visuelle grâce notamment à une meilleure capacité de généralisation et des estimations plus fines des formes.

De cette étude nous concluons donc que, si le but est d'avoir la carte de profondeur la plus précise possible à partir d'une entrée RGB monoculaire, il est utile d'utiliser une architecture de réseau nativement multiéchelles où différents niveaux de contexte sont intégrés lors du calcul de la prédiction. Ainsi l'estimation de profondeur sera localement plus fine, tout comme les résultats constatés dans la littérature de la segmentation sémantique.

Chapitre 5

Évaluation de l'incertitude prédictive

Tandis que les applications du *deep learning* sont de plus en plus répandues, la question d'évaluer la fiabilité de leurs prédiction devient une question centrale dans la communauté de l'apprentissage machine. Ce domaine, connu sous le nom d'*incertitude prédictive*, fait l'objet de travaux par des groupes de recherches développant notamment des approches Bayésiennes pour le *deep learning*. Malheureusement pour le moment, le véritable objectif de l'incertitude prédictive semble avoir été perdu de vue. En effet, les approches Bayésiennes sont uniquement évaluées en termes de performance brute des prédictions tandis que la qualité des estimations d'incertitudes elles-mêmes n'est pas quantifiée.

Dans ce chapitre, nous adressons ce problème. Nous porterons d'abord notre attention sur des métriques existantes, développées dans la communauté des prévisions (par exemple la météo) et conçues pour évaluer à la fois la 'sharpness' et la calibration d'une incertitude prédictive. La 'sharpness' se rapporte à la concentration de la distribution prédictive et la calibration correspond à la consistance entre l'incertitude prédite et l'erreur véritable commise. Nous analysons en particulier le comportement de ces métriques sur des problèmes de régression lorsque des CNN profonds sont impliqués. Plusieurs approches de *deep learning* récentes permettent en effet de calculer une sortie assimilable à une incertitude et sont donc applicables dans ce contexte.

Le second aspect important de ce chapitre est la proposition d'une nouvelle métrique qui décrit une incertitude moyenne pondérée. Elle est basée sur le calcul d'un coefficient d'échelle calculé de sorte que les intervalles centrés sur les prédictions englobent les vérités terrain. Contrairement aux métriques classiques, la métrique que nous proposons ne requiert pas d'a priori sur la distribution, permet de classer efficacement les différentes méthodes d'estimation et est paramétrable pour être robuste aux données aberrantes (outliers). Elle est donc directement applicable et mieux adaptée aux méthodes récentes d'estimation d'incertitude en *deep learning*. Cette métrique est évaluée et comparée avec celles existantes sur un exemple jouet ainsi que sur le problème d'estimation de profondeur monoculaire où nous montrons qu'elle reflète bien la corrélation avec l'erreur réelle commise par les réseaux.

5.1 Introduction

De nos jours, nous sommes capables d'apprendre aux machines une grande variété de tâches. Depuis l'avènement du *deep learning* ces dernières années, nous sommes également capables d'atteindre - et même parfois de surpasser - les performances humaines, *e.g.* pour des tâches de vision Kokkinos (2015); Lu et Tang (2015). Cependant, bien que le *deep learning* nous permette d'atteindre des performances impressionnantes, nous manquons toujours de moyens de fournir une confiance dans les prédictions fournies par un réseau donné. Tandis que la question de la robustesse des algorithmes de *deep learning* aux attaques adversaires a été très activement étudiée dans la littérature récente (voir par exemple Papernot et al. (2016)), la

question de l'estimation de l'incertitude des prédictions faites par les réseaux de neurones a reçu beaucoup moins d'attention. Pourtant, pour d'innombrables applications, il est critique de savoir si la sortie d'un algorithme est fiable ou non.

Dans ce chapitre nous prenons pour exemple l'estimation de profondeur à partir d'images monoculaires. Pour ce genre de tâche, les algorithmes d'estimation sont entraînés à régresser une carte de profondeur depuis une image RGB. Si nous voulons utiliser de tels algorithmes comme outils de mesure, il est nécessaire d'avoir, en plus de la prédiction de profondeur, une carte d'incertitude fournissant une estimation de la précision de la prédiction pour chaque pixel. La prédiction de l'incertitude est en effet une information très précieuse, dépendante du contenu de l'image. Il est probable que l'estimation de profondeur soit moins précise sur des régions peu texturées par exemple, ou sur des images très sombres par rapport à des images bien exposées.

Dans la littérature du *deep learning*, des techniques récentes ont émergé pour estimer des incertitudes correspondantes aux prédictions des réseaux Gal et Ghahramani (2016); Lakshminarayanan et al. (2017); Kendall et Gal (2017) ; nous y reviendrons par la suite. Cependant, il arrive que ces incertitudes prédictives ne soient évaluées que par leur influence sur les performances de prédiction du réseau, ou visuellement en regardant les cartes d'incertitudes obtenues Kendall et Gal (2017). Ce type d'évaluation contraste avec d'autres domaines comme la prévision (typiquement la prévision météo). Dans ce domaine, des méthodes statistiques sont employées dans le but de prédire un évènement futur en se basant sur des données du passé et du présent. Les notions de risque et d'incertitude sont centrales dans la prévision, c'est pourquoi des outils y ont été établis pour évaluer directement l'incertitude prédictive Uusitalo et al. (2015); Gneiting et al. (2007); Gneiting et Raftery (2007). Nous pensons que la communauté de vision par ordinateur devrait profiter de ces métriques d'évaluation pour évaluer la qualité des estimations d'incertitudes en elles-mêmes.

Dans ce chapitre, nous présentons d'abord une revue des métriques existantes permettant d'évaluer la qualité des incertitudes prédictives. Dans un second temps, nous effectuons une analyse approfondie sur le comportement de ces métriques sur des problèmes de régression lorsque des CNN profonds sont impliqués et pour plusieurs approches récentes de prédiction d'incertitude. Pour finir, nous proposons dans ce chapitre une nouvelle métrique d'évaluation qui est plus adaptée à l'évaluation de l'incertitude relative à l'erreur et directement applicable aux pratiques actuelles en *deep learning*. Cette métrique est évaluée et comparée à celles existantes sur un exemple jouet ainsi que dans le contexte d'estimation de profondeur à partir d'images monoculaires.

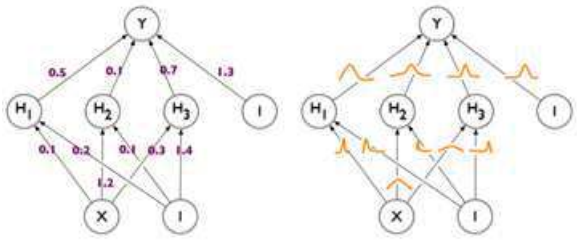


FIGURE 5.1 – Réseau de neurones classique (poids déterministes) vs réseau de neurones probabiliste (distribution pour chaque poids)

5.2 Littérature associée

Bien que cela n'ait pas été particulièrement étudié dans le contexte de l'estimation de profondeur, avoir des estimations d'incertitude associées aux prédictions devient de plus en plus important dans le domaine du *deep learning*. Pour ce faire, une manière classique de procéder est de transformer les réseaux de neurones déterministes, dans lesquels chacun des paramètres est un scalaire, par des réseaux de neurones Bayésiens probabilistes, où chaque paramètre est remplacé par une distribution de probabilité (voir figure 5.1). Étant donné que l'inférence devient intraitable, plusieurs articles reposent sur des méthodes d'approximation des solutions. Parmi les techniques les plus populaires, on trouve la *Probabilistic BackPropagation (PBP)* Hernández-Lobato et Adams (2015), le *Monte-Carlo Dropout* Gal et Ghahramani (2016), mais également quelques techniques non Bayésiennes comme les *Deep Ensembles* Lakshminarayanan et al. (2017) ou encore l'apprentissage d'un coût d'atténuation proposée par Kendall et Gal (2017) pour capturer l'incertitude dite "aleatoric" dans les données.

Cependant, cette littérature manque toujours de métriques appropriées pour évaluer la qualité des incertitudes. En effet, on trouve régulièrement que les évaluations quantitatives sont effectuées sur plusieurs ensembles de données, ceux classiquement utilisés pour évaluer les tâches apprises, mais avec des comparaisons de performances moyennes uniquement en termes de RMSE. C'est la qualité de la prédiction qui est mesurée et non la qualité de l'estimation d'incertitude. De plus, des résultats qualitatifs permettent de commenter visuellement les cartes d'incertitudes. Cependant, à notre connaissance, la littérature du *deep learning* ne propose pas de méthode d'évaluation quantitative spécifique pour les estimations d'incertitude en elles-mêmes.

En revanche, évaluer l'incertitude prédictive a été une préoccupation depuis des décennies dans d'autres domaines, menant au développement de métriques bien établies. Le challenge PASCAL d'évaluation de l'incertitude prédictive Quinonero-Candela et al. (2006) propose deux métriques adaptées aux tâches de régression, qui sont toujours communément utilisées. La MSE normalisée vise à évaluer la qualité d'une prédiction et utilise la variance empirique des estimations de test pour normaliser les scores. Ce n'est donc pas un moyen direct d'évaluer l'in-

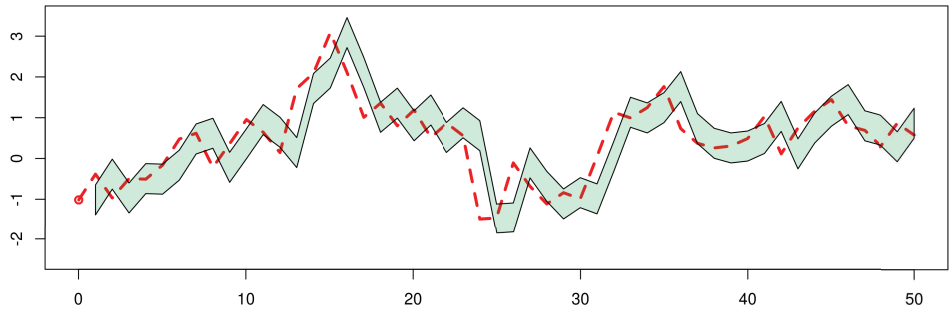


FIGURE 5.2 – En rouge le signal réel et en vert la prédiction probabiliste. La calibration correspond au décalage moyen entre les deux courbes et la 'sharpness' à l'étalement moyen de la distribution prédite

certitude prédictive. La seconde métrique, la Negative Log Predictive Density moyenne (NLPD), prend directement en compte l'estimation d'incertitude et pénalise à la fois les prédictions sur confiantes et sous-confiantes.

Une analyse plus approfondie peut être trouvée dans la littérature de la prévision. Uusitalo et al. (2015) dressent un aperçu des différentes familles de méthodes existantes d'estimations d'incertitudes pour des modèles déterministes, telles que :

- l'évaluation d'un expert : un expert dans le domaine évalue la confiance d'une méthode en utilisant ses connaissances ;
- l'analyse de la sensibilité du modèle : détermine comment la réponse d'un modèle change en fonction des entrées et des paramètres du modèle ;
- l'émulation de modèle : consiste à approximer un modèle complexe par une fonction de faible degré pour ensuite appliquer plus facilement une analyse de sensibilité ;
- la variabilité spatio-temporelle : utilise la variabilité dans le temps et dans l'espace comme une estimation de la variance ;
- le multi modèle : un ensemble de modèles est utilisé pour décrire le même domaine ; et finalement les approches basées sur les données, où suffisamment de données sont disponibles pour directement évaluer l'incertitude statistique liée aux sorties du modèle.

Gneiting et al. (2007) proposent d'évaluer l'incertitude prédictive d'une prévision en termes de calibration et de 'sharpness'. La calibration se réfère à la cohérence statistique entre les distributions prédictives et les observations tandis que la 'sharpness' se réfère à la concentration des distributions prédictives, comme illustré en figure 5.2. Plusieurs outils graphiques sont proposés pour évaluer visuellement si les prévisions probabilistes sont calibrées et sharp. Cependant, le plus intéressant pour nous repose dans l'évaluation quantitative jointe à la fois de la calibration et de la 'sharpness'. Il se trouve que justement, des fonctions évaluant la qualité d'une prévision en termes de calibration et de 'sharpness' existent et sont appelées scoring rules Gneiting et Raftery (2007). Une scoring rule mesure la précision d'une prédiction probabiliste en assignant un score basé sur la distribution.

Toutes ces métriques impliquent donc que les prédictions soient des distributions de probabilités, ce qui est rarement le cas en pratique. Au lieu de cela, les méthodes existantes de *deep learning* prédisent généralement une incertitude sous la forme d'un scalaire représentant la variance de quelques prédictions (*e.g.* dans le cas de l'échantillonnage de Monte-Carlo). Nous pouvons facilement convertir les paires prédictions/incertitudes en distribution de probabilité, mais il faut alors choisir un a priori sur la distribution. Nous estimons que, dans des limites de variation raisonnables, cet a priori n'est pas d'une importance critique, car il ne change pas radicalement le classement entre les différentes méthodes de prédiction d'incertitude, comme nous allons le voir par la suite.

Dans ce contexte, nous proposons une nouvelle métrique pour évaluer les couples prédictions/incertitudes. Nous appelons notre métrique la **Mean Rescaled Confidence Interval (MeRCI)**. Elle est basée sur le calcul d'un coefficient d'échelle tel que l'intervalle centré sur la prédiction englobe l'observation de la vérité terrain. Elle ne nécessite aucun a priori sur la distribution, permet de classer efficacement les différentes méthodes d'estimation et est paramétrable pour être robuste aux données aberrantes (outliers).

5.3 Métriques d'évaluation classiques pour l'incertitude prédictive

Supposons une configuration de régression impliquant un jeu de données de N échantillons dénotés $\mathcal{T} = \{(x_i, y_i^*)\}$ où $x_i \in \mathbb{R}^d$ est le vecteur d'entrée de dimension d et $y_i^* \in \mathbb{R}$ le scalaire à prédire. Dans un contexte de régression déterministe standard, le régresseur est une fonction définie par $f_\theta : \mathbb{R}^d \to \mathbb{R}$, où $f_\theta(x)$ est un estimateur de la vraie fonction $f(x)$, et $\hat{y} = f_\theta(x)$ est la prédiction faite par le modèle. Apprendre un tel régresseur consiste à trouver les paramètres θ minimisant un coût empirique définie par $\frac{1}{N}\sum_{i=1}^{N} L(f_\theta(x_i), y_i^*)$ où $L(\cdot,\cdot)$ représente la proximité entre ses deux arguments (*e.g.* la différence au carré). Dans un contexte probabiliste, on prédit une distribution entière P_y au lieu d'une prédiction déterministe. Cette distribution a une densité de probabilité $p_y(x)$ et une fonction de répartition F_y.

En pratique, P_y est souvent caractérisée par sa valeur moyenne, généralement utilisée comme la prédiction $\hat{y}$, et par son écart type σ. Différentes approches pour calculer ce genre de prédictions seront examinées dans la section 5.5.

Dans la littérature de la prévision, on peut trouver plusieurs métriques pour évaluer la qualité d'une distribution prédictive, ou prévision. Nous présentons ici les 5 plus populaires. Chacune d'elles est calculée comme une espérance empirique :

$$\frac{1}{N}\sum_{i=1}^{N} S(P_{y_i}, y_i^*) \tag{5.1}$$

Selon la variante considérée, cela peut expliquer la 'sharpness', la calibration, ou les deux. Dans les paragraphes suivants, nous nous référerons à S comme la fonction d'adéquation.

5.3.1 Coverage

Le coverage est défini pour un niveau de confiance α fixé (*e.g.* 95%) comme le ratio des observations qui appartiennent effectivement à l'intervalle de confiance associé. Soit $CI_\alpha(P_{y_i})$ l'intervalle de confiance à α pour la prévision, la fonction d'adéquation est alors la fonction indicatrice de l'intervalle de confiance :

$$S_{cov}(P_{y_i}, y_i^*) = \mathbb{1}_{\{y_i^* \in CI_\alpha(P_{y_i})\}} \tag{5.2}$$

Le coverage est minimum (0%) si aucune des valeurs n'est dans l'intervalle de confiance correspondant, et peut atteindre 100% si toutes les valeurs sont dans l'intervalle. Par conséquent, le coverage ne représente que la calibration, et peut être arbitrairement amélioré en faisant des prédictions non sharp (en d'autres termes en étant intentionnellement pessimiste sur les prédictions).

5.3.2 Strictly proper scoring rules

Dans la littérature de la prévision, les fonctions d'adéquation qui sont orientées positivement sont connues comme étant des scoring rules. Une scoring rule est dite strictly proper si en moyenne, elle distingue la vraie distribution comme étant la prévision optimale au sens suivant :

$$\mathbb{E}_{Y \sim P_y^*}[S(P_y^*, Y)] < \mathbb{E}_{Y \sim P_y^*}[S(P_y, Y)] \tag{5.3}$$

pour toue prévision P_y différente de la distribution vérité terrain P_y^*.

Voici trois scoring rules communes et populaires :

$$S_{Quadratic}(P_y, y^*) = 2p_y(y^*) - \int_{-\infty}^{+\infty} p_y^2(x)dx \tag{5.4}$$

$$S_{Logarithmic}(P_y, y^*) = \log(p_y(y^*)) \tag{5.5}$$

$$S_{Spherical}(P_y, y^*) = \frac{p_y(y^*)}{\sqrt{\int_{-\infty}^{+\infty} p_y^2(x)dx}} \tag{5.6}$$

5.3.3 CRPS

Une autre métrique populaire est le Continuous Ranked Probability Score (CRPS). C'est une mesure de distance entre une distribution probabiliste et une observation déterministe. La fonction d'adéquation CRPS entre la prévision P_y (de fonction de répartition F_y) et y^* est définie de la manière suivante :

$$S_{crps}(P_y, y^*) = \int_{-\infty}^{+\infty} \Big(F_y(x) - H(x - y^*)\Big)^2 dx \tag{5.7}$$

où $H(t) = \mathbb{1}_{\{t>0\}}$ est la fonction de Heaviside.

Le CRPS est exprimé dans la même unité que la variable observée et se réduit à l'erreur absolue moyenne si la prévision est déterministe. Il est minoré par 0 et, contrairement aux métriques précédentes, il est orienté négativement.

5.4 Métrique proposée

La section précédente introduit plusieurs métriques pour évaluer la qualité d'une prévision. Pour être applicable, elle nécessite que la prédiction soit définie par une fonction de densité. Cependant, la plupart des méthodes de régression en *deep learning* ne permettent pas de calculer directement une distribution, mais une valeur scalaire $\hat{y}$ associée à un écart type σ. Par conséquent, pour évaluer la qualité de ces méthodes avec les métriques existantes, nous devons choisir une distribution centrée autour de la valeur prédite avec la bonne variance. Par simplicité, on peut utiliser un a priori Gaussien mais ce choix est complètement arbitraire. Nous présentons ici une métrique qui s'affranchit de cette contrainte en établissant un score basé uniquement sur les paramètres estimés $\hat{y}$ et σ grâce au régresseur.

5.4.1 Mean Rescaled Confidence Interval

Nous cherchons une métrique qui reflète la corrélation entre la vraie erreur et l'estimation de l'incertitude. De plus, nous voulons que cette métrique soit indépendante au changement d'échelle parce que nous considérons que, en pratique, les estimations d'incertitudes ne donnent qu'un aperçu de la magnitude relative des erreurs. Nous la voulons également robuste aux données aberrantes (outliers). Inspiré par la notion de calibration et avec l'invariance aux changements d'échelle à l'esprit, nous proposons de mettre à l'échelle tous les σ_i, correspondants aux prédictions faites à chaque pixel i, par un facteur commun λ de sorte que les incertitudes rescalées $\tilde{\sigma}_i = \lambda\sigma_i$ soient compatibles avec les erreurs observées, c'est à dire $|\hat{y}_i - y_i^*| \leq \tilde{\sigma}_i$. La robustesse aux données aberrantes est obtenue en relâchant cette contrainte de sorte qu'elle ne soit vérifiée que par une certaine proportion des échantillons. Nous choisissons arbitrairement de fixer cette valeur à 95 %. Nous autorisons ainsi 5% de données aberrantes lors de l'évaluation. En d'autres termes, nous cherchons le plus petit facteur λ^{95} tel que 95 % des erreurs observées soient plus petites que les prédictions d'incertitudes (mises à l'échelle) correspondantes.

Ainsi, le Mean Rescaled Confidence Interval que nous proposons est une incertitude moyenne calculée sur les incertitudes estimées λ-rescalées et moyennées sur le nombre de points N évalués :

$$MeRCI = \frac{1}{N}\sum_{i=1}^{N} \lambda^{95}\,\sigma_i \tag{5.8}$$

Pour déterminer le facteur d'échelle, nous calculons d'abord tous les ratios $\lambda_i = \frac{|\hat{y}_i - y_i^*|}{\sigma_i}$ et ensuite le $95^{\text{ème}}$ percentile qui correspond exactement à la valeur désirée.

5.4.2 Comportement

MeRCI est intrinsèquement insensible au changement d'échelle et nous permet donc de comparer des méthodes qui prédisent des gammes d'incertitude très différentes. En outre, certaines valeurs pertinentes peuvent facilement être calculées pour cette métrique. D'une part, le score MeRCI est toujours minoré par un score correspondant à une prédiction d'incertitude idéale, appelée **oracle**. Cet oracle prédit simplement une incertitude égale à la vraie erreur observée, ce qui mène à un $\lambda^{95} = 1$ et un score MeRCI qui se réduit alors à l'erreur absolue moyenne. D'autre part, toute prédiction d'incertitude qui est moins bonne qu'un prédicteur d'incertitude **constant** est inutile. En effet, avoir une incertitude uniforme constante pour l'ensemble des prédictions est en fait équivalent à ne prédire aucune incertitude ; dans les deux cas, il est impossible de démarquer des régions plus ou moins certaines. Par conséquent, le score MeRCI obtenu par un prédicteur constant fournit également une forme de borne supérieure. Il se réduit à l'erreur absolue commise pour le 95e percentile.

Dans sa construction, le score MeRCI suit la piste décrite par Gneiting et al. (2007). Il reflète la meilleure 'sharpness' possible sous la contrainte de calibration de l'estimation d'incertitude, et ce dans le sens où il est consistant avec 95 % des échantillons. De plus, l'oracle est le prédicteur d'incertitude le plus sharp possible, une fois que les valeurs estimées $\hat{y}_i$ sont fixées.

5.5 Méthodes pour l'incertitude prédictive

Nous utilisons plusieurs méthodes pour calculer, pour une tâche donnée, à la fois une prédiction et l'estimation d'incertitude correspondante. Tout d'abord, nous prenons les méthodes les plus utilisées et populaires de la littérature du *deep learning*, parmi lesquelles on trouve les techniques Bayésiennes. **Monte-Carlo Dropout (MCD)** Gal et Ghahramani (2016) est une des techniques les plus populaires utilisées dans le domaine du *deep learning* pour estimer des incertitudes associées aux prédictions du réseau, en raison de sa faible contrainte. Au moment du test, le dropout est laissé activé pour effectuer un échantillonnage de Monte-Carlo. La prédiction finale est la moyenne des différentes estimations et leur écart-type est utilisé comme mesure d'incertitude. Kendall et Gal (2017) de leur côté, proposent une fonction de coût qui leur permet de directement régresser ces deux valeurs d'intérêt, à savoir la prédiction et l'écart-type. Ils montrent que cela peut être assimilé à un coût d'atténuation appris et que cette manière d'apprendre l'écart-type peut modéliser l'incertitude aléatoire qui correspond au bruit dans les données d'entrée. Nous appelons cette méthode **Learned Attenuation**.

Nous explorons également des techniques ensemblistes populaires. Dans l'apprentissage

par ensembles, plusieurs algorithmes sont utilisés pour obtenir une performance prédictive. Ici, les différentes sorties déterministes sont combinées (généralement moyennées) pour obtenir la prédiction finale et leur écart-type est associé au degré d'incertitude de la prédiction. Une approche d'ensemble classique consiste à entraîner plusieurs réseaux sur la même tâche, mais avec différentes initialisations aléatoires. C'est l'approche employée dans Deep Ensembles Lakshminarayanan et al. (2017), et nous référons à cette méthode par **Multi Inits (MI)**. Une autre manière ensembliste populaire d'estimer l'incertitude est le **Bagging** Breiman (1996). Cela implique d'utiliser plusieurs sous-sets de l'ensemble de données pour entraîner plusieurs réseaux.

Bien que nous considérons uniquement les problèmes de régression dans cette étude, les cadres de classification permettent dans une certaine mesure de bénéficier a posteriori d'une distribution sur l'ensemble possible des résultats. C'est pourquoi nous essayons aussi de discrétiser nos vérités terrain pour changer le problème de régression en classification. Cela permet d'obtenir des distributions en sortie sans avoir besoin d'a priori paramétrique. L'estimation finale est l'$arg\,max$ des log probabilités calculées par le réseau tandis que l'écart-type de la distribution peut être considéré comme l'estimation d'incertitude. Nous le dénotons **Discretization**.

Ensuite, inspirés par l'état de l'art, nous proposons également plusieurs techniques pour prédire des incertitudes dans un cadre de *deep learning*. D'abord, nous proposons de considérer le réseau à plusieurs étapes de son apprentissage comme étant autant de réseaux différents qu'il y a d'époques. En admettant cela, on peut simuler une méthode d'ensemble sans coût supplémentaire avec une procédure d'entraînement standard. Nous nous référons à cette méthode par **Multi Epochs (ME)**. De manière similaire à avoir plusieurs initialisations, nous proposons **Multi Networks (MN)**. Cette méthode est également comparable à une approche ensembliste où cette fois non seulement l'initialisation diffère d'un réseau à un autre, mais également l'architecture des réseaux. Comme dans les méthodes d'ensemble, pour ces deux dernières approches, nous moyennons les sorties pour obtenir la prédiction et utilisons l'écart-type comme mesure d'incertitude. Finalement, nous considérons que puisque nous voulons mesurer la corrélation entre l'incertitude et l'erreur, nous pouvons essayer d'apprendre directement l'erreur d'un réseau entraîné avec un autre réseau. La sortie de ce réseau qui apprend l'erreur sera donc l'incertitude estimée, associée à la prédiction du réseau initial. Cette technique est désignée par **Learned Error (LE)**.

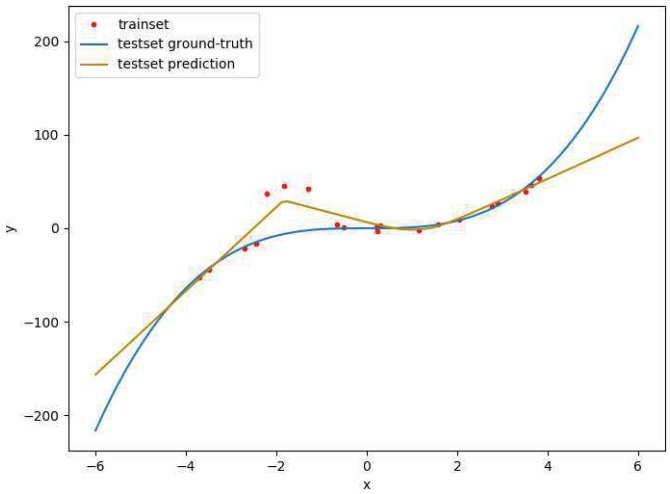


FIGURE 5.3 – L'exemple jouet

5.6 Validation expérimentale : comportement des métriques sur un exemple jouet

Dans cette section, nous voulons comprendre comment les différentes métriques que nous avons présentées se comportent sur des problèmes pratiques de machine learning et en particulier sur des tâches de régression. Dans le but d'avoir une intuition sur cette question, nous commençons avec un exemple jouet sur lequel nous analysons le comportement des différentes métriques.

5.6.1 Protocole expérimental

L'exemple jouet que nous utilisons est un simple cas de régression en 1D. Les données consistent en 20 points d'apprentissage, échantillonnés uniformément sur l'intervalle $[-4; 4]$. Les vérités terrain correspondantes sont générées par $y = x^3 + \epsilon$ où $\epsilon \sim \mathcal{N}(0, 3^2)$. Nous ajoutons un biais important pour chaque point d'entraînement dans l'intervalle $[-2.3; -1.3]$ pour simuler des données aberrantes, comme illustré en figure 5.3. Un ensemble de données similaire a déjà été utilisé pour évaluer la qualité d'incertitudes prédictives dans Hernández-Lobato et Adams (2015) et Lakshminarayanan et al. (2017).

Étant donné x, nous apprenons à régresser y en utilisant un réseau de neurones avec une couche cachée composée de 100 neurones. Nous utilisons une ReLU pour la non-linéarité et un dropout de 0.2 sur la couche cachée. Les prédictions du réseau sur la base de test sont illustrées en figure 5.3. Comme attendu, nous voyons que les prédictions sont moins précises pour des données de test à l'extérieur de la distribution des données d'entraînement ainsi que dans l'intervalle où se situaient les données aberrantes. Nous sommes intéressés par la comparaison de différentes techniques d'estimation d'incertitude sur ces données. Pour que cela soit possible, nous conservons les prédictions du réseau fixées pour toutes les expériences.

A partir de là, nous faisons varier la manière dont les estimations d'incertitude sont calculées avec 4 techniques d'estimation d'incertitude différentes : Multi Inits, Bagging, Monte-Carlo Dropout et Multi Epochs.

Nous utiliserons cet ensemble de données pour discuter de la manière dont les différentes métriques sont biaisées en faveur d'estimations d'incertitude plutôt optimistes ou pessimistes ; à quel point elles sont robustes aux données aberrantes ; et enfin nous vérifierons qu'elles classent correctement les différentes techniques d'estimation d'incertitude évaluées.

En ce qui concerne la procédure d'apprentissage, nous utilisons l'optimiseur Adam avec les paramètres $\beta_1 = 0.9$, $\beta_2 = 0.999$ et $\epsilon = 1\text{e}{-8}$ pour minimiser une erreur L2. Le pas d'apprentissage est à $1\text{e}{-1}$ pour le cas de l'underfitting et à $1\text{e}{-4}$ pour celui de l'overfitting. Les deux expériences ont un weight decay de $5\text{e}{-4}$. Nous utilisons un jeu de validation de 20 échantillons, générés de la même manière que le jeu 'entraînement et grâce auquel nous utilisons du early-stopping à l'entraînement. Nous choisissons arbitrairement d'utiliser 5 initialisations pour Multi Inits, 5 ensembles de données générés selon la même distribution que le jeu d'entraînement pour le Bagging, 50 prédictions pour le Monte-Carlo Dropout, les 50 dernières époques de l'entraînement pour le Multi Epochs et un réseau avec une couche cachée et 3 neurones pour Learned Error.

5.6.2 Sensibilité à l'optimisme et au pessimisme

Nous commençons par étudier le cas simple d'un estimateur d'incertitude constant, prédisant toujours la même valeur d'incertitude σ quelle que soit la donnée d'entrée. Pour ce faire, nous traçons l'évolution de chaque métrique au fur et à mesure que σ évolue d'une valeur trop optimiste (faible incertitude σ) à une valeur trop pessimiste (forte incertitude σ). En pratique, nous choisissons σ dans l'intervalle $[0; 100]$ (voir figure 5.4). Nous constatons que les métriques classiques sont grandement impactées par le changement d'échelle de σ, affichant de fortes variations entre ses valeurs extrêmes. Nous observons que, quelle que soit la métrique considérée, la valeur optimum de σ est plutôt basse, ce qui signifie qu'elle dépend plus de la calibration de la méthode que de la métrique en elle-même.

Comme attendu, nous confirmons que le coverage est uniquement sensible à la calibration. Pour cette raison le score peut être arbitrairement amélioré en augmentant simplement la valeur de σ. Étant donné qu'il est complètement indépendant de la 'sharpness', nous préférons ne pas l'utiliser dans nos analyses à venir. Le CRPS est plutôt sensible à la 'sharpness' étant donné que son score continue à se détériorer à mesure que σ augmente. Les autres scoring rules réagissent plus fortement à la calibration. Enfin, nous voyons bien que le score MeRCI est constant quelle que soit la valeur de σ ce qui montre qu'il n'est influencé ni par les prédictions excessivement confiantes ni celles excessivement pessimistes.

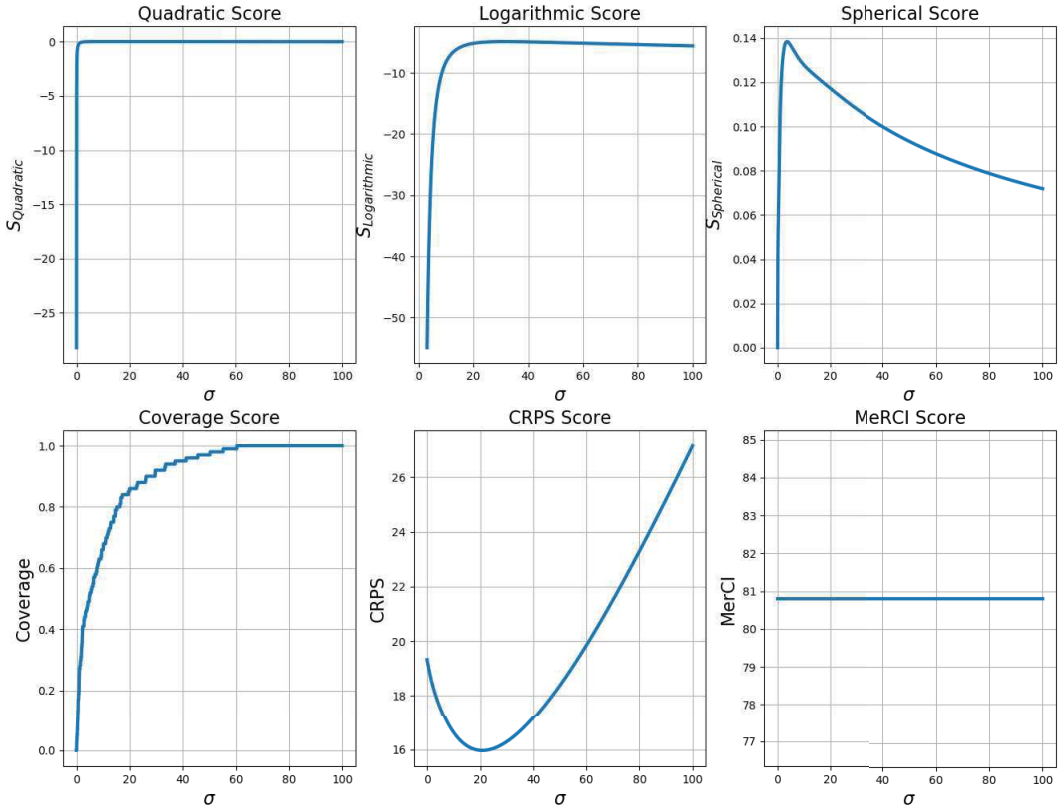


FIGURE 5.4 – Évolution des scores pour chaque métrique en fonction d'un estimateur d'incertitude constant

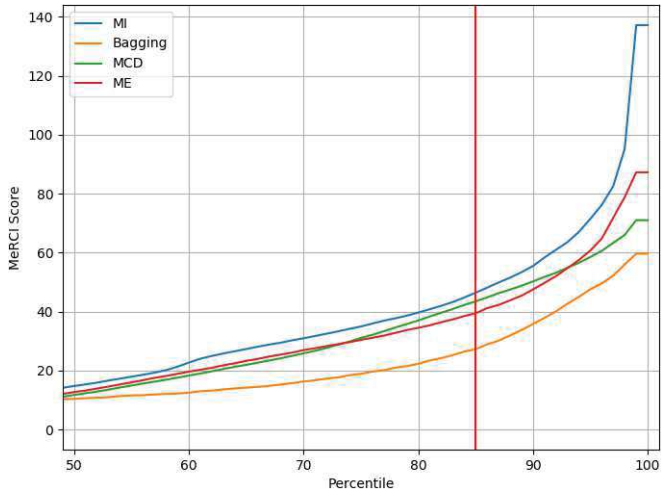


FIGURE 5.5 – Évolution du score MeRCI pour chaque méthode en fonction du percentile choisi. La ligne rouge marque le pourcentage réel de données aberrantes

5.6.3 Robustesse aux données aberrantes

Contrairement aux métriques classiques, MeRCI peut être paramétrée pour tolérer plus ou moins de données erronées via la valeur du percentile choisi. Pour mieux comprendre ce qui se passe lors de la variation du percentile, nous traçons l'évolution de MeRCI par rapport au percentile choisi en figure 5.5. Pour réaliser la comparaison la plus juste possible au sens

Table 5.1 – Évaluation quantitative sur l'exemple jouet

	à maximiser			**à minimiser**	
Method	Q_s	L_s	S_s	CRPS	MeRCI
Oracle	-	-	-	-	19
Constant	-	-	-	-	81
Multi Inits	-0.65	-6.0	0.07	19.0	64
Bagging	-0.02	-4.8	0.13	16.8	53
Monte-Carlo Dropout	-0.03	-4.9	0.11	17.1	60
Multi Epochs	-28	-6.9	0.0	19.3	81

statistique, nous calculons 1000 tirages indépendants pour chacune des méthodes comparées et nous affichons la courbe médiane d'évolution de la MeRCI. Dans cet exemple jouet, le taux de données corrompues est connu à l'avance (il est marqué par la ligne verticale rouge sur la figure). Comme attendu, l'évolution du score MeRCI augmente doucement à mesure que le nombre d'inliners considérés augmente. Par ailleurs, nous voyons que MeRCI est robuste aux données aberrantes puisqu'il classe les différentes techniques d'estimation d'incertitude dans le même ordre pourvu qu'il considère suffisamment de données (ici à partir du 74e percentile) et ce jusqu'à ce que les données aberrantes n'influencent le score en prenant trop de poids dans la décision finale (92e percentile). En pratique, le percentile devrait être choisi selon le taux de données aberrantes supposé. Cependant il est tout de même à noter que connaître le nombre exact d'outilers n'est pas nécessaire pour obtenir un score MeRCI pertinent. En effet, nous constatons qu'il n'y a pas de changement brutal du classement autour du vrai nombre de données aberrantes présents dans le jeu de données (la ligne rouge). C'est pourquoi, dans nos expériences futures, nous choisissons d'utiliser la MeRCI avec un percentile de 95 pour nous donner assez de latitude pour gérer jusqu'à 5% d'éventuelles données aberrantes.

5.6.4 Résultats

Finalement, nous discutons des scores obtenus pour chacune des techniques d'estimation d'incertitude. Nous nous attendons à ce que Bagging soit la meilleure technique puisqu'elle bénéficie de sources de données extérieures au jeu d'entraînement. Nous nous attendons également à ce que Multi Epochs soit la pire technique puisque dans cette configuration, il est très probable que le réseau converge dans un minimum local. Ainsi, nous pensons qu'il n'y aura que très peu de variances informatives entre les époques après convergence.

Nous montrons les résultats obtenus en table 5.1. Nous mettons également en évidence le classement relatif des différentes techniques d'estimation d'incertitude en fonction de chacune des métriques en utilisant un jeu de couleurs : du jaune pour la meilleure technique jusqu'au rouge pour la moins bonne. Pour la visibilité, le moins bon score est en gris. Conformément à cette table, toutes les métriques s'accordent sur le classement des différentes méthodes.

Comme nous l'attendions, elles classent toutes la méthode Multi Epochs en dernière position. Il est intéressant de constater que cette technique est clairement marquée comme étant non informative par le score MeRCI (obtenant un score aussi mauvais que la prédiction d'incertitude constante). Enfin, les meilleures méthodes sont Bagging et Monte-Carlo Dropout avec des scores très proches pour chaque métrique.

Dans une configuration simple comme cet exemple jouet, nous montrons que toutes les métriques classiques ont un comportement sain dans leur capacité à classer différentes techniques d'estimation d'incertitude. Nous montrons également que la métrique que nous proposons suit correctement ces traces. Nous pouvons alors passer à une évaluation avec une configuration plus réaliste.

5.7 Analyse : le cas d'application réel d'estimation de profondeur à partir d'images monoculaires

Nous allons maintenant travailler sur un jeu de données réelles avec l'étude d'une tâche de vision par ordinateur basée sur la régression : l'estimation de profondeur monoculaire.

5.7.1 Protocole expérimental

Cette section présente les expériences sur l'estimation de profondeur à partir d'images monoculaires issues de la base de données NYU Depth v2 (Nathan Silberman et Fergus, 2012). Nous utilisons le même ensemble de 16K images proposé dans le chapitre 4. De plus, nous utilisons la même architecture mono échelle (SSN dans le chapitre 4) avec un ResNet200 dilaté pré entraîné sur la tâche de classification d'ImageNet pour l'encodeur suivi par 4 modules d'up-projection puis une dernière convolution pour le décodeur. Nous ajoutons également du dropout avec une probabilité de 0.2 après les 3 premiers modules d'up-projection dans le but de pouvoir effectuer un échantillonnage de Monte-Carlo au moment du test pour l'expérience de Monte-Carlo Dropout.

Pour que les expériences soient comparables, la procédure d'apprentissage est la même pour chaque réseau entraîné à la tâche d'estimation de profondeur. Nous utilisons une Mini-Batch Gradient Descent avec un batch de 12 images et l'optimiseur SGD pour minimiser une erreur L1, excepté pour l'expérience Learned Error où nous minimisons le coût atténué proposé par Kendall et Gal (2017) et pour Discretization où nous utilisons une Cross Entropy. Le pas d'apprentissage est fixé à $1e{-}3$, le weight decay à $5e{-}4$. Nous entraînons tous nos réseaux durant 120 époques (la convergence est empiriquement toujours observée à ce moment), excepté pour Discretization où l'entraînement dure 60 époques, parce que nous avons empiriquement constaté que c'était un bon compromis entre capacité de généralisation et temps de calcul. Pour l'expérience Learned Error, nous utilisons la même architecture que pour le prédicteur

et l'entraînons également pour 120 époques. Nous utilisons 5 initialisations pour le Multi Inits, 5 sous-ensembles de tout le jeu d'entraînement pour le Bagging, 50 prédictions pour le Monte-Carlo Dropout, les 50 dernières prédictions du jeu d'entraînement pour le Multi Epochs. Nous utilisons également la technique Multi Networks, avec une architecture mono échelle et 4 architectures multi échelles différentes, comme proposé au chapitre 4 pour l'estimation de profondeur monoculaire.

5.7.2 Analyse : distributions de l'incertitude

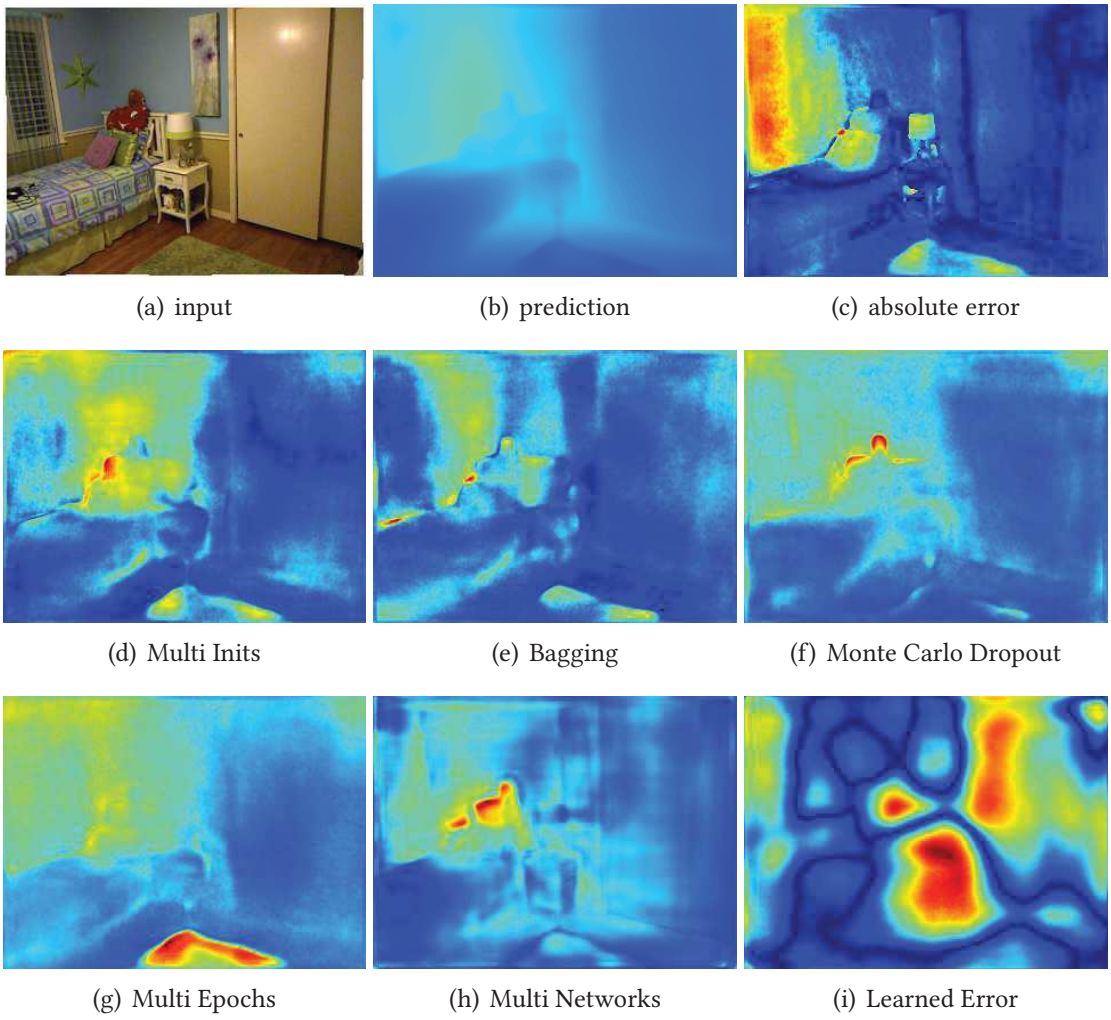
	minimum	médiane	moyenne	maximum	écart-type
MI	$4.9\ 10^{-4}$	$6.3\ 10^{-2}$	$8.3\ 10^{-2}$	1.3	$7\ 10^{-2}$
Bagging	$3.3\ 10^{-4}$	$6.1\ 10^{-2}$	$7.8\ 10^{-2}$	1.3	$6.2\ 10^{-2}$
MCD	$7.9\ 10^{-3}$	$2.8\ 10^{-2}$	$3.7\ 10^{-2}$	$4.3\ 10^{-1}$	$2.8\ 10^{-2}$
ME	$1.5\ 10^{-2}$	$5.5\ 10^{-2}$	$6.7\ 10^{-2}$	$8.7\ 10^{-1}$	$4.2\ 10^{-2}$
MN	$8.6\ 10^{-4}$	$1.2\ 10^{-1}$	$1.4\ 10^{-1}$	1.2	$8.5\ 10^{-2}$
LE	$1.2\ 10^{-10}$	$4.1\ 10^{-2}$	$5.3\ 10^{-2}$	$8.5\ 10^{-1}$	$4.8\ 10^{-2}$

Table 5.2 – Statistiques sur les différentes distributions d'incertitude obtenues pour chaque technique dans le cas de l'estimation de profondeur monoculaire.

Nous présentons dans le tableau 5.2 certaines statistiques obtenues dans le cas de l'estimation de profondeur monoculaire et calculées pour chaque technique d'estimation d'incertitude pour un prédicteur commun. Nous voulons montrer ici que différentes techniques d'estimation peuvent avoir des distributions totalement différentes avec différentes plages de valeurs. Nous voyons par exemple que Multi Inits, Bagging et Multi Networks ont des distributions étendues, leurs valeurs minimales se situant autour de 10^{-4} et leur maximum à environ 1.3. D'autre part, nous constatons que des méthodes telles que Multi Epochs ou Monte-Carlo Dropout présentent des intervalles plus petits avec une concentration plus élevée autour de la valeur moyenne. Pour cette raison, il apparaît qu'il faut prendre en compte cette hétérogénéité pour choisir une métrique appropriée si l'objectif est de comparer différentes techniques d'estimation de l'incertitude.

5.7.3 Résultats : analyse qualitative d'un échantillon

Nous nous intéressons d'abord à l'évaluation qualitative des résultats obtenus sur l'estimation de profondeur monoculaire dans le cas où le prédicteur est commun pour chaque méthode. En figure 5.6, nous montrons un exemple d'une image de test 5.6(a), de la prédiction faite par le prédicteur commun 5.6(b) et finalement de l'erreur absolue entre la prédiction et la vérité terrain 5.6(c). Notons que, si l'on se réfère à la carte d'erreur, quelques régions peuvent être définies comme étant plus notablement incertaines à savoir : les zones correspondant au tapis, à la lampe de chevet, à l'oreiller et à la fenêtre.



(a) input (b) prediction (c) absolute error

(d) Multi Inits (e) Bagging (f) Monte Carlo Dropout

(g) Multi Epochs (h) Multi Networks (i) Learned Error

FIGURE 5.6 – Un cas du jeu de test de NYU Depth v2. Première ligne : l'image de test utilisée en entrée aux côtés de la carte de profondeur prédite ainsi que la carte d'erreur correspondante ; lignes restantes : estimations d'incertitudes par 6 différentes méthodes

TABLE 5.3 – Scores des méthodes et classement selon chacune des métriques pour une image spécifique du jeu de test de NYU Depth v2

Method	Q_s	L_s	S_s	CRPS	MeRCI
Multi Inits	-3.4	-1.9	0.87	0.12	0.40
Bagging	-2.7	-1.6	0.90	0.12	0.55
Monte Carlo Dropout	-7.5	-3.3	0.61	0.13	0.37
Multi Epochs	-2.3	-1.6	0.87	0.12	0.34
Multi Networks	1.8	0.25	1.3	0.10	0.41
Learned Error	-67	-3.1	0.80	0.13	2.3

La table 5.3 montre le classement relatif des méthodes pour cette image en particulier en fonction de chaque métrique. Nous voyons que les métriques classiques de l'état de l'art s'accordent globalement sur le classement et désignent Multi Networks comme étant la meilleure méthode. D'un autre côté, MeRCI classe Multi Networks à la position 4. Selon MeRCI, la méthode d'estimation d'incertitude la plus corrélée avec la véritable erreur est Multi Epochs, laquelle est classée entre la 2e et la 4e position selon les autres métriques. De plus, le Monte-Carlo Dropout a un mauvais classement d'après les scoring rules classiques (classé 5e ou 6e) tandis que c'est la deuxième meilleure méthode lorsque l'on utilise MeRCI. Contrairement à l'exemple jouet, nous observons cette fois un clair désaccord entre les différentes métriques. Nous cherchons à approfondir ces observations pour comprendre cette discordance par une analyse visuelle.

Nous affichons les 6 estimations d'incertitude obtenues avec les différentes méthodes en figure 5.6. Seules deux méthodes montrent une grande incertitude dans la région correspondant à la fenêtre à savoir Multi Epochs et Monte-Carlo Dropout. Cette région est clairement la plus importante à marquer comme incertaine puisque c'est celle présentant le plus d'erreurs d'après la carte d'erreur que nous voyons en figure 5.6(c). De plus, Multi Epochs affiche également une grande incertitude à l'endroit où se situe le tapis tandis que le Monte-Carlo Dropout rate cet endroit. C'est la raison pour laquelle MeRCI classe ces méthodes comme les deux premières en avantageant Multi Epochs. Dans le même temps, nous voyons que Multi Networks, qui est classé premier par toutes les métriques classiques, a cependant une faible corrélation avec l'erreur réelle. En effet, il affiche une forte incertitude autour du coussin, mais rate toutes les autres localisations importantes. Multi Networks capture également une variance sur certains contours de l'image RGB qui ne sont pas présents dans la vérité terrain de profondeur, ce qui est donc inconsistant avec l'erreur commise. Comme nous l'avons vu dans le chapitre 4, cela est dû au fait que certains des différents réseaux impliqués dans les estimations d'incertitude de Multi Networks produisent des sorties plus saillantes, mais au prix de certains artefacts transférés depuis l'image RGB. Finalement, bien que nous voyons que Learned Error échoue complètement à capturer des informations pertinentes sur l'erreur, certaines scoring rules ne la classent pas à la dernière position. Elle est au contraire correctement classée par MeRCI.

5.7.4 Résultats : analyse qualitative

Les images présentées dans les figures 5.7 et 5.8 sont des résultats obtenus sur le jeu de test de NYU depth v2. Nous considérons le cas où le prédicteur est commun et seule la technique d'estimation de l'incertitude diffère d'une méthode à l'autre. Nous affichons de gauche à droite : l'image RGB, la prédiction de profondeur faite par le réseau, la vérité terrain, l'erreur absolue entre la prédiction et la vérité terrain, puis les estimations des incertitudes faites respectivement par Multi Inits, Bagging, Monte-Carlo Dropout, Multi Epochs, Multi Networks et Learned Error.

Nous avons uniquement sélectionné des échantillons contenant des zones erronées dans

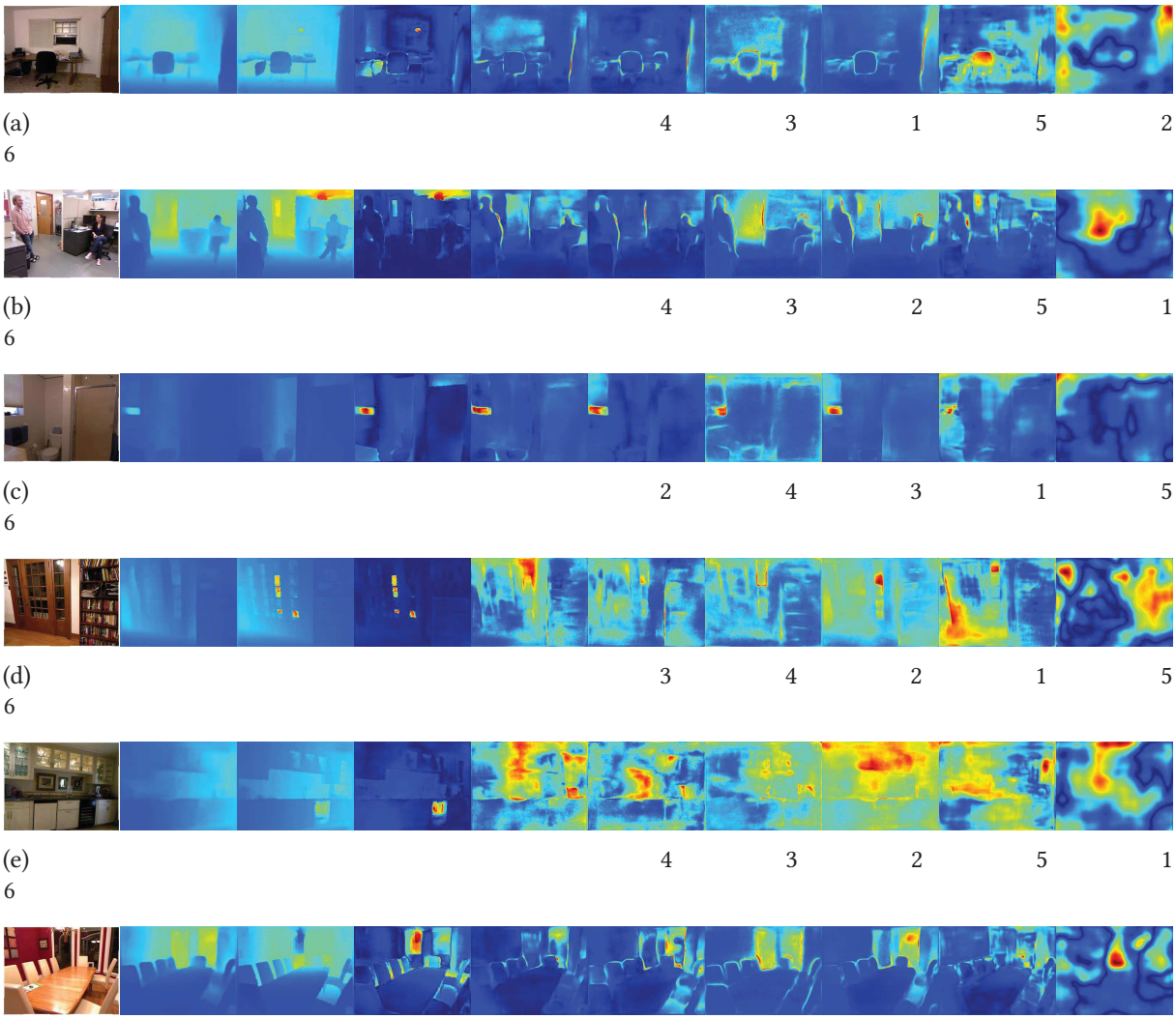


FIGURE 5.7 – Plusieurs résultats sur le jeu de données de test de NYU Depth v2 (lignes). Les colonnes de gauche à droite représentent : l'image d'entrée, la prédiction, la vérité terrain, l'erreur absolue puis les estimations d'incertitude MI, Bagging, MCD, ME, MN et LE. Pour chaque exemple, les scores MeRCI sont calculés et le classement des méthodes est affiché en dessous des estimations d'incertitude.

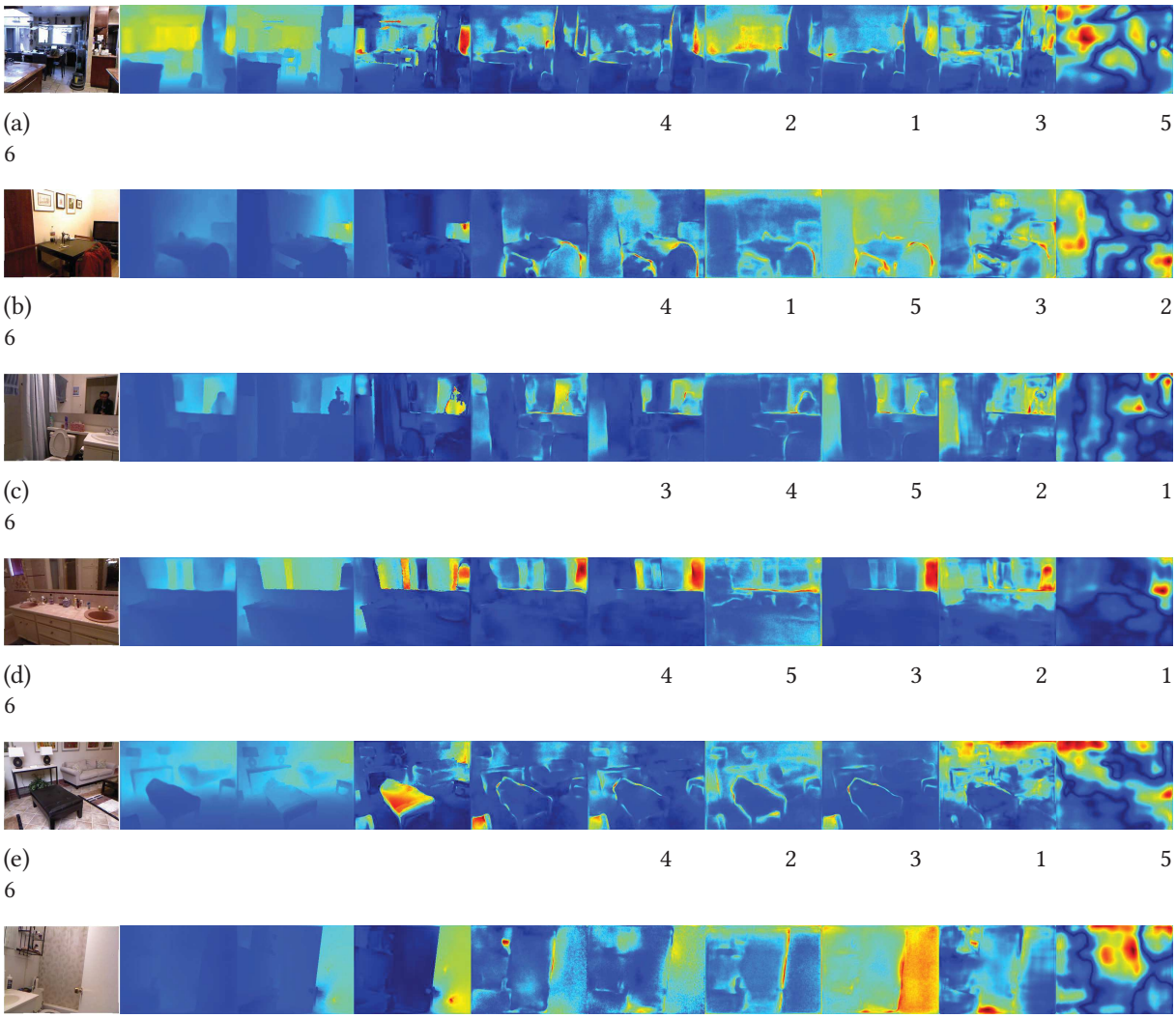


FIGURE 5.8 – Plusieurs résultats sur le jeu de données de test de NYU Depth v2 (lignes). Les colonnes de gauche à droite représentent : l'image d'entrée, la prédiction, la vérité terrain, l'erreur absolue puis les estimations d'incertitude MI, Bagging, MCD, ME, MN et LE. Pour chaque exemple, les scores MeRCI sont calculés et le classement des méthodes est affiché en dessous des estimations d'incertitude.

TABLE 5.4 – Évaluation quantitative sur le jeu de test de NYU Depth v2. A des fins de comparaison, la prédiction de profondeur est la même pour tous les estimateurs

	higher is better									lower is better			
Method	Q_s			L_s			S_s			CRPS			MeRCI
	U	G	L	U	G	L	U	G	L	U	G	L	
Oracle	-	-	-	-	-	-	-	-	-	-	-	-	0.356
Constant	-	-	-	-	-	-	-	-	-	-	-	-	1.11
MI	-3.1	-3.0	-4.3	-4.7	-3.4	-2.8	0.56	0.56	0.50	0.32	0.32	2.5	1.22
Bagging	-3.1	-3.0	-4.4	-4.8	-3.5	-2.9	0.55	0.56	0.50	0.32	0.32	2.5	1.18
MCD	-7.7	-7.5	-10	-5.7	-4.8	-4.3	0.40	0.41	0.36	0.34	0.34	2.5	1.10
ME	-2.9	-2.7	-4.1	-4.9	-3.7	-3.0	0.54	0.54	0.48	0.32	0.32	2.5	0.91
MN	-0.45	-0.37	-1.1	-3.3	-1.9	-1.6	0.73	0.74	0.67	0.30	0.30	2.5	1.18
LE	-137	-134	-169	-5.2	-4.3	-3.8	0.46	0.46	0.41	0.33	0.33	2.5	4.31

la vérité terrain. Ces images sont classées en fonction de leur ratio de valeurs aberrantes : la première image de la figure 5.7 est celle qui présente la plus petite proportion d'erreurs dans la vérité terrain (en l'occurrence à l'endroit où nous voyons la fenêtre), tandis que la dernière image de la figure 5.8 est celle avec le plus haut pourcentage d'outliers (toute la porte est erronée).

Nous avons calculé le score MeRCI (percentile 95) pour chaque technique et classé les résultats. Nous avons affiché ce classement sous chaque image, pour chaque exemple. Le but de cette analyse est double : nous voulons à la fois montrer les résultats obtenus par MeRCI pour plus d'échantillons de test, mais aussi analyser son comportement dans un cas réel où des valeurs aberrantes sont présentes dans la vérité terrain.

Nous voyons que lorsque le taux de données aberrante est faible, comme dans la figure 1.a, le paramètre de percentile permet de ne considérer que les pixels "valides" pour le calcul du score. Ainsi, le classement des différentes techniques est basé sur le reste de l'estimation de l'incertitude. Ici, la meilleure méthode met en évidence les contours des objets et ne répond pas fortement autour de la fenêtre.

Le classement reste fiable et non biaisé par le ratio de données aberrantes de l'image tant que celui ci est en accord avec le percentile choisi. Cependant, lorsque le pourcentage de valeurs aberrantes devient trop élevé, le score de la méthode est basé sur la similitude avec la vérité terrain et donc avec des données erronées. Ceci est clairement visible dans la figure 2.f, où la deuxième technique met en évidence les pixels de la porte comme étant mal prédits, ce qui est en accord avec la vérité terrain terrain qui s'avère finalement être fausse pour ces pixels tandis que la prédiction était bonne.

5.7.5 Résultats : évaluation quantitative pour un prédicteur commun

Les résultats quantitatifs sont présentés en table 5.4. Étant donné que nous voulons étudier l'impact des différents a priori de distribution sur les différentes métriques, nous calculons analytiquement les valeurs des scoring rules pour trois densités de probabilité différentes : avec un a priori Uniforme (noté **U**), avec un a priori Gaussien (noté **G**) et avec un a priori de Laplace (noté **L**). Nous constatons que, bien que les a priori choisis on des queues de distribution très différentes, le choix de la densité de probabilité n'influence que très peu le classement relatif des différentes méthodes. On voit par exemple que dans le cas d'une distribution de Laplace, il devient même impossible de fournir un classement entre les méthodes avec la CRPS (qui par ailleurs ne montre que de faibles écarts pour tous les a priori choisis).

Tous ces scores sont basés sur le calcul de distributions prédictives et pourtant nous constatons que la forme de la distribution n'a pas d'impact majeur sur l'évaluation de la qualité de la méthode. D'un autre côté, MeRCI ne repose sur aucun a priori et elle est donc bien plus directe pour juger les différentes techniques, notamment celles issues du *deep learning*.

De même que pour l'évaluation qualitative d'une image que nous venons d'effectuer, nous voyons que Multi Networks est de loin la meilleure méthode selon les métriques classiques de l'état de l'art. D'après (Gneiting et al., 2007), cela signifie qu'elle à la meilleure 'sharpness' étant donné la calibration. Cependant, MeRCI montre que cette technique d'estimation d'incertitude est moins corrélée à l'erreur réelle qu'une prédiction d'incertitude constante (et donc non informative).

Ici encore, MeRCI met en évidence Multi Epochs comme étant la méthode donnant la meilleure prévision (en fait c'est même la seule à avoir un meilleur score que celui obtenu avec une incertitude constante). Ce classement privilégié est dû au fait que l'entraînement d'un réseau conséquent sur une tâche complexe et avec une routine de minimisation par gradient stochastique produit naturellement de oscillations autour d'un minimum. De plus, au cours des dernières époques, le réseau se concentrera sur l'apprentissage de ce qui est encore améliorable, c'est-à-dire les régions présentant les plus fortes erreurs. Ainsi, l'utilisation de l'écart type des estimations obtenues au cours des dernières époques permet de capturer ces variations et de mettre en évidence les régions les plus sujettes aux erreurs.

5.7.6 Résultats : évaluation quantitative pour des prédicteurs propres

Dans la Table 5.5 nous montrons les résultats de l'évaluation quantitative sur le jeu de test de la base de données NYU Depth v2, où le score de chaque méthode est calculé en utilisant son propre couple prédiction/incertitude. L'utilisation de prédicteurs propres pour chaque méthode nous permet de comparer les scores réels de chaque technique, en fonction de leur

TABLE 5.5 – Évaluation quantitative sur le jeu de test de NYU Depth v2. Pour chaque méthode, nous calculons son propre couple prédiction/incertitude.

	à maximiser			**à minimiser**	
Method	Q_s	L_s	S_s	CRPS	MeRCI
Multi Inits	-3.0	-3.4	0.57	0.31	1.20
Bagging	-3.0	-3.5	0.55	0.32	1.19
Monte Carlo Dropout	-7.7	-5.0	0.37	0.37	1.20
Multi Epochs	-2.8	-3.7	0.53	0.32	0.91
Multi Networks	-0.63	-2.3	0.66	0.32	1.20
LE	-81	-4.6	0.43	0.33	4.70
Learned Attenuation	-1.8	-2.9	0.62	0.31	1.43
Discretization	0.15	-2.2	0.40	0.88	1.60

calibration et de leur 'sharpness'. En effet, fixer la prédiction pour qu'elle soit identique pour chaque méthode biaise l'analyse de la 'sharpness' sous contrainte de calibration : les méthodes basées sur une moyenne de leurs estimations présentent des calibrations différentes.

Cela nous permet également d'inclure d'autres techniques d'estimation d'incertitude dont nous avons parlé, mais qui n'étaient pas comparables aux précédentes parce qu'elles sont basées sur des fonctions de coût différentes. Nous pouvons donc maintenant ajouter Learned Attenuation et Discretization à cette comparaison. Puisque nous avons montré dans la section 5.7.5 que l'a priori sur la distribution prédictive n'avait pas d'impact majeur sur le classement des méthodes, nous avons choisi de ne présenter ici que les résultats calculés avec un a priori Gaussien.

Les résultats sont très similaires à ceux présentés dans le tableau 5.4 où le prédicteur était commun à toutes les méthodes. Cela signifie que, pour cette étude, les calibrations propres calculées pour chaque méthode sont proches les unes des autres. Par conséquent, nous voyons que Multi Networks est la meilleure méthode selon les métriques classiques et Monte-Carlo Dropout est la dernière trois fois sur quatre. D'autre part, MeRCI met en avant Multi Epochs comme étant le meilleur estimateur pour cette tâche et Learned Error comme étant le pire. Ces résultats sont cohérents avec ceux obtenus dans la section 5.7.5. Parmi les deux nouvelles techniques évaluées, on constate que Discretization atteint de très bons scores pour chacune des métriques classiques tandis qu'elle est classée avant-dernière par MeRCI. La méthode Multi Epochs reste la meilleure selon notre métrique.

5.8 Conclusions

Ce chapitre a permis de dresser un bilan des métriques existantes utilisées pour évaluer l'incertitude prédictive ainsi qu'analyser le comportement de ces différentes métriques sur des

tâches de régression, d'abord dans le cas d'un exemple jouet et ensuite sur le cas concret de l'estimation de profondeur monoculaire. Nous avons ensuite proposé une nouvelle métrique, le Mean Rescaled Confidence Interval (MeRCI), également dédiée à l'évaluation de l'incertitude prédictive. Nous avons d'abord montré que la métrique proposée a un comportement sain via l'exemple jouet puisqu'elle classe correctement les différentes méthodes évaluées, tout en donnant de la latitude pour être robuste aux données aberrantes si nécessaire (grâce à son paramètre de percentile). Ensuite, nous montrons que la métrique proposée est étroitement liée à l'erreur réelle de prédiction et par conséquent plus fiable que les métriques précédentes, dans le cas particulier de l'estimation de profondeur monoculaire. De plus, nous avons vu que contrairement aux autres métriques, MeRCI a l'avantage d'être directement applicable aux tâches de régression par *deep learning* sans nécessiter d'a priori sur la forme de la distribution prédite.

Chapitre 6

Complétion de profondeur et estimation de confiance

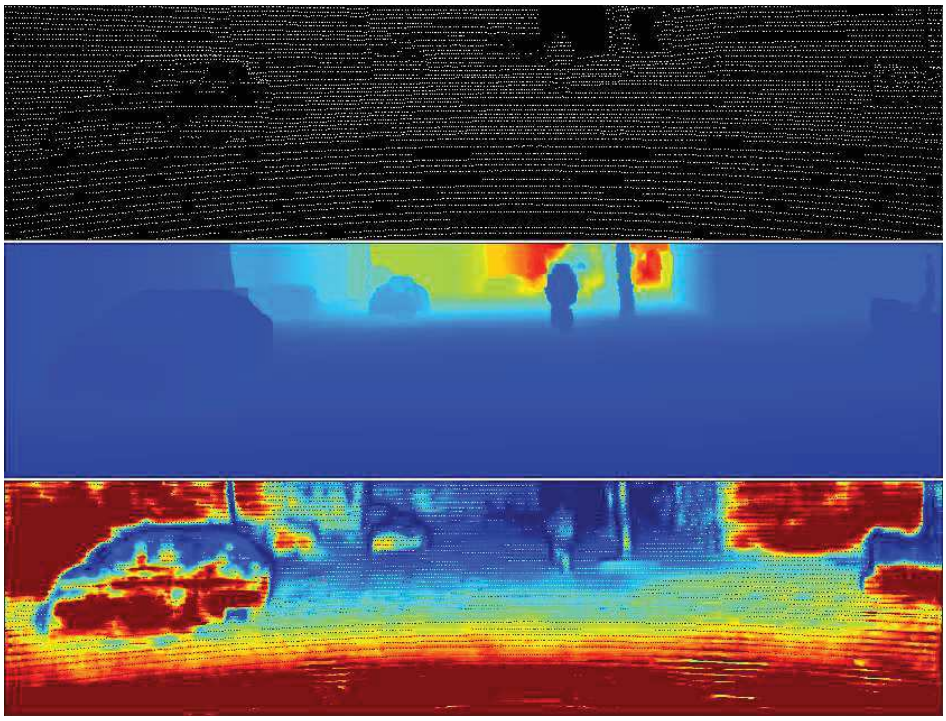


FIGURE 6.1 – Un exemple de notre algorithme de complétion de profondeur. En n'utilisant qu'un scan LiDAR en entrée (haut) nous sommes capables de prédire une carte de profondeur dense (milieu) ainsi qu'une carte de confiance associée (bas)

Tandis que les réseaux de neurones convolutionnels actuels sont conçus pour opérer sur des données denses, les applications de vision par ordinateur requièrent souvent l'utilisation d'entrées parcimonieuses, telles que des scans LiDAR. Par conséquent, les CNN profonds doivent être repensés de manière à gérer correctement cette parcimonie. Suivant cette direction, nous proposons dans ce chapitre un nouveau bloc de convolution qui vise à la fois à exploiter la parcimonie des données d'entrée tout en propageant une estimation de confiance au travers du réseau s'il existe un a priori sur la certitude des données fournies en entrée, comme dans le cas des scans LiDAR. En fin de compte, cela nous permet d'obtenir à la fois une prédiction de profondeur dense ainsi qu'une carte de confiance correspondante, par pixel. Nous présentons un ensemble d'expériences sur le jeu de données KITTI Depth Completion et nous démontrons empiriquement que notre stratégie de complétion de profondeur permet non seulement d'atteindre les performances de l'état de l'art mais produit également des cartes de confiances pertinentes. Nous évaluons les estimations de confiance obtenues à la fois quantitativement et qualitativement, démontrant qu'elles sont informatives et corrélées à l'erreur véritable commise par le réseau.

6.1 Introduction

Pour de nombreuses applications de vision par ordinateur, comme le tracking vidéo ou la conduite autonome, connaître la géométrie de la scène est un enjeu majeur. Par exemple, dans le cadre d'un véhicule autonome, des capteurs type LiDAR sont souvent utilisés pour mesurer la distance des objets rencontrés. Tandis que ces capteurs produisent des sorties très précises, les cartes de profondeur obtenues sont par nature parcimonieuses et donc difficiles à exploiter avec les architectures de réseaux de neurones actuels qui reposent sur des données denses.

Différents types de parcimonies peuvent être rencontrées sur des données de profondeur et nous pouvons globalement les classer en deux catégories : les cartes de profondeur avec des zones entières de données non observées au milieu d'une annotation dense, on les rencontre par exemple en stéréovision lorsqu'on cherche la profondeur d'une région partiellement occultée Wang et al. (2008). Les méthodes qui visent à prédire ces valeurs non observées appartiennent au domaine de l'inpainting de profondeur Xue et al. (2017). D'un autre côté, une parcimonie structurée peut être observée en sortie de certains capteurs comme dans le cas du LiDAR. La prédiction de profondeur utilisant ce type de données parcimonieuses en entrée appartient au domaine de la complétion de profondeur.

Dans ce chapitre, nous nous intéressons en particulier au second cas, à savoir l'utilisation de scans LiDAR parcimonieux comme entrée d'un algorithme de densification (ou complétion) de profondeur. Avec les avancées du *deep learning*, des travaux récents ont émergé et permettent d'adresser cette tâche de complétion. Par exemple, une astuce fréquemment utilisée est d'encoder les données de profondeur manquantes en entrée pour artificiellement densifier le LiDAR et l'utiliser ensuite comme une image avec des approches de *deep learning* classiques Ma et Karaman (2018). On trouve également d'autres approches spécifiques à la gestion de données parcimonieuses en entrée Uhrig et al. (2017).

En parallèle, il est intéressant de noter que dans le cas de la complétion de profondeur, les entrées utilisées sont en fait une partie de la vérité terrain que nous cherchons à modéliser. Par conséquent, nous aimerions tirer parti de ces données pour propager une estimation de confiance le long du réseau, en commençant par assigner une confiance maximum pour les points LiDAR présents en entrée et minimum pour les autres points sur lesquels nous n'avons pas d'information.

Ainsi, nous proposons une architecture de réseau basée sur un nouveau bloc de convolution spécifique, inspiré par des travaux précédents de *deep learning* sur des CNN parcimonieux. Notre bloc, que l'on appelle la convolution ajustée, est capable de prendre des données parcimonieuses en compte et de propager une confiance le long du réseau en fonction de la certitude a priori des données d'entrée (voir Figure 6.1). Nos contributions sont les suivantes :

— Nous proposons une architecture comprenant notre nouveau bloc de convolution, la

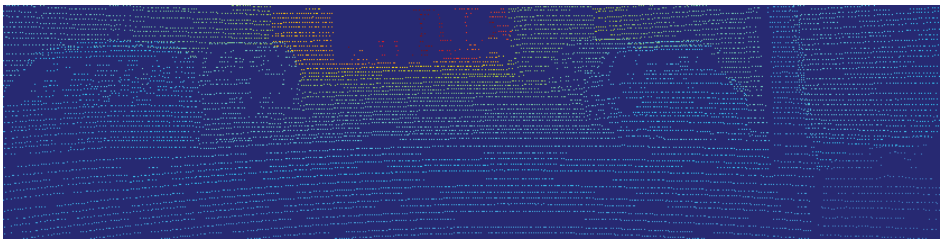


FIGURE 6.2 – Un exemple de scan LiDAR artificiellement densifié où les pixels non observés par le LiDAR ont été encodés avec des 0. La colormap va de 0 mètre (en bleu) à 80 mètres (en rouge)

convolution ajustée, conçue spécifiquement pour des entrées parcimonieuses. Nous démontrons empiriquement que cette nouvelle architecture atteint les performances de l'état de l'art pour la tache de complétion de profondeur en utilisant uniquement des scans LiDAR en entrée. Nous proposons également une fusion avec des descripteurs RGB pour la tâche de complétion de profondeur guidée qui améliore encore un peu les résultats.

— Pour l'apprentissage nous proposons une fonction de coût spécifique inspirée par Kendall et Gal (2017); Gurevich et Stuke (2017). Cette fonction, en combinaison avec notre convolution ajustée, nous permet de prédire à la fois une carte de profondeur dense ainsi qu'une carte de confiance par pixel.

— Nous proposons plusieurs critères d'évaluation pour analyser nos prédictions de confiance (incluant la MeRCI) démontrant ainsi la pertinence de nos résultats en termes de corrélation avec l'erreur réelle du réseau.

Ces contributions sont empiriquement évaluées sur le benchmark KITTI Depth Completion en utilisant un ensemble de métriques largement utilisées pour l'évaluation de cette tâche.

Le reste du chapitre est structuré de la manière suivante : la section 6.2 introduit les travaux relatifs au domaine de complétion de profondeur ainsi que ceux abordant la notion d'incertitude dans les réseaux de neurones. Elle est suivie par la section 6.3 où nous décrivons en détail l'approche que nous proposons, en particulier notre architecture basée sur la convolution ajustée et permettant d'obtenir des paires de prédictions de profondeur et de confiance. Nous présentons ensuite nos configurations expérimentales et nos résultats, y compris des analyses quantitatives et qualitatives pour à la fois pour la profondeur et pour la confiance en section 6.4. Enfin, nous concluons en section 6.7.

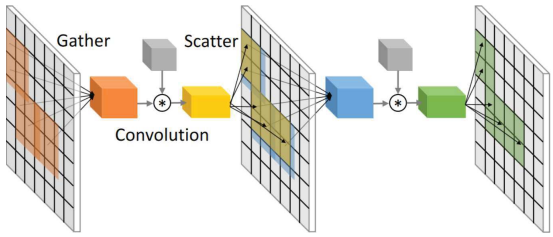


FIGURE 6.3 – Schéma de la convolution proposée par Ren et al. (2018) pour gérer des données parcimonieuses. Une convolution standard est appliquée aux sous localisations les plus denses qui sont d'abord regroupées ensemble. Les résultats de la convolution sont ensuite replacés aux endroits correspondants du descripteur de sortie

6.2 Littérature associée

6.2.1 La complétion de profondeur

Dans la littérature de complétion de profondeur par méthodes de *deep learning*, nous avons trouvés deux approches principales pour gérer la parcimonie des données d'entrée : d'un côté les méthodes utilisant des convolutions standards, de l'autre celles qui utilisent des convolutions spécifiques, spécifiquement conçues pour gérer les données parcimonieuses.

Tout d'abord, il est important de noter que les convolutions standards ont été initialement créées pour opérer sur des données denses. Ainsi, une modification est nécessaire pour les rendre compatibles avec des données parcimonieuses. Une astuce classique consiste à encoder chaque pixel non observé par une valeur unique caractéristique (par exemple 0, voir figure 6.2). En utilisant cette astuce, certaines méthodes utilisent directement des convolutions standards sur ces données artificiellement densifiées Ma et Karaman (2018); Jaritz et al. (2018). Ces méthodes comptent sur la phase d'entrainement du réseau pour que les convolutions apprennent d'elle même comment gérer la parcimonie des données d'entrée.

Une autre façon de faire consiste à utiliser un type spécifique de convolution, dédiée au filtrage des données parcimonieuses. A notre connaissance, de telles convolutions sont déclinées sous deux formes différentes dans la littérature. La première utilise la parcimonie des données principalement pour réduire le coût computationnel de la convolution. Par exemple, Ren et al. (2018) proposent une décomposition par blocs des descripteurs parcimonieux. Ils sélectionnent ensuite uniquement les sous localisations les plus denses sur lesquels ils appliquent des convolutions standards avant de réinjecter les résultats dans les descripteurs de sortie (voir figure 6.3). Cela permet de réduire le nombre total d'opérations et par conséquent d'abaisser la complexité computationnelle. Cependant, la décomposition par bloc s'accommode difficilement de grande variations de densité sur les données.

Une deuxième catégorie vise principalement à obtenir les meilleures performances possible, en utilisant éventuellement la parcimonie comme source d'information supplémentaire. Une

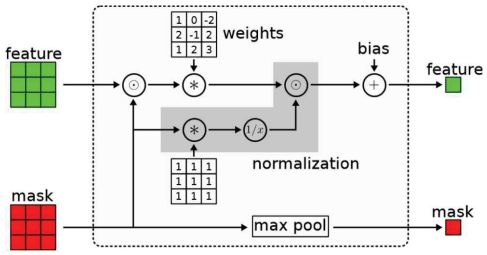


FIGURE 6.4 – Schéma de la Sparse Convolution proposée par Uhrig et al. (2017)

convolution spécifique, appelée Sparse Convolution, a d'abord été introduite par Uhrig et al. (2017) pour adresser ce problème. Ils proposent un bloc de Sparse Convolution (figure 6.4) qui prend à la fois des descripteurs issus d'un signal (d'un scan LiDAR en entrée par exemple) ainsi qu'un masque binaire indiquant les localisations d'unités valides (positions des pixels observés). La convolution prend uniquement en compte l'information associée aux localisations valides. Le masque binaire est propagé pour permettre de ne pas perdre la connaissance sur la position initiale des pixels non valides, en particulier si un biais est présent dans la convolution. Chaque sortie de convolution est ensuite normalisée en fonction du nombre d'entrées valides dans la fenêtre de convolution. Ainsi, les valeurs de sortie restent dans le même intervalle quel que soit le nombre d'entrées valides observées. Cependant, étant donné que le masque binaire est propagé en utilisant des max pooling, il finit par devenir complètement dense. Sa sortie ne peut donc pas être utilisée comme une mesure de confiance correspondant aux prédictions de profondeur.

Une autre opération a été introduite en 1993 par Knutsson et Westin (1993), appelée la Normalized Convolution. Dans ce papier, les auteurs proposent une opération de convolution spécifique capable de gérer des signaux incomplets et incertains. Ils formalisent leur opération dans un contexte théorique sur lequel nous reviendrons par la suite. L'approche globale est similaire à celle utilisée pour la Sparse Convolution, avec à la fois un signal d'entrée et un masque de confiance indiquant la position des pixels valides (considérés sûrs). Les stratégies de convolution et de normalisations, elles, diffèrent légèrement et seront expliquées en détail dans la section 6.3.

L'opération proposée par a été étendue pour être compatible avec des approches de *deep learning* par Eldesokey et al. (2018) et Hua et Gong (2018). Dans ces travaux, le masque de confiance utilisé en entrée est binaire et est soit propagé à travers le réseau en utilisant des max pooling Hua et Gong (2018), comme c'est fait pour la Sparse Convolution, soit en utilisant des convolutions standards Eldesokey et al. (2018). Le dernier cas permet alors de propager un descripteur continu à travers le réseau qui peut être assimilé à une carte de confiance. Ainsi, lorsqu'une partie des données de la vérité terrain est utilisée comme entrée, comme c'est le cas pour la complétion de profondeur, la Normalized Convolution est adaptée et permet de calculer à la fois une prédiction de profondeur ainsi que la carte de confiance correspondante.

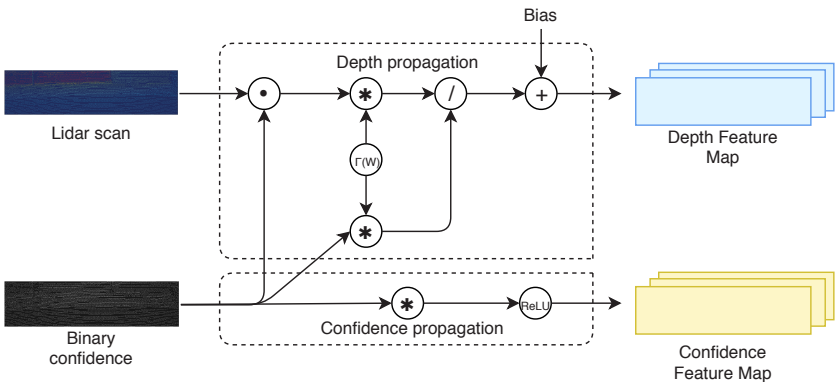


FIGURE 6.5 – Notre convolution ajustée. Nous schématisons ici la première convolution ajustée du réseau qui prend donc en entrée un scan LiDAR et un masque binaire indiquant l'emplacement des pixels valides du scan LiDAR

Enfin il est également intéressant de noter que des études comparatives entre les convolutions classiques, la Sparse Convolution et la Normalized Convolution ont été menées sur des réseaux de neurones peu profonds Uhrig et al. (2017); Hua et Gong (2018). Elles révèle que la Sparse Convolution et la Normalized Convolution montrent de meilleurs résultats que les convolutions classiques pour la tâche de complétion de profondeur. Cependant, rien n'indique que cette déclaration reste valable pour des architectures plus profondes, comme en attestent les résultats d'autres publications Ma et Karaman (2018); Jaritz et al. (2018).

Outre le design de la convolution, la complétion de profondeur peut également être améliorée en utilisant des images RGB pour guider l'apprentissage. En effet, les contours des objets ou les variations de textures peuvent servir d'indices complémentaires à l'estimation de la profondeur. C'est pourquoi on trouve plusieurs méthodes proposant la fusion de descripteurs extraits depuis des images RGB avec des descripteurs issus de scans LiDAR pour améliorer les performances de complétion. Différentes sortes de fusion ont été proposées : au début du réseau, là où les descripteurs sont encore d'assez bas niveau Ma et Karaman (2018) ; au milieu du réseau avec des descripteurs de plus haut nieau sémantique Li et al. (2016); Hua et Gong (2018) ou encore à différentes étapes et échelles de l'architecture Hui et al. (2016); Huang et al. (2018); Wang et al. (2018). Cependant, la valeur ajoutée de la fusion RGB est conditionnée par la densité des informations de profondeur utilisées en entrée. Il y a un compromis entre le coût supplémentaire lié à la fusion RGB et l'amélioration des performances. Avec un LiDAR à 64 faisceaux Geiger et al. (2013), les méthodes de fusion démontrent que le guidage RGB apporte une amélioration systématique mais limitée Jaritz et al. (2018); Huang et al. (2018) sur le benchmark KITTI Depth Completion.

6.2.2 L'incertitude dans les réseaux de neurones

La confiance en une prédiction est une notion importante qui existe depuis des décennies et nous pensons qu'elle devrait aller de pair avec tout type de prédicteur. Cependant, la littérature du *deep learning* manque encore de méthodes appropriées pour estimer une telle confiance en parallèle de la prédiction spécifique à la tâche demandée au réseau. La confiance pouvant être considérée comme l'inverse de l'incertitude, nous discutons ici de travaux récents qui traitent du problème de l'estimation de l'incertitude en *deep learning*.

Selon Der Kiureghian et Ditlevsen (2009), l'incertitude d'une prédiction peut être divisée en deux différents types : l'incertitude dite aléatoire et l'incertitude dite épistémique. En supposant que les paramètres d'un réseau soient des variables générées à partir d'une distribution de probabilité, l'incertitude épistémique évalue la variance des prévisions en fonction de différents échantillonnages des paramètres. L'incertitude aléatoire, elle, modélise le bruit des observations pour un ensemble donné de paramètres.

En *deep learning*, le formalisme Bayésien est bien connu pour modéliser l'incertitude épistémique en transformant les réseaux de neurones déterministes classiques, pour lesquels chaque paramètre est un scalaire, en modèles probabilistes où les paramètres sont remplacés par des distributions de probabilité. Cependant l'entrainement des réseaux Bayésiens est plus long et, en pratique, l'inférence exacte est intraitable. Malgré les nombreuses méthodes développées pour gérer l'entrainement et approximer l'inférence Gal et Ghahramani (2016); Blundell et al. (2015); Graves (2011); Kingma et al. (2015), le formalisme Bayésien a toujours un impact non négligeable sur le coût computationnel et sur le temps d'inférence. En outre, Kendall et Gal (2017) soutiennent que la valeur ajoutée de l'incertitude épistémique est limitée lorsque des régimes de données volumineux sont impliqués dans l'entrainement. Dans les applications de *deep learning* où des capteurs dédiés sont capables de générer directement des vérités terrain pour superviser des tâches d'apprentissage telles que la complétion de profondeur, il n'est pas rare d'être dans un tel régime de fonctionnement. Par conséquent, la plupart des prédictions du réseau pourraient être décrites par une incertitude aléatoire. Celle-ci peut être régressée directement durant la procédure d'apprentissage. Par exemple, Gurevich et Stuke (2017) proposent une méthode pour quantifier l'incertitude aléatoire à l'aide de deux réseaux distincts, un régresseur et un quantifieur d'incertitude, en interaction via une fonction de coût jointe. Par ailleurs, la fonction de coût proposée par Kendall et Gal (2017) peut être considérée comme un cas particulier de cette perte jointe.

Inspirés par ces travaux et avec la volonté de modéliser l'incertitude aléatoire de manière dense sur toute la prédiction, nous proposons un seul réseau, basé sur une convolution spécifique et capable d'estimer à la fois une carte de profondeur dense à partir de données LIDAR parcimonieuses ainsi qu'une carte de confiance associée.

6.3 Méthode

Dans cette section nous expliquons comment nous avons conçu la convolution ajustée que nous proposons. Pour ce faire, nous commençons d'abord par décrire les détails de l'opération proposée par Knutsson et Westin (1993). Ensuite, nous présentons l'architecture que nous avons choisi d'implémenter d'après les résultats présents sur le benchmark KITTI Depth Completion et la fonction de coût particulière que nous avons utilisée pour obtenir à la fois des prédicions de profondeur et de confiance. Enfin, nous expliquons comment effectuer la fusion RGB pour apprendre la tâche de complétion guidée dans notre cas.

6.3.1 La Normalized Convolution

Pour comprendre clairement le cadre mathématique dans lequel s'inscrivent les méthodes de l'état de l'art ainsi que nos contributions, nous devons dans un premier temps commencer par expliquer la Normalized Convolution telle qu'elle est définie par Knutsson et Westin (1993). Dans cet article de 1993, le but de Knutsson et Westin (1993) est de proposer une méthode de filtrage permettant l'interpolation de signaux incomplets et incertains.

Il arrive que la représentation des signaux incomplets puisse être une source d'ambiguïté sur le signal. En effet, l'encodage d'une information manquante ne doit pas être confondu avec celui d'une information connue d'intensité particulière. Par exemple dans le cas d'une image incomplète avec une partie des pixels manquants, il faut pouvoir distinguer les pixels inconnus, souvent encodés par des 0, des pixels connus mais dont l'intensité lumineuse est nulle, et donc également encodés par des 0.

C'est pour adresser cette problématique dans le cas du filtrage que Knutsson et Westin (1993) proposent une opération de convolution particulière où le filtre de convolution et le signal sont décomposés en faisant apparaître une certitude sur la donnée. Considérons X un signal d'entrée et W les paramètres d'une convolution, alors la convolution U standard peut s'écrire de la façon suivante :

$$U = W \circledast X \tag{6.1}$$

où $\circledast$ représente l'opération de convolution. Knutsson et Westin (1993) proposent de réécrire cette opération comme suit :

$$U = aB \circledast cT \tag{6.2}$$

avec $W = aB$ et $cT = X$. Ici, le signal d'entrée X est décomposé comme le produit d'un tenseur T représentant le signal et d'une fonction scalaire positive c représentant la confiance, ou certitude, associée à chaque valeur de T. De même, les paramètres de la convolution W

sont décomposés comme le produit de B, un tenseur représentant la base de l'opérateur de filtrage de la convolution, par a une fonction scalaire positive représentant ce qu'ils appellent l'applicabilité, soit l'équivalent d'une certitude associée à chaque valeur de B.

Puisque le nombre de données valides que l'on convolue est inconnu et varie lorsque le signal d'entrée est incomplet, Knutsson et Westin (1993) proposent de normaliser cette convolution et forment ainsi la Normalized Convolution U_N définie par :

$$U_N = \frac{aB \circledast cT}{aBB^* \circledast c} \tag{6.3}$$

où B^* est le conjugué de B (pour traiter les cas où la base B choisie est complexe). Notons que dans le cas où il n'y a pas d'incertitude cela revient à normaliser U par les poids de la convolution W.

Originalement pour effectuer cette opération, la fonction d'applicabilité et la base étaient choisies a priori en fonction de l'application visée tandis que la confiance c dépendait des connaissances a priori sur les données d'entrée. En *deep learning*, on veut apprendre les paramètres de cette Normalized Convolution pour qu'ils s'adaptent au mieux à la tâche à optimiser. Il faut donc contraindre les paramètres des convolutions à être égaux au produit aB. En pratique on se place dans le cas simple où la base B est fixée et égale à un tenseur de 1, ce qui nous laisse à apprendre la fonction d'applicabilité a lors de la phase d'entrainement Hua et Gong (2018); Eldesokey et al. (2018). Il faut donc contraindre les poids à être positifs.

6.3.2 Notre convolution ajustée

Inspirés par ces travaux, nous présentons notre convolution ajustée, que nous illustrons en figure 6.5. La fonction d'applicabilité a correspond donc aux paramètres du réseau, c'est à dire l'ensemble des poids W. Parce que l'applicabilité doit rester une fonction positive Knutsson et Westin (1993), la positivité des poids de la convolution doit être garantie. Ainsi, nous appliquons une fonction softplus $\Gamma(x) = \ln(1 + \exp(x))$ sur les poids de la convolution, suivant Eldesokey et al. (2018). On a alors $A = \Gamma(W)$. D'après l'équation (6.3), notre propagation de profondeur devient donc :

$$Y_N = \frac{\Gamma(W) * (cT)}{\Gamma(W) * c} \tag{6.4}$$

Nous ajoutons également un terme de biais pour augmenter la capacité de notre modèle que nous plaçons après la convolution définie en (6.4) pour ne pas perturber la normalisation. Comme discuté en section 6.2, les scans LiDAR en entrée sont densifiés artificiellement en encodant les pixels non observés. Admettant que toutes les mesures LiDAR ont la même certitude(maximum), nous construisons un masque de confiance binaire contenant des 1 aux emplacements valides (dont la profondeur est connue) et des 0 ailleurs. Puisqu'il n'y a pas de

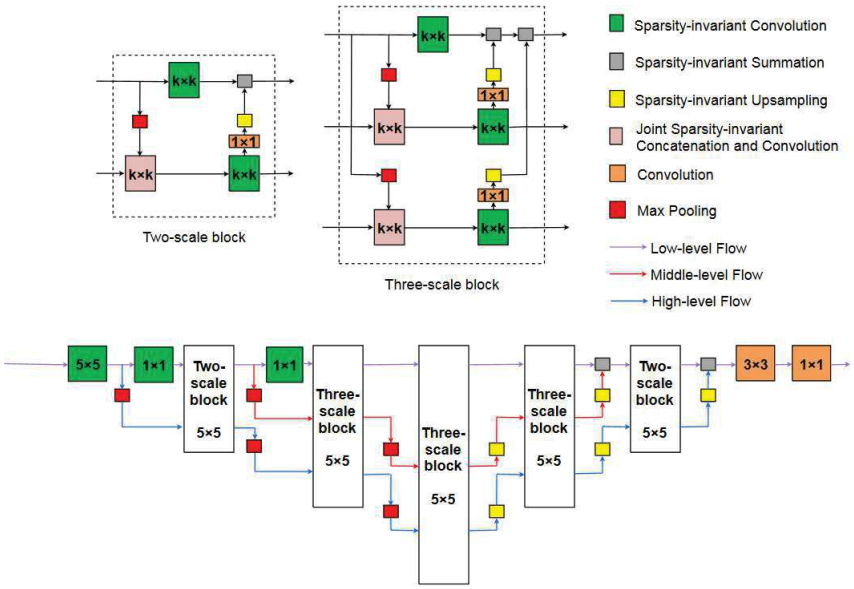


FIGURE 6.6 – Schéma de l'architecture du réseau HMS-Net proposé par Huang et al. (2018)

biais sur la branche de confiance et l'entrée considérée est simplement binaire, nous optons pour l'utilisation d'une convolution classique pour la propagation de la confiance, en utilisant une activation ReLU pour garantir sa positivité.

6.3.3 L'architecture

Avec ces deux flux en parallèle, la convolution ajustée a un coût computationnel plus élevé qu'une convolution standard. Pour cette raison, et également parce que la valeur ajoutée de convolutions spécialement dédiées aux données parcimonieuses n'a pas été prouvée dans le cas d'architectures très profondes Ma et Karaman (2018); Jaritz et al. (2018), nous choisissons d'utiliser une architecture peu profonde pour effectuer la tâche de complétion de profondeur. Nous utilisons le réseau HMS-Net Huang et al. (2018) qui, au moment où nous écrivons, est la meilleure implémentation peu profonde sur le benchmark KITTI Depth Completion qui n'utilise que des scans LiDAR en entrée. C'est une architecture à 3 différentes échelles combinant des descripteurs calculés à différents champs réceptifs, illustrée en figure 6.6.

Nous adaptons ce réseau pour notre utilisation. Tout d'abord, le HMS-Net a été conçu avec des Sparse Convolutions et des convolutions standards : nous substituons donc chacune de ces convolutions par des convolutions ajustées. De plus, nous effectuons les sous-échantillonnage en utilisant des couches de max pooling standard pour la branche correspondant aux descripteurs de confiance. Pour les descripteurs de profondeur, nous sélectionnons les emplacements correspondant aux maximums de confiance. Lorsque nous utilisons des couches de sur-échantillonnage nous effectuons des interpolations bilinéaires. Il n'y a pas de sous-échantillonnage sur les branches du HMS-Net correspondant aux descripteurs de bas niveau tandis qu'on trouve 2 sous-échantillonnages sur la branches des descripteurs de niveau intermé-

diaire et 4 sur celle de haut niveau. Les auteurs du HMS-Net estiment en effet que l'estimation de profondeur est une tâche qui implique principalement des descripteurs de bas niveau. Ainsi, la branche qui contient les descripteurs de bas niveau est la principale, tandis que les deux autres sont utilisés pour apporter de l'information multi-échelles et toutes sont fusionnées ensemble avant la couche finale permettant de produire les prédictions.

6.3.4 La fonction de coût

Pour pouvoir obtenir un couple profondeur/confiance en sortie, nous proposons également une nouvelle fonction de coût pour l'apprentissage, que nous appellons fonction de coût jointe. Inspirés par Kendall et Gal (2017); Gurevich et Stuke (2017), nous apprenons à la fois à régresser la profondeur et à modéliser l'inverse de l'incertitude aléatoire (autrement appelée confiance). Soit S l'ensemble des points valides de la vérité terrain, $log(C_{(u,v)})$ la log-confiance prédite, $y_{(u,v)}$ la vérité terrain de profondeur et $\hat{y}_{(u,v)}$ la profondeur prédite. Nous définissons notre fonction de coût jointe $\mathcal{L}$ comme suit :

$$\mathbb{L}_p = \left\| y_{(u,v)} - \hat{y}_{(u,v)} \right\|_p \tag{6.5}$$

$$R = log(C_{(u,v)}) \tag{6.6}$$

$$\mathcal{L} = \frac{1}{card(S)} \sum_{u,v \epsilon S} C_{(u,v)} \cdot \mathbb{L}_p - \lambda \cdot R \tag{6.7}$$

où $\mathbb{L}_p$ est l'erreur de régression, R un terme de régularisation et λ son coefficient de pondération. La régularisation permet de prévenir le cas où les confiances en sortie sont égales à 0. La norme p est à remplacer par l'erreur de régression souhaitée. Dans la suite, nous nous intéressons au cas des normes $\mathbb{L}_1$ et $\mathbb{L}_2$.

Le terme de gauche est donc le produit de la confiance par l'erreur de régression. À travers cette multiplication, la confiance agit comme une pondération sur l'erreur de régression (et donc sur le gradient de l'erreur) ce qui impacte la vitesse d'apprentissage, à la fois globalement et relativement. D'abord globalement, parce que lorsque λ décroît, la valeur de la confiance moyenne décroît également donc la vitesse d'apprentissage diminue globalement. Et relativement parce que plus l'entropie de la distribution de la confiance est grande, plus l'impact sur la vitesse d'apprentissage va être varié en fonction des localisations spatiales. Le choix de lambda contrôle donc la confiance moyenne et l'entropie de la distribution, impactant ainsi l'apprentissage. Nous discuterons plus en détail des conséquences du choix de λ par la suite.

En pratique, nous prédisons la log confiance pour améliorer la stabilité de l'apprentissage. Aussi, nous trouvons intéressant de maintenir les sorties de confiance dans l'intervalle $[0, 1]$ pour faciliter l'interprétation des résultats. Ainsi, nous plaçons une activation $(-1) \times ReLU$

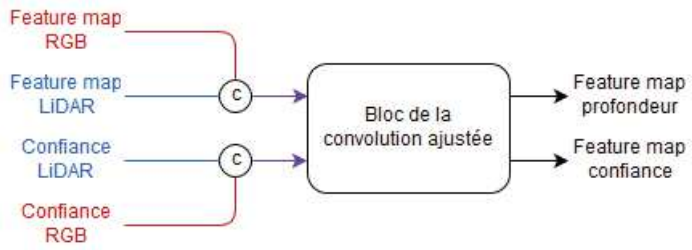


FIGURE 6.7 – Schéma de la fusion par concaténation des descripteurs RGB et LiDAR

sur notre dernière couche pour obtenir une log confiance négative, ce qui nous permet de produire une sortie finale de confiance dans l'intervalle $[0, 1]$.

6.3.5 La fusion RGB

Finalement, nous nous intéressons aux possibilités de fusion RGB dans notre cas de figure particulier où nous utilisons des convolutions ajustées. En effet comme nous l'avons vu, l'utilisation d'images RGB dans l'apprentissage peut améliorer les performances de la complétion de profondeur. Avec des convolutions standards, il y a de multiples exemples de fusions de descripteurs issus d'images RGB et de scans LiDAR. Cependant de nouvelles contraintes apparaissent avec l'utilisation de la convolution ajustée puisqu'il faut aussi prendre en compte les descripteurs de confiance. Nous identifions alors deux configurations différentes possibles avec la convolution ajustée :

— Considérer que la confiance propagée depuis l'entrée LiDAR représente également la confiance à associer aux descripteurs RGB au moment de la fusion dans le réseau

— Considérer que les descripteurs RGB ont une confiance propre

Dans le premier cas, la confiance associée aux descripteurs RGB au moment de la fusion est celle calculée pour les descripteurs de profondeur à partir des entrées LiDAR. Cependant dans la convolution ajustée, l'opération de convolution est pondérée par la confiance au travers de la multiplication préalable entre le masque de confiance et les descripteurs (voir figure 6.4). Nos descripteurs RGB seront alors pondérés par la confiance LiDAR, mettant l'accent sur des régions déjà très sûres pour les descripteurs issus du LiDAR et n'ayant pas de réalité sémantique pour les descripteurs issus des images RGB. Nous avons tout de même essayé cette méthode pour valider expérimentalement cette hypothèse ; nous constatons que le réseau apprend progressivement à réduire l'importance des descripteurs issus des images RGB et ne montre finalement aucune amélioration des résultats.

Dans le second cas, nous considérons une confiance spécifique pour les descripteurs issus des images RGB, donnant une information de certitude sur ces données. Différentes stratégies

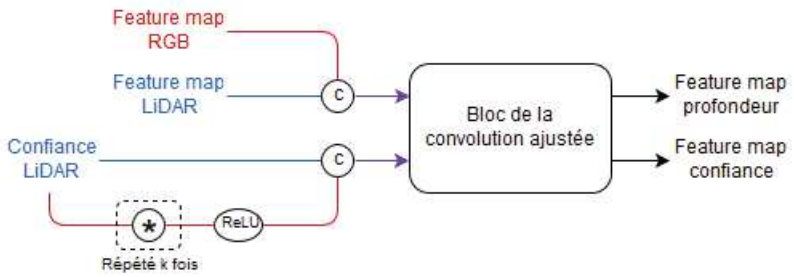


FIGURE 6.8 – Schéma de notre fusion des descripteurs RGB et LiDAR

de fusion peuvent alors être considérées, comme l'addition ou la concaténation des descripteurs (illustrée en figure 6.7) avant d'utiliser des convolutions ajustées.

Lors de la fusion, les contributions des descripteurs issus des images RGB et des entrées LiDAR sont conditionnées par leurs confiances respectives. Cependant, la modélisation de la confiance à partir d'images RGB est un problème différent et plus complexe qu'avec des scans LiDAR où un a priori peut facilement être choisi (les points LiDAR étant une portion de la vérité terrain). Au contraire, il n'y a pas de certitude a priori associée aux images RGB pour l'estimation de profondeur.

Nous simplifions le problème en considérant non plus la carte de confiance des descripteurs RGB comme un indicateur de certitude sur les données mais simplement comme une pondération à calculer sur les contributions des descripteurs lors de la fusion. Nous voulons ainsi permettre au réseau d'apprendre les régions où favoriser les contributions RGB là où les confiances calculées avec le LiDAR sont les plus faibles.

Ainsi, nous avons conçu la fusion comme illustré en figure 6.8. Nous injectons la confiance calculée depuis les scans LiDAR comme référence pour la branche de confiance RGB. Par ce biais, nous aspirons à ce que que le réseau apprenne à partir de la confiance LiDAR à obtenir une carte de pondération des descripteurs RGB. Pour ce faire nous plaçons un certain nombre de convolutions suivies d'activation ReLU pour maintenir la positivité des descripteurs. Nous fixons empiriquement ce nombre de convolutions à 3 convolutions 3x3 suivie d'une convolution 1x1.

Pour l'estimation des descripteurs issus d'images RGB, nous utilisons le réseau DSP proposé au chapitre 4, voir figure 4.11. Cependant nous apportons quelques légères modifications à l'architecture pour réduire sa complexité computationnelle : nous utilisons un ResNet35 au lieu du ResNet200 proposé plus tôt et nous utilisons des déconvolutions simples pour le décodeur plutôt que les blocs d'up-projection. Nous entraînons d'abord ce réseau sur une tâche d'estimation de profondeur avant de l'intégrer à l'architecture globale pour que les descripteurs qu'il délivre soient déjà spécialisés dans la caractérisation de la profondeur. Nous connectons ce réseau au HMS-Net après le dernier *Three-scale block* 6.6 en utilisant la stratégie de fusion

illustrée figure 6.8. En effet, une fusion trop proche de la dernière couche réduit la capacité du réseau à intégrer les descripteurs LiDAR et RGB. Une position de fusion trop proche de l'entrée du HMS-Net augmente l'hétérogénéité entre les deux sources, car les descripteurs RGB sont denses tandis que les descripteurs du LiDAR sont encore parcimonieux.

6.4 Expériences

Nous nous intéressons maintenant aux performances de complétion de profondeur que l'on obtient et à l'impact de la propagation de confiance sur l'apprentissage. De même, nous démontrons la pertinence de nos prédictions de confiance à l'aide de différentes analyses quantitatives et qualitatives, incluant la MeRCI. Enfin, nous comparons la convolution ajustée à la convolution standard et à la Sparse Convolution sur la même architecture, en analysant les performances obtenues pour l'estimation de profondeur. Nos expériences sont basées sur le benchmark KITTI Depth Completion.

6.4.1 Bases de données

La bases de données KITTI Depth Complétion Uhrig et al. (2017) fournit des scans LiDAR bruts (64 lignes de LiDAR) projetées sur un plan 2D avec des vérités terrains semi-denses qui lui sont associées. Ces vérités terrains ont été calculées en utilisant à la fois une accumulation de scans LiDAR ainsi que les résultats de reconstructions stéréos via un algorithme de semi-global matching Hirschmuller (2008). Sur la base du jeu de données brut de KITTI, les auteurs ont générés 85k vérités terrain semi-denses pour l'entrainement (issues d'une paire stéréo), 4k pour la validation ainsi que 3k pour le test. Nous utilisons ici la moitié des images disponibles due l'ensemble d'entrainement, correspondant aux images issues de la caméra de gauche. Nous évaluons notre architecture ainsi que notre approche d'apprentissage sur deux jeux de données différents : les séquences d'images brutes de KITTI Geiger et al. (2013), avec les mêmes ensembles d'entrainement, de validation et de test de KITTI que ceux proposés par Eigen et al. (2014), et les séquences synthétiques de Virtual KITTI Gaidon et al. (2016) où une séquence a été isolée pour l'utiliser comme ensemble de test tandis que les quatre autres constituent l'ensemble d'entrainement.

6.4.2 Métriques d'évaluation

Sur le benchmark associé, toutes les implémentations proposées sont évaluées selon les métriques suivantes :

— L'erreur quadratique moyenne (rmse) en mm : $\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2}$

— L'erreur absolue moyenne (mae) en mm : $\frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$

— L'erreur quadratique moyenne de l'inverse de la profondeur (irmse) en km^{-1} :

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\hat{y}_i}-\frac{1}{y_i}\right)^2} \tag{6.8}$$

— L'erreur absolue moyenne de l'inverse de la profondeur (irmae) en km^{-1} :

$$frac1N\sum_{i=1}^{N}\left|\frac{1}{\hat{y}_i}-\frac{1}{y_i}\right| \tag{6.9}$$

Où $\hat{y}_i$ est la profondeur prédite au pixel i et y_i la vérité terrain associée, avec un nombre total de N pixels. la rmse et la irmse sont plus sensibles aux outliers puisque les erreurs sont au carré. De plus, les grandes valeurs de profondeur ayant potentiellement de plus grandes erreurs absolues, les erreurs rmse et mae sont plus sensibles aux erreurs en arrière-plan, tandis que les erreurs irmse et imae sont plus sensibles aux erreurs au premier plan.

Nous utilisons également la MeRCI, proposée au chapitre précédent, où l'incertitude σ est ici remplacée par la confiance C selon $\sigma = \frac{1}{C}$. On a alors :

$$MeRCI = \frac{1}{N}\sum_{i=1}^{N}\frac{\lambda^{99}}{C_i} \tag{6.10}$$

Notons que nous choisissons ici de prendre en considération le $99^{\text{ème}}$ percentile λ^{99} des ratios $\lambda_i = \frac{|\hat{y}_i - y_i^*|}{\sigma_i}$ puisque la vérité terrain est acquise avec un LiDAR, que nous considérons très fiable. C'est pourquoi nous choisissons de n'autoriser que 1% d'outliers sur les ensembles évalués.

En outre, pour rendre les résultats plus facilement interprétables, nous affichons directement le ratio $\frac{MeRCI}{MeRCI_{constant}}$ où $MeRCI_{constant}$ est le score MeRCI calculé pour un prédicteur de confiance constant. Ainsi, un ratio inférieur à 1 correspond à une prédiction de confiance meilleure qu'un prédicteur constant, ce qui équivaut à ne pas avoir de confiance du tout (voir chapitre 5).

6.4.3 Protocole expérimental

Pour l'apprentissage, les paramètres du réseau sont initialisés aléatoirement. Nous utilisons l'algorithme de Batch Gradient Descent avec des batch de 8 images, l'optimiseur Adam et un pas d'apprentissage constant de $1 \cdot 10^{-3}$ sur 50 époques. Comme les auteurs du HMS-Net, nous n'appliquons pas de weight decay. Nous avons essayé différentes valeurs de λ sur lesquelles nous reviendrons en par la suite.

Pour les expériences avec la fusion RGBG, nous pré entraînons d'abord le réseau monoculaire RGB (DSP, voir figure 4.11) sur la base de données Virtual KITTI. Nous utilisons l'optimiseur

Paramètres de la fonction de coût		Distribution de confiance en sortie		
Valeur de λ	Norme choisie pour l'erreur	Moyenne	Écart type	Entropie
$\lambda = 50$	erreur $\mathbb{L}_2$	1	-	0
$\lambda = 1.6 \cdot 10^{-2}$	erreur $\mathbb{L}_2$	0.59	0.41	2.56
$\lambda = 2$	erreur $\mathbb{L}_1$	1	-	0
$\lambda = 6 \cdot 10^{-2}$	erreur $\mathbb{L}_1$	0.59	0.34	3.41

TABLE 6.1 – Étude de l'impact de l'estimation de confiance pendant l'apprentissage. Les statistiques de distribution des confiances obtenues en sortie sont faites pour des apprentissages avec différentes valeurs de λ.

Adam et un pas d'apprentissage de $2.5e-4$ que l'on diminue de 30% toutes les 15 époques jusqu'à atteindre $1e-5$. Le weight decay est de $5e-4$. L'entrainement dure 250 époques et minimise un coût $\mathbb{L}_1$. Une fois que ce réseau est entraîné, nous remplaçons la dernière convolution par une nouvelle qui produit non pas un mais 16 descripteurs, pour pouvoir appliquer la fusion décrite dans la section 6.3. Ensuite, nous utilisons encore une fois l'optimiseur Adam et un pas d'apprentissage constant de $5e-4$, sans weight decay. Nous utilisons notre fonction de coût jointe avec une erreur $\mathbb{L}_1$ et $\lambda = 6e-2$ pour entraîner sur KITTI Depth Completion pendant 50 époques.

6.5 Résultats

Comme nous l'avons vu en Section 6.3.4, le paramètre λ contrôle l'importance de la pénalisation. Celle-ci est minimale lorsque $C = 1$. En définissant une valeur de λ élevée, nous pouvons contraindre le réseau à générer des valeurs de confiance maximum (égales à 1) en sortie. Dans ce régime, la confiance ne pondère donc pas l'erreur de régression. Le réseau apprend alors à minimiser simplement l'erreur de régression $\mathbb{L}_p$ tout en étant contraint de générer des valeurs de confiance à 1. Nous pouvons donc dans ce cas nous comparer aux méthodes dont l'apprentissage est fait uniquement avec une erreur de régression standard (comme une $\mathbb{L}_2$).

Nous avons testé différentes valeurs de λ pour atteindre ce régime et avons finalement choisi $\lambda = 50$ pour une erreur de régression $\mathbb{L}_2$ et $\lambda = 2$ pour une erreur de régression $\mathbb{L}_1$ (voir table 6.1). Ces valeurs de λ sont différentes en raison des différentes plages de valeurs d'erreur obtenues lors de l'optimisation selon les normes $\mathbb{L}_1$ ou $\mathbb{L}_2$. Les deux valeurs que nous avons choisies ici nous permettent d'obtenir des confiances constantes uniformes et égales à 1 en sortie.

De plus pour la suite, nous cherchons également à produire une confiance correctement répartie sur l'intervalle $[0; 1]$. Comme précédemment, nous avons testé différentes valeurs de λ et choisi $\lambda = 6 \cdot 10^{-2}$ et $\lambda = 1.6 \cdot 10^{-2}$ pour les erreurs de régression respectives $\mathbb{L}_1$ et $\mathbb{L}_2$ (voir

Méthode	rmse	mae	irmse	imae	MeRCI
NCONV-CNN Eldesokey et al. (2018)	1370	380	-	-	-
HMS-Net (Convolution standard) Huang et al. (2018)	1137.42	315.32	-	-	-
HMS-Net (Sparse Convolution) Huang et al. (2018)	**994.14**	262.41	-	-	-
Notre méthode ($\mathbb{L}_2$, $\lambda = 50$)	1110.06	271.97	4.07	1.08	1
Notre méthode ($\mathbb{L}_2$, $\lambda = 1.6 \cdot 10^{-2}$)	1150.60	280.06	**3.49**	1.05	2.21
Notre méthode ($\mathbb{L}_1$, $\lambda = 2$)	1177.15	**237.91**	5.74	**0.94**	1
Notre méthode ($\mathbb{L}_1$, $\lambda = 6 \cdot 10^{-2}$)	1220.75	247.93	3.92	**0.95**	0.45

Table 6.2 – Performances des complétions de profondeur sur le jeu de validation de KITTI Depth Completion

lignes 2 et 4 de la Table 6.1). Ces deux configurations nous permettront par ailleurs d'étudier l'impact de la confiance sur l'apprentissage.

6.5.1 Estimation de profondeur : analyse

Nous reportons nos résultats ainsi que ceux de méthodes de l'état de l'art dans la table 6.2. NCONV-CNN Eldesokey et al. (2018) est la seule méthode figurant au benchmark KITTI Depth Completion qui permette de prédire une confiance continue, comme nous l'avons vu en Section 6.2. Cependant, les auteurs utilisent une architecture multi-échelles différente, des Normalized Convolution ainsi qu'une fonction de coût spécifique mais qui, comme la nôtre, prend en compte à la fois des estimations de profondeur et de confiance. Nous constatons d'abord que nous obtenons de meilleures performances que la seule méthode existante produisant à la fois des paires profondeur et confiance sur le jeu de données de validation de KITTI Depth Completion.

Nous reportons également les résultats du HMS-Net Huang et al. (2018). Dans leur papier, les auteurs analysent l'impact du type de convolutions utilisé sur les performances en utilisant soit des Sparse Convolution soit des convolutions classiques. Pour la comparaison avec les résultats du papier original du HMS-Net, nous pouvons considérer uniquement notre architecture entraînée selon une $\mathbb{L}_2$ avec $\lambda = 50$. en effet comme nous l'avons vu, un λ élevé comme celui ci revient à minimiser une erreur de régression $\mathbb{L}_2$ seule. Ainsi nous voyons que notre convolution ajustée permet d'obtenir des résultats similaires au HMS-Net original utilisant des Sparse Convolution et de meilleurs résultats que celui entraîné avec convolutions classiques.

6.5.2 Estimation de profondeur : impact de la fonction de coût jointe

Nous discutons maintenant des résultats mis en évidence par les 4 dernières lignes de la table 6.2, correspondant à nos différents apprentissages.

Nous voyons que pour nos apprentissages utilisant la fonction de coût jointe non saturée

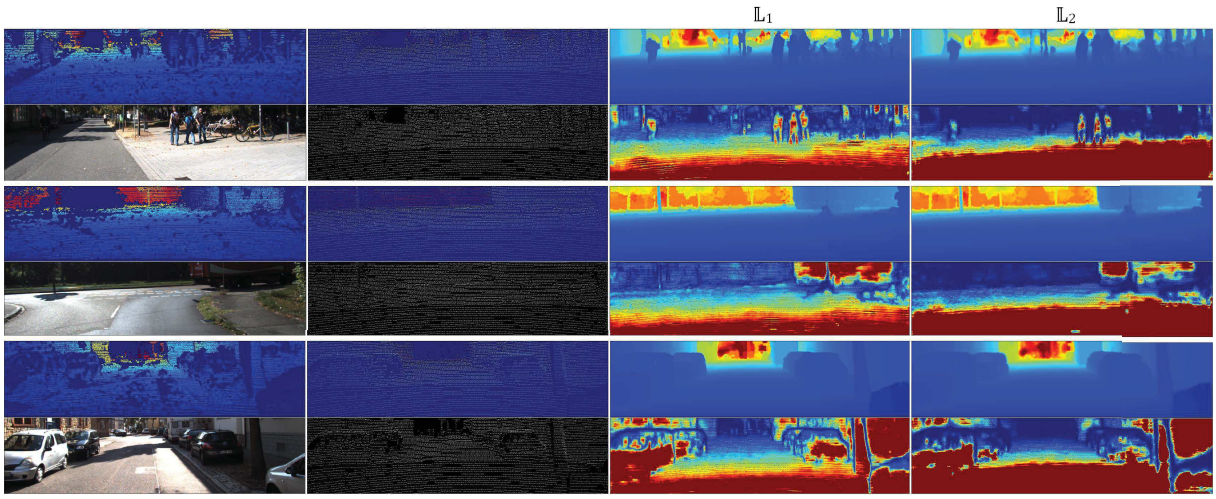


FIGURE 6.9 – Prédictions de confiance $\mathbb{L}_1$ et $\mathbb{L}_2$. Les deux colonnes de gauche contiennent les données d'entrée : vérité terrain, scan LiDAR, masque binaire. L'image RGB est également affichée comme un support visuel mais elle n'est pas utilisée dans nos réseaux ici. Les deux colonnes de droite contiennent les prédictions basées sur les normes $\mathbb{L}_1$ et $\mathbb{L}_2$: estimation de profondeur en haut et estimation de confiance en dessous

(lignes 5 et 7) les erreurs sont plus faibles sur le premier plan (d'après les scores de irmse et imae) et plus élevées en arrière plan (rmse et mae). En effet lorsque la confiance est distribuée dans l'intervalle $[0, 1]$ et n'est pas constante, les régions où la confiance est la plus élevée bénéficient comme nous l'avons vu d'une accélération de la vitesse d'apprentissage, tandis que celle ci diminue pour les régions avec des confiances faibles.

Au global, nous constatons que l'apprentissage de la confiance n'améliore que peu les estimations de profondeur (bien qu'on puisse s'attendre à des améliorations au premier plan). Toutefois, améliorer les scores de rmse ou de mae n'est pas évident sur des données d'extérieur avec de grandes distances. En effet dans notre cas, la vitesse d'apprentissage étant plus faible dans les régions correspondant à l'arrière-plan (jusqu'à stagnation de l'erreur lors d'un apprentissage très long), il est difficile d'améliorer drastiquement les scores moyens sur l'image.

Enfin nous voyons que sur la figure 6.9 nos estimations de profondeur sont très saillantes et ont l'air visuellement très proche de l'image RGB et de la vérité terrain. Les différences sont minces et nous constatons, comme nous pouvions l'attendre, que l'apprentissage basé sur la norme $\mathbb{L}_2$ produit des sorties un peu moins nettes sur les contours d'objets.

6.5.3 Estimation de confiance : analyse qualitative

Dans cette partie nous nous intéressons aux cartes de confiance obtenues en sortie, illustrées figure 6.9.

D'abord, on note que la confiance obtenue en sortie est répartie sur l'intervalle $[0, 1]$ et si nous la comparons avec l'entrée binaire correspondante, nous remarquons qu'elle ne suit

pas simplement la densité du scan LiDAR. En effet, on note une diminution progressive de la confiance à mesure que la profondeur augmente, ce qui est cohérent avec la fonction de coût jointe que l'on optimise puisque les erreurs $\mathbb{L}_1$ ou $\mathbb{L}_2$ sont potentiellement plus grandes à l'arrière-plan. Nous voyons également que la confiance correspondant aux positions des points LiDAR donnés en entrée est beaucoup plus élevée. C'est un comportement attendu et qui est désirable étant donné que ces points LiDAR font partie de la vérité terrain. Certains objets sont reconnaissables, par exemple sur les premières estimations de confiance on peut identifier des piétons, sur la seconde un camion ou encore des voiture sur la troisième. Sur ces objets, on trouve plusieurs scans LiDAR avec de faibles variations de profondeur. Nous nous attendons donc à une confiance plus élevée à ces endroits car la tâche de complétion de profondeur est plus facile.

Au contraire, la confiance est plus faible sur les régions où l'on trouve des surfaces réfléchissantes. Comme nous pouvons le constater sur la troisième prédiction, la confiance sur les fenêtres de voiture est faible : à cause de la transparence le LiDAR ne fournit que peu de données. Dans le même temps, nous voyons que la zone correspondant aux fenêtres de la camionnette blanche a une confiance plus élevée parce que les fenêtres sont opaques. Il est connu que le LIDAR est sujet aux erreurs sur les surfaces réfléchissantes. On peut observer cette propriété sur le masque binaire : les vitres de voiture correspondent à des trous dans les mesures. Les vérités terrain de profondeur sont également plus rares sur les surfaces réfléchissantes, car elles sont en partie basées sur l'accumulation de scans LIDAR. Ainsi, les réseaux ne peuvent pas apprendre l'erreur attendue correspondante. Enfin on voit que les contours des objets ont également une confiance moindre car les ruptures brusques de profondeur augmentent les erreurs potentielles. La confiance que nous prédisons semble donc informative et proportionnelle à l'erreur. Nous étudierons plus en détail la relation entre erreur et confiance dans la prochaine section.

6.5.4 Estimation de confiance : comparaison $\mathbb{L}_1$ et $\mathbb{L}_2$

Bien que les prédictions de confiance aient l'air pertinentes pour nos deux réseaux appris à la fois avec les normes $\mathbb{L}_1$ et $\mathbb{L}_2$ dans la fonction de coût jointe, nous constatons d'après la table 6.1 que la confiance obtenue avec la $\mathbb{L}_1$ a une entropie plus élevée. Cela suggère que la distribution de confiance est mieux répartie lorsque l'on utilise une $\mathbb{L}_1$.

De plus, on voit sur la figure 6.9 que les prédictions de confiance faites avec la $\mathbb{L}_1$ apportent visuellement plus de détails (les jambes des piétons par exemple). On constate aussi que les prédictions de confiance en arrière plan sont principalement nulles lorsqu'elles sont apprises avec la $\mathbb{L}_2$. Aussi, la proportion des prédictions égales à 1 ou proches de 0 est plus grande que pour la version $\mathbb{L}_1$, donc la confiance est effectivement mieux distribuée comme l'indique l'entropie. Ainsi, les prédictions de confiance entraînées avec une $\mathbb{L}_1$ dans la fonction de coût jointe semblent visuellement être de meilleure qualité.

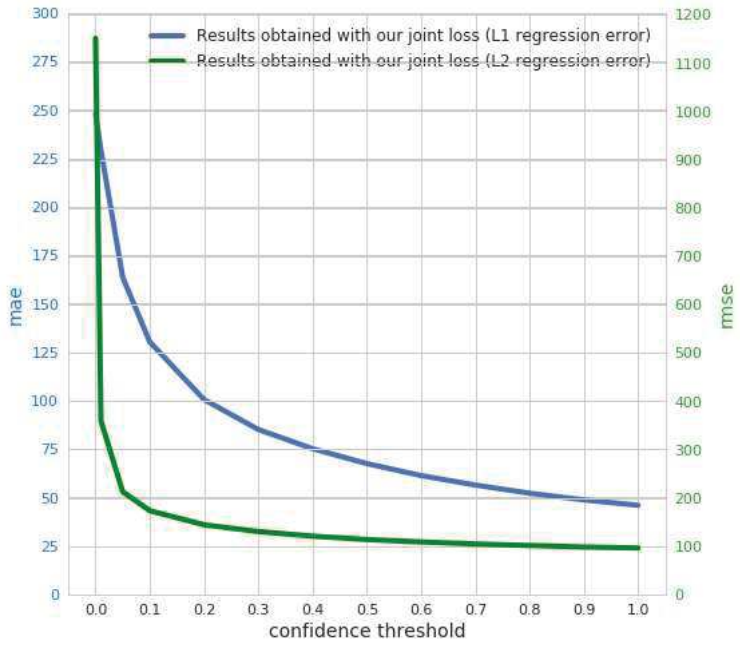


FIGURE 6.10 – Évolution de l'erreur en fonction du seuil de confiance choisi. La rmse est calculée pour l'implémentation entraînée avec une $\mathbb{L}_2$ dans la fonction de coût jointe et la MAE pour l'implémentation basée sur la $\mathbb{L}_1$

C'est un résultat que nous pouvons vérifier avec la MeRCI. En effet, la table 6.2 nous indique que les prédictions de confiance obtenues pour un apprentissage basé sur l'erreur $\mathbb{L}_2$ sont moins bonnes qu'un prédicteur constant. Par contre pour le réseau entraîné avec une $\mathbb{L}_1$ dans la fonction de coût jointe le score MeRCI obtenu est très bon ce qui indique que les cartes de confiance sont bien corrélées à l'erreur. Les entrainements avec des valeurs de λ élevées ayant été choisis pour produire des confiances uniformes et constante à 1, le ratio $MeRCI/MeRCI_{constant}$ est à 1.

Nous pensons que c'est dû au gradient de chacune des normes. En effet, nous savons que les régions où la confiance est forte présente des erreurs relativement faibles. Avec une erreur de régression $\mathbb{L}_2$, l'action de la confiance sur la vitesse d'apprentissage est atténuée par le gradient. Les régions avec des confiances plus faible (c'est à dire des erreurs plus grandes) profitent de gradients pus forts, tandis que les régions avec une grande confiance (et donc une erreur plus petite) ont des gradients faibles. Par rapport à la $\mathbb{L}_1$, l'apprentissage sera moins ciblé sur les régions très confiantes à cause de l'action du gradient de la $\mathbb{L}_2$. Nous pouvons donc nous attendre à des estimations de confiance de meilleure qualité avec une $\mathbb{L}_1$ puisque la relation entre la confiance et l'inverse de l'erreur est plus forte.

6.5.5 Estimation de confiance : analyse quantitative

Sur la Figure 6.10, nous traçons les résultats en rmse et mae nos deux réseaux entraînés respectivement avec les erreurs $\mathbb{L}_1$ et $\mathbb{L}_2$ dans la fonction de coût jointe, en fonction de

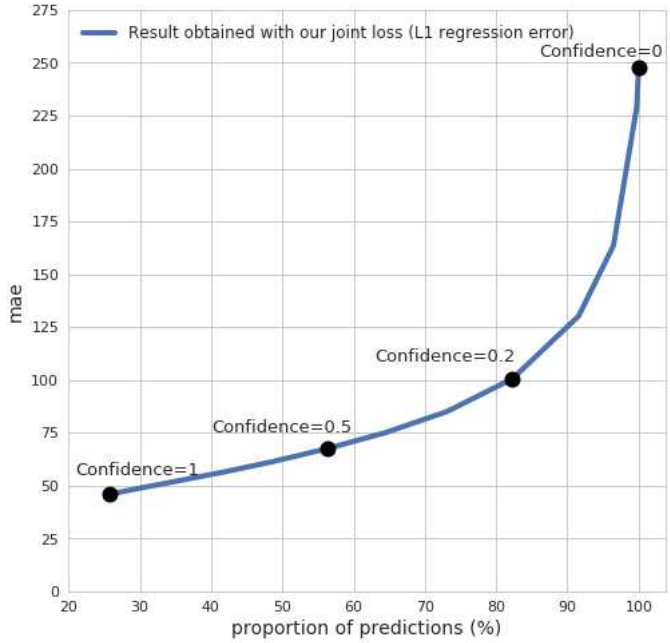


FIGURE 6.11 – Pour une mae donnée, nous renvoyons la proportion des prédictions impliquées dans le calcul. 4 marqueurs indiquent des valeurs de confiance caractéristiques. Seuls les pixels valides de la vérité terrain sont considérés pour le calcul de la proportion des points.

la confiance : pour un seuil de confiance donné, nous ne considérons que les prédictions correspondant aux confiances au dessus de ce seuil. On voit que les sorties de confiance sont inversement corrélées aux erreurs, allant respectivement de mae $= 247.9$ mm et rmse $= 1150.8$ mm pour une confiance≥ 0, c'est à dire quand tous les points sont considérés, à mae $= 45.9$ mm et rmse $= 94.3$ mm pour les points dont la confiance $= 1$.

La Figure 6.11, est la même que la courbe bleue précédente mais nous affichons en plus en abscisse la proportion de points qui sont impliqués pour une confiance donnée. Ainsi, nous lisons que les prédictions avec une confiance≥ 0.5 ont une mae de 67.4 mm et représentent 56% des prédictions. La valeur de confiance 0.2 marque le changement sur la pente de la courbe et nous voyons que 82% des points ont une erreur plus de deux fois plus petite que la mae globale. Ce qu'il faut retenir de cette analyse est que l'on vérifie une propriété importante de notre carte de confiance puisqu'on constate qu'une importante proportion de prédictions présente des erreurs très faibles relativement à l'erreur globale.

6.5.6 Estimation de confiance : analyse par sous-sets des points

Comme la vérité terrain est générée par une accumulation de scans LiDAR Uhrig et al. (2017), un partie des points LiDAR utilisés en entrée est également contenue dans la vérité terrain. Statistiquement pour un couple d'observations donné (scan LiDAR, vérité terrain), 14.87% des points de la vérité terrain sont donnés en entrée. Comme nous l'avons vu en Section 6.5.3, ces points ressortent sur la carte de confiance prédite car ils obtiennent des confiances beaucoup

-	$Prop_1 = \frac{S_i}{All}$	$MAE_1\ (S_i)$	$Prop_2 = \frac{(S_i \mid C=1)}{S_i}$	$MAE_2\ (S_i \mid C=1)$
$Subset_1(S_1)$	14.87%	122.49	57.10%	26.79
$Subset_2(S_2)$	85.13%	269.83	20.32%	55.30
Ratio $\frac{S_1}{S_2}$	0.17	0.45	2.81	0.48

TABLE 6.3 – Le $Subset_1$ désigne les points présents à la fois dans la vérité terrain et dans l'entrée. Le $Subset_2$ désigne les points de la vérité terrain qui ne sont pas présent dans le scan LiDAR d'entrée. All désigne l'ensemble des prédictions, S_i un sous ensemble particulier de prédictions et C la confiance prédite

Méthode	rmse (mm)	mae (mm)	irmse (km^{-1})	imae (km^{-1})	MeRCI
LiDAR, erreur $\mathbb{L}_1$, $\lambda = 6.10^{-2}$	1220	247	3.92	0.95	0.45
LiDAR+RGB, erreur $\mathbb{L}_1$, $\lambda = 6.10^{-2}$	1169	243	3.55	0.93	0.45

TABLE 6.4 – Comparaisons des performances qualitatives obtenues sur l'ensemble de validation de KITTI Depth completion avec et sans la fusion RGB

plus élevées que leurs voisins. Idéalement, nous nous attendons à ce que les confiances prédites sur ces points soient égales à 1 et que l'estimation de profondeur soit parfaites sur ces points. C'est pourquoi nous comparons à la fois l'erreur de profondeur et la distribution de confiance sur ces points avec le reste des points de la vérité terrain dans la Table 6.3.

Si nous considérons uniquement les points contenus à la fois dans l'entrée et dans la vérité terrain, nous voyons que 57% ont une confiance de 1, c'est à dire 3 fois plus que pour le reste des points. Nous remarquons également que l'erreur mae pour l'estimation de profondeur est environ deux fois plus faible, avec seulement 26.79m d'erreur pour les points dont la confiance $= 1$. Ces résultats démontrent que, bien qu'elle puisse être encore améliorée, la propagation de l'information est saine via notre architecture.

6.5.7 Fusion RGB

Nous étudions maintenant l'apport de la fusion RGB pour guider l'apprentissage. Comme nous l'avons vu les images RGB peuvent apporter de l'information supplémentaire (sémantique ou contour des objets par exemple) utile pour la tâche. Nous entraînons donc notre réseau en suivant l'architecture décrite dans la section 6.3 avec la fusion RGB.

Résultats quantitatifs

Le tableau 6.4 résume les résultats quantitatifs obtenus sur KITTI Depth Completion, évalués sur les pixels valides de la vérité terrain. Pour chaque méthode, nous sélectionnons la meilleure epoch d'après le score obtenu en irmse. Nous constatons que, bien que relativement faible, la fusion RGB montre une amélioration selon chacune des métriques. Le guidage RGB contribue

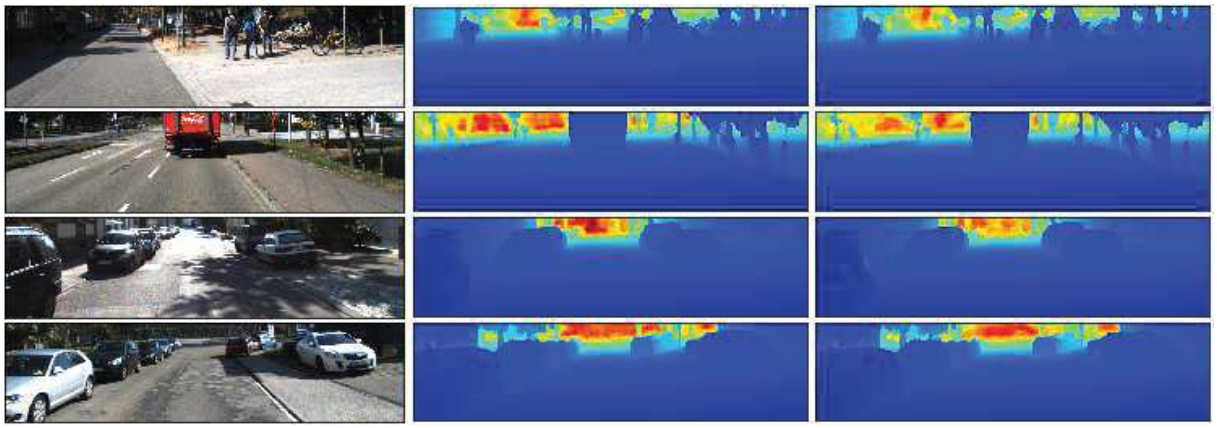


FIGURE 6.12 – Comparaison qualitative des résultats de complétion simple avec ceux de la complétion guidée. A gauche l'image RGB correspondant aux prédictions, au milieu la prédiction de la complétion guidée RGB et à droite la complétion simple issue d'un scan LiDAR

donc clairement à améliorer les résultats bien que ces améliorations n'affectent qu'une faible partie de l'ensemble de données et ont donc peu d'impact sur les résultats moyens obtenus dans l'analyse quantitative.

On note également que nos résultats sur la MeRCI restent identiques, ce qui indique que les prédictions de confiance obtenues avec le guidage RGB ne sont pas dégradées et sont aussi informatives que celles obtenues pour la complétion simple.

Résultats qualitatifs

Tout d'abord, nous nous intéressons aux résultats obtenus avec cet apprentissage en termes qualitatifs. Pour ce faire nous comparons les résultats issus d'un apprentissage n'utilisant que des scans LiDAR en entrée (que nous appellerons complétion simple) et d'un apprentissage utilisant à la fois des scans LiDAR et les images RGB correspondantes (que nous appellerons complétion guidée). Des échantillons de test sont affichés en figure 6.12. En comparant les estimations de profondeur, nous notons que la complétion guidée permet d'apporter plus de détails, en particulier sur les contours des objets. Par exemple, nous voyons que sur la complétion simple, les contours de la voiture à gauche de la 3ème image sont flous, tandis qu'on les voit plus nettement sur l'estimation issue de la complétion guidée. Nous voyons également à la 4ème image que la profondeur estimée sur le véhicule militaire montre un trou dans le pare-brise dans le cas de la complétion simple qui n'est pas présente dans la complétion guidée. On peut aussi constater les différences de contours estimés par les deux algorithmes sur les voitures et troncs d'arbres de la première image. Cependant nous constatons que les estimations de profondeur sont globalement très similaire et ce sont les détails qui sont plus ou moins fins.

La base de données KITTI a été acquise en utilisant un Velodyne haut de gamme avec

Méthode	rmse (mm)	mae (mm)	irmse (km^{-1})	imae (km^{-1})
Ma et Karaman (2018) LiDAR seul	954	288	3.21	1.35
Ma et Karaman (2018) LiDAR + RGB	814	249	2.80	1.25
Jaritz et al. (2018) LiDAR seul	1035	248	2.60	0.98
Jaritz et al. (2018) LiDAR + RGB	917	234	2.17	0.95
Huang et al. (2018) LiDAR seul	937	258	2.93	1.14
Huang et al. (2018) LiDAR + RGB	841	253	2.73	1.13

Table 6.5 – Performances de méthodes de l'état de l'art sur l'ensemble de test du benchmark KITTI Depth Completion avec et sans la fusion RGB

64 couches. Avec ces 64 couches, le Velodyne permet de fournir beaucoup d'information sur presque toute la scène. Nous pensons que l'apport du guidage RGB est conditionné par la qualité du LiDAR (répartition spatiale et densité). En effet, le LiDAR fournit déjà une information de profondeur de haute précision (elle est d'ailleurs utilisée comme vérité terrain) et l'estimation de profondeur à partir d'image RGB est, comme nous l'avons vu, une tâche complexe. C'est pourquoi nous estimons que plus la qualité du LiDAR sera grande et moins importante sera la contribution de la fusion RGB.

6.6 Comparaison entre l'estimation et la complétion de profondeur

Enfin, bien que très différentes, nous voulons comparer les méthodes d'estimation de profondeur monoculaire avec la complétion de profondeur à partir de scans LiDAR uniquement.

Sur le benchmark de KITTI Depth Completion, il n'y a pas de comparaison entre un réseau entraîné avec uniquement des images RGB contre un réseau entraîné avec des scans LiDAR puisque l'objectif du challenge est la complétion de profondeur. Cependant, nous avons noté que le guidage RGB n'apportait que peu d'amélioration quantitative à la méthode de complétion que nous proposons, tout comme à celles de l'état de l'art (voir table 6.5).

La carte de profondeur prédite par un tel algorithme est donc principalement expliquée par le LiDAR fourni en entrée. Ces résultats sont confirmés par l'analyse visuelle où l'on voit que les contributions RGB sur la prédiction de profondeur sont faibles en comparaison de ce que l'on obtient avec le LiDAR : dans le but de minimiser le coût de régression il semble donc que le réseau apprenne à s'appuyer davantage sur la branche de complétion (à partir des scans LiDAR) que sur la branche d'estimation (à partir d'images RGB).

Ajoutons à cela que Ma et Karaman (2018) ont étudié les performances de leur réseau en considérant soit uniquement des images RGB en entrée, soit uniquement des cartes de profondeur parcimonieuses, sur la base de données KITTI (voir figure 6.13). Nous constatons

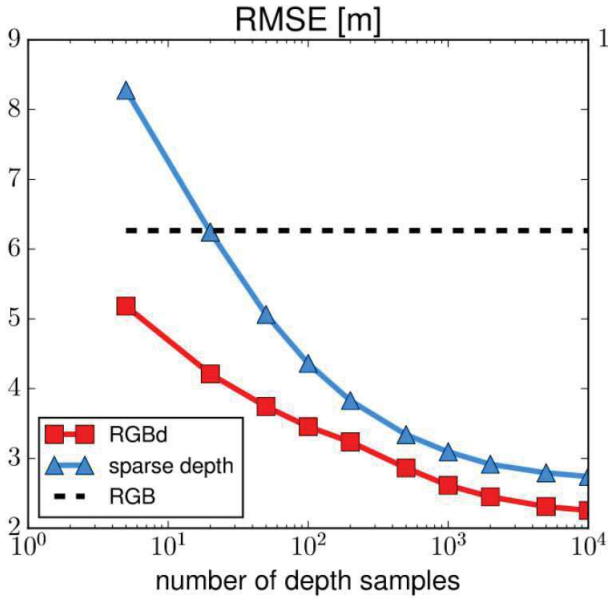


FIGURE 6.13 – Erreur rmse en fonction du nombre de points de profondeur pris en compte pour un réseau avec des entrées RGB seules (en pointillés), un réseau avec des entrées RGB+D (en rouge) et un réseau avec des entrées de profondeur uniquement (en bleu)

Méthode	rmse (mm)	mae (mm)	irmse (km^{-1})	imae (km^{-1})	MeRCI
Estimation de profondeur (DSP)	3660	1538	8.90	5.16	-
Complétion de profondeur (notre meilleure architecture)	1220	247	3.92	0.95	0.45

TABLE 6.6 – Comparaison des résultats obtenus par l'estimation de profondeur et la complétion de profondeur sur la base de données KITTI pour notre méthode

que le réseau entraîné avec des profondeurs uniquement en entrée obtient de meilleures performances que celles obtenues avec des images RGB uniquement (en pointillés) avec une vingtaine de points seulement. Nous constatons également que la contribution de la fusion rgb (l'écart entre les courbes rouges et bleues) décroît à mesure que le nombre d'échantillons augmente. Ces conclusions sont à modérer par le fait que les données d'entrée sont des cartes de profondeur denses échantillonnées à la volée durant l'apprentissage pour simuler des données parcimonieuses. Néanmoins, il s'agit tout de même d'une tâche de complétion de profondeur puisque les entrées sont une portion de la vérité terrain.

Les travaux précédents laissent penser que la complétion de profondeur donne de meilleurs résultats que, logiquement, l'estimation de profondeur. Nous voulons le vérifions expérimentalement et reprenons donc le réseau d'estimation de profondeur monoculaire que nous avons utilisé pré entraîné sur KITTI pour la fusion RGB (voir 6.3). Nous obtenons les résultats présentés en table 6.6.

Nous constatons effectivement que la complétion de profondeur obtient des résultats significativement meilleurs en termes d'erreur. Visuellement, les sorties produites par l'estimation de profondeur monoculaire sont pertinentes mais restent assez floues (contours d'objets peu

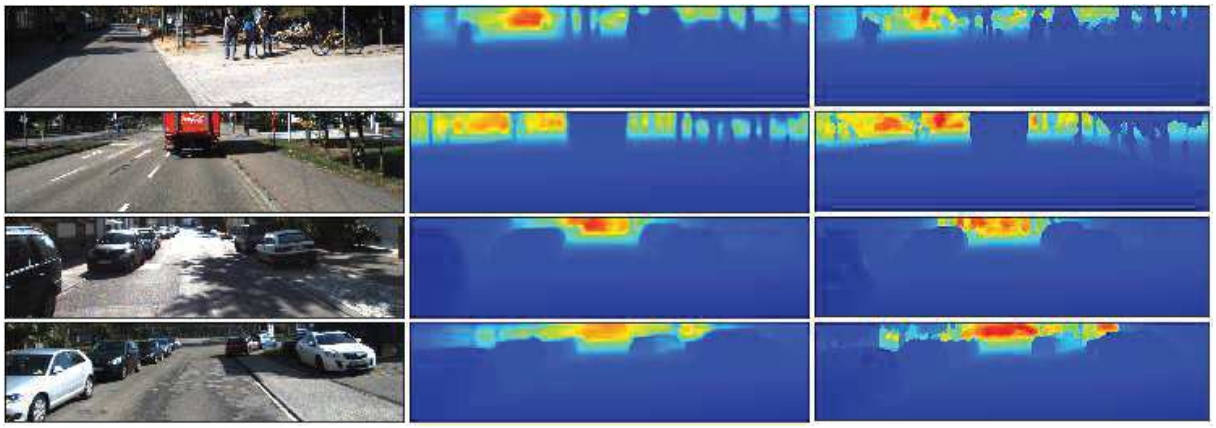


FIGURE 6.14 – Comparaison qualitative des résultats d'estimation de profondeur monoculaire avec ceux de la complétion simple. L'image RGB correspondante aux prédictions à gauche, l'estimation de profondeur monoculaire au milieu et la complétion de profondeur issue d'un scan LiDAR à droite

précis). Cependant nous notons que la complétion de profondeur peut montrer quelques aberrations, comme par exemple sur la voiture de la 3ème ligne figure 6.14, dû au fait que la vérité terrain semi-dense ne pénalise pas les pixels isolés des surfaces transparentes comme les vitres de voiture et freine l'apprentissage de structures sémantiques globales, ce qui n'est pas le cas pour l'estimation de profondeur.

6.7 Conclusions

Dans ce chapitre, nous avons présenté un nouveau bloc, la convolution ajustée, conçue pour gérer les données parcimonieuses compatibles avec les approches récentes de *deep learning*. Nous avons utilisé cette convolution ajustée pour proposer une architecture répondant à la tâche de la complétion de profondeur avec et sans guidage RGB.

Sur la base du cadre mathématique de la convolution normalisée, nous avons également conçu notre bloc de telle sorte à ce qu'il prenne en compte la certitude des données d'entrée. Nous tirons parti de cette information pour propager une estimation de confiance sur l'ensemble du réseau. Par conséquent, nous sommes en mesure de produire à la fois une prédiction de profondeur dense ainsi qu'une carte de confiance associée. Nous évaluons notre approche sur le jeu de données KITTI Depth Completion et montrons que nos estimations de profondeur atteignent les performances de l'état de l'art. D'autre part, nous démontrons, à la fois quantitativement et qualitativement, que nos prédictions de confiance sont fiables et fortement liées à l'erreur de prédiction de profondeur.

Chapitre 7

Conclusions et perspectives

7.1 Conclusions

Dans cette thèse, nous nous sommes intéressés à différentes questions que pose l'estimation de la profondeur à partir d'images monoculaires. Notre objectif était d'obtenir les meilleures estimations possible, tant en termes d'erreur moyenne sur les images, de qualité visuelle des prédictions et également de qualité de l'incertitude associée lorsque cela est possible.

Dans un premier temps, nous avons commencé par présenter les travaux de la littérature qui se sont penchés sur la question de l'estimation de profondeur de manière générale. Nous avons passé en revue différents types de méthodes, de l'acquisition de la profondeur par des capteurs spécifiques dédiés jusqu'à l'estimation algorithmique de la profondeur à partir d'images. Nous avons vu comment, dans ce dernier cas, les méthodes classiques permettaient de calculer une profondeur exacte basée sur une résolution géométrique du problème à partir de plusieurs vues de la scène. Les limitations de ces techniques, par exemple la nécessité de disposer de plusieurs caméras à l'acquisition ou bien les lacunes qu'elles montrent pour des régions peu texturées de l'image, nous ont amenés à nous poser le problème de l'estimation de profondeur par apprentissage statistique. En particulier, nous nous sommes intéressés aux méthodes de *deep learning* qui ont récemment connu un important essor grâce à la disponibilité croissante de base de données annotées ainsi que de la puissance de calcul grandissante des ordinateurs. Ces éléments ont ainsi permis une formidable avancée dans de nombreux domaines comme celui de la vision par ordinateur. C'est pourquoi nous avons choisi de concentrer notre étude sur les méthodes d'estimation de profondeur monoculaire à partir d'une seule image et basées sur l'apprentissage de réseaux de neurones convolutionnels profonds.

De très nombreux hyperparamètres rentrent en compte à la fois dans la définition et dans l'optimisation des modèles de *deep learning*. C'est pourquoi par la suite, nous avons proposé d'étudier un certain nombre de ces hyperparamètres pour tenter de comprendre leur influence sur la capacité de généralisation d'un CNN entraîné à l'estimation de profondeur monoculaire. Nous avons proposé un plan d'expériences visant à analyser à la fois l'importance d'éléments architecturaux des réseaux, mais aussi à analyser l'influence des paramètres d'optimisation dans l'apprentissage.

Tandis que nous avons montré que chacun des paramètres avait un intérêt particulier dans l'apprentissage, cette étude nous a permis de mettre en avant deux éléments majeurs essentiels pour obtenir des estimations de profondeur de qualité : les descripteurs $\mathcal{RGB}$ et la diversité des données d'entraînement. En effet, nous avons vu que les meilleures architectures étaient composées d'encodeurs profonds et élaborés permettant d'apprendre de meilleures représentations de l'image. En particulier, on constate des performances bien supérieures lorsque l'encodeur est pré entraîné pour des tâches sémantiques. Nous pensons que c'est parce que les indices monoculaires les plus importants sont basés sur la connaissance a priori des objets de la scène. D'autre part nous observons qu'augmenter le nombre d'images d'entraînement, quand

c'est possible, ou, le cas échéant, qu'appliquer un algorithme d'augmentation des données, permet d'améliorer significativement les performances de généralisation du réseau.

Cette étude nous a permis de faire un tour d'horizon de la majorité des hyperparamètres régissant l'apprentissage d'un CNN et de les analyser sous le prisme de l'estimation de profondeur. Néanmoins, nous avons choisi de regarder leur influence de manière isolée en fixant autant que possible les autres paramètres à une valeur de référence. Pour aller plus loin, nous pensons qu'il serait pertinent de s'intéresser à la combinatoire de ces hyperparamètres pour tenter de comprendre comment ils interagissent et d'observer dans quelle proportion cela modifie le comportement de l'apprentissage.

Puisque nous avons montré que la représentation apprise par l'encodeur avait une importance majeure dans la qualité de l'estimation finale, nous avons décidé de nous intéresser aux travaux effectués dans la littérature de la classification d'objets et de la segmentation sémantique qui comptent parmi les encodeurs les plus performants. Ainsi, nous avons travaillé sur l'architecture de nos réseaux pour leur donner la capacité d'intégrer une analyse multi échelle de l'image comme c'est le cas dans les réseaux conçus pour la segmentation sémantique.

Inspirés par cet état de l'art, nous avons proposé plusieurs architectures de CNN aptes à apprendre des descripteurs multi échelle de l'image et adaptés à la tâche d'estimation de profondeur. Nous avons mené une étude comparative de ces architectures que nous avons mises en parallèle avec un réseau mono échelle classique. Nous avons montré que donner la capacité au réseau d'extraire des caractéristiques multi échelle de l'image améliorait de manière consistante les performances de l'estimation de profondeur obtenue, pour un petit jeu d'entraînement. En outre, nous avons sélectionné une architecture en particulier pour l'entraîner sur un jeu de données de plus grande envergure. Cela nous a permis d'instaurer un nouvel état de l'art pour l'estimation de profondeur monoculaire et d'obtenir des prédictions de profondeur très fine, de meilleure qualité visuelle.

Cependant, la contrepartie inhérente à la capacité de représentation accrue des réseaux profonds est leur très grand nombre de paramètres. En effet, il n'est pas rare que les CNN avec les meilleures performances de classification aient plusieurs centaines de couches et plusieurs dizaines de millions de paramètres à optimiser. C'est également le cas pour les architectures que nous comparons et pour lesquelles la complexité est encore augmentée par l'intégration de stratégies d'analyse multi échelle. Ici pour notre étude nous ne nous sommes donc pas limités à la taille démesurée de ces réseaux. En pratique, il serait intéressant de travailler sur la possibilité d'obtenir des résultats équivalents pour une taille de réseau donnée, en guidant l'apprentissage comme c'est le cas ici avec l'ajout d'une analyse multi échelle de l'image.

Nous nous sommes ensuite placés dans le cas où notre algorithme serait capable de produire, en plus d'une prédiction de profondeur, une estimation de l'incertitude de prédiction. Nous nous sommes donc intéressés à l'état de l'art de l'estimation d'incertitude. Dans ce contexte,

nous avons d'abord proposé une revue des méthodes existantes permettant l'évaluation de l'incertitude prédictive. Nous avons notamment recensé plusieurs métriques issues de la littérature de la prévision et avons discuté de leurs limitations dans le cas de l'évaluation de méthodes de *deep learning*.

En parallèle, nous avons proposé une nouvelle métrique adaptée à ces méthodes et permettant d'évaluer la qualité de l'estimation d'incertitude elle-même en jugeant de sa corrélation avec l'erreur réelle commise par le réseau. Nous avons là encore effectué une analyse comparative des méthodes classiques de l'état de l'art avec la méthode proposée. Nous avons montré sur un exemple jouet que notre métrique avait un comportement sain et qu'elle était robuste aux données aberrantes. Ensuite, pour le cas concret de l'estimation de profondeur monoculaire, nous avons montré que notre métrique est étroitement liée à l'erreur réelle de prédiction et qu'elle est donc plus fiable pour juger de la pertinence des méthodes évaluées. En outre, elle s'avère mieux adaptée aux méthodes récentes de *deep learning* que les métriques classiques puisqu'elle ne nécessite pas d'a priori sur la forme de la distribution prédite.

La vocation principale de cette métrique est de pouvoir évaluer n'importe quelle méthode d'apprentissage statistique d'estimation d'incertitude et de la classer relativement à d'autres méthodes. Par ailleurs, nous montrons que des techniques différentes d'estimation d'incertitude peuvent produire des résultats dont les intervalles sont très différents, qui ne sont donc pas nativement comparables. Pour pallier à ce problème, nous avons donc pris le parti d'adopter une mesure insensible à l'échelle des incertitudes. Bien que cela nous permette de rationaliser l'évaluation de tout type de méthodes, nous ne sommes pas capables avec cette métrique de donner une mesure de l'erreur moyenne absolue d'un algorithme.

Enfin, nous nous sommes intéressés au cas applicatif de la complétion de profondeur à partir de scans LiDAR, avec ou sans image $\mathcal{RGB}$ de la scène. Dans ce cadre, nous avons proposé un nouveau bloc de convolution permettant de gérer des données parcimonieuses et nous l'avons intégré au sein d'un réseau particulier, de taille limitée et avec une architecture multi échelle. Le bloc de convolution que nous proposons utilise également l'information de certitude inhérente au LiDAR fourni en entrée et permet, lorsque son apprentissage est fait avec la fonction de coût appropriée, de propager une estimation de confiance au sein du réseau. Au final, cela nous permet de prédire à la fois une carte de profondeur dense de la scène, mais aussi une carte de confiance associée à ces prédictions. Nous montrons que notre approche permet non seulement d'obtenir des performances de complétion de profondeur au niveau de l'état de l'art, mais qu'elle permet également de produire des cartes de confiance pertinentes. En effet, nous analysons nos résultats par différentes méthodes en incluant la métrique proposée au chapitre précédent et nous montrons que nos mesures de confiance sont bien corrélées aux erreurs réelles commises par le réseau.

Dans ce chapitre, nous tentons de mettre en pratique les conclusions tirés précédemment.

Pourtant, tandis que le but de ce genre d'algorithme serait typiquement d'être implémenté dans systèmes autonomes embarqués, nous nous rendons compte que leur intégration pour des applications concrètes est le plus grand défi. En effet, cela implique d'abord de prendre en compte des contraintes de taille de mémoire ou de temps d'exécution qui nécessitent de travailler sur la réduction des coûts computationnels des réseaux. D'autre part, nous pensons que le calcul systématique de la confiance associée aux prédictions est une composante indispensable pour le passage aux applications réelles, que ce soit pour l'amélioration de la prise de décision ou pour la sécurité du système. La convolution que nous proposons utilise le fait que les données d'entrée sont en fait une partie de la vérité terrain pour estimer la confiance, mais cette méthode n'est pas adaptable à l'estimation de profondeur monoculaire à partir d'images. Il y a encore trop peu de travaux dans la littérature sur ce sujet et nous pensons qu'il serait intéressant de travailler à la généralisation de cette approche.

7.2 Perspectives

Nous pensons que nos travaux pourraient être étendus dans deux directions de recherches prometteuses principales :

— l'utilisation d'images de synthétiques pour l'apprentissage
— la consolidation temporelle des résultats

7.2.1 Images synthétiques

Tout d'abord, nous savons que l'inconvénient principal intrinsèque aux méthodes d'apprentissage statistique profond est la nécessité d'avoir à disposition une grande quantité d'images annotées pour l'apprentissage. Nous avons vu par ailleurs que le nombre d'images impliquées dans l'entraînement était effectivement déterminant dans les performances du modèle. C'est pourquoi nous pensons que l'une des pistes de recherche les plus intéressantes qui pourraient être explorées serait l'utilisation d'images synthétiques pour l'apprentissage. En effet, il existe maintenant de plus en plus de bases de données d'images synthétiques photoréalistes dans des environnements variés (intérieurs ou extérieurs) ainsi que d'outils permettant de générer ce type d'images. L'avantage de type de représentation est double : il est facile de générer rapidement de grandes quantités de données annotées et surtout il est possible de contrôler l'environnement pour simuler un grand nombre de situations, possibles ou non. Ainsi, l'apprentissage se fait avec des données statistiquement plus variées et couvrant potentiellement mieux la distribution de l'espace possible.

Cependant, le passage d'un entraînement sur des données synthétiques à une application avec des données réelles n'est pas trivial : il faut apprendre aux réseaux à combler l'écart existant entre les domaines. C'est ce qui est appelé l'adaptation de domaine dans la littérature.

Une piste possible est de travailler sur la représentation apprise par l'encodeur pour la rendre invariante à la modalité d'entrée. Ainsi, nous espérons que les mêmes caractéristiques seront extraites à partir d'une image réelle ou d'une image synthétique de la même scène. Dans le cas de l'estimation de profondeur, où nous avons vu que la performance est corrélée avec la capacité du réseau à apprendre les meilleurs descripteurs sémantiques de l'image, nous pensons que l'enjeu pourrait être de taille. En effet il existe déjà de nombreuses bases de données annotées pour l'estimation de profondeur, mais les environnements restent peu variés (intérieurs et milieux urbains) et les conditions d'acquisitions sont souvent idéales (luminosité, conditions météorologiques). Il est donc très probable que nos algorithmes appris avec ces données produisent des résultats loin de leurs performances habituelles dans des conditions moins favorables. La maîtrise de l'adaptation de domaine avec des images synthétiques pourrait aider à corriger ce problème.

7.2.2 Consolidation temporelle et spatiale

La plupart des architectures profondes actuelles n'utilisent pas plusieurs images comme c'est le cas pour les approches classiques d'estimation de profondeur. À la place, c'est souvent une seule image fixe qui est utilisée. En effet, l'estimation de profondeur à partir d'une seule image a pris une place prépondérante dans les études récentes aux vues de ses avantages et des résultats des méthodes de *deep learning*. Cependant nous constatons qu'en pratique l'acquisition d'images multiples tend à devenir une norme dans les applications et les systèmes basés sur la vision. Les collections d'images personnelles et publiques augmentent sans cesse, avec de plus en plus de redondances et de recouvrement entre les images. Par exemple de nos jours, la prise d'une simple photo avec un smartphone implique souvent l'acquisition de plusieurs images avant de les combiner, parfois même à partir de capteurs différents.

Les cas pratiques offrent donc souvent des informations supplémentaires utiles qu'il convient de prendre en compte dès la conception de l'architecture pour en tirer parti et améliorer la qualité et la robustesse des estimations. Par ailleurs, dans un cas réel l'utilisation d'une caméra se fera souvent en continu. Nous pensons que le principal défaut des méthodes que nous avons proposées jusqu'ici est qu'elles ne prennent pas en compte cet aspect temporel pour la consolidation des résultats.

Nous pensons donc qu'un axe de travail pertinent pour faire suite à nos travaux serait de trouver comment intégrer l'utilisation de plusieurs images issues d'une même caméra, mais acquises à des instants différents (consolidation temporelle), ou bien de plusieurs images représentant plusieurs vues de la scène (consolidation spatiale). En effet, ajouter ces contraintes à l'apprentissage devrait aider le réseau à apprendre des estimations plus robustes dans le temps et dans l'espace. De plus, l'utilisation de vidéo permettrait d'extraire de nouveaux types d'indices de profondeur, comme le parallaxe, et ainsi améliorer la qualité des estimations

obtenues. Nous proposons des travaux préliminaires allant en ce sens, avec une architecture et une procédure d'optimisation originales, en annexe de ce manuscrit.

Annexe : Consolidation temporelle et spatiale

Comme nous l'avons vu, la connaissance de l'information de profondeur est d'une importance cruciale dans la compréhension de scène dans de nombreux contextes applicatifs tel que celui des véhicules autonomes. Là où l'inférence de profondeur à partir d'une seule image fixe a pris une place prépondérante dans les études récentes avec le résultat des méthodes de *deep learning*, les cas pratiques offrent souvent des informations supplémentaires utiles qu'il convient de prendre en compte dès la conception de l'architecture pour en tirer parti et améliorer la qualité et la robustesse des estimations. Ainsi, nous proposons ici un réseau complet entièrement convolutionnel permettant d'exploiter plusieurs images, soient issues de séquences temporelles monoculaires soit issues d'une paire stéréoscopique. Une nouvelle procédure d'apprentissage prenant en compte l'optimisation multi-échelles est également proposée. En effet, comme nous l'avons vu l'utilisation d'informations multi-échelles tout au long du réseau est d'une importance primordiale pour une estimation précise de la profondeur et améliore considérablement les performances. Ces contributions nous permettent d'obtenir des résultats au niveau de l'état de l'art à la fois sur des données synthétiques, en utilisant la base de données Virtual KITTI, mais également sur des images réelles sur le jeu de données KITTI.

7.3 Introduction

La plupart des architectures profondes actuelles n'utilisent pas des couples d'images comme le font des approches classiques, mais apprennent la profondeur à partir d'une seule image. Cependant, nous constatons qu'en pratique, l'acquisition d'images multiples tend à devenir une norme dans les applications et les systèmes basés sur la vision. Les collections d'images personnelles et publiques augmentent sans cesse, avec encore plus de redondances et d'overlapping entre images. Par exemple de nos jours la prise d'une simple photo avec un smartphone implique souvent l'acquisition de plusieurs images avant de les combiner, parfois même à partir de capteurs différents. C'est pourquoi nous nous intéressons ici au développement d'une architecture profonde spécifique pour l'estimation de profondeur à deux entrées, permettant soit d'effectuer une prédiction à partir d'une séquence temporelle d'images monoculaires soit à partir d'images issues de capteurs stéréo.

Nous passerons d'abord en revue les approches récentes permettant l'estimation de profondeur à partir d'un couple ou d'une séquence d'images pour tenter de comprendre pourquoi elles sont si efficaces. Une fois ces approches synthétisées et en se basant sur nos conclusions, nous proposons une nouvelle architecture. Notre but est de couvrir deux problématiques principales de l'estimation de profondeur :

— Estimer la profondeur à partir de deux images stéréoscopiques

— Estimer la profondeur à partir de deux images consécutives issues d'une même séquence

Afin de comparer l'efficacité de notre approche avec les méthodes de l'état de l'art, nous évaluerons nos performances sur deux jeux de données académiques différents :

— la base de données KITTI (images réelles, vérité terrain parcimonieuse)

— la base de données Virtual KITTI (images synthétiques, vérité terrain dense)

L'un des principaux avantages de ces deux jeux de données est de fournir des images issues de séquences de conduite, illustrant l'efficacité avec laquelle ces méthodes peuvent être utilisées pour des véhicules autonomes.

7.3.1 Notations

Vision stéréoscopique

Dans le cas stéréoscopique, c'est à dire lorsque les images proviennent de différentes caméras calibrées, une notion spécifique est mise en jeu. La calibration implique un alignement des droites épipolaires le long de l'axe x. Cette notion est appelée **disparité**. La disparité est définie par l'équation implicite suivante :

$$I_l(x, y) = I_r(x - \rho(x, y), y) \tag{7.1}$$

où ρ représente la fonction de disparité. Le principal avantage ici est que cette formule relie la disparité à la profondeur. En effet, nous pouvons déduire d'une transformation spatiale que :

$$\rho(x, y) = \frac{fb}{Z(x, y)} \tag{7.2}$$

où f est la dsitance focale des deux caméras et b la distance entre les deux caméras. Les équations 7.1 et 7.2 apportent donc un moyen indirect de résoudre le problème d'estimation de profondeur, par l'estimation préalable de la disparité.

Vision monoculaire temporelle

Un concept similaire à la disparité existe également dans le cas séquentiel. Il est connu sous le nom de **flot optique**. Le flot optique relie deux images issues de la même caméra à différents instants. Comme pour la disparité, le flot optique est défini par l'équation implicite suivante :

$$I_t(x, y) = I_{t+1}(x + V_x(x, y), y + V_y(x, y)) \tag{7.3}$$

où V_x et V_y représente le flot optique le long des deux axes.

La première différence entre ces deux concepts est la différence de dimensions. En effet, la calibration des caméras dans le cas stéréoscopique limite l'équation 7.1 le long d'un seul axe au lieu des deux dimensions présentes dans l'équation 7.3. C'est pourquoi il est généralement plus simple d'estimer la disparité plutôt que le flot optique. La seconde différence est qu'il n'existe pas de formule directe reliant le flot optique et la profondeur.

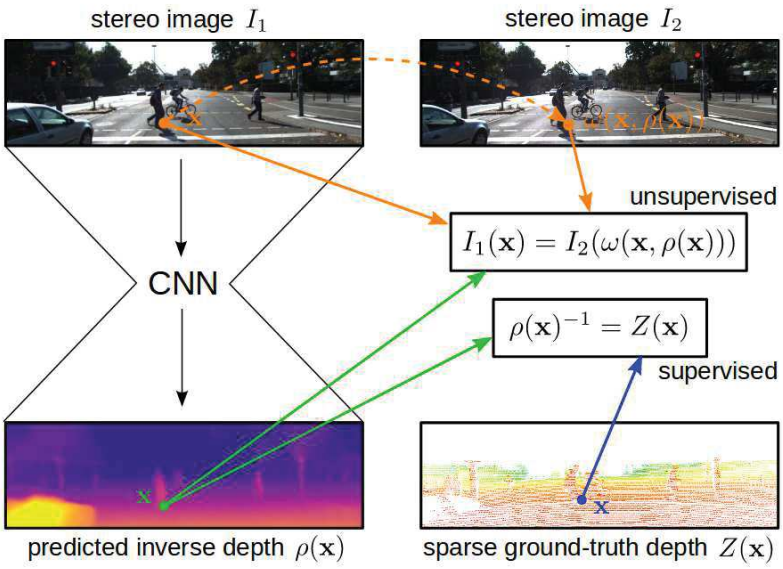


FIGURE 7.1 – Illustration de l'architecture du réseau proposé par Kuznietsov et al. (2017)

L'étude du flot optique peut alors sembler inutile mais puisque les deux images proviennent de la même caméra, étant donné le déplacement de la caméra (c'est à dire la rotation R et la translation T) il devient possible de relier le flot optique et les coordonnées 3D des points qui dépendent, eux, de la profondeur.

7.4 Littérature associée

7.4.1 Estimation de profondeur contrainte

Consistance spatiale

Dans la littérature récente, on trouve également des méthodes d'estimation de profondeur basées sur l'information stéréoscopique. Deux publications ont notamment abordé ce problème.

Kuznietsov et al. (2017) proposent une architecture atteignant les meilleures performances de l'état de l'art sur le jeu de données KITTI. Sa structure est similaire à celle de Laina et al. (2016). La principale différence se trouve dans la fonction de coût utilisée pour l'entrainement. En effet, ils utilisent le fait que deux images issues d'une paire stéréo sont disponibles à l'entrainement pour ajouter une contrainte dans la fonction de coût. D'après l'équation 7.2, si la profondeur est connue on peut retrouver la disparité et ainsi reconstruire l'autre image de la paire en utilisant l'équation 7.1. La fonction de coût qu'ils introduisent se base sur cette propriété et est la suivante :

$$\mathcal{L} = ||F(I_l) - Z'|| + ||I_r(x - \rho(x, y), y) - I_l|| \tag{7.4}$$

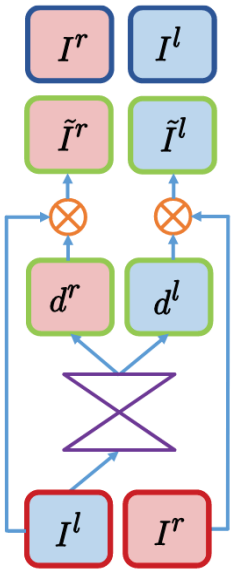


FIGURE 7.2 – Schéma de l'architecture proposée par Godard et al. (2017)

où F représente la fonction apprise par le réseau, Z' la vérité terrain et ρ la disparité calculée à partir l'estimation de profondeur du réseau.

Cette fonction de coût oblige le modèle non seulement à prédire la profondeur avec des valeurs de pixels proches de la vérité terrain, mais également une profondeur conduisant à une reconstruction stéréo consistante. Il est intéressant de noter que nous pouvons n'utiliser que le second terme de la fonction de coût $||I_r(x - \rho(x, y), y) - I_l||$, ce qui permet alors d'entrainer le modèle sans avoir besoin de la vérité terrain Z'. Le modèle peut alors apprendre à estimer la profondeur avec uniquement des paires d'images stéréoscopiques RGB, de manière non supervisée.

D'autres travaux ont été publiés récemment par Godard et al. (2017) où des images stéréoscopiques sont utilisées de manière non supervisée pour entrainer le réseau à l'estimation de profondeur. Leur architecture a une structure similaire aux précédents avec un ResNet50 pour la partie encodeur et ensuite une partie décodeur spécifique. La procédure d'apprentissage est légèrement différente puisque elle prend en compte à la fois les prédictions faites à partir de l'image de gauche et de l'image de droite de la paire stéréo pour rendre les estimations de profondeur plus consistantes. Ce réseau est entrainé de manière non supervisée, sans vérité terrain de profondeur. La figure 7.2 illustre la manière dont le réseau est entrainé. Tout d'abord, le réseau estime deux cartes de disparités différentes d^l et d^r qui seront utilisées pour reconstruire l'image de droite à partir de l'image de gauche et vice versa. La fonction de coût quant à elle est composée de trois termes différents. Le premier pénalise les erreurs de reconstruction de chacune des images de la paires stéréo. Il est définit par l'équation suivante :

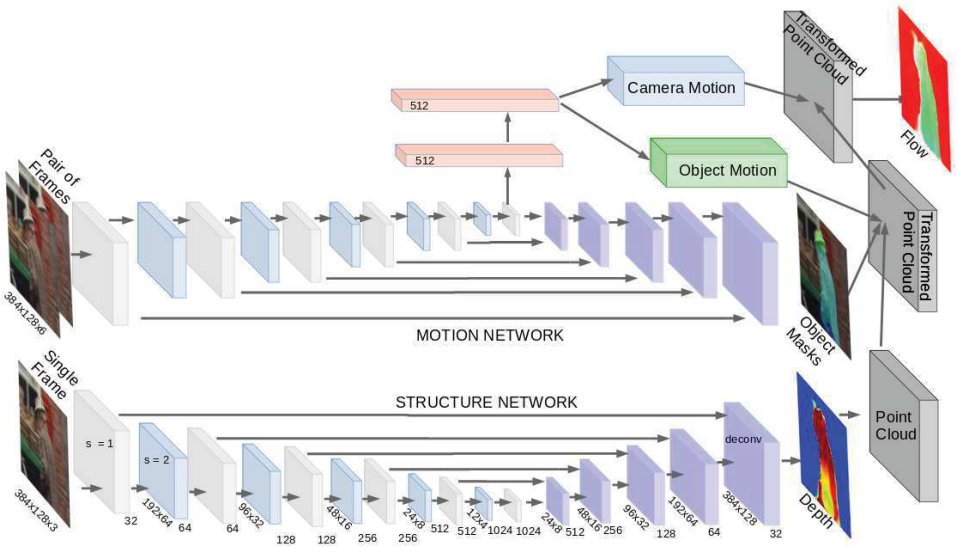


FIGURE 7.3 – Architecture du réseau Sfm-Net introduit par Vijayanarasimhan et al. (2017)

$$\mathcal{L}_{reconstruction} = ||I_r(x - d^r(x,y), y) - I_l|| + ||I_l(x - d^l(x,y) - I_r|| \tag{7.5}$$

Le second terme s'assure de la consistance des disparités prédites :

$$\mathcal{L}_{disparites} = ||d^l(x,y) - d^r(x + d^l(x,y), y)|| \tag{7.6}$$

Enfin le dernier terme de la fonction de coût permet d'affiner les prédictions en s'assurant que les gradients de l'estimation de profondeur soient superposables à ceux de l'image I :

$$\mathcal{L}_{lissage} = ||\partial_x d||e^{-|\partial_x I|} + ||\partial_y d||e^{-|\partial_y I|} \tag{7.7}$$

La fonction de coût globale est donc la somme de ces trois différents termes, à la fois pour les disparités gauche et droite. Au moment où nous écrivons, ce réseau permet d'obtenir les meilleurs résultats en estimation de profondeur non supervisée sur le jeu de données KITTI.

Consistance temporelle

A partir de la même idée, d'autres travaux récents se sont penchés sur l'utilisation de l'information temporelle (utilisation de séquences d'images) pour améliorer leur performances d'estimation de profondeur.

Vijayanarasimhan et al. (2017) ont publié une architecture qui utilise l'information séquentielle pour améliorer leurs prédictions. La figure 7.3 illustre cette architecture. Elle est différente

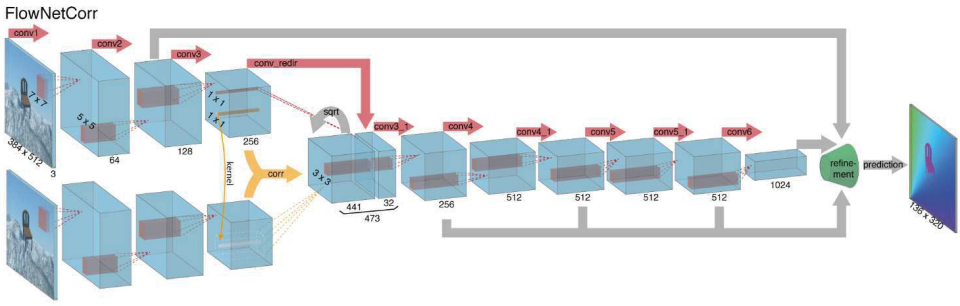


FIGURE 7.4 – L'architecture de Flownet, introduite par Dosovitskiy et al. (2015)

des précédentes architectures dont nous avons parlé. En effet, nous trouvons ici d'abord un premier réseau qui permet de prédire la profondeur à partir d'une seule image. Un second réseau prend une paire d'images consécutives en entrée et prédit à la fois un masque des objets de la scène ainsi que le déplacement de la caméra et de chaque objet entre les deux frames. L'idée est ensuite de calculer le flot optique à partir de ces estimations de profondeur et de mouvement pour ajouter une contrainte à l'apprentissage dans la fonction de coût, comme c'était le cas pour l'estimation à partir de paire d'images stéréo en utilisant la disparité.

Pour ce faire, on peut calculer les coordonnées 3D des points en chaque pixel étant donné les paramètres de la caméra, d'après l'équation 2.1. A partir de ces points 3D, l'équation 2.2 nous permet de calculer la projection dans le repère image au temps $t+1$. Les matrices de rotation et de translation sont obtenues par composition des prédictions de déplacement des objets et de la caméra. Une fois que la projection dans l'image $t+1$ a été calculée, le flot optique peut être déduit en comparant le déplacement de chaque pixels entre les deux frames.

La fonction de coût contient un terme de pénalisation sur l'erreur de reconstruction utilisant le flot optique :

$$\mathcal{L} = ||F(I_t) - Z'|| + ||I_{t+1}(x + V_x(x,y), y + V_y(x,y)) - I_t|| \tag{7.8}$$

où V correspond au flot optique estimé, Z' la vérité terrain et I_t l'image au temps t. De la même manière que pour les travaux précédents, ce réseau peut être entrainé de manière non supervisée en supprimant le premier terme de la fonction de coût basé sur Z'.

Zhou et al. (2017) proposent également une architecture permettant de calculer la profondeur à partir d'images issues d'une séquence vidéo. Ils présentent une architecture et une procédure d'entrainement similaires mais ne prédisent ni le masque des objets de la scène, ni la carte de déplacement des objets. Ainsi, ce réseau nécessite que le seul déplacement entre deux instants différents soit celui de la caméra, contrairement à celui de Vijayanarasimhan et al. (2017) qui intègre également le mouvement des objets.

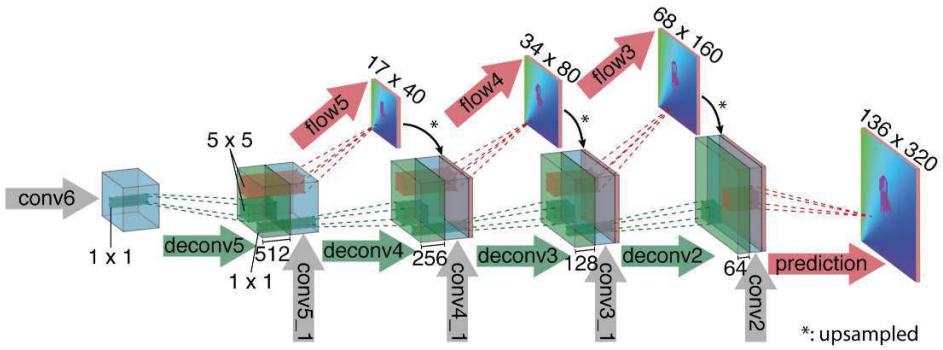


FIGURE 7.5 – Architecture du module de raffinement introduite par Dosovitskiy et al. (2015)

Flot optique

Tous ces réseaux ont un point commun : ils n'utilisent qu'une seule image à la fois en entrée. A notre connaissance, il n'y a pas de réseau de l'état de l'art utilisant un couple d'images pour l'estimation de profondeur. Cependant il existe des tâches pour lesquelles deux images sont nécessaires. Par exemple, Dosovitskiy et al. (2015) proposent une architecture permettant de prédire le flot optique à partir d'une paire d'images successives, appelé FlowNet. Ce réseau, illustré en figure 7.4, présente deux spécificités dans son architecture. La première est la présence d'un module appelé module de corrélation. Les deux images sont d'abord encodées indépendamment puis on calcule la corrélation entre les descripteurs (en jaune sur la figure 7.4). Cette opération de corrélation peut être globalement comprise comme un produit scalaire entre les descripteurs le long des canaux. Nous reviendrons plus en détails sur cette opération dans la section suivante.

Les descripteurs issus de la corrélation sont ensuite eux-mêmes encodés puis injectés dans un module appelé module de raffinement (illustré en figure 7.5) qui caractérise la seconde spécificité de ce réseau puisqu'il permet de produire quatre différentes cartes de flot optique en sortie du réseau et ce à différentes résolutions. Chacune de ces cartes de flot optique est alors réinjectée dans le réseau pour aider au décodage de la résolution suivante, jusqu'à obtenir la sortie finale.

D'après ce que nous avons vu des techniques utilisées dans l'état de l'art, nous retenons plusieurs points majeurs :

— Les architectures récentes les plus efficaces sont toutes basées sur une architecture type encodeur-décodeur. La partie d'encodage est généralement effectué en utilisant des blocs type ResNet

— On trouve souvent une approche du global vers le local, par exemple dans le cas des réseaux de Eigen et al. (2014) et FlowNet Dosovitskiy et al. (2015) qui affinent leurs

prédictions avant de produire la sortie finale

— Les performances sont meilleures lorsque de l'information stéréoscopique ou temporelle est disponible et utilisée lors de l'entrainement. La plupart du temps ces informations sont prises en compte dans la fonction de coût utilisée. D'un autre côté, FlowNet prend directement l'information temporelle en entrée et utilise une opération spécifique dans son architecture : la corrélation.

Nous nous inspirons de tous ces résultats pour proposer une architecture spécifique à l'estimation de profondeur à partir soit d'une paire d'images stéréoscopiques soit d'une paire d'images successives issues d'une même séquence.

7.5 Méthode

Nous présentons dans cette section un nouveau réseau conçu pour l'estimation de profondeur à partir d'un couple d'images, stéréoscopiques ou séquentielles. Nous appelons ce réseau le *Multi Scale Depth Optimization Strategy Network* (ou réseau MSDOS). Nous proposons trois contributions principales aux réseaux existants de l'état de l'art au travers de cette architecture : une structure en pyramide inspirée par les méthodes classiques de calcul de la disparité, un nouveau module de décodage et enfin une stratégie d'apprentissage spécifique.

7.5.1 Architecture

L'architecture globale du réseau MSDOS est présentée en figure 7.6. Le modèle peut être décomposé en trois modules macroscopiques indépendants.

Le premier module transpose une approche de corrélation pyramidale classique au travers de plusieurs encodeurs à différentes résolutions d'image. Chaque image est d'abord encodée à différentes échelles puis les descripteurs résultants sont concaténés ensemble. Ensuite vient l'opération de corrélation que l'on applique entre les descripteurs des deux images. Nous utilisons le même module de corrélation que celui proposé dans FlowNet par Dosovitskiy Dosovitskiy et al. (2015). Nous mélangeons les descripteurs issus de la corrélation avec des descripteurs d'une des branches pour conserver l'information monoculaire. Afin de renforcer la robustesse de la corrélation, la même opération est effectuée sur des descripteurs capturés à différentes résolutions (en rouge et vert sur la figure 7.6). Pour chaque sortie intermédiaire un module Depth Encoder-Decoder (DED) est employé pour prédire une carte de profondeur à la résolution voulue. Il est composé d'une ResNet Proj, visant à augmenter le nombre de descripteurs et à diminuer la résolution. Deux ResNet Skip sont ensuite ajoutées pour augmenter la profondeur. Le décodage suit ensuite le schéma de l'up-projection originale proposé par Laina et al. (2016) pour recouvrer la résolution spatiale d'entrée du module..

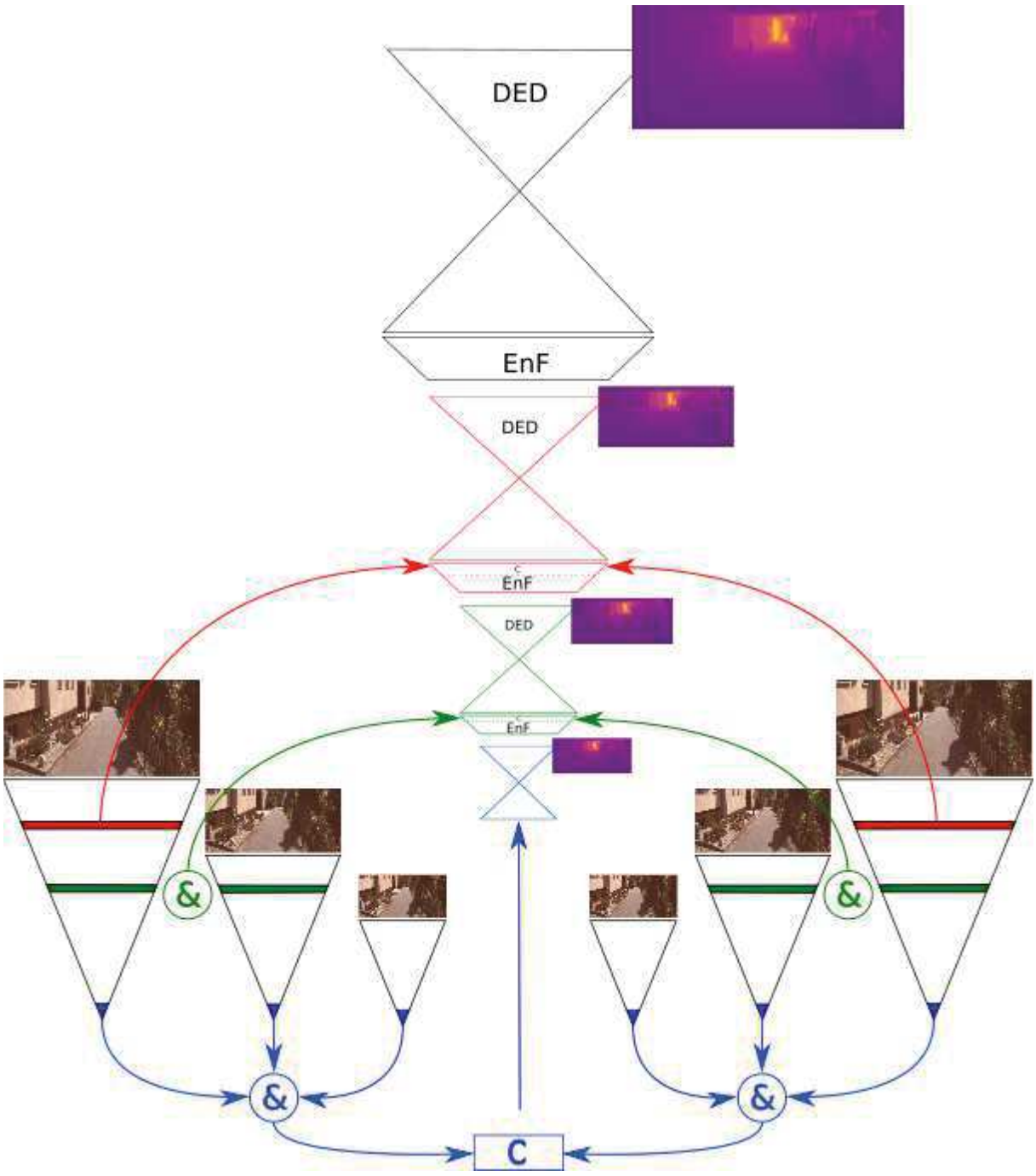


FIGURE 7.6 – Architecture globale du réseau MSDOS : estimation de profondeur d'abord globale puis locale à l'aide d'un encodage pyramidal. Des opérations de corrélation sont effectuées à plusieurs résolutions (en bleu, vert et rouge sur la figure) et intégrées successivement dans les modules EnF (Expand and Fuse) correspondants. Le symbole '&' est utilisé pour représenter la concaténation des descripteurs et 'c' est utilisé pour représenter l'opération de corrélation

Image $1:1$	Image $1:2$	Image $1:4$
$Conv_2^7(3,64)$ $Conv_2^5(64,64)$		
Corr. haute résolution		
$Conv_2^5(64,64)$ $Resnet_1(64,256)$	$Conv_2^7(3,64)$ $Conv_2^5(64,64)$	
Corr. résolution intermédiaire	Corr. résolution intermédiaire	
$Resnet_2(256,512)$	$Conv_2^5(64,64)$ $Resnet_1(64,256)$	$Conv_2^7(3,64)$ $Conv_2^5(64,64)$
Corr. basse résolution	Corr. basse résolution	Corr. basse résolution

Table 7.1 – Architecture des encodeurs pour les différentes résolutions d'entrée où $Conv_s^k(channels_{in}, channels_{out})$ représente une convolution de stride s, de noyau carré de taille k avec $channels_{in}$ descripteurs en entrée et qui renvoie $channels_{out}$ descripteurs de sortie. Toutes les convolutions sont suivies d'une étape de batch normalisation et d'une ReLU pour la non linéarité. $Resnet_i$ représente le i^{eme} bloc du ResNet50 qui est lui même composé de 4 blocs principaux

Entre chaque bloc DED se trouve un module d'up-sampling que nous appelons Expand and Fuse (EnF). Il prend les descripteurs courants ainsi que la dernière estimation de profondeur effectuée à une résolution intermédiaire et double leur taille. Le tout est ensuite concaténé avec le résultat de la corrélation correspondante et avec une des deux images RGB d'entrée, sous échantillonnée à la même résolution.

Une dernière séquence EnF+DED permet d'affiner la carte de profondeur à la moitié de la résolution des image d'entrée. Pour des raisons computationnelles nous choisissons de procéder ici à une décomposition pyramidale à trois niveaux de résolution différents, bien que cette méthode puisse s'appliquer pour autant de sous-résolutions que souhaité.

7.5.2 Encodage multi-échelles

Les entrées stéréoscopiques ou séquentielles sont d'abord sous-échantillonnées selon un schéma de décomposition pyramidale, chaque niveau étant la moitié de la résolution du précédent. Les images résultantes sont encodées jusqu'à la taille de descripteurs souhaitée. Cette partie de l'encodage est en partie composée de modules provenant du ResNet50 pré-entrainé sur ImageNet, pour faciliter la convergence. Ce processus d'encodage est donc très similaire aux réseau existants de l'état de l'art pour l'estimation de profondeur monoculaire. La table 7.1 présente l'architecture spécifique pour l'encodeur de chacune des résolutions en entrée. Notons que des encodages plus profonds sont effectués lorsque les résolutions sont plus élevées afin d'obtenir les mêmes dimensions finales.

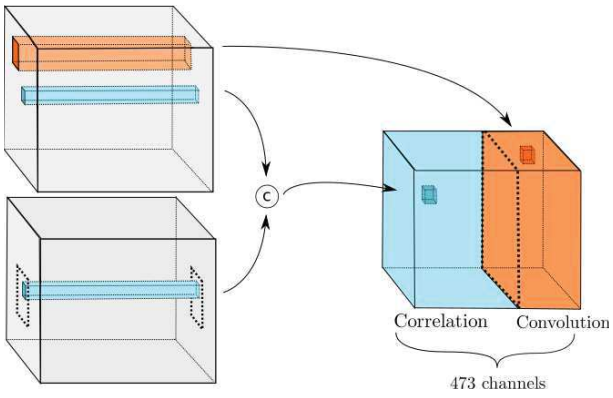


FIGURE 7.7 – Détail de la couche de corrélation constituée du produit scalaire entre les descripteurs des deux images. Le nombre de descripteurs de sortie est arbitrairement fixé à 473, quelle que soit la taille de la corrélation ; les descripteurs supplémentaires sont monoculaires et proviennent du résultat d'une convolution sur les descripteurs d'une des deux branches.

Nous considérons les deux images d'entrée séparément et concaténons les descripteurs de résolution égales ensemble de la manière suivante : tous les encodeurs contribuent à fournir des descripteurs de faible résolution et de haut niveau d'abstraction (en bleu sur la figure 7.6), deux encodeurs sur trois contribuent aux descripteurs intermédiaires (en vert) et enfin un seul fournit les descripteurs de bas niveau (en rouge).

Une fois concaténée, les descripteurs provenant de chaque image sont ensuite mises en correspondance à chaque résolution via un module de corrélation, identique à celui de FlowNet Dosovitskiy et al. (2015). Ce module effectue des comparaisons par patch et n'a pas de poids pouvant être entraîné.

Dosovitskiy et al. (2015) définit la corrélation de deux patchs de taille Ω, centrée en x_1, et x_2 pour les descripteurs f_1 et f_2 comme suit :

$$C(x_1, x_2) = \sqrt{\sum_{o \in \Omega} f_1(x_1 + o).f_2(x_2 + o)} \tag{7.9}$$

Cette opération est répétée autour de x_2, dans un voisinage censé contenir le déplacement entre les images fournies en entrée. La zone de recherche doit donc dépendre de la tâche (et du niveau de sous-échantillonnage effectué pour calculer les descripteurs). Il est donc plus simple à définir dans le cas de l'utilisation de paires stéréoscopiques puisqu'on peut avoir une connaissance a priori de la disparité entre les deux caméras fixes. D'un autre côté, avec l'utilisation d'images séquentielles la disparité varie avec la vitesse de déplacement de la caméra (dans le cas où elle est montée sur un véhicule par exemple) et il devient alors plus complexe de choisir la bonne taille de voisinage .

Ce faisant, la sortie de la couche de corrélation est dimensionnée par la taille du voisinage

Résolution d'entrée	taille	Nb. de descripteurs de corrélation	Nb. de descripteurs monoculaires
Basse résolution	(7,7)	49	424
Résolution moyenne	(11,11)	121	352
Haute résolution	(21,21)	441	32

TABLE 7.2 – Taille de chaque corrélation et taille des sorties correspondantes pour les différents niveaux de résolution présents dans l'architecture

exploré. Par exemple, le calcul de la corrélation dans une zone de 7×7 pixels donnera 49 descripteurs de corrélation. Dans le cadre de notre étude multi-échelles, le nombre de sorties de la couche de corrélation est arbitrairement fixé à 473 descripteurs, quelle que soit la taille de la corrélation. Les descripteurs supplémentaires sont monoculaires et sont le résultat d'une convolution sur les derniers descripteurs d'une des deux images d'entrée (voir figure 7.7).

Le tableau 7.2 récapitule la taille des corrélations utilisées pour entraîner le réseau sur la base de données Virtual KITTI pour la tâche d'estimation de profondeur à partir de paires d'images issues de séquences temporelles.

7.5.3 Estimation globale puis raffinement local

L'idée directrice pour la conception de notre architecture est la prédiction multi-échelles. Comme nous l'avons expliqué précédemment, les entrées sous-échantillonnées successives et leurs corrélations à différents niveaux d'encodage permettent d'obtenir un encodeur multi-échelles. Par ailleurs, le module de raffinement, c'est-à-dire la partie décodage de notre réseau, intègre également une stratégie multi-échelles avec une prédiction grossière que l'on affine au fur et à mesure, inspirée de Dosovitskiy et al. (2015). Après une première étape d'encodage-décodage, le réseau produit une carte de profondeur grossière à faible résolution, puis nous l'affinons à l'aide du résultat de corrélation suivant, et ainsi de suite. A la fin, nous obtenons quatre cartes de profondeur à différentes résolutions.

Les résultats des deux corrélations suivantes sont successivement injectés dans le module de raffinement, concaténés avec la carte de profondeur précédente et l'image d'entrée de référence sous-échantillonnée à la bonne résolution. Comme toutes ces informations proviennent de différentes sources, un encodeur peu profond a été ajouté à chaque fois après l'incorporation de ces nouvelles informations hétérogènes. Ce module vise à les homogénéiser afin d'améliorer la qualité de nos prédictions.

En pratique, les données corrélées à chaque résolution sont mieux codées, décodées et mises à l'échelle, en alternant les modules DED et EnF. Bien que le module DED ne change pas la résolution globale des descripteurs, il permet d'ajouter une branche de sortie dédiée qui transforme les descripteurs en carte de profondeur.

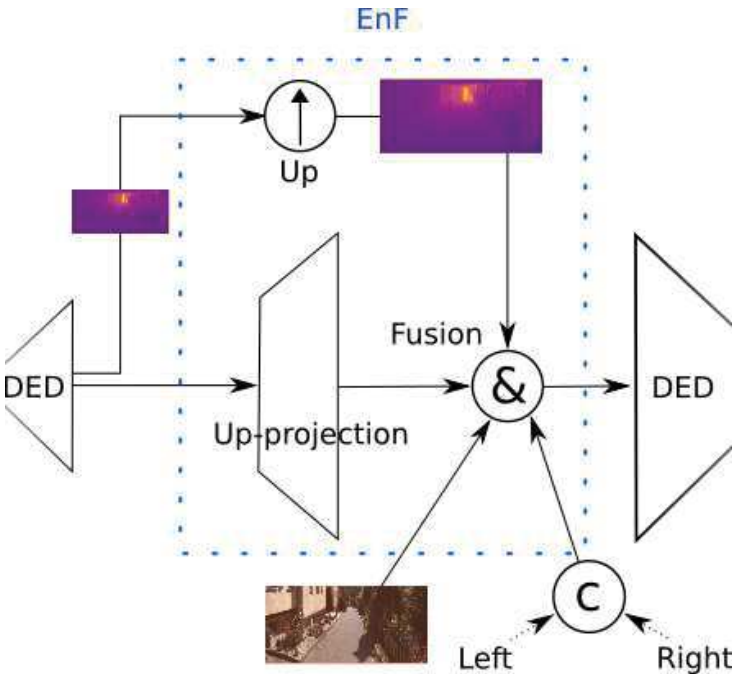


Figure 7.8 – Détails du module Expand and Fuse (EnF)

Le module Expand and Fuse (EnF) est à l'origine de toutes les prédictions de résolution supérieure. Pour ce faire, il combine trois principales fonctionnalités. Tout d'abord, une couche de sur-échantillonnage redimensionne la carte de profondeur précédente d'un facteur deux. Un module d'up-projection similaire à celui proposé par Laina et al. (2016) multiplie par deux la résolution des descripteurs en entrée. Enfin, une étape de fusion concatène la carte de profondeur et les descripteurs mise à l'échelle avec les sorties de la corrélation correspondante ainsi que l'image d'entrée de référence préalablement sous-échantillonnée (voir figure 7.8)

La séquence globale du décodeur est la suivante. A partir de la couche de corrélation de plus bas niveau, un premier module DED fournit une carte de profondeur grossière. Ensuite, trois couples EnF-DED additionnels raffinent la prédiction jusqu'à atteindre la moitié de la résolution de l'image d'origine. Il est à noter que le dernier module EnF n'utilise pas de résultat de corrélation. La table 7.3 résume les différentes opérations mises en œuvre dans le décodeur pour affiner la prédiction.

7.5.4 Procédure d'entrainement multi-échelles

Pour la procédure d'entrainement, nous proposons une nouvelle approche qui consiste à entrainer le réseau à apprendre séquentiellement chaque résolution de sortie. Nous commençons par la sortie de plus basse résolution plutôt que d'entrainer toutes les résolutions en même temps ou en ayant simplement pour but de produire la résolution complète finale. Cette approche s'inspire d'approches pyramidales couramment utilisée en traitement d'images, ainsi que

Module	Nb. canaux d'entrée	Nb. canaux de sortie	Échelle
Up projection	1024	512	x2
Up projection	512	256	x4
Concatenation	256	516	x4
Resnet Proj	516	1032	x2
Resnet Skip	1032	1032	x2
Resnet Skip	1032	1032	x2
Up projection	1032	516	x4
Up projection	516	128	x8
Concatenation	128	196	x8
Resnet Proj	196	392	x4
Resnet Skip	392	392	x4
Resnet Skip	392	392	x4
Up projection	392	196	x8
Up projection	169	64	x16
Concatenation	64	68	x16
Resnet Proj	68	136	x8
Resnet Skip	136	136	x8
Resnet Skip	136	136	x8
Up projection	136	68	x16
Convolution	68	1	x16

TABLE 7.3 – Résumé des différentes opérations dans le décodeur et de l'évolution du nombre de canaux

des résultats de Lin et al. (2017b) pour la détection d'objets. Elle a pour objectif d'entrainer notre réseau à la prédiction d'une structure globale de la carte de profondeur puis de l'affiner progressivement à l'aide des images, des prédictions de cartes de profondeur intermédiaires et des résultats de corrélation.

Pour traiter cet entrainement multi-échelles, nous introduisons également une fonction de coût évolutive qui dépend de la résolution cible à entrainer :

$$L_i = \sum_{k=1}^{i} \frac{1}{2^{2i-1}} ||Z_i - Z'_i||_2^2 \tag{7.10}$$

où Z'_i est la vérité terrain de profondeur, Z_i la prédiction à la résolution i avec $i = 1$ représentant la plus faible résolution et $i = 4$ la plus grande. Les différents coefficients ont été calculés pour attribuer le même poids à l'erreur pour chaque pixel de chaque carte de profondeur. La phase d'entrainement commence par l'entrainement de L_1 uniquement, pendant 2 epochs, puis L_2 pendant 2 epochs, 5 epochs avec L_3 et finalement l'entrainement de L_4.

7.6 Expériences

Nous évaluons notre architecture ainsi que notre approche d'apprentissage sur deux jeux de données différents : les séquences d'images brutes de KITTI Geiger et al. (2013) et les séquences synthétiques de Virtual KITTI Gaidon et al. (2016). Nous utilisons les mêmes ensembles d'entrainement, de validation et de test de KITTI que ceux proposés par Eigen et al. (2014). D'autre part, Virtual KITTI offre des cartes de profondeur parfaites et denses pour chaque image des séquences monoculaires synthétiques disponibles. Une séquence a été isolée pour l'utiliser comme ensemble de test tandis que les quatre autres constituent l'ensemble d'entrainement.

L'expérimentation du modèle sur ces deux jeux de données différents permet d'analyser son comportement selon différentes approches, telles que l'entrainement avec des labels parcimonieux ou denses, ou encore l'estimation de profondeur à partir de données stéréoscopiques (KITTI) ou temporelles (Virtual KITTI) ce qui nous permet de montrer le comportement du réseau proposé dans différents contextes.

7.6.1 Protocole expérimental

Toutes les couches de la partie encodeur du réseau sont issues du ResNet50 pré-entrainé sur ImageNet. La partie décodeur est initialisée aléatoirement selon l'initialisation Xavier Glorot et Bengio (2010). L'optimisation a été effectuée avec l'optimiseur Adam avec les paramètres 0.9 et 0.99. Le pas d'apprentissage est initialisé à 1.10^{-4} et on lui applique une décroissance exponentielle toutes les 15 epochs.

Pour KITTI, des crops de taille 480×192 sont utilisés pour l'entraînement. Ils sont de taille 320×256 pour Virtual KITTI. Les images ne sont pas redimensionnées au moment du test, l'estimation de profondeur se fait donc à pleine résolution. Chaque apprentissage a été réalisée selon la procédure d'entrainement multi-échelles proposée précédemment.

7.6.2 Métriques d'évaluation

Le modèle proposé pour chaque jeu de données a été évalué selon les critères rmse, rel, Threshold ainsi qu'avec la *Squared Relative Difference* (SRD) définie comme suit :

$$SRD : \frac{1}{T}\sum_{i=1}^{N}\frac{(\hat{y}_i - y_i)^2}{y_i} \tag{7.11}$$

Toutes les profondeurs ont été plafonnées à 80 mètres.

7.6.3 Résultats : Virtual KITTI

Pour évaluer l'efficacité de notre modèle sur des cartes de profondeur parfaites et denses, il a d'abord été entrainé en utilisant le jeu de données Virtual KITTI. Comme il n'y a pas de

FIGURE 7.9 – Illustration de l'approche multi-échelle sur Virtual KITTI. De haut en bas on trouve : une des images d'entrée, la vérité terrain puis les quatre prédictions de profondeur, de la résolution la plus faible à la plus élevée.

Approche	RMSE	ARD	SRD
Laina et al. (2016)	14.110	0.627	8.036
Réseau MSDOS	8.612	0.459	4.153

TABLE 7.4 – Résultats quantitatifs de test du réseau de Laina et al. (2016) entrainé sur Virtual KITTI et de notre méthode. La profondeur de la vérité terrain a été plafonnée à 80 m pour être similaire au jeu de données KITTI

paires stéréoscopiques disponibles dans ce jeu de données, des images consécutives issues de la même séquence ont été utilisées pour l'apprentissage du réseau. N'ayant pas de référence pour se comparer dans l'état de l'art et afin d'étudier l'efficacité du modèle proposé, nous nous comparons au meilleur réseau de l'état de l'art pour l'estimation de profondeur monoculaire, à savoir celui de Laina et al. (2016). Nous ré-entrainons ce modèle selon la procédure d'apprentissage proposée dans l'article original mais sur la base de données Virtual KITTI. Nous obtenons ainsi un résultat de référence auquel se comparer.

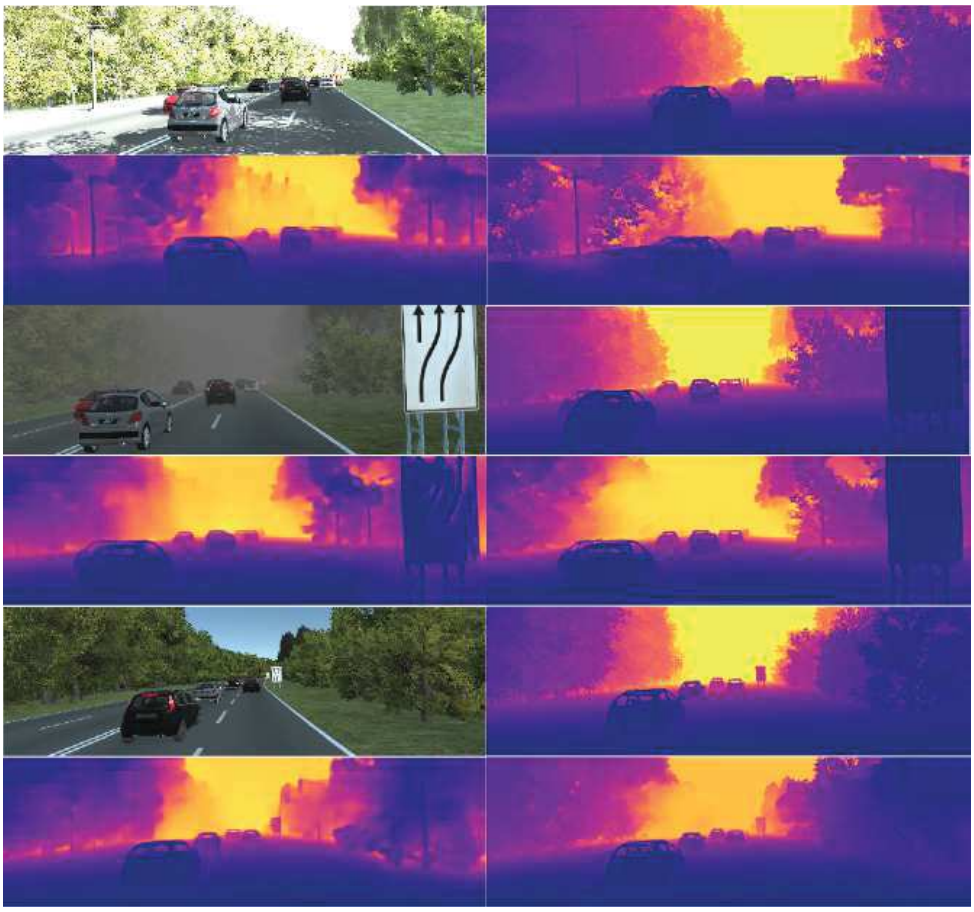


FIGURE 7.10 – Illustration des résultats de notre meilleur modèle sur Virtual KITTI. Pöur chaque image de test on trouve : en haut à gauche une des images RGB donnée en entrée ; en haut à droite la vérité terrain de profondeur plafonnée à 80m ; en bas à gauche la prédiction par le réseau de Laina et al. (2016) ; en bas à droite la prédiction de notre modèle

La table 7.4 résume les différents scores obtenus sur le jeu de test de Virtual KITTI. Puisque la base de données ne contient pas de paires stéréo, le modèle a été entrainé et testé uniquement avec des paires d'images séquentielles. La figure 7.10 illustre différentes prédictions sur des données de test faites par notre modèle et par celui de Laina et al. (2016). Les résultats sont meilleurs et affichent une amélioration de près de 40% entre les deux réseaux. Visuellement, les estimations de profondeur du modèle que nous proposons sont beaucoup plus détaillées que celles estimées à partir du réseau de Laina et al. (2016). Par exemple, des détails fins tels que les ramifications des différents arbres apparaissent dans les estimations de notre modèle. Il convient tout de même de rappeler que nous comparons nos résultats issus d'un couple d'images à ceux d'un réseau monoculaire, faute d'autres référence de l'état de l'art.

	à minimiser			à maximiser		
Approche	RMSE	ARD	SRD	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen et al. (2014)	7.156	0.215	-	0.692	0.899	0.967
Liu et al. (2016)	6.986	0.217	1.841	0.647	0.882	0.961
Godard et al. (2017)	5.849	0.141	1.369	0.818	0.929	0.966
Kuznietsov et al. (2017) (supervisé)	4.815	0.122	0.763	0.845	0.957	0.987
Kuznietsov et al. (2017) (semi-supervisé)	4.621	0.113	0.741	0.862	0.960	0.986
Réseau MSDOS (entrées stéréoscopiques)	3.102	0.098	0.520	0.914	0.969	0.987
Réseau MSDOS (pré-entrainé sur Virtual Kitti)	**3.050**	**0.091**	**0.472**	0.921	0.968	0.987
Réseau MSDOS (entrées séquentielles)	4.612	0.157	1.082	0.813	0.932	0.972

Table 7.5 – Résultats quantitatifs des méthodes de l'état de l'art et de notre méthode sur le jeu de test de KITTI, selon le split proposé par Eigen et al. (2014). Les meilleurs résultats sont indiqués en gras

7.6.4 Résultats : KITTI

Nous évaluons également notre réseau MSDOS sur le jeu de données KITTI où seule une vérité terrain parcimonieuse est disponible. Nous pouvons évaluer selon deux cas d'application différents puisque la base de données KITTI fournit à la fois des paires stéréo et des informations séquentielles.

La plupart des réseaux de neurones profonds de l'état de l'art conçus pour l'estimation de profondeur utilisent une seule image pour la prédiction. L'autre image de la paire stéréoscopique est généralement utilisée dans la partie semi-supervisée de la fonction de coût pour la renforcer et ainsi obtenir de meilleures prédictions, plus robustes. Afin de comparer nos performances avec ces travaux, le réseau MSDOS a d'abord été entrainé en utilisant des paires stéréo. Pour ce faire, nous utilisons deux stratégies d'initialisation différentes : tout d'abord la même stratégie que celle décrite pour l'entrainement avec les données de Virtual KITTI mais aussi une seconde où nous utilisons les coefficients du modèle pré-entraîné avec Virtual KITTI.

La table 7.5 résume les performances quantitatives obtenues par les méthodes de l'état de l'art et par notre approche pour l'estimation de profondeur à partir de paire d'images stéréoscopiques. Nous montrons ici que notre modèle pré-entrainé sur les données de Virtual KITTI à partir de couples d'images séquentielles obtient les meilleurs résultats pour chacune des métriques d'évaluation.

La figure 7.11 illustre les prédictions du réseau MSDOS sur le jeu de test de KITTI. Pour une meilleure comparaison visuelle avec la vérité terrain de profondeur, nous avons densifié la vérité terrain parcimonieuse issue du LiDAR par une triangulation de Delaunay suivie d'une interpolation linéaire. Un filtre bilatéral est ensuite appliqué pour améliorer la netteté. Une chose intéressante que nous notons dans ces résultats est le fait que les prédictions de notre modèle MSDOS semblent visuellement mieux correspondre aux images RGB que la vérité terrain densifiée, bien que seuls les points valides de la vérité terrain aient été utilisés pour l'entraînement du réseau.

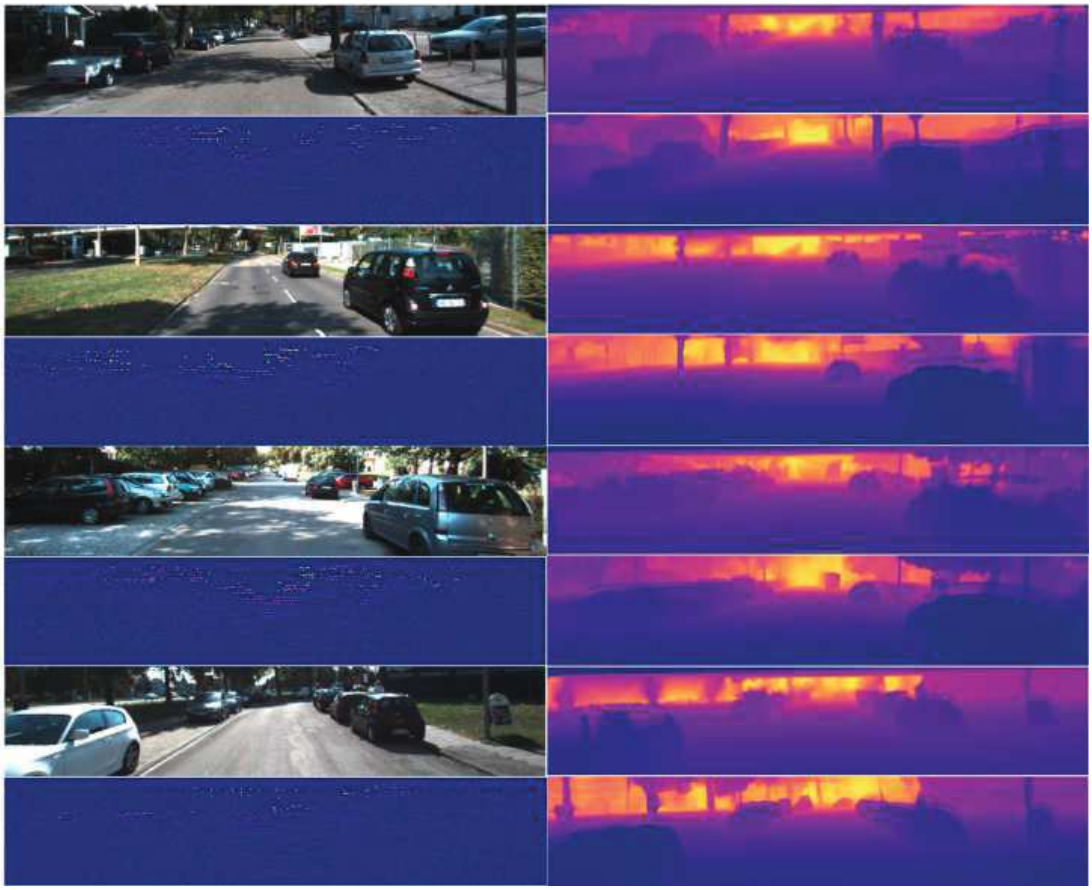


FIGURE 7.11 – Illustration des résultats de notre meilleur modèle sur KITTI. Pour chaque image on trouve : en haut à gauche une des deux images RGB de la paire stéréo en entrée ; en bas à gauche la vérité terrain de profondeur parcimonieuse ; en haut à droite la vérité terrain dense obtenue par une triangulation de Delaunay sur la distribution parcimonieuse et une interpolation linéaire suivie d'un filtre bilatéral pour améliorer la netteté ; en bas à droite notre estimation de profondeur en utilisant le réseau MSDOS

Enfin, nous nous intéressons aux résultats dans le cas de l'utilisation d'une paire d'images séquentielles. C'est pourquoi, pour comparer les performances du réseau MSDOS dans le cas où les images d'entrée proviennent de la même caméra, le modèle a été entrainé en utilisant les séquences d'images disponibles dans la base de données KITTI. Les résultats obtenus sont affichés dans la table 7.5.

Nous voyons que notre modèle produit également des résultats au niveau de l'état de l'art dans cette configuration, bien que moins bons que ceux obtenus en utilisant des paires stéréoscopiques. Cet écart de performances peut s'expliquer par le fait qu'une plus grande quantité d'informations 3D est contenue dans une paire stéréo que dans une paire séquentielle (dans le cadre d'une caméra montée sur un véhicule, il ne sera pas rare, par exemple, qu'un déplacement temporel induise plus de régions visibles dans une seule des deux vues).

7.7 Conclusion

L'énorme quantité de données disponibles de nos jours et le développement de nouveaux capteurs permettent de concevoir des algorithmes de *deep learning* permettant d'estimer la profondeur avec une grande précision. Cependant, la plupart des méthodes récentes n'utilisent que des images provenant d'une seule caméra. Nous nous intéressons ici à l'amélioration que pourrait apporter l'utilisation de plusieurs images d'une même caméra mais acquises à des instants différents, ou de plusieurs images issues d'une paire stéréoscopique. C'est pourquoi nous proposons une nouvelle méthode d'estimation de profondeur à partir de plusieurs entrées que nous comparons avec des algorithmes basés sur une seule image.

Pour ce faire, nous proposons une nouvelle architecture : le réseau MSDOS. Il permet d'obtenir de meilleures performances que celles de l'état de l'art en matière d'estimation de profondeur multi-oculaires sur deux jeux de données parmi les plus utilisés pour des environnements extérieurs : KITTI et Virtual KITTI. Les résultats montrent une amélioration à la fois quantitative mais aussi en qualité visuelle.

Pour obtenir de tels résultats, nous utilisons différents modules dans le réseau MSDOS qui ont été inspirés d'architectures de l'état de l'art ainsi qu'une procédure de décodage et d'apprentissage spécifiques. Cependant, étant donné qu'il n'existe encore que peu de systèmes de prédiction de profondeur multi-entrées, il est difficile d'estimer l'écart entre les performances de ces systèmes et celui des systèmes à une seule entrée.

Ces contributions peuvent facilement être adaptées à d'autres tâches telles que la segmentation sémantique ou l'estimation du flot optique. Par conséquent, nous pensons que l'une des extensions principales de ces travaux consiste à adapter une architecture similaire à d'autres compétitions de *deep learning*. Une autre extension possible serait d'explorer le potentiel du réseau MSDOS à travers un apprentissage non supervisé, comme c'est le cas de plusieurs réseaux de neurones profonds récents pour l'estimation de profondeur.

Bibliographie

Michael Riis Andersen, Thomas Jensen, Pavel Lisouski, Anders Krogh Mortensen, Mikkel Kragh Hansen, Torben Gregersen, et Peter Ahrendt. Kinect depth sensor evaluation for computer vision applications. *Aarhus University, Technical Report*, 2012.

V. Aslantas. A depth estimation algorithm with a single image. *Optics express*, 2007.

Christian Banz, Sebastian Hesselbarth, Holger Flatt, Holger Blume, et Peter Pirsch. Real-time stereo vision system using semi-global matching disparity estimation : Architecture and fpga-implementation. *SAMOS*, 2010.

Jonathan T. Barron et Jitendra Malik. Shape, albedo, and illumination from a single image of an unknown object. *CVPR*, 2012.

Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2) :157–166, 1994.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, et Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv :1505.05424*, 2015.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2) :123–140, 1996.

I Bulthoff, HH Bulthoff, et P Sinha. Top-down influences on stereoscopic depth-perception. *Nature Neuroscience*, 1(3), 1998.

Yuanzhouhan Cao, Zifeng Wu, et Chunhua Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

Ayan Chakrabarti, Jingyu Shao, et Greg Shakhnarovich. Depth from a single image by harmonizing overcomplete local network predictions. In *Advances in Neural Information Processing Systems*, pages 2658–2666, 2016.

Liang-Chieh Chen, George Papandreou, Florian Schroff, et Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv :1706.05587*, 2017.

Alejo Concha, Wajahat Hussain, Luis Montano, et Javier Civera. Manhattan and piecewise-planar constraints for dense monocular mapping. *RSS*, 2010.

Armen Der Kiureghian et Ove Ditlevsen. Aleatory or epistemic ? does it matter ? *Structural Safety*, 31(2) :105–112, 2009.

N. Mohd Disa. Lidar : A review on generating digital true orthophoto. *CSPA*, 2011.

M. Domanski, J. Konieczny, M. Kurc, A. Luczak, J. Siast, O. Stankiewicz, et K. Wegner. Fast depth estimation on mobile platforms and fpga devices. *3DTV-CON*, 2015.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, et Thomas Brox. Flownet : Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

David Eigen et Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

David Eigen, Christian Puhrsch, et Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

Abdelrahman Eldesokey, Michael Felsberg, et Fahad Shahbaz Khan. Propagating confidences through cnns for sparse data regression. *arXiv preprint arXiv :1805.11913*, 2018.

David F Fouhey, Wajahat Hussain, Abhinav Gupta, et Martial Hebert. Single image 3d without a single 3d image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1053–1061, 2015.

Adrien Gaidon, Qiao Wang, Yohann Cabon, et Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016.

Yarin Gal et Zoubin Ghahramani. Dropout as a bayesian approximation : Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

Varun Ganapathi, Christian Plagemann, Daphne Koller, et Sebastian Thrun. Real time motion capture using a single time-of-flight camera. *CVPR*, 2010.

Yaroslav Ganin et Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv :1409.7495*, 2014.

Andreas Geiger, Philip Lenz, et Raquel Urtasun. Are we ready for autonomous driving ? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Andreas Geiger, Philip Lenz, Christoph Stiller, et Raquel Urtasun. Vision meets robotics : The kitti dataset. *The International Journal of Robotics Research*, 32(11) :1231–1237, 2013.

Stuart Geman, Elie Bienenstock, et René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1) :1–58, 1992.

Jason Geng. Structured-light 3d surface imaging : a tutorial. *Advances in Optics and Photonics*, 3(2) :128–160, 2011.

Xavier Glorot et Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

Tilmann Gneiting et Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477) :359–378, 2007.

Tilmann Gneiting, Fadoua Balabdaoui, et Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 69 (2) :243–268, 2007.

Clément Godard, Oisin Mac Aodha, et Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

Abhinav Gupta, Alexei A. Efros, et Martial Hebert. Blocks world revisited : Image understanding using qualitative geometry and mechanics. *ECCV*, 2010.

Pavel Gurevich et Hannes Stuke. Learning uncertainty in regression tasks by artificial neural networks. *arXiv preprint arXiv :1707.07287*, 2017.

Simon Hadfield et Richard Bowden. Exploiting high level scene cues in stereo reconstruction. *ICCV*, 2015.

Jungong Han, Ling Shao, Dong Xu, et Jamie Shotton. Enhanced computer vision with microsoft kinect sensor : A review. *IEEE Transactions on Cybernetics*, 2013.

Miles Hansard, Seungkyu Lee, Ouk Choi, et Radu Horaud. Time of flight cameras : Principles, methods, and applications. *hal-00725654*, 2012.

Richard Hartley et Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

Kaiming He, Jian Sun, et Xiaoou Tang. Single image haze removal using dark channel prior. *TPAMI*, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, et Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Varsha Hedau, Derek Hoiem, et David Forsyth. Thinking inside the box : Using appearance models and context based on room geometry. *ECCV*, 2010.

José Miguel Hernández-Lobato et Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, et Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv :1207.0580*, 2012.

Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2) :328–341, 2008.

Derek Hoiem, Alexei Efros, et Martial Hebert. Geometric context from a single image. *ICCV*, 2005.

Stefan Hrabar. An evaluation of stereo and laser-based range sensing for rotorcraft unmanned aerial vehicle obstacle avoidance. *Journal of Field Robotics*, 2012.

Jiashen Hua et Xiaojin Gong. A normalized convolutional neural network for guided sparse depth upsampling. In *IJCAI*, pages 2283–2290, 2018.

Zixuan Huang, Junming Fan, Shuai Yi, Xiaogang Wang, et Hongsheng Li. Hms-net : Hierarchical multi-scale sparsity-invariant network for sparse depth completion. *arXiv preprint arXiv :1808.08685*, 2018.

Tak-Wai Hui, Chen Change Loy, et Xiaoou Tang. Depth map super-resolution by deep multi-scale guidance. In *European Conference on Computer Vision*, pages 353–369. Springer, 2016.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcomb, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et Andrew Fitzgibbon.

Kinectfusion : Real-time 3d reconstruction and interaction using a moving depth camera. *UIST*, 2011.

Omid Hosseini Jafari, Oliver Groth, Alexander Kirillov, Michael Ying Yang, et Carsten Rother. Analyzing modular cnn architectures for joint depth prediction and semantic segmentation. *arXiv preprint arXiv :1702.08009*, 2017.

Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, et Fawzi Nashashibi. Sparse and dense data with cnns : Depth completion and semantic segmentation. In *2018 International Conference on 3D Vision (3DV)*, pages 52–60. IEEE, 2018.

E. Kardas. Monocular cues in depth perception.{Online} : http ://peace. saumag. edu/faculty/-kardas/courses/gp weiten/c4sandp. *MonoCues. html*, 2005.

Kevin Karsch, Ce Liu, et Sing Bing Kang. Depth transfer : Depth extraction from video using non-parametric sampling. *IEEE transactions on pattern analysis and machine intelligence*, 36 (11) :2144–2158, 2014.

Alex Kendall et Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision ? In *Advances in Neural Information Processing Systems*, pages 5580–5590, 2017.

Nitish Shirish Keskar et Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv :1712.07628*, 2017.

Kourosh Khoshelham et Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 2012.

Diederik P Kingma et Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

Diederik P Kingma, Tim Salimans, et Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

Hans Knutsson et C-F Westin. Normalized and differential convolution. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*, pages 515–523. IEEE, 1993.

Iasonas Kokkinos. Surpassing humans in boundary detection using deep learning. *CoRR*, 2015.

Andreas Kolb, Erhardt Barth, Reinhard Koch, et Rasmus Larsen. Time-of-flight cameras in computer graphics. *Computer Graphics Forum*, 2010.

Janusz Konrad, Meng Wang, et Prakash Ishwar. 2d-to-3d image conversion by learning depth from examples. *CVPRW*, 2012.

Adarsh Kowdle, Noah Snavely, et Tsuhan Chen. Recovering depth of a dynamic scene using real world motion prior. *ICIP*, 2012.

Alex Krizhevsky, Ilya Sutskever, et Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 2012a.

Alex Krizhevsky, Ilya Sutskever, et Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012b.

Yevhen Kuznietsov, Jörg Stückler, et Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6647–6655, 2017.

L'ubor Ladický, Jianbo Shi, et Marc Pollefeys. Pulling things out of perspective. *CVPR*, 2014.

Koonchun Lai, Liefeng Bo, Xiaofeng Ren, et Dieter Fox. Detection-based object labeling in 3d scenes. *ICRA*, 2012.

Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, et Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

Balaji Lakshminarayanan, Alexander Pritzel, et Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6405–6416, 2017.

Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, et Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

David C. Lee, Abhinav Gupta, Martial Hebert, et Takeo Kanade. Estimating spatial layout of rooms using volumetric reasoning about object and surfaces. *NIPS*, 2010.

Anat Levin, Dani Lischinski, et Yair Weiss. Colorization using optimization. In *ACM transactions on graphics (tog)*, volume 23, pages 689–694. ACM, 2004.

Bo Li, Chunhua Shen, Yuchao Dai, Anton van den Hengel, et Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127, 2015.

Jun Li, Reinhard Klein, et Angela Yao. A two-streamed network for estimating fine-scaled depth maps from single rgb images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Larry Li. Time-of-flight camera - an introduction. *Technical White Paper, Texas Instruments*, 2014.

Yijun Li, Jia-Bin Huang, Narendra Ahuja, et Ming-Hsuan Yang. Deep joint image filtering. In *European Conference on Computer Vision*, pages 154–169. Springer, 2016.

Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, et Serge Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017a.

Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, et Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017b.

Beyang Liu, Stephen Gould, et Daphne Koller. Single image depth estimation from predicted semantic labels. *CVPR*, 2010.

Fayao Liu, Chunhua Shen, et Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5162–5170, 2015.

Fayao Liu, Chunhua Shen, Guosheng Lin, et Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10) :2024–2039, 2016.

Angeles Lopez, Elena Garces, et Diego Gutierrez. Depth from a single image through user interaction. *CEIG*, 2014.

Chaochao Lu et Xiaoou Tang. Surpassing human-level face verification performance on lfw with gaussianface. In *AAAI*, pages 3811–3819, 2015.

Fangchang Ma et Sertac Karaman. Sparse-to-dense : Depth prediction from sparse depth samples and a single image. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

Warren S McCulloch et Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4) :115–133, 1943.

Moritz Menze et Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Jeff Michels, Ashutosh Saxena, et Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *ICML*. ACM, 2005.

Rahul Mohan. Deep Deconvolutional Networks for Scene Parsing. *arXiv*, 2014.

Pushmeet Kohli Nathan Silberman, Derek Hoiem et Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

Art B Owen. A robust hybrid of lasso and ridge regression. *Contemporary Mathematics*, 443(7) : 59–72, 2007.

Sinno Jialin Pan et Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10) :1345–1359, 2010.

N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, et A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 372–387, 2016.

Nadia Payet et Sinisa Todorovic. Scene shape from texture of objects. *CVPR*, 2011.

Cristiano Premebida, Goncalo Monteiro, Urbano Nunes, et Paulo Peixoto. A lidar and vision-based approach for pedestrian and vehicle detection and tracking. *ITSC*, 2007.

Joaquin Quinonero-Candela, Carl Edward Rasmussen, Fabian Sinz, Olivier Bousquet, et Bernhard Schölkopf. Evaluating predictive uncertainty challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 1–27. Springer, 2006.

Srikumar Ramalingam et Matthew Brand. Lifting 3d manhattan lines from a single image. *ICCV*, 2013.

Eduardo Ramos-Diaz, Victor Gonzalez-Huitron, Volodymyr I. Ponomaryov, et Araceli Hernandez-Fragoso. 2d to 3d conversion implemented in different hardware. *Real-Time Image and Video Processing*, 2015.

Ali S. Razavian, Hossein Azizpour, Josephine Sullivan, et Stefan Carlsson. Cnn features off-the-shelf : an astounding baseline for recognition. *CVPRW*, 2014.

Mengye Ren, Andrei Pokrovsky, Bin Yang, et Raquel Urtasun. Sbnet : Sparse blocks network for fast inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8711–8720, 2018.

Olaf Ronneberger, Philipp Fischer, et Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015.

Anirban Roy et Sinisa Todorovic. Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5506–5514, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, et Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3) :211–252, 2015. doi : 10.1007/s11263-015-0816-y.

Ashutosh Saxena, Sung H. Chung, et Andrew Y. Ng. Learning depth from single monocular images. *NIPS*, 2005.

Ashutosh Saxena, Sung H. Chung, et Andrew Y. Ng. 3-d depth reconstruction from a single still image. *IJCV*, 2007.

Ashutosh Saxena, Min Sun, et Andrew Y. Ng. Make3d : Learning 3d scene structure from a single still image. *TPAMI*, 2008.

D. Scharstein, R. Szeliski, et R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proc.s IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001.

Daniel Scharstein et Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002.

Alexander G. Schwing et Raquel Urtasun. Efficient exact inference for 3d indoor scene understanding. *ECCV*, 2012.

Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, et Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *CVPR*, 2006.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, et Rob Fergus. Indoor segmentation and support inference from rgbd images. *ECCV*, 2012.

Karen Simonyan et Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, et Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1) :1929–1958, 2014.

Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, et Andreas Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017.

Laura Uusitalo, Annukka Lehikoinen, Inari Helle, et Kai Myrberg. An overview of methods to evaluate uncertainty of deterministic models in decision support. *Environmental Modelling & Software*, 63 :24–31, 2015.

J. Vaze et J. Teng. High resolution lidar dem - how good is it ? *MODSIM*, 2007.

Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, et Katerina Fragkiadaki. Sfm-net : Learning of structure and motion from video. *arXiv preprint arXiv :1704.07804*, 2017.

Benzhang Wang, Yiliu Feng, et Hengzhu Liu. Multi-scale features fusion from sparse lidar data and single image for depth completion. *Electronics Letters*, 2018.

Jingwei Wang, Hao Xu, et C-C Jay Kuo. Single-image depth inference based on blur cues. *APSIPA ASC*, 2012.

Liang Wang, Hailin Jin, Ruigang Yang, et Minglun Gong. Stereoscopic inpainting : Joint color and depth completion from stereo images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, et Alan L Yuille. Towards unified depth and semantic prediction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2015a.

Wei Wang, Vincent W Zheng, Han Yu, et Chunyan Miao. A survey of zero-shot learning : Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2) :13, 2019.

Xiao Wang et Han Wang. Markov random field modeled range image segmentation. *Pattern Recognition Letters*, 2004.

Xiaolong Wang, David Fouhey, et Abhinav Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–547, 2015b.

Andreas Wedel, Uwe Franke, Jens Klappstein, Thomas Brox, et Daniel Cremers. Realtime depth estimation and obstacle detection from monocular video. *DAGM*, 2006.

Qingqing Wei. Converting 2d to 3d : A survey. *Research Assignment, Delft University of Technology*, 2005.

Bing Wu et Zijiang J. Ooi, Teng Lengand He. Perceiving distance accurately by a directional process of integrating ground information. *Nature*, 2004.

Yalin Xiong et Steven A. Shafer. Depth from focusing and defocusing. *CVPR*, 1993.

Bing Xu, Naiyan Wang, Tianqi Chen, et Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv :1505.00853*, 2015.

Hongyang Xue, Shengming Zhang, et Deng Cai. Depth image inpainting : Improving low rank matrix completion with low gradient regularization. *IEEE Transactions on Image Processing*, 26(9) :4311–4320, 2017.

Ryo Yonetani, Akisato Kimura, Hitoshi Sakano, et Ken Fukuchi. Single image segmentation with estimated depth. *BMVC*, 2012.

Fisher Yu et Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv :1511.07122*, 2015.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, et Jianxiong Xiao. Lsun : Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv :1506.03365*, 2015.

Fisher Yu, Vladlen Koltun, et Thomas A. Funkhouser. Dilated residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 636–644, 2017.

Stella X. Yu, Hao Zhang, et Jitendra Malik. Inferring spatial layout from a single image via depth-ordered grouping. *CVPRW*, 2008.

Wenjun Zeng. Microsoft kinect sensor and its effect. *MultiMedia*, 2012.

Guofeng Zhang, Jiaya Jia, Tien-Tsin Wong, et Hujun Bao. Consistent depth maps recovery from a video sequence. *TPAMI*, 2009.

Tinghui Zhou, Matthew Brown, Noah Snavely, et David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017.

Quanwen Zhu, Long Chen, Qingquan Li, Ming Li, Andreas Nüchter, et Jian Wang. 3d lidar point cloud based intersection recognition for autonomous driving. *IV*, 2012.

Shaojie Zhuo et Terenc Sim. On the recovery of depth from a single defocused image. *CAIP*, 2009.

Laurent Zwald et Sophie Lambert-Lacroix. The berhu penalty and the grouped effect. *arXiv preprint arXiv :1207.6868*, 2012.