REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre IVA3/M2/2021

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Image et Vie Artificielle (IVA)**

Deep learning pour la reconstruction des images HDRs

Par :
BAHRI AICHA

Soutenu le 06/07/2021 devant le jury composé de :

Nom Prénom	grade	Président
BABAHENINI Djihane	MCB	Rapporteur
Nom Prénom	grade	Examinateur

Année universitaire 2020-2021

Remerciement

”Au début, nous remercions Allah qui nous a aidés à accomplir ce travail, et qui a été avec nous à tous les moments de notre chemin d’étude. A mes chers parent ma mère et mon père pour leur Patience, leur amour, leur soutien et leur Encouragement.

Je tiens à exprimer mon vif remerciement **Dr BABAHENINI Djihane** ma superviseure pour-son aide, ses conseils et ses encouragements depuis le premier jour jusqu’à maintenant, sans vous je ne pourrais pas présenter ce travail.

Mes remerciements vont également aux membres du jury d’avoir accepté d’évaluer mon travail.”

Résumé

La reconstruction HDR(haute gamme dynamique) est un sujet d'intérêt dans le passé et dans les années actuelles pour la communauté des chercheurs. Ce problème a beaucoup de solutions proposées avec differentes techniques classiques et modernes. L'une des techniques les plus récentes est l'apprentissage profond que nous trouvons beaucoup de contributions l'ont utilisé comme une solution a ce type de tâche et montre aussi de bons résultats. Deep learning pour la reconstruction HDR utilise l'architecture VGG (groupe de géométrie visuelle) basé sur le modèle CNN pour générer image les images HDRs à partir d'une image LDR (plage dynamique faible) en une seule exposition.

Mots-clés : HDR, LDR, Deep Learning, le réseau VGG-19, CNN.

Abstract

HDR (high dynamic range) reconstruction is a subject of past and present interest for the research community. This problem has been the subject of many proposed solutions with different classical and modern techniques. One of the most recent techniques is deep learning, which has been used in many contributions as a solution to this type of task and which gives good results. Deep learning for HDR reconstruction using VGG (visual geometry group) architecture based on CNN model to generate HDR image from LDR(Low Dynamic Range) image single exposure.

Mots-clés: HDR, LDR, Deep Learning, VGG-19 network, CNN.

ملخـص

إعادة اعمار الصورة النطاق الديناميكي العالي تشكل موضوعا مهما في الأعوام الماضية والحالية بالنسبة لمجتمع البحوث. ولهذا المشكلة العديد من الحلول المقترحة بتقنيات كلاسيكية وحديثة مختلفة. وإحدى أحدث التقنيات هي التعليم العميق باننا نجد اسهامات عديدة قد استخدمناها كحل لهذا النوع من المهام كما انها تظهر نتائج جيدة. ويستخدم التعليم العميق لإعادة بناء صورة النطاق الديناميكي العالي الهندسة المعمارية لمجموعة الهندسة البصرية القائمة على نموذج الشبكة الصبية المعقدة لتوليد صورة النطاق الديناميكي العالي انطلاقا من صورة النطاق الديناميكي المنخفض تعرّض واحد.

الكلمات المفتاحية : الصورة النطاق الديناميكي العالي. إعادة اعمار الصورة النطاق الديناميكي العالي. صورة النطاق الديناميكي المنخفض. التعليم العميق. الشبكة العصبية المعقدة. مجموعة الهندسة البصرية.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les images à haute dynamique de luminance (HDR images, High Dynamic Range) se sont imposé ces dernières années, avec l'émergence de l'intelligence artificielle et l'apprentissage automatique (Machine Learning) dérivée l'apprentissage profondeur (DL, deep learning) qui a été développée pour créer des algorithmes capables d'apprendre et de s'améliorer de manière autonome.

Les images HDRs ouvrant un nouveau domaine de concurrence pour les fabricants des systèmes basés sur le deep learning pour générer des images HDRs de haute qualité à partir des images de basse gamme dynamique (LDRs, Low Dynamic Range). Cette nouvelle branche d'imagerie est nécessaire pour différents domaines d'application. Elle permet de fournir des images d'une grande qualité, riches en détails donc en informations.

On constate l'intérêt, voir même la nécessité de l'utilisation de l'imagerie à grande gamme de dynamique dans divers domaines :

— **La vidéosurveillance :** La sécurité dans les magasins, sociétés, routes, banques et aéroports, et d'autres lieux, est assurée en partie ou exclusivement (dans les horaires non travaillés) par la vidéosurveillance. La vidéosurveillance est très contraignante la nuit où les scènes observées par les caméras de surveillance sont illuminées par des projecteurs. Une personne malveillante assise dans l'ombre ou derrière une source de lumière est difficilement reconnaissable par les caméras conventionnelles. On retrouve les mêmes problèmes avec les plaques d'immatriculation des voitures lorsque les phares de cette dernière sont allumés en face de l'objectif de la caméra.

— **Les systèmes autonomes de conduite :** Prenant l'exemple de l'entrée et la sortie d'un tunnel, les caméras utilisées dans les systèmes de décision des voitures autonomes récupèrent des images totalement saturées blanches de la sortie du tunnel ou des images complètement noires de l'entrée du tunnel (car la différence d'exposition entre l'intérieure et l'extérieur de tunnel est très grande), ce qui fait que la décision dans ces circonstances sera prise en se basant sur des informations incomplètes, voire erronées.

— **Le médical :** L'imagerie médicale est un recours obligatoire dans le diagnostic de certaines maladies. Bien que les systèmes d'imagerie médicale (scanner, IRM, écho, etc.) soient différents de ceux de l'imagerie grand public, l'imagerie à grande gamme de dynamique

peut être très utile dans ce domaine. Elle permet par exemple d'augmenter le contraste entre les tissus cancéreux et les tissus sains dans le cas de diagnostic du cancer.

— **L'astrophysique :** Les objets astrophysiques ont des intensités lumineuses très écartées. Prenant l'exemple de l'éclipse solaire, la surface de la lune capturée par une caméra conventionnelle est totalement noire, car l'intensité de la lumière du soleil situé derrière est beaucoup plus élevée que celle réfléchie de la surface de la lune. De même dans le cas de la photographie d'une nébuleuse, si on essaye d'avoir des détails du nuage de la poussière, le cœur de cette dernière se sature complètement en blanc. Il est impossible d'obtenir simultanément les détails du nuage et du cœur de la nébuleuse.

L'objectif principal de ce travail consiste à concevoir automatiquement un système basé sur les réseaux de neurones convolutifs (CNN, Convolutional Neural Networks) pour générer des images HDRs à partir d'une image de basse gamme dynamique (LDRs, Low Dynamic Range).

Le présent mémoire est organisé en quatre chapitres dont les thèmes sont donnés ci-dessous : dans un premier chapitre, nous avons introduit le rendu HDR à partir de système visuel humain jusqu'à l'affichage sur écran LDR en utilisant la technique de tone mapping. Dans le deuxième chapitre nous présentons le deep learning, les domaines d'application et les différentes architectures et nous avons décrit une synthèse des travaux connexes. Dans le troisième chapitre, nous aons présenté la conception générale et détaillée de notre système. Dans le quatrième chapitre nous avons présenté les étapes de l'implémentation de notre système et les outils utilisés pour développer notre application, et nous avons terminé notre mémoire par une conclusion générale.

Chapitre 1

Les images HDRs

1.1 Introduction

Les images à haute gamme dynamique (HDR images, pour High Dynamic Range) se sont imposées ces dernières années comme une manière de dépasser les limites des images 8 bits, que nous manipulons sur les ordinateurs. Cette évolution touche à la fois l'acquisition des images et leur restitution visuelle, ce dernier problème ayant donné lieu à de nombreuses publications sur les opérateurs de tone mapping (TMO). La question centrale est comment afficher des images HDRs sur des écrans LDRs ? et de comparer le rendu visuel d'une image HDR avec la même image de basse gamme dynamique (LDR, pour Low Dynamic Range).

1.2 La colorimétrie

La colorimétrie est la science de la mesure de la couleur. Bien que chaque personne perçoit les couleurs légèrement différentes. La CIE définit également un observateur de base pour la perception des couleurs.
Pour cela, elle a procédé à une expérimentation, où le sujet doit reproduire la couleur donnée sur l'écran en ajustant les intensités des trois couleurs primaires (rouge, vert et bleu). Chaque intensité peut être comprise entre -1 et 1. Figure 1.1 montre les courbes correspondantes obtenues à partir de différentes expérimentations.[1]
La longueur d'onde de la couleur donnée à l'origine est en abscisse, et Les trois couleurs primaires doivent être données pour obtenir l'intensité correspondante en ordonnée. La couleur donnée initiale $C\lambda$ en fonction des intensités $\overline{r}(\lambda)$, $\overline{g}(\lambda)$ et $\overline{b}(\lambda)$ des trois couleurs primaires R (red), G (green) et B (blue) peut donc être obtenue par la formule suivante :

$$C_\lambda = \overline{r}(\lambda)R + \overline{g}(\lambda)G + \overline{b}(\lambda)B \tag{1.1}$$

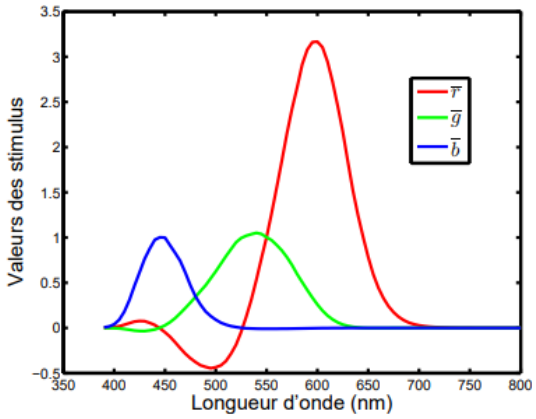


FIGURE 1.1 – Courbes $\overline{r}(\lambda)$,$\overline{g}(\lambda)$ et $\overline{b}(\lambda)$ de correspondance. [1]

De plus, il n'est pas pratique d'utiliser l'espace colorimétrique de la valeur Parfois, il est négatif, et CIE lui correspond, dans lequel chaque couleur doit avoir un coefficient positif. Les fonctions résultantes sont nommées $\overline{x}(\lambda)$,$\overline{y}(\lambda)$ et $\overline{z}(\lambda)$ et comme le montre la figure 1.2

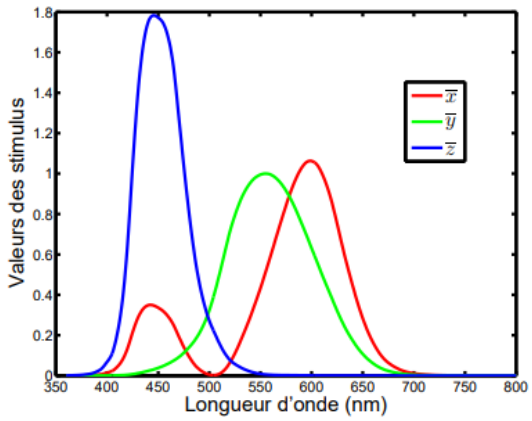


FIGURE 1.2 – Courbes $\overline{x}(\lambda)$,$\overline{y}(\lambda)$ et $\overline{z}(\lambda)$ de correspondance. [1]

Avec ce nouvel espace, toute couleur donnée peut être basée sur Les formules suivantes, X, Y et Z définissent alors un nouvel espace colorimétrique :

$$C_\lambda = \overline{x}(\lambda)X + \overline{y}(\lambda)Y + \overline{z}(\lambda)Z \tag{1.2}$$

1.3 Le système visuel humain

Les informations visuelles sont reçues par les yeux, puis transmises au cerveau après divers traitements préliminaires dans la rétine. Le niveau de traitement le plus élevé est effectué dans le cerveau Zone visuelle située à l'arrière des deux hémisphères cérébraux (voir Figure 1.3). selon En recherche physiologique[2], l'aire corticale dédiée au traitement de l'information visuelle

occupe une place importante. De plus, les connexions à d'autres régions du cerveau font du système visuel un ensemble très complexe. La figure 1.3 présente les principaux traitements traitement rétinien, puis après transmission des informations par le nerf optique, et finalement le traitement passé par la première zone du cortex visuel primaire.[3]

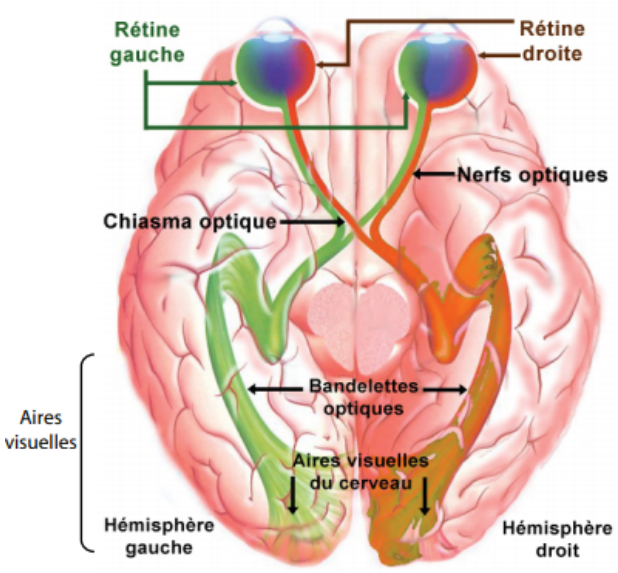


FIGURE 1.3 – Schéma global du système visuel humain.[3]

1.3.1 La vision

La rétine est tapissée d'un grand nombre de cellules nerveuses cônes et des BÂTONNETS (environ 100 millions dans la rétine humaine). Ces cellules photosensible convertit la tâche lumineuse capturée par l'objectif en une série d'impulsions codées remonte le nerf optique jusqu'à le "centre visuel" est situé dans la région occipitale du cerveau.

1.3.2 La rétine

L'œil est le capteur du système visuel. Il se compose d'un système d'optique adaptative Les éléments suivants (voir Figure 1.4) :

— La barrière protectrice transparente entre la cornée, l'environnement extérieur et l'œil lui-même. C'est l'un des premiers systèmes de focalisation.

— L'ouverture et lentille qui permettent respectivement de moduler la quantité de lumière incidente Réglez la distance focale du système optique.

— La rétine est tapissée de photorécepteurs, qui sont un endroit pour recevoir des informations lumineuses.[4]

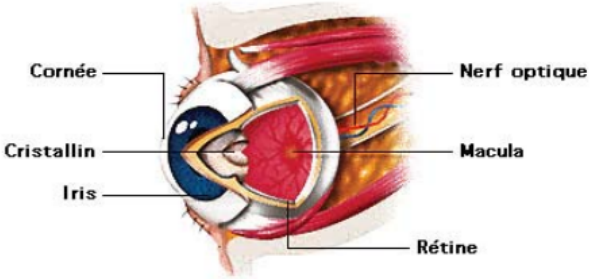


FIGURE 1.4 – schéma général de l'oeil.[4]

1.3.3 Les bâtonnets

Les bâtonnets représentent 95% des photorécepteurs rétiniens, soit environ 100 à 120 millions d'unités. Ils sont très sensibles aux capteurs de lumière et permettent de voir dans des conditions de faible luminosité, du crépuscule jusque dans l'obscurité (Vision noire). Ce type de vision n'a qu'une échelle de gris : ils ne peuvent pas percevoir La couleur, car ils sont également sensibles aux longueurs d'onde comprises entre 400 et 650 nm. Les bâtonnets sont également des cellules très sensibles aux mouvements [1].

1.3.4 Perception de la luminosité

Le système visuel humain n'est pas la capacité de percevoir toute la dynamique de la luminiance du monde réel en même temps. A chaque Instant, un photorecepteur présente une dynamique de 1 à 50 Environ, (soit environ 1,7 log unités). Cependant, le système visuel est capable de s'adapter à des ambiances correspondent à une plage de luminosité simultanée supérieure, typiquement de 1 à 6000 environ (soit 3,7 log unités) [5] grâce à l'adaptation de la réponse neuronale de la rétine, au travers des connexions entre les différentes couches et les différents types de cellules dans ces couches, et ce, de façon très rapide.

Les mécanismes ci-dessous contribuent également à augmenter la plage d'adaptation :

— la variation de l'ouverture de la pupille (entre 1,5 et 8 mm), le temps de réaction est de l'ordre de la seconde.

— la modification de la sensibilité des photorécepteurs, grâce à la possibilité de moduler la concentration en pigment : plus l'intensité lumineuse est importante, plus cette concentration diminue, et donc moins le photorécepteur est sensible, le temps d'adaptation est de l'ordre de la minute, et peut atteindre une heure pour une complète adaptation à l'obscurité ;

Grâce à ces mécanismes, tout se passe comme si les photorécepteurs étaient capables de translater leur dynamique vers la luminance moyenne locale qu'ils reçoivent t [6], et grâce à cela, le système visuel humain peut fonctionner entre 10^{-6} et 108 cd/m^2 (soit 14 log unités) (voir figure 1.5).

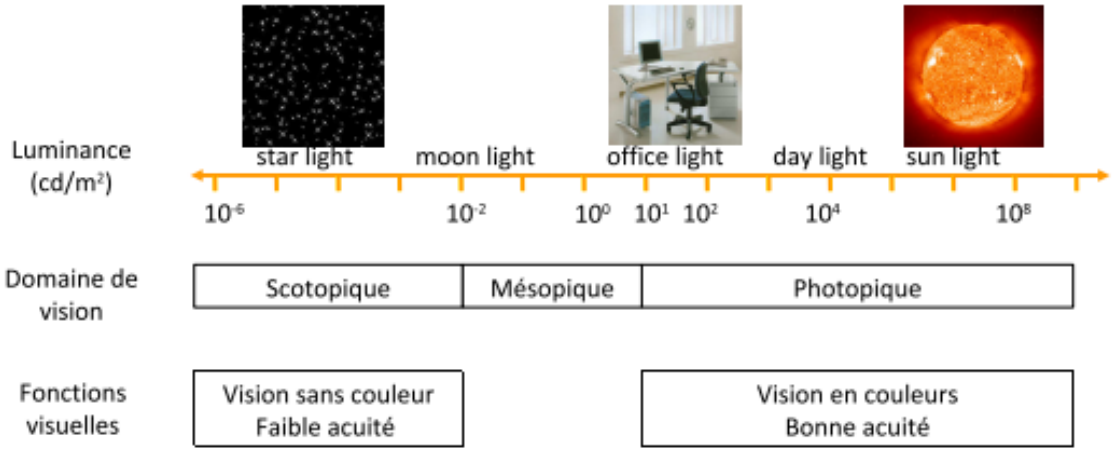


FIGURE 1.5 – Vision scotopique et vision mésopique (figure adaptée de [7])

Dès que le niveau lumineux dépasse 10 cd/m^2, on parle de «vision photopique» : les bâtonnets sont saturés, et seuls les cônes participent à la vision.

Pour des niveaux d'éclairement faibles (inférieurs à $10^{-2}cd/m^2$), les cônes ne sont pas assez sensibles, et la vision est assurée par les bâtonnets : on parle de «vision scotopique ».

La vision mésopique (crépusculaire), représente la vision dans des conditions d'éclairage moyen de $10^{-2}cd/m^2$ et $10cd/m^{-2}$. Dans cette plage de luminance, les bâtonnets et les cônes sont simultanément actifs[8].

En conséquence, la sensibilité spectrale, ou plus précisément la « fonction d'efficacité lumineuse spectrale » ne sera pas la même en fonction du niveau de luminosité (figure 1.6) :

D'autre part, la relation entre la luminance perçue L_v (luminance visuelle) et la luminance énergétique L (en cd/m²) n'est pas linéaire, on a plutôt la relation : $L_v \infty L^{1/3}$ [9].

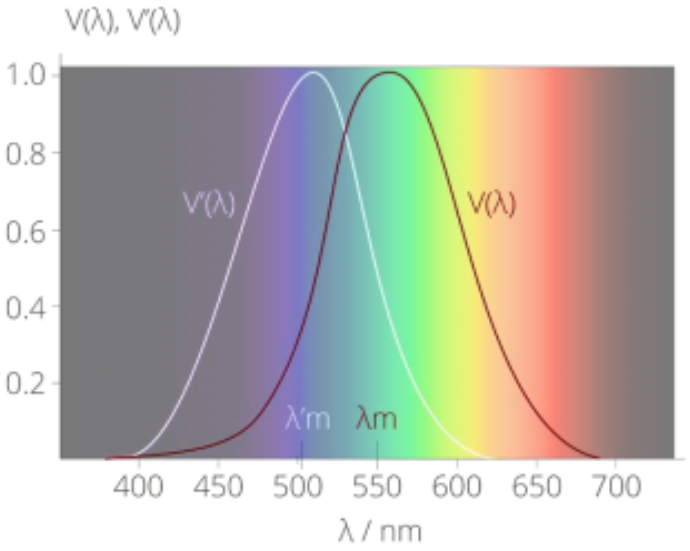


FIGURE 1.6 – Fonctions d'efficacité lumineuse relatives.[9]

1.4 Les images HDRs

Le dynamique d'une image est l'étendue de la gamme de couleurs ou de niveaux de gris que peuvent prendre les pixels. Ce dynamique dans les images HDRs est plus grande que celle qui peut être affichée par un écran standard d'ordinateur.

1.4.1 Principe

Les techniques HDR telles qu'elles sont perçues et utilisées aujourd'hui trouvent leur origine dans l'imagerie numérique et les images générées par ordinateur (notamment les travaux du professeur Paul Debevec à l'ICT Graphic Lab de Californie en 1997). S'ensuivent tout un tas de logiciels, de filtres, de techniques et autres artefacts informatiques permettant de littéralement créer des images à l'exposition optimale à partir de plusieurs photos.
Pour créer une photo HDR à grande dynamique lumineuse, il vous faudra en entrée plusieurs photographie de la même scène, exposées différemment (voir la figure 1.7). Le principe est finalement très simple. En effet, parfois nos photos sont sous-exposées, parfois elles sont un peu sur-exposées, parfois très sur-exposées, etc. : pourquoi ne pas prendre le meilleur de plusieurs photos exposées différemment pour en créer une à l'exposition « parfaite » ? C'est ça, le HDR.[10]

FIGURE 1.7 – Les images capturées avec un temps d'exposition croissant sont présentées de gauche à droite [11].

1.4.2 Comparaison avec les images LDRs

La luminosité en monde réel est comprise entre 10^{-6} et 10^{6} cd/m^2.
la figure 2.12 précise la luminosité HDR dans le monde réel.

FIGURE 1.8 – Luminosité HDR dans le monde réel.[12]

La figure 1.9 illustre la luminosité LDR dans un écran d'ordinateur.

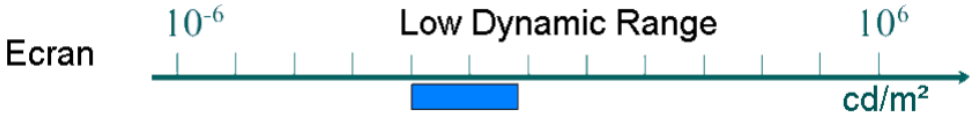


FIGURE 1.9 – Luminosité LDR dans un écran d'ordinateur.[12]

Ce que l'on fait habituellement est de transférer la luminosité en monde réel vers la luminosité sur un écran d'ordinateur (voir la figure 1.10) et c'est là qu'on fait la différence entre les images LDRs et les images HDRs. La table 1.1 montre une comparaison entre les images LDRs et les images HDRs.

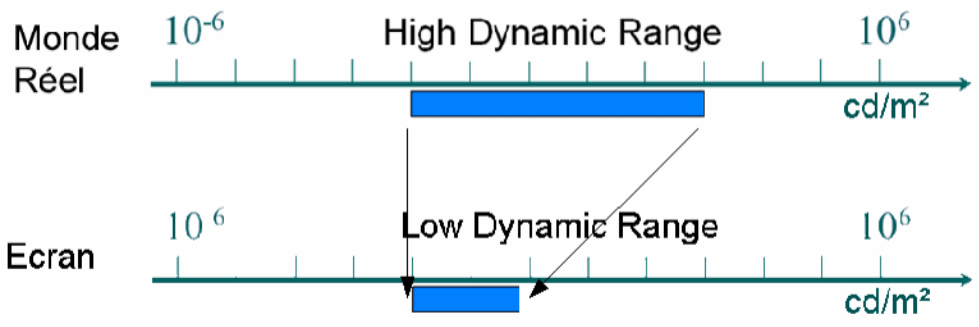


FIGURE 1.10 – Transfert monde réel vars les écran.[12]

Images HDR	Images LDR
Codées sur 16-32 bits par canal	Codées sur 8 bits par canal
Codage en valeurs flottantes	Codage en valeurs entières
Informations correspondantes aux valeurs réelles de luminance.	Informations correspondantes aux couleurs à afficher à l'écran ou sur le papier photo.

TABLE 1.1 – Comparaison des images HDRs avec les images LDRs.

1.4.3 Construction des images HDRs

Les images HDR peuvent être obtenues par deux différents procédés. Dans le premier, les images HDR sont générées de manière séparée ou isolée de système d'acquisition (le système d'acquisition dans ce cas est traditionnel ou de faible gamme de dynamique LDR : acronyme anglais de Low Dynamic Range) en utilisant l'une des techniques de la génération HDR (qu'on trouve dans l'état de l'art) implémentée en logiciel (la technique utilisée reproduit des images HDR à partir d'images LDR , c'est l'une des solutions les plus adaptées et les moins coûteuses, permettant d'augmenter la gamme dynamique des images acquises par les capteurs d'image conventionnels) . Le deuxième procédé consiste à obtenir les images HDR directement à partir du système d'acquisition. Ce dernier peut intégrer un capteur d'image permettant des acquisitions HDR natives (très contraignant : coût élevé, verrou technologique et peu fiable) ou garder l'utilisation d'un capteur LDR et générer des images HDR par la reconstruction en post traitement au sein du système d'acquisition par l'intégration d'une des techniques de la génération HDR à partir d'images LDR. Bien que plusieurs algorithmes de la génération HDR à partir d'images LDR sont facilement intégrables au sein d'un système d'acquisition, atteindre certaines performances (faible latence, cadence images élevée) s'avèrent difficiles car les algorithmes doivent

être adaptés à l'architecture matérielle du système et aux conditions de l'acquisition.[13]
Problèmes : On ne préserve pas vraiment les détails (compression) et ne peut pas avoir à la fois :

- Des objets très lumineux.
- Des objets très sombres.
- Voir les détails dans les deux cas.
- On ne peut pas éclaircir ou assombrir réellement une image.

Parmi les logiciels permettant d'afficher les images HDRs

- HDRShop : Windows, gratuit pour utilisation non comerciale.
- Luminance HDR : Unix, MacOS X, gratuit, Open Source.
- Desktop RADIANCE : Windows, gratuit, propriétaire.
- Photosphere : MacOS X

Formats des images HDRs :
Après la création d'image de grande gamme dynamique et afin de faciliter leurs manipulation (sauvegarde sous un support physique, transfert d'une machine à l'autre,..etc), trois différents formats sont présentés dans l'état de l'art (HDR, EXR et TIFF). Les formats les plus complets et flexibles sont le format EXR et le format TIFF. Ces deux formats permettent plusieurs encodages et plusieurs types de compression avec une possibilité de représenter des données HDR brutes sans compression ou avec compression. Les deux formats contient des champs de méta-données complets avec une possibilité d'insérer de nouveaux paramètres par l'utilisateur[13].
Le tableau 1.2 donne les différents formats HDR et leurs caractéristiques et points clés.

Format	Encodage	Compression	Méta-donnée	support
HDR	RGBE. XYZE.	run-length	calibration, espace couleur, +à définir par l'utilisateur	open source software (Radianc)
TIFF	IEEE RGB. LogLuv24. LogLuv32.	pas de compression. pas de compression. run-length.	calibration, espace couleur, +à définir par l'utilisateur	public domain library (libtiff).
EXR	Half RGB	wavelet, ZIP	calibration, espace couleur, fenêtrage, +à définir par l'utilisateur	open source library (OpenEXR)

TABLE 1.2 – Les formats HDR[14].

1.5 Tone mapping

Nos écrans n'étant pas prévu pour afficher des pixels autrement qu'en 24 bits (8 bits par couleurs), il est nécessaire d'appliquer des corrections à l'image HDR si on souhaite la visualiser. Le Tone mapping est un procédé qui consiste à passer d'une image HDR à une image à faible dynamique (LDR), en gardant tous les détails que l'on souhaite : il s'agit de compresser la gamme dynamique à quelque chose de plus facile à percevoir, en attribuant à certaines couleurs une autre couleur. Pour cela, il existe une multitude d'algorithmes différents.[15]

1.5.1 Principe

Les images HDRs, avec leur grande dynamique, ne peuvent pas être affichées directement sur un moniteur classique. En effet, un moniteur classique ne peut afficher que des luminances allant, par exemple, de 0 à 300 cd/m^2. Ceci ne convient pas à l'image HDR qui elle peut contenir des luminances allant de 0 à 10^6 cd/m^2. Ce constat montre qu'il est nécessaire de trouver une fonction qui permette de compresser la dynamique de l'image HDR afin de l'afficher sur un moniteur LDR.

L'opérateur de Tone Mapping a pour but de trouver une fonction pour convertir une image HDR en une image LDR. Cependant, trouver ce type d'opérateur n'est pas trivial car il doit minimiser l'erreur de perception entre l'image HDR et l'image LDR après l'application de l'opérateur de Tone Mapping.[16]

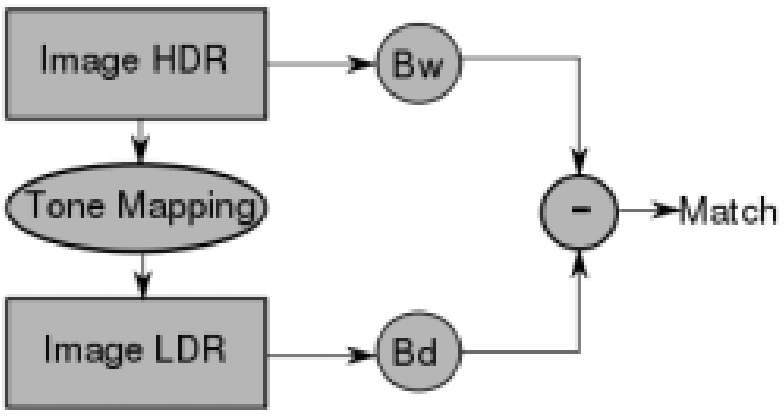


FIGURE 1.11 – B_w correspond à la luminosité perçue par l'œil directement. B_d correspond à la luminosité perçue par l'œil sur l'écran. L'opérateur de Tone Mapping cherche à avoir la même luminosité pour les deux images[16].

Dans leur fonctionnement, les opérateurs de Tone mapping jouent le rôle de l'adaptation visuelle humaine. En effet, leur but est de transformer les valeurs physiques de l'éclairement et de nous donner une perception correcte de celles-ci à travers le système d'affichage (écran).

Cependant, ce processus n'est pas le seul qui intervient dans la formation de notre perception. La plupart des opérateurs de Tone mapping se compose de deux modèles (figure 1.12) :

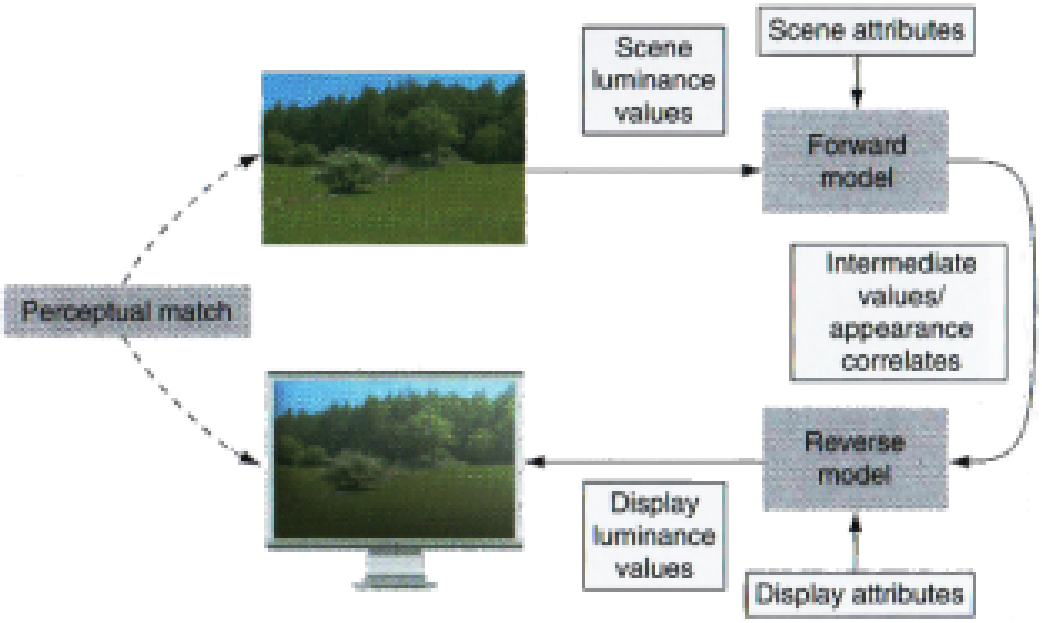


FIGURE 1.12 – – En haut, se trouve l'image HDR, celle-ci contient les valeurs de luminance de la scène. En bas, se trouve l'image LDR. L'opérateur de Tone Mapping qui va compresser l'image HDR doit prendre en compte les spécificités de la scène (Forward model) et aussi les conditions de visualisation de l'image (Reverse model)[16].

Le modèle direct (Forward model) effectue une simple compression sous la forme d'une sigmoïde, il s'inspire du modèle proposé par Naka-Rushton. Soit L_a la luminance d'adaptation de l'image HDR, L_w la luminance de l'image HDR et n une constante :

$$V(x,y) = \frac{L_w^n(x,y)}{L_w^n(x,y) + L_a^n(x,y)} \tag{1.3}$$

Le modèle inverse (Reverse model) effectue une mise à l'échelle du résultat du Forward model (V(x,y)) pour qu'il puisse correspondre à la luminance du moniteur. Soit $L_{d,mean}$ la médiane de la luminance du moniteur, m une constante du moniteur et L_d la luminance affichable :

$$L_d(x,y) = \left(\frac{V(x,y)L_{d,mean}^m(x,y)}{1 - V(x,y)}\right)^{1/m} \tag{1.4}$$

1.5.2 Types

Pour calculer la luminance d'adaptation de l'image HDR, les opérateurs de Tone Mapping se décomposent en plusieurs groupes :

— **Opérateur global** ce type d'opérateur calcule une seule luminance d'adaptation pour toute l'image.

— **Opérateur local** ce type d'opérateur calcule la luminance d'adaptation pour chaque pixel de l'image en fonction de son voisinage.

D'autre part, exception faite des opérateurs de Tone mapping travaillant dans l'espace colorimétrique HSV(Human visual system), il est généralement accepté qu'un opérateur de Tone mapping ne travaille que sur la luminance. Cependant, pour revenir à l'espace colorimétrique RGB, il faut effectuer la transformation suivante :

$$\begin{pmatrix} R_d \\ G_d \\ B_d \end{pmatrix} = \begin{pmatrix} L_d(\frac{R_w}{L_w})^s \\ L_d(\frac{G_w}{L_w})^s \\ L_d(\frac{B_w}{L_w})^s \end{pmatrix} \tag{1.5}$$

avec L_d la luminance résultante de l'opération de Tone mapping, (L_w,R_w ,G_w ,B_w) la luminance et les valeurs colorimétriques de l'image HDR et $s \in [0, 1]$ le paramètre de saturation [16].

1.6 État de l'art sur le rendu HDR

Grande gamme dynamique HDR est une technique introduite afin de remédier à cette limitation. Il existe deux grandes classes de génération d'images à grande gamme dynamique. La première classe cherche à introduire des améliorations au niveau du capteur d'image afin d'obtenir un capteur d'image nativement de grande gamme dynamique. Dans ce cas, l'image HDR est générée directement par le capteur [17]. Cependant, le coût du système d'acquisition HDR est très important puisque il nécessite la conception et la fabrication d'un capteur spécifique. La seconde classe maintient l'utilisation d'un ou plusieurs capteurs à faible gamme dynamique. Dans le cas d'utilisation de plusieurs capteurs, les images à différents temps d'expositions sont capturées simultanément à l'aide de différents capteurs visualisant la même scène [18]. Chacun des capteurs est configuré avec un temps d'intégration différent.

1.6.1 Création de contenu HDR à partir de multiples expositions LDR

C'est une solution bas coût et flexible permettant la création d'image HDR à partir d'une liste d'images LDR capturées en utilisant différent temps d'exposition comme illustré par la figure 1.13. La dynamique désirée peut être paramétrée ou ajustée selon le besoin en agissant sur le nombre d'images LDR à utiliser pour créer une seule image HDR. Pour acquérir la gamme dynamique complète d'une scène, il est possible de prendre une liste idéale d'images LDR allant d'une image sous exposée (complètement noir) à une image surexposée (saturée blanche) avec un pas d'exposition adapté. Cependant, l'utilisation d'un grand nombre d'images LDR peut compromettre les performances du système final (fréquence d'images, résolution et complexité

de l'implantation logiciel/matérielle). En principe, il existe deux méthodes qui permettent la génération HDR en exploitant des images issues d'un ou de plusieurs capteurs d'image LDR. La première méthode est réalisée dans le domaine de l'image, elle génère des image pseudo-HDR (HDR-like en anglais)[19]. La deuxième méthode consiste à fusionner les différentes expositions LDR dans le domaine de la radiance.
Dans cette rubrique, on ne détaille que la génération d'image HDR dans le domaine de la radiance, car ce mode offre la possibilité de recouvrir la gamme dynamique réelle de la scène capturée, ce qui ne peut être réalisé en utilisant le mode de génération dans le domaine de l'image qui génère des images contenant plus de détails mais de faible gamme dynamique.

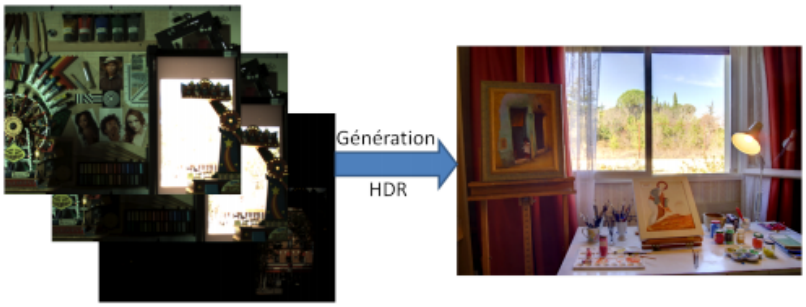


FIGURE 1.13 – Création d'image HDR à partir de multiples expositions LDR.[13]

1.6.1.1 La génération HDR dans le domaine de la radiance

La création d'images HDR par la fusion de multiples expositions LDR dans le domaine de la radiance est basée sur la récupération de la carte de radiance. Cette dernière est construite en utilisant la courbe inverse de la réponse de système d'acquisition. La courbe inverse sert à convertir les pixels des images LDR en valeurs de radiance en les préparant à l'étape de fusion de données en utilisant un algorithme de création HDR.[13]

1.6.1.2 Acquisition de multiples expositions

L'acquisition des différentes expositions (images LDRs à différents temps d'expositions) peut se faire de deux manières différents. La première utilise plusieurs capteurs d'images LDR visualisant une même scène soit avec des angles de vue différentes soit en utilisant des diviseurs de faisceau (miroirs semi-réfléchissants) comme le montre la figure 1.14. La seconde méthode utilise un seul capteur d'image LDR, les images LDR à différents temps d'exposition sont capturées d'une manière séquentielle en faisant varier le temps d'intégration de capteur à chaque acquisition.[13]

FIGURE 1.14 – – La visualisation d'une seule scène avec plusieurs capteurs d'images via un diviseur de faisceau.[20]

Méthode de Mann :

C'est la toute première méthode proposée en 1995 par Mann et al. Elle permet de fusionner les intensités lumineuses réelles de plusieurs expositions en une seule image HDR où la valeur de pixel HDR est flottante. La courbe de réponse est reconstruite automatiquement à partir d'une liste d'images LDR à différentes expositions. Ensuite la courbe estimée est utilisée en combinaison avec une fonction de pondération pour fusionner les données. La fonction de pondération utilisée est la dérivée de la fonction de réponse du système $f^{'-1}$.[21]

Méthode de Debevec et Malik :

ont proposé un algorithme qui relie la valeur du pixel Z et le rapport de l'irradiance E et du temps d'exposition Δt par la fonction de réponse f () donnée par l'équation 1.6 :

$$z_{ij}^{p} = f(E_{ij}\Delta t_p). \tag{1.6}$$

La méthode de Debevec et Malik est basée sur le fait que la valeur de l'exposition reste la même si on divise la valeur de l'irradiance E et on double le temps d'exposition Δt. La courbe de réponse est déduite par la résolution d'un ensemble d'équations linéaires par la décomposition en valeurs singulières (SVD) tout en supposant que la courbe de réponse est continue. Ils utilisent une fonction de pondération triangulaire simple de type "chapeau", pour fusionner les valeurs d'irradiance récupérées à partir des différentes expositions disponibles. Leur fonction de pondération est basée sur l'hypothèse que les pixels de milieu de la plage de dynamique du capteur (proches de 128 dans le cas d'un capteur à sortie codée en 8 bits) sont les plus fiables et les mieux exposés. [22]

1.7 Conclusion

Dans ce chapitre visait à élucider ce qu'est les images HDRs aussi qu'une vision générale sur le système visuel humain, souligner sur principe de tone mapping ,finalement nous avons présenté l'état de l'art sur le rendu HDR. Le prochain chapitre, traite les détails de Deep Learning.

Chapitre 2

Apprentissage profond (Deep Learning)

2.1 Introduction

L'apprentissage profond (deep learning) est une forme d'intelligence artificielle, dérivée de l'apprentissage automatique(Machine Learning), qui a été développée pour créer des algorithmes capables d'apprendre et de s'améliorer de manière autonome. Avec l'émergence de l'intelligence artificielle, l'essor des objets interconnectés et l'amélioration des capacités technologiques, un grand nombre d'études ont tenté de simuler le comportement humain et de reproduire ses capacités cognitives. Dans ce cas, le deep learning, littéralement "deep learning", est actuellement l'un des domaines les plus étudiés.

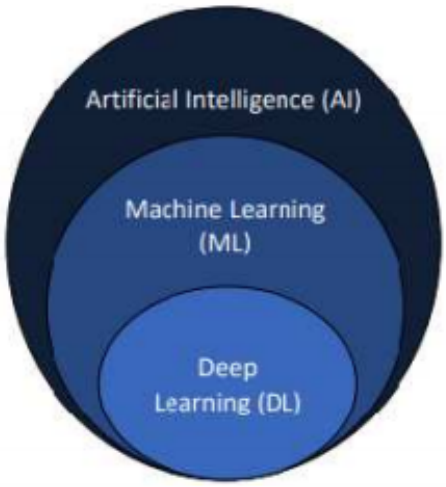


FIGURE 2.1 – Relation entre AI, ML et DL[23].

Au début de ce chapitre, nous posons des questions, qu'est-ce que le deep learning ? pour quelles applications on a appliqué ? qu'est-ce qu'une architectures de réseaux de neurones profonds ? et surtout comment cela représenter pour la reconstruction HDR ?

2.2 Définition de l'apprentissage profond (deep learning)

Le deep learning est un sous-domaine de l'intelligence artificielle. Le terme désigne toutes les techniques d'apprentissage automatique, c'est-à-dire une forme d'apprentissage basée sur des méthodes mathématiques utilisées pour modéliser les données. Pour mieux comprendre ces technologies, il faut remonter aux origines de l'intelligence artificielle en 1950.

Ce type de réflexion donnera naissance à l'apprentissage automatique, c'est-à-dire que les machines communiquent et fonctionnent sur la base d'informations stockées.

L'apprentissage profond est un système avancé basé sur un énorme réseau de neurones artificiels sont inspirés par les neurones du cerveau humain ,ils sont constitués de plusieurs neurones artificiels chaque neurones connecter par des autre neurones. Plus le nombre de neurones est élevé, plus le réseau est profond est-ce ça deep learning .Au parti du cerveau humain, chaque neurone reçoit environ 100 000 signaux électriques des autres neurones. Chaque neurone actif peut exciter ou inhiber les neurones qui lui sont connectés. Au parti d'un réseau artificiel, le principe est similaire les signaux transmet entre les neurones. Toutefois, au lieu d'un signal électrique, le réseau de neurones assigne s'appeler poids, à différents neurones par des couche. La dernière couche de neurones répond à ces signaux.

Ces neurones sont connectés les uns aux autres pour traiter et stocker des informations, comparer tout problème ou situation avec des situations similaires dans le passé, analyser des solutions et résoudre les problèmes de la meilleure façon.

2.3 Pour quoi le deep learning ?

Les modèles d'apprentissage en profondeur ont tendance peuvent souvent gérer de grandes quantités de données(2.2), tandis que les modèles d'apprentissage automatique plus traditionnels cessent de s'améliorer après avoir atteint un point de saturation. Au fil des ans, avec l'avènement du big data et des composants informatiques de plus en plus puissants, de puissants algorithmes d'apprentissage en profondeur à forte intensité de données ont remplacé la plupart des autres méthodes.

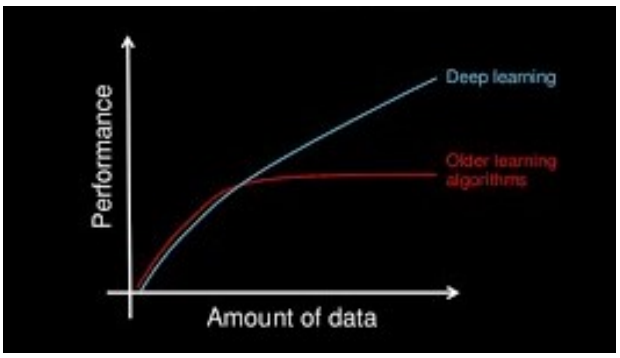


FIGURE 2.2 – Deep learning évoluent avec la quantité de données par pour ancien algorithme d'apprentissage.[24]

2.2 étant donné que nous sommes à l'ère du big data, nous voulons utiliser pleinement les données, donc la capacité à améliorer les performances lorsqu'il y'a une grande quantité de données (le passage à l'échelle) est un vrai rêve.

2.4 Domaines d'application de l'apprentissage profond

Il y'a quelques années, nous n'aurions jamais imaginé des applications d'apprentissage profond pour nous apporter des voitures autonomes et des assistants virtuels. Mais aujourd'hui, ces créations font partie de notre quotidien. Deep Learning continue de nous fasciner avec ses possibilités infinies. Pensez à un monde sans accidents de la route ni cas de rage routière, chaque chirurgie est réussie sans causer la perte de vies humaines en raison d'erreurs chirurgicales. Pensez à un monde où aucun enfant n'est défavorisé et où même ceux qui ont des limitations mentales ou physiques peuvent jouir de la même qualité de vie que le reste de l'humanité. L'apprentissage profond fait globalement en explorant et en résolvant les problèmes humains dans tous les domaines comme suit :

2.4.1 Voitures autonomes (Self-Driving Cars)

Un cycle régulier de test et de mise en œuvre typique des algorithmes d'apprentissage profond garantit une conduite sécuritaire avec une exposition de plus en plus grande à des millions de scénarios. Les données des caméras, des capteurs (voir 2.3) et de la géocartographie aident à créer des modèles succincts et sophistiqués pour naviguer dans la circulation, identifier les sentiers, la signalisation, les voies réservées aux piétons et les éléments en temps réel comme le volume de la circulation et les blocages routiers.

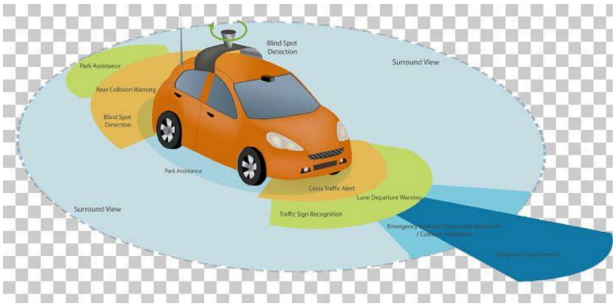


FIGURE 2.3 – Voitures autonomes.[25]

2.4.2 Virtuels assistants

L'application la plus populaire de deep learning est les assistants virtuels consiste à Google Assistant. Chaque interaction avec ces assistants leur donne l'occasion d'en apprendre davantage sur votre voix et votre accent, vous offrant ainsi une expérience secondaire d'interaction humaine. Les assistants virtuels utilisent l'apprentissage profond pour en savoir plus sur leurs sujets allant de vos préférences un repas manger à vos endroits les plus visités ou vos chansons préférées.

2.4.3 Reconnaissance visuelle

Dans deep learning les images peuvent être triées en fonction des lieux détectés dans les photographies, les visages, une combinaison de personnes, ou selon les événements, les dates, etc et recherche d'une photo particulière à partir d'une bibliothèque photo.

2.4.4 Soins de santé

Aide au diagnostic précoce, précis et rapide des maladies mortelles, la compréhension de la génétique pour prédire le risque futur de maladies et d'épisodes de santé négatifs sont quelques-uns des projets d'apprentissage profond qui prennent de la vitesse dans le domaine de la santé comme :

— « Analyse d'images de pathologie numérique avec apprentissage profond » L'avènement de l'imagerie sur les lames virtuelles en pathologie numérique (DP : digital pathology) a permis de faire progresser le diagnostic assisté par ordinateur (CAD : computer-aided diagnostics) des tissus, via l'analyse d'images numériques.[26]

— « Apprentissage profond pour la segmentation de l'IRM tissus cérébraux » Diverses approches d'analyse cérébrale reposent sur une segmentation précise des données anatomiques des régions. Il permet de détecter la pathologie, de préparer la planification médicale et les chirurgies.[27]

— « Segmentation sémantique d'images médicales 3D par deep learning » La segmentation sémantique d'images médicales 3D consiste à assigner à chaque voxel d'un volume

d'entrée (e.g. CT-scan, IRM, MRI) une étiquette sémantique correspondant par exemple à l'appartenance à un organe donnée (voir 2.4). C'est une tâche pour laquelle la mise en place d'outils d'intelligence artificielle (IA) revêt des enjeux cruciaux pour l'aide au diagnostic.[28]

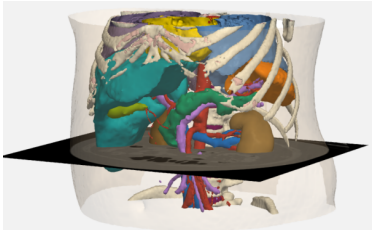


FIGURE 2.4 – Segmentation d'images médicales 3D.[28]

2.4.5 Colorisation des images en noir et blanc

La colorisation d'image est le processus de prise d'images en niveaux de gris (en entrée) et de production d'images en couleurs (en sortie) qui représente les couleurs et les tons sémantiques de l'entrée.La figure 2.5 montre ça.

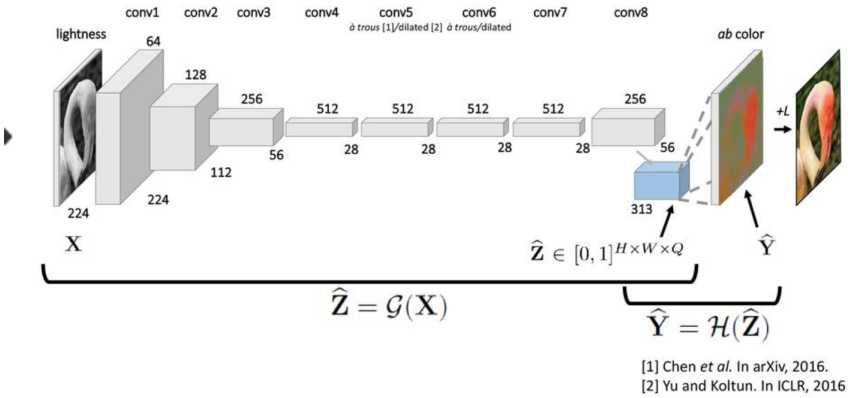


FIGURE 2.5 – Architecture CNN pour la colorisation.[29]

2.4.6 Agriculture

L'intelligence artificielle a le potentiel de révolutionner le domaine de l'agriculture. L'apprentissage en profondeur a permis aux agriculteurs de déployer des équipements capables d'identifier et de distinguer les cultures et les mauvaises herbes. Cette fonction permet à la machine de désherbage d'appliquer spécifiquement des herbicides sur les mauvaises herbes en plus d'autres plantes. Les machines agricoles qui utilisent l'apprentissage en profondeur peuvent même optimiser le rendement de chaque plante grâce à l'utilisation ciblée d'herbicides, d'engrais, de fongicides, de pesticides et de produits biologiques. L'apprentissage en profondeur peut réduire

l'utilisation d'herbicides et améliorer la production agricole, mais il peut également être étendu à d'autres activités agricoles, telles que la fertilisation, l'irrigation et la récolte.

2.5 Architectures de réseaux de neurones profonds

Il existe un grand nombre de variantes d'architectures profondes. La plupart d'entre elles sont dérivées de certaines des architectures parentales originale. Il y a deux types apprentissage que classifiées architectures :

- **Supervisé**
  - Les réseaux de neurones convolutifs utiliser pour computer vision
  - Les réseaux de neurones récurrents utiliser pour séries temporelles
- **Non-supervisé**
  - Cartes auto-adaptives utiliser pour détection de features
  - Machines de Boltzmann profondes utiliser pour système de recommandation
  - Auto-encodeurs utiliser pour système de recommandation

2.5.1 Les réseaux de neurones convolutifs

Depuis pas mal d'années les réseaux de neurones convolution prennent vraiment le dessus par rapport aux réseaux de neurones artificiels , CNN est la plus en plus utilisé et populaire ça évolue énormément. Par exemple, on peut penser aux voitures autonomes cette voitures utilisent énormément les réseaux de neurones à convolution par ce qu'elles ont beaucoup d'images à reconnaître. Cette technologie développée par Yann Lecun le père des réseaux de neurones à convolution, il va servir pour classifier des images on va mettre une image en entrée, et elle va passer dans le réseau qui est déjà entraînée et on ressort la catégorie de l'image et on la divise en deux blocs (comme illustre la figure 2.6) :

- **Le premier bloc : feature extraction,** il applique des opérations de filtrage par convolution, c'est à dire il filtre l'image avec convolution et renvoie des images appelées " feature maps " ensuite normalisées ,poooling ,flattening, finalement les valeurs des dernières feature maps sont concaténées dans un vecteur ,ce vecteur permet de définir la sortie.
- **Le second bloc : classification,** les valeurs du vecteur en entrée sont transformées (avec plusieurs combinaisons linéaires et fonctions d'activation) pour renvoyer un nouveau vecteur en sortie. Ce dernier vecteur contient autant des éléments représentent les probabilités que l'image appartienne à la classe . Chaque élément est donc compris entre 0 et 1, et la somme de tous vaut 1. Ces probabilité sont calculées par la dernière couche de ce bloc (et donc du réseau), qui utilise une fonction logistique (classification binaire) ou une fonction softmax (classification multi-classe) comme fonction d'activation .

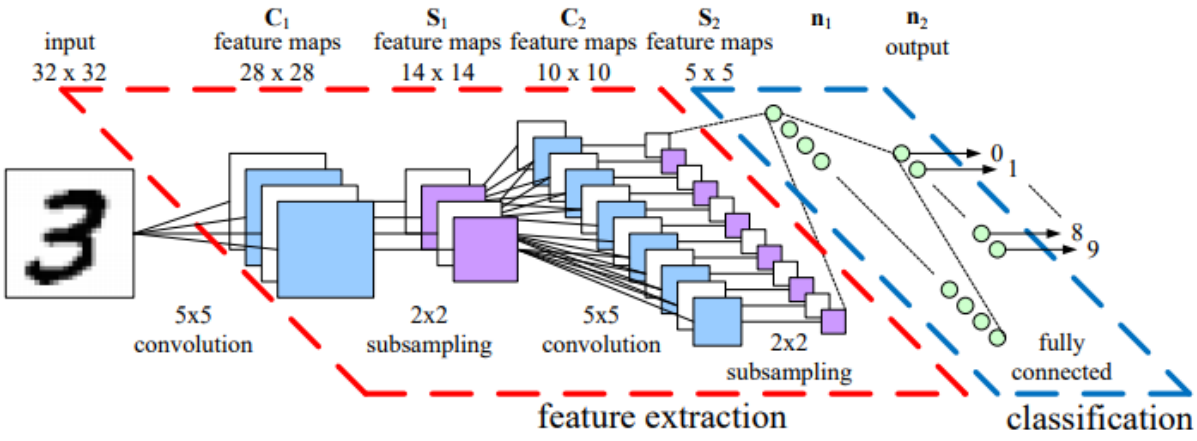


FIGURE 2.6 – Un exemple d'architecture CNN pour une tâche de reconnaissance de chiffres manuscrits.[30]

L'architecture réseau CNN est représentée sur la figure 2.6. Le traitement commence par des couches de feature extraction et se finit par des couches de classification entièrement connectées.

2.5.2 Réseau de neurones récurrents

Un réseau neuronal récurrent RNN est un réseau de neurones dans lequel l'information peut se propager dans deux directions, y compris de la couche profonde à la première couche (voir 2.7). En cela, ils sont plus proches du vrai fonctionnement du système nerveux, qui n'est pas à sens unique. Ces réseaux possèdent des connexions récurrentes au sens où elles conservent des informations en mémoire : ils peuvent prendre en compte à un instant **t** un certain nombre d'états passés. C'est l'un des algorithmes les plus avancés dans le monde de l'apprentissage supervisé . L'architectures de ce réseau est représentée par :

— un seul neurone en entrée et on a plusieurs sorties

— plusieurs entrées et par contre on a qu'une seule sortie

— plusieurs entrées et on a plusieurs sorties.

FIGURE 2.7 – Schématique Réseau de neurones récurrents [31]

2.5.3 Cartes auto-adaptatives (SOMs)

La Carte auto-organisatrice ou carte de Kohonen est un algorithme de classification développé par Teuvo Kohonen learning dès 1982. L'objectif principal d'une carte auto adaptative est de réduire la dimension, c'est qu'ils ont un jeu de données multidimensionnelles avec énormément de dimensions énormément de colonnes de variables. Alors évidemment ils ont aussi beaucoup de lignes mais vraiment le but est de réduire la dimension du jeu de données, c'est à dire de réduire le nombre de colonnes, à la fin on finit avec une carte en deux dimensions qui représente les données (voir2.8). Donc l'objectif principal est vraiment de réduire le nombre de colonnes en perdant bien entendu le moins d'informations possibles.

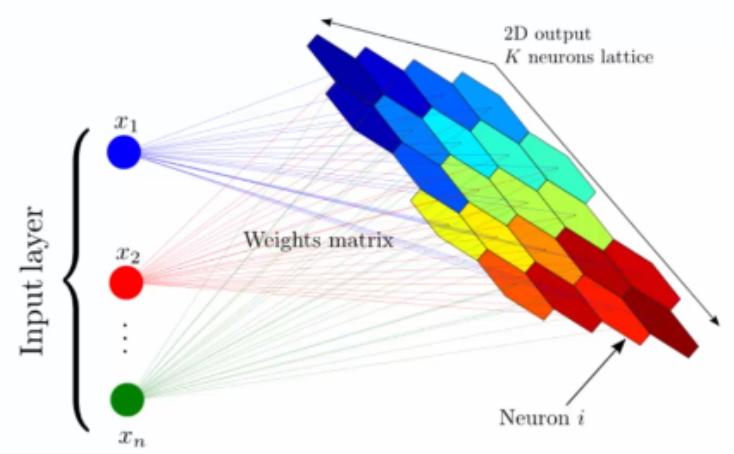


FIGURE 2.8 – Schématique réseau de neurones Cartes auto-adaptatives.[32]

L'algorithme d'apprentissage non supervisé et la méthode de quantification vectorielle , regroupement des informations en classes tout en respectant la topologie de l'espace des observations définit à priori d'une notion de voisinage entre les classes des observations voisines dans l'espace des données appartiennent après classement à la même classe ou à des classes voisines et les compressions de données multidimensionnelles tout en préservant leurs caractéristiques.

2.5.4 Machines de Boltzmann

Le fait que le fonctionnement des machines à Boltzmann est vraiment différent, on a deux couleurs (voir 2.9) n'y a pas de couche de sortie, on a la couche d'entrée qui est en bas et on a la couche de caché et tous ont connecté à tous par ce que c'est l'apprentissage non supervisé, l'information va dans tous les sens n'ont pas de direction, l'information elle part nodes Visible jusqu'au nodes caché , elle revient dans les nodes visibles et ainsi de suite. Il est possible d'empiler plusieurs couches de machines de Boltzmann restreintes pour créer des réseaux profonds qui sont plus performants. L'algorithme va générer des états dans ce système, dans la machine de Boltzmann asynchrone, les neurones modifient leur état un à un [33] .

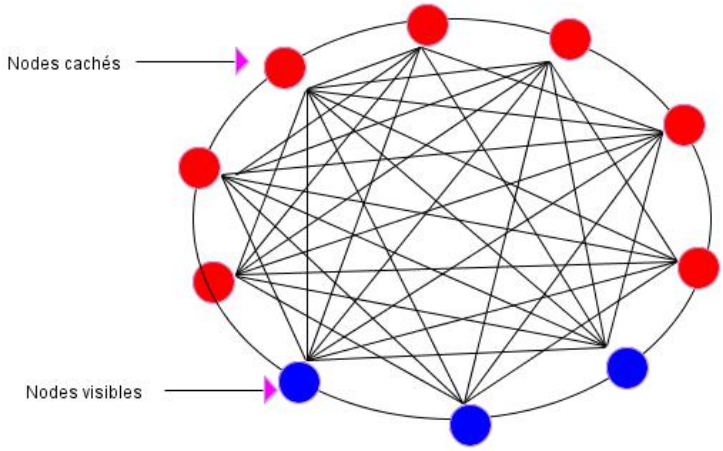


FIGURE 2.9 – Schématique réseau de neurones Machines de Boltzmann.

2.5.5 Auto-encodeurs

Les auto-encodeurs sont des algorithmes d'apprentissage non supervisé à base de réseaux de neurones artificiels, qui permettent de construire une nouvelle représentation d'un jeu de données. Généralement, celle-ci est plus compacte, et présente moins de descripteurs, ce qui permet de réduire la dimension du jeu de données. L'architecture d'un auto-encodeur est constitué de deux parties : l'encodeur et le décodeur(voir 2.10).
L'encodeur est constitué par un ensemble de couches de neurones, qui traitent les données afin de construire de nouvelles représentations dites "encodées". À leur tour, les couches de neurones du décodeur, reçoivent ces représentations et les traitent afin d'essayer de reconstruire les données de départ.

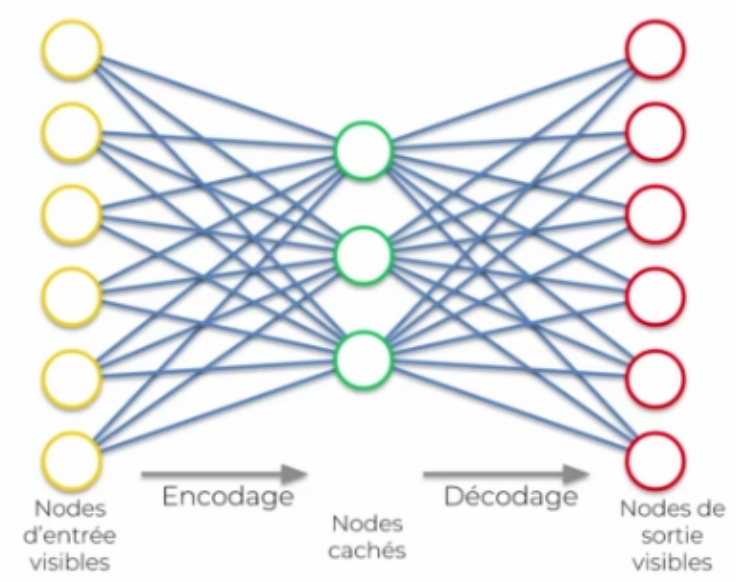


FIGURE 2.10 – Schématique réseau de neurones Auto-encodeurs. [34]

2.6 Deep learning pour la reconstruction des images HDRs

L'apprentissage profond est au cœur des progrès récents de la perception visuelle, permettant de transformer de grandes quantités de données d'entraînement sous une forme compacte, pouvant être ensuite utilisés en ligne pour fournir des connaissances à priori dans un nouvel environnement.
Représenter la forme 2D dans des cadres d'apprentissage profond d'une manière précise, efficace et compacte reste un défi ouvert, l'apprentissage profond a trouvé des solutions pour les problèmes de la vision par ordinateur(computer vision) comme :

— L'extraction des profondeurs.

— Classification/recognition des objets.

— Détection des objets.

— Reconstruction d'image.

2.6.1 Travaux connexes

Il existe plusieurs travaux pour reconstruire des images HDRs à partir des images LDRs.

Gabriel Eilertsen et al [35] a été proposé une nouvelle méthode de reconstruction des images HDRs à partir des images d'entrées (LDRs),en estimant les informations manquantes dans les parties claires de l'image,telles que les hautes lumières, perdues en raison de la saturation du capteur de la caméra. Un réseau de neurones entièrement convolutif (CNN) sous la forme d'un auto-codeur à gamme dynamique hybride. Auto-codeur hybride de plage dynamique. Comme pour les architectures d'auto-encodeurs profonds, l'image d'entrée LDR est transformée par un réseau d'encodeurs pour produire une représentation compacte du contexte spatial de l'image.

Avantage :

— L'utilisation de CNN est facile à comprendre et donne des résultats de bonne qualité.

Limite :

— Quand on augmente les expositions, on observe quelques petites régions noires.

FIGURE 2.11 – Résultat de la reconstruction des images HDR.[35]

Nima Khademi et al [36] propose une approche basée sur l'apprentissage pour produire une image HDR de haute qualité à partir de trois images LDR exposées différemment d'une scène dynamique. Le système d'apprentissage génère une image HDR, qui est alignée sur l'image de référence, mais qui contient des informations provenant des deux autres images.
Par exemple, dans la figure 2.12 les détails de la table sont saturés dans l'image de référence, mais sont visibles dans l'image avec l'exposition(exposure) la plus courte.
La méthode consiste à aligner d'abord les images LDR d'entrée en utilisant la méthode de flux optique de[37] sur l'image de référence (exposition moyenne) et utilise ensuite les images LDRs alignées comme entrée de ce système de fusion HDR basé sur le deep learning pour produire une image HDR de haute qualité qui est ensuite mise en correspondance avec le tonemap pour produire l'image finale.

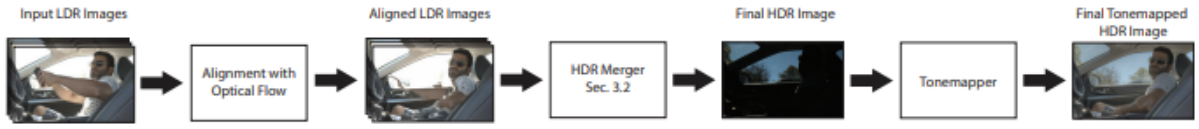


FIGURE 2.12 – Illustration de la méthode Deep High Dynamic Range Imaging of Dynamic Scenes.[36]

Avantage :

— Méthode simple.

— Ils présentent et comparent trois architectures de système différentes pour modéliser le processus de fusion HDR.

Limite :

— Dans certains cas, en raison du mouvement de la caméra, les images de faible et de forte exposition ne sont pas disponibles.

Kenta Moriwaki et al [38] ont proposé une méthode qui permet la reconstruction HDR à partir d'une seule image LDR, les méthodes conventionnelles inverse-tone-mapping (iTM) se concentrent sur les zones saturées et extrapolent l'intensité lumineuse des régions environnantes sur la base d'une heuristique locale. Comme ces heuristiques ne sont pas universelles, elles échouent souvent à récupérer les zones sur - et sous- exposées ou introduisent des artefacts non naturels.
Plus récemment, des méthodes basées sur l'apprentissage profond ont été appliquées à la reconstruction HDR basée sur une seule image.

Avantage :

— La méthode produit des résultats de très bonne qualité à celle des méthodes existantes et permet de récupérer les régions sur- et sous-exposées.

Limite :

— lorsque les zones saturées sont trop grandes, le générateur peine à repeindre les régions.

— Approfondissement du site le réseau, la collecte de plus grands ensembles de données.

FIGURE 2.13 – Résultat de la reconstruction HDR.[38]

Zeeshan Khan et al[39]La génération d'images à haute gamme dynamique (HDR) à partir d'une image à faible gamme dynamique (LDR) à exposition unique a été étudiée. A partir LDR à exposition unique rendue possible grâce aux progrès récents de l'apprentissage profond (deep learning).
Différents réseaux neuronaux convolutifs (CNN) à la source ont été proposés pour l'apprentissage des représentations LDR vers HDR.
Pour mieux utiliser la puissance des CNN, en exploitant l'idée de rétroaction(feedback), dans laquelle les caractéristiques initiales de bas niveau sont guidées par les caractéristiques de haut niveau en utilisant un état caché d'un réseau neuronal récurrent. Contrairement à un seul passage vers l'avant dans un conventionnel, la reconstruction de LDR vers le HDR dans un réseau de rétroaction est apprise sur plusieurs itérations. Les auteurs ont proposé une nouvelle méthode feedback network (FHDR) pour la reconstruction des images LDRs. La figure 2.15 illustre l'architecture de cette méthode.

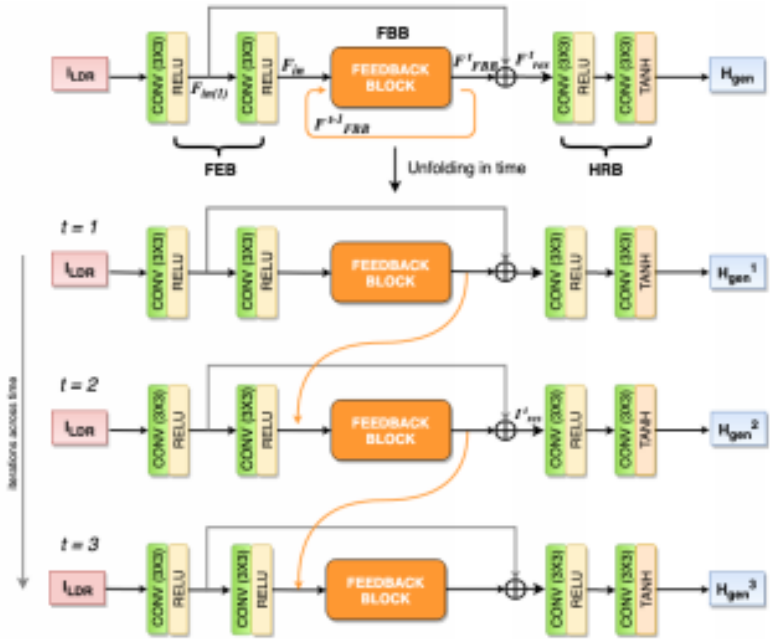


FIGURE 2.14 – FHDR Architecture.[39]

FIGURE 2.15 – Résultat de FHDR.[39]

Avantage :

— Le réseau FHDR utilise les connexions de rétroaction et donc il améliore la netteté globale de l'image et il réalise une reconstruction encore meilleure.

Limite :

— Le réseau n'accepte pas les images de basse résolution(low resolution images).

Yu-Lun Liu et al [40]La récupération d'une image à haute gamme dynamique (HDR) à partir d'une seule image d'entrée à faible gamme dynamique (LDR) est un défi en raison des détails manquants dans les régions sous-exposées ou surexposées causés par la quantification et la saturation des capteurs de la caméra, les auteurs propose d'utiliser le deep learning pour faire ça. Ils ont modéliser le pipeline de formation d'images HDR-to-LDR comme étant l'élément suivant :
(1) dynamic range clipping.
(2) non-linear mapping from a camera response function.
(3) quantization.
La figure suivante 2.16 illustre le résultat obtenu :

FIGURE 2.16 – Résultat de Single-Image HDR Reconstruction by Learning to Reverse the Camera Pipeline.[40]

Avantage :

— Ils ont utilisé quatre architectures de réseaux CNN de quantization-Net,Linearization-Net, Hallucination-Net,Refinement-Net.

Limite :

— Ne sont pas conçues spécifiquement pour le traitement des images bruyantes en basse lumière (handling noisy low-light images).

2.6.2 Comparaison entre les travaux

Travaille	Méthode	Avantage	Limite
[35]	Entraîner le CNN et rassemble un grand ensemble de données d'images HDR.	L'utilisation de CNN est facile à comprendre et donne des résultats de bonne qualité.	Quand on augmente les expositions, on observe quelques petites régions noires.
[36]	Alignment with optical flow. Alignment LDR image. HDR Merger. finale HDR image. Tonemapper après finale hdr	Méthode simple. Ils présentent et comparent trois architectures de système différentes pour modéliser le processusde fusion HDR.	Dans certains cas, en raison du mouvement de la caméra, les images de faible et de forte exposition ne sont pas disponibles.
[38]	The hybrid loss, HDR-reconstruction loss, Adversarial loss.	La méthode produit des résultats de très bonne qualité à celle des méthodes existantes etpermet de récupérer les régions sur- et sous-exposées	lorsque les zones saturées sont trop grandes, le générateur peine à repeindre les régions. Approfondissement du site le réseau, la collecte de plus grands ensembles de données.

TABLE 2.1 – Bilan des travaux récents (1).

[39]	Le premier bloc est le bloc de Feature Extraction block (FEB) suivi du bloc de Feedback block (FBB) et finalement d'un bloc de reconstruction HDR (HRB)	Le réseau FHDR utilise les connexions de rétroaction et donc il améliore la netteté globale de l'image et il réalise une reconstruction encore meilleure.	Le réseau n'accepte pas les images de basse résolution(low resolution images).
[40]	Dynamic range clipping. Non-linear mapping from a camera response function.Quantization	ils ont utilisé quatre architectures de réseaux CNN de quantization-Net,Linearization-Net, Hallucination-Net,Refinement-Net.	Ne sont pas conçues spécifiquement pour le traitement des images bruyantes en basse lumière(handling noisy low-light images).

TABLE 2.2 – Bilan des travaux récents (2).

Travaille	Nombre des images LDRs	Architecture	Tone mapping
[35]	Une image	Auto-encodeurs	oui
[36]	3 images	CNN	oui
[38]	Une image	CNN	inverse-tone-mapping
[39]	Une image	FHDR	oui
[40]	Une image	CNN plus ensemble des méthodes : dynamic range clipping. non-linear mapping from a camera response function. quantization.	oui

TABLE 2.3 – Bilan des travaux récents (3)

2.7 Conclusion

Dans ce chapitre visait à élucider ce qu'est l'apprentissage profond aussi qu'une vision générale sur l'apprentissage profond, souligner ses applications dans le monde actuel et architectures de réseaux de neurones profonds ,finalement nous avons présenté quelques travaux qui utilise le deep learning pour la reconstruction des images HDRs.
Dans le chapitre suivant , nous allons présenté le schéma de conception globale et détaillée de notre système.

Chapitre 3

Conception

3.1 Introduction

Dans les chapitres précédents, nous avons présenté la notion des images HDRs, le deep learning ainsi que l'utilisation de deep learning pour reconstruire des images HDRs. Ce chapitre est consacré à la conception de notre système, nous avons décrit les objectifs et un schéma de conception global et détaillé afin d'expliquer les différentes étapes permettent la génération des images HDRs à partir d'une seule image LDR, tous en utilisant le deep learning.

3.1.1 Objectif

Notre objectif est de prédire les valeurs des pixels saturés à partir d'une image LDR produite par n'importe quel type de caméra, afin de produire l'image HDR finale.

3.2 Détails de la conception

Cette section sera orientée vers l'étude conceptuelle et la conception de notre système. Dans un premier temps, nous présenterons l'architecture générale qui met en évidence les principaux processus du système, et ensuite nous présenterons l'architecture détaillée où chaque processus sera expliqué indépendamment.

3.2.1 Architecture générale

Pour reconstruire une image HDR à partir d'une seule image LDR en utilisant l'architecture de CNN profonds : (voir la figure 3.1) :

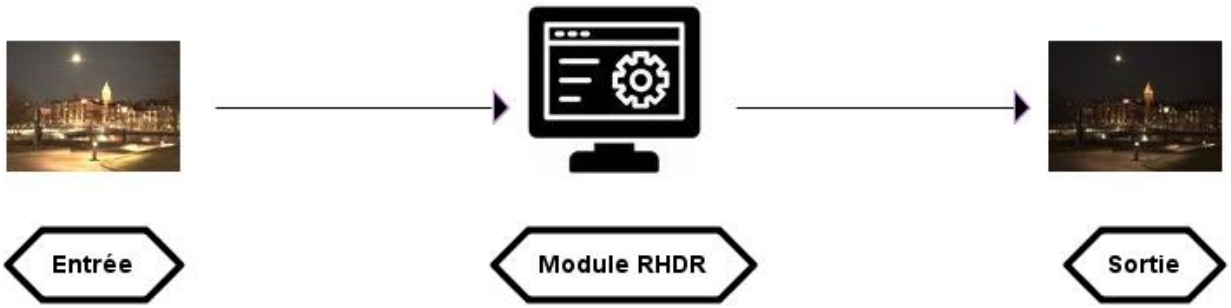


FIGURE 3.1 – L'architecture générale de notre système.

— L'entrée : la base de données (images LDRs).

— Le module de la reconstruction HDR (RHDR).

— La sortie : image HDR.

3.2.2 Architecture détaillée

Après avoir élaboré les spécifications pour la structure générale du projet, nous arrivons au le cœur de la matière avec sa conception détaillée en plusieurs étapes à partir de la lecture et la collecte des données jusqu'à la mise en œuvre (voir la figure 3.2)

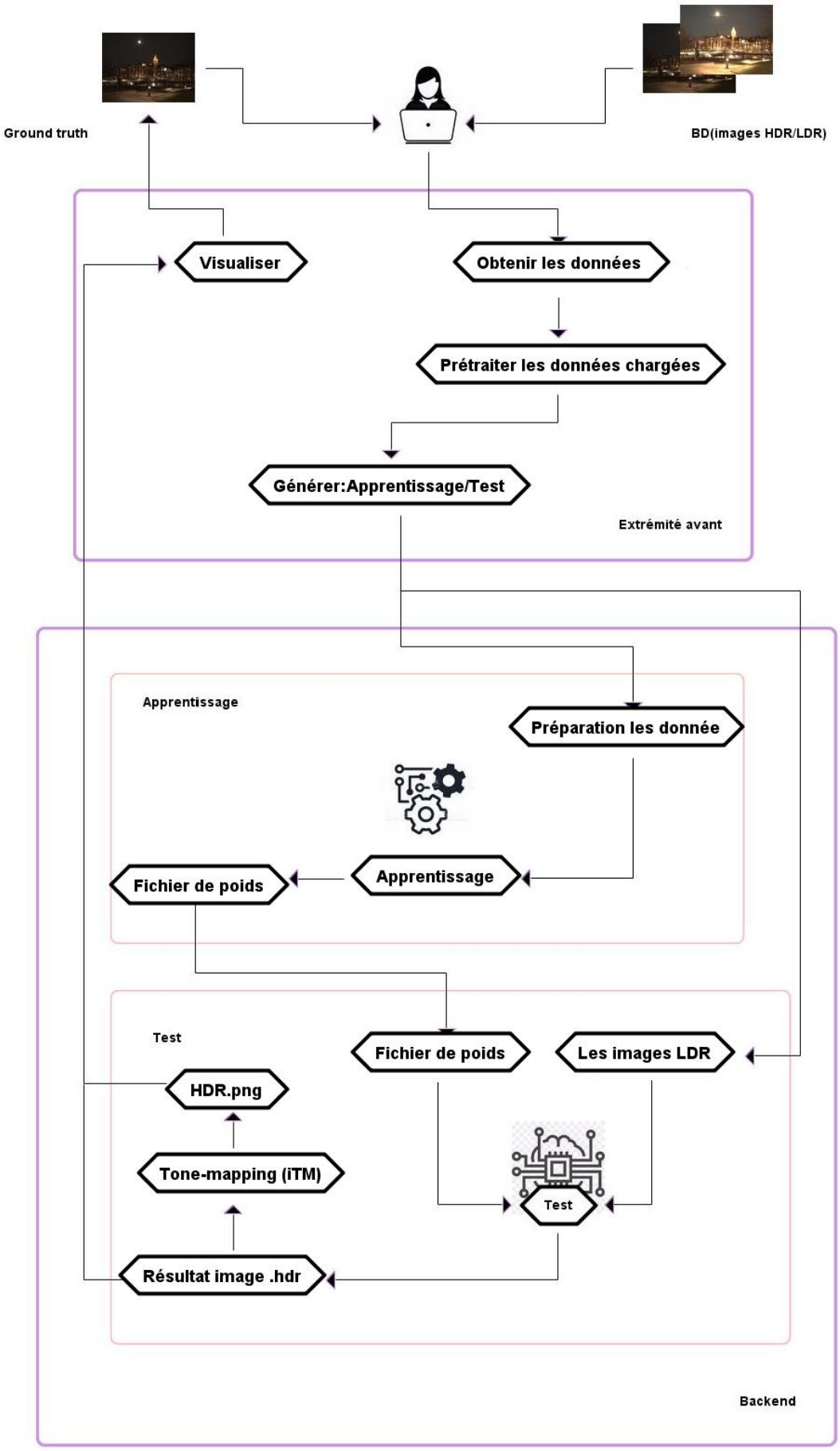


FIGURE 3.2 – L'architecture détaillée de notre système.

3.2.2.1 Lecture et collecte des données

Les données sont une clé très importante dans le processus d'apprentissage, c'est la première étape pour commencer à construire notre module, sélectionner et collecter les données appropriées pour le test et partie apprentissage.

— La base de données d'images : au début de notre travail, notre systèmes reçois une base de données d'images comme entrée qui sont de type LDR, on l'a divisé en deux sous bases de données (base d'apprentissage et base de test) (voir la figure 3.3) :
  — Base de données pour l'apprentissage : elle a composé d'un 74 dossiers. Chaque dossier contient une image LDR et leur image HDR correspondante (ground truth).
  — Base de données pour le test : il se compose de 8 images LDRs.

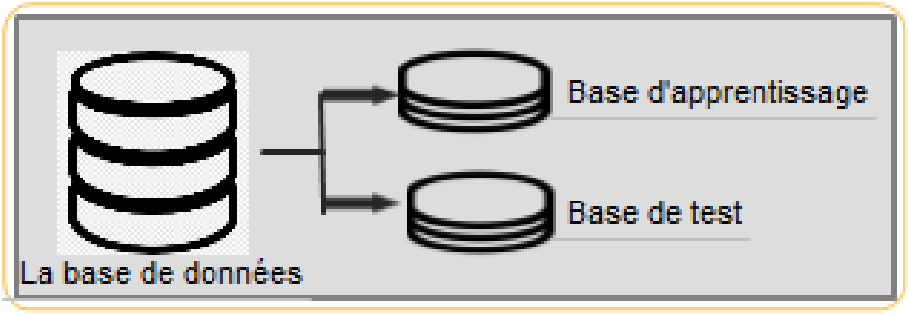


FIGURE 3.3 – La base de données.

Pré-traitement :

Avant de fournir nos données à la CNN, certaines techniques de traitement d'images peuvent être appliquées pour préparer les données et construire des modèles d'apprentissage en profondeur. Elles ont Utilisé pour préparer les données et créer des modèles d'apprentissage en profondeur. Dans notre travail, l'étape de pré-traitement comprend deux opérations : la normalisation et l'extraction des patches (voir la figure 3.4).

— **La normalisation :** La normalisation de lot (en anglais batch normalization) est une étape qui normalise le lot x_i un choix de paramètres Γ, β. En notant μ_B, σ_B^2 la moyenne et la variance de ce que l'on veut corriger du lot[41], on a :

$$x_i = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{3.1}$$

— **L'extraction des patches :**une étape la plus efficace pour former des modèles d'apprentissage profond, n'utilise pas l'image entière comme entrée, mais extraient l'image et utilisent seulement un petit nombre de patchs.

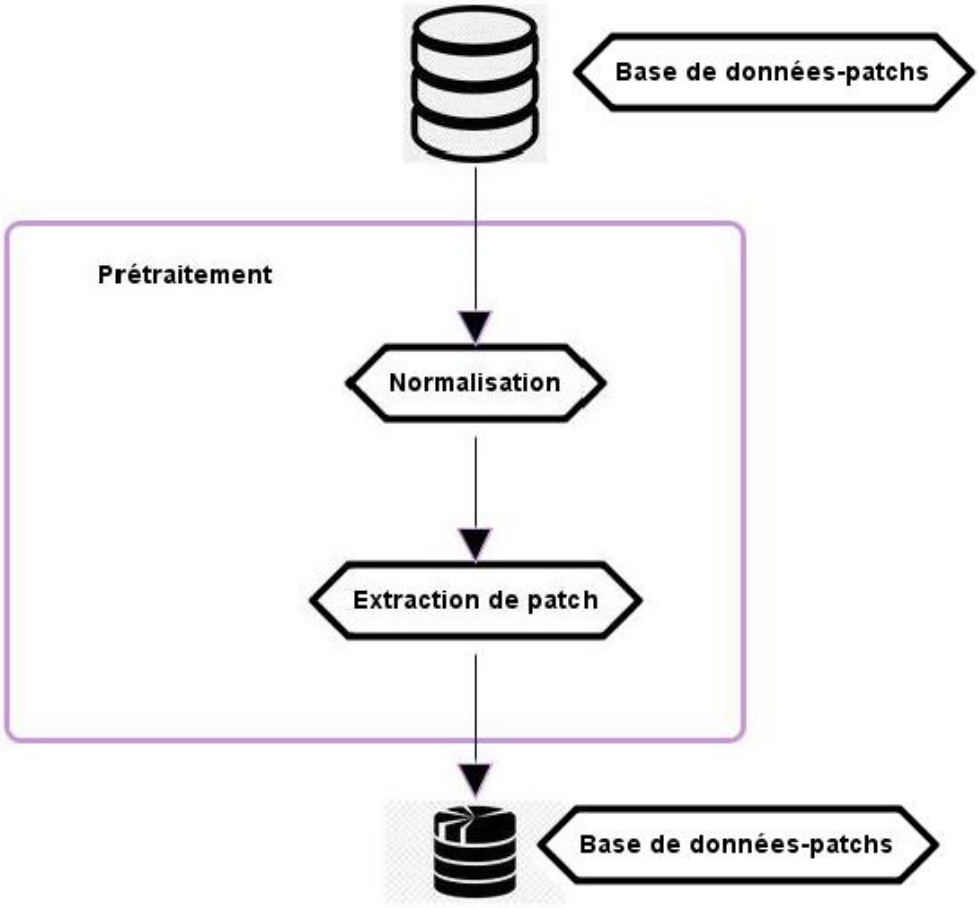


FIGURE 3.4 – Opérations de pré-traitement.

3.2.2.2 Processus d'apprentissage

Après la préparation des données d'entraînement, nous mettons en place l'architecture de notre modèle VGG-19(pour Visual Geometry Group) qui est basé sur le modèle CNN, et nous insérons les données d'apprentissage (les images LDRs et HDRs Ground truth). Afin que notre modèle peut être appris à partir des ensembles des données et des paramètres (plusieurs itérations et époques) appliquées à notre système, les meilleurs poids générés sont sauvegardés dans un ficher des poids, le modèle est automatiquement formé afin qu'il soit capable d'exécuter la tâche(voir la figure 3.5).

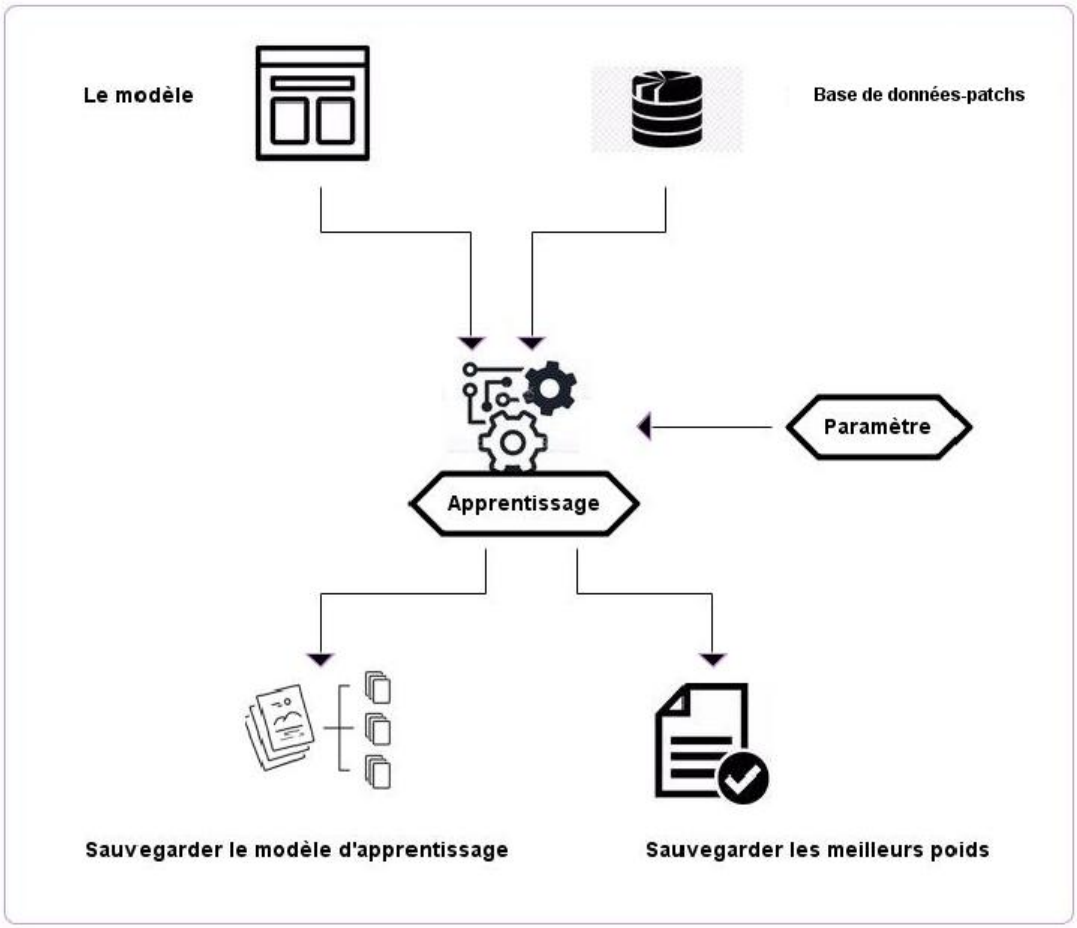


FIGURE 3.5 – Illustration du processus d'apprentissage.

3.2.2.3 Processus de tests

Pour tester notre réseau, nous passons à la phase de test. Nous chargeons le modèle et les poids sauvegardés de l'étape d'apprentissage, les images LDRs qui ont déjà préparé et les paramètres (gamma,width,height). Afin de voir si notre modèle est bien entraîné, on a obtenu les résultats des images hdr(extension .hdr), ensuite pour la visualiser sur un écran LDR, nous avons utiliser une méthode de Tone-mapping (iTM) (voir la figure 3.6).

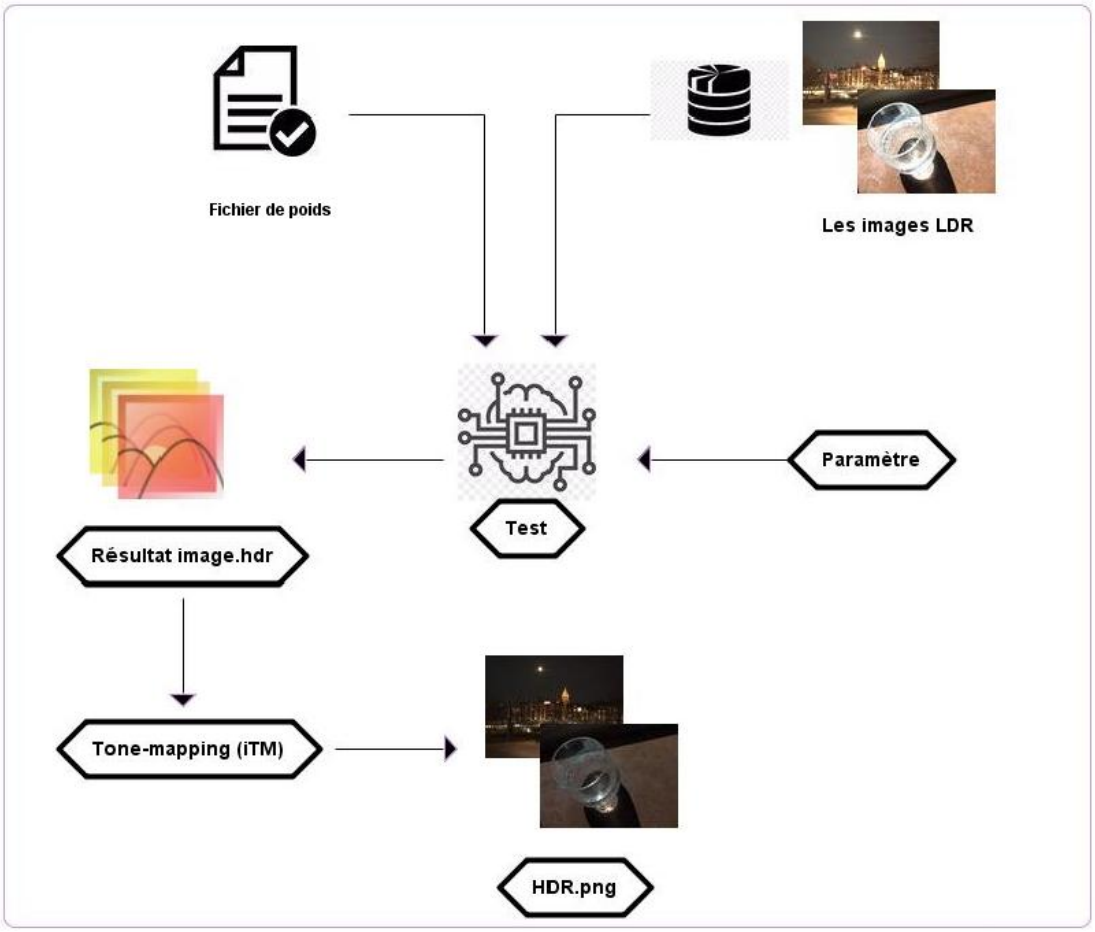


FIGURE 3.6 – Illustration du processus de tests.

3.3 Système de reconstruction des images HDRs à base d'un réseau neuronal convolutif

3.3.1 Architecture VGG-19 de notre système

Le réseau VGG est un réseau neuronal profond à fonctionnement multi- couches. Le VGGNet est basé sur le modèle CNN et il est appliqué sur le jeu de données ImageNet. Le VGG-19 est utile en raison de sa simplicité car des couches de convolution sont montées au sommet pour augmenter le niveau de profondeur. Pour réduire la taille du volume, des couches de regroupement max ont été utilisées comme gestionnaires dans VGG-19.[42]

L'image en entrée est de taille (largeur 1024 –hauteur 768). Elle est passé d'abord à la première couche de convolution, Cette couche est composée de 32 filtres de taille 3x3, la fonction d'activation ReLU est utilisée pour forcer les neurones à retourner des valeurs positives. Cette convolution produit 32 features maps de taille 32x32 chacune, ensuite, les 32 feature maps sont présentées en entrée à la deuxième couche de convolution. Le même processus est retourné trois

fois (conv+Relu).

Le Average pooling est appliqué ensuite pour réduire la taille de l'image et des paramètres.

À la sortie de cette couche, nous obtenons les feature maps. On répète les mêmes opérations sur les couches de convolutions deux fois après la couche Average pooling et la fonction ReLU. Après quatre couches de convolution, Average pooling, et la fonction ReLU, ensuite les convolution, average pooling, et Relu se répètents huit même. À la sortie de la dernière couche Pooling, nous utilisons un réseau de neurones composé d'une couche FC(Fully connected). Le nombre de neurones dans cette couche est égal au nombre de classes dans la base d'images. La sortie de neurone de cette couche entièrement connectée sont converties en probabilités par l'intermédiaire d'une fonction d'activation 'Leaky ReLU'. La figure 3.7 précise toutes les phases précédentes.

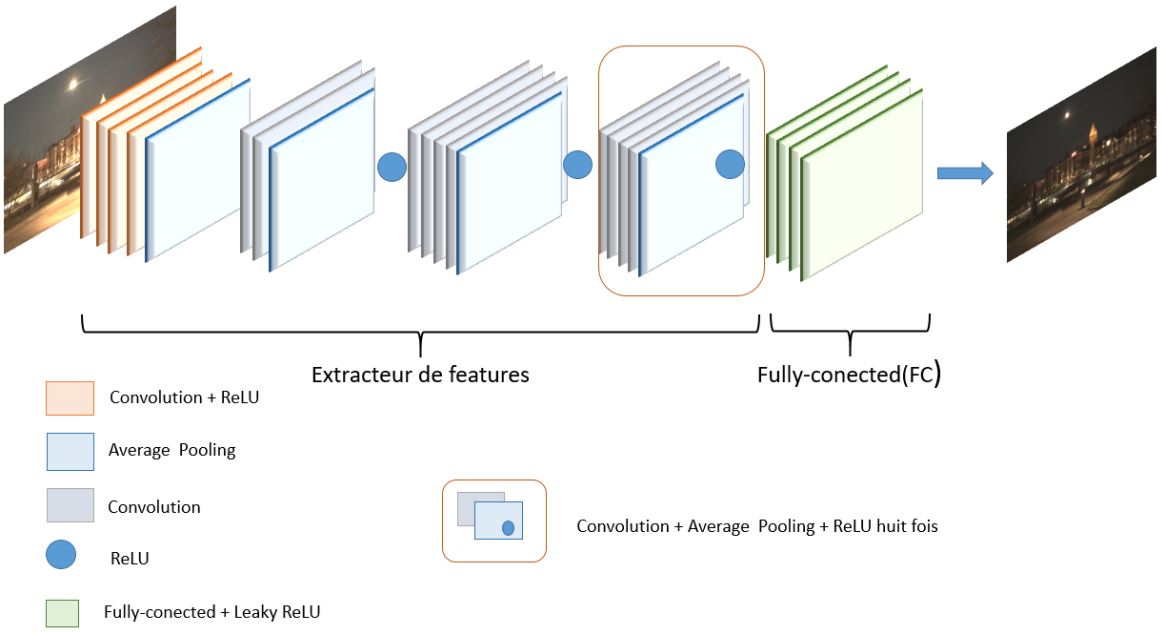


FIGURE 3.7 – L'architecture de modèle VGG-19 utilisé dans notre système pour la reconstruction HDR.

3.3.2 Les couches de réseaux de neurones convolutionnels

Il existe quatre types de couches pour un réseau de neurones convolutif : la couche de convolution, la couche de pooling, la couche de Flattening et la couche complètement connectée. Le CNN traite les images de manière séquentielle en utilisant les couches suivantes :

3.3.2.1 Couche convolutionnelle (CONV)

L'opération de convolution va prendre notre feature detector(kernel ou bien de filtres) et applique ces filtres sur l'image d'entrée, on obtient comme résultat une nouvelle image qui s'appelle feature map.

L'opérateur de convolution est défini par :

$$(W * X)(i, j) = \sum_m \sum_n X(m, n)W(i - m, j - n) \tag{3.2}$$

Où :

W :matrice feature detector.

X :image d'entrée.

La figure 3.8 illustre un feature detector qui est une matrice carrée de W(3x3).

0	1	2
2	2	0
0	1	2

FIGURE 3.8 – Exemple de la matrice feature detector.

La figure 3.9 montre comment le processus convolutionnel fonctionne. On a comme entrée une matrice (image) X(5x5), on va multiplier les valeurs de pixel de la matrice X (en bleu foncé) avec le feature detector terme à terme et après on additionne tous les résultats, la valeur finale est ajoutée dans une feature map (pixel en vert foncé), et on rempli toute la matrice feature map avec la même manière. Si on réitère ce processus ce qu'on obtient énormément de feature map ça s'appelle la couche de convolution.

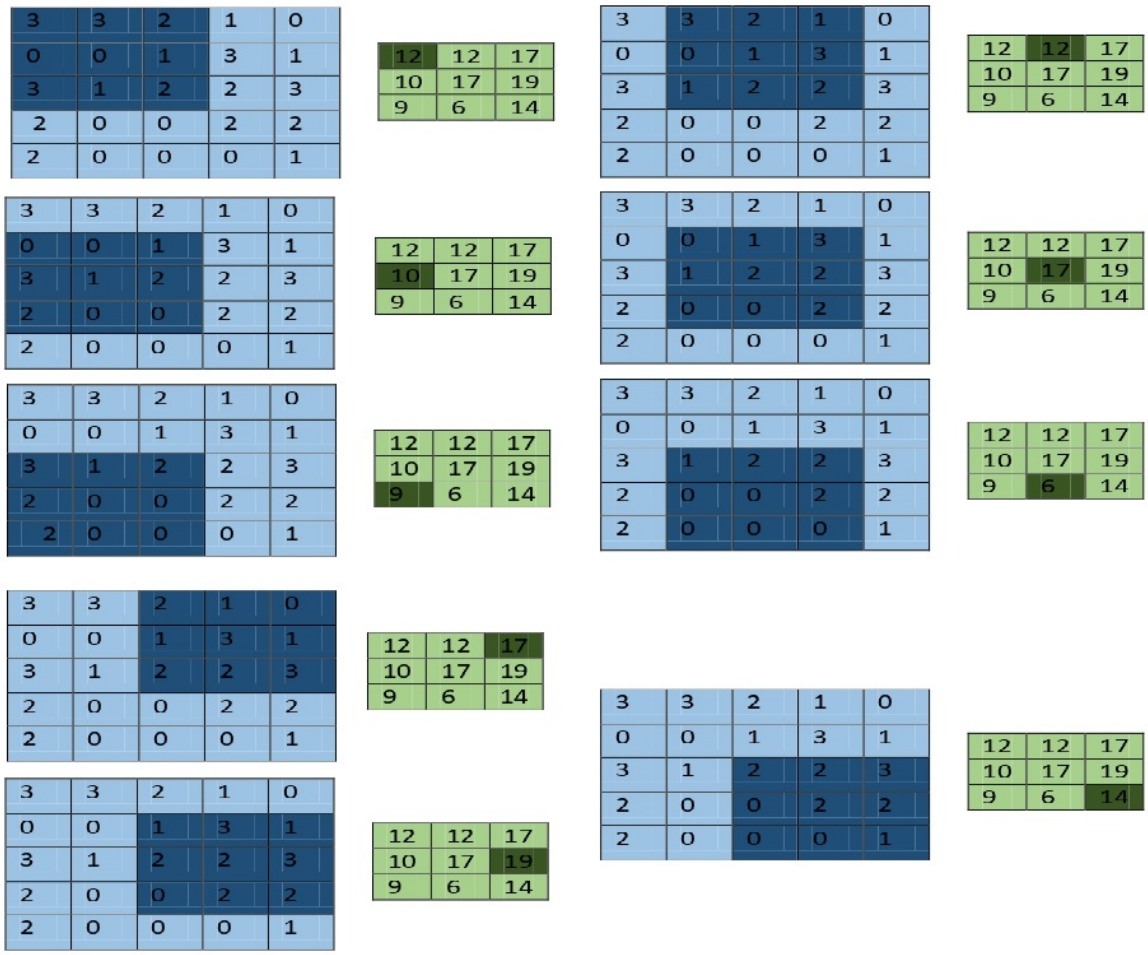


FIGURE 3.9 – La marche de processus convolutionnelle.

Premièrement on a réduit la taille de l'image à gauche figure 3.9, on a une matrice 5x5 et comme résultat on obtient une matrice 3x3.
C'est une étape importante par ce qu'en réduisant la taille de l'image et bien tout simplement elle sera plus facile à traiter par ce qu'on garde seulement la partie intéressante.

3.3.2.2 Couche de mise en commun (Pooling)

La couche de mise en commun ("POOL" en anglais) est une opération de sous-échantillonnage, généralement appliquée après la couche convolutive [43]. Mise en commun permet de réduiser la taille de l'espace de l'image intermédiaire, réduisant ainsi les paramètres en réseau. Par conséquent, il est courant d'insérer la couche de mise en commun régulièrement Contrôlé entre les deux couches convolutives consécutives de l'architecture CNN sur ajustement.
Les types de pooling les plus populaires sont le max et l'average pooling, où les valeurs maximales et moyennes sont prises respectivement [43]. Dans ce travail on a utilisé l'average pooling.
La figure 3.10 précise les caractéristiques de l'average pooling.

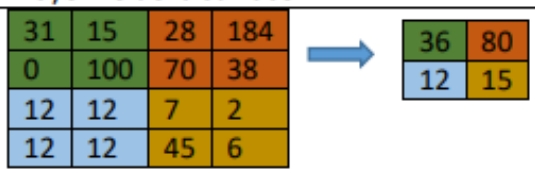
Type	Average pooling
But	Chaque opération de pooling sélectionne la valeur moyenne de la surface
Illustration	
Commentaires	- Sous-échantillonne la feature map. - Utilisé dans LeNet

FIGURE 3.10 – les caractéristiques de l'average pooling.

3.3.2.3 Couche Flattening

Jusqu'ici on a obtenu pooled feature map ça va être d'aplatir complètement cette matrice, on prend les valeurs une par une, ligne par ligne, on les aligné dans une colonne comme dans la figure 3.11, on va placer ça dans un réseau de neurones. À la fin on va voir un vecteur énorme qui contient toutes les valeurs les unes après les autres et ça va constituer une couche d'entrée pour le réseau de neurones.

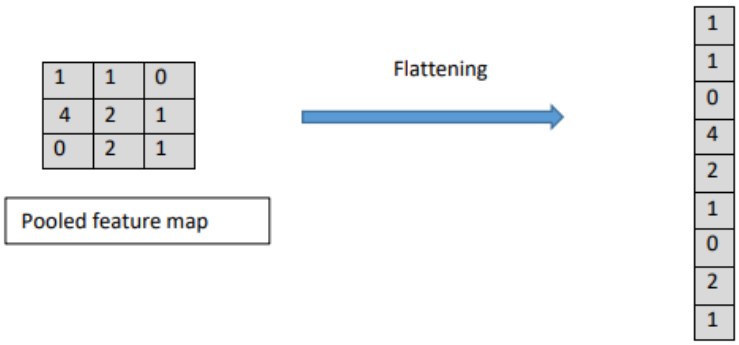


FIGURE 3.11 – Opération flattening.

3.3.2.4 Couche entièrement connecté (Fully connected)

La dernière étape est l'étape de flattening, on avait créer un gros vecteur vertical et ça on avait dit qu'on allait nous en servir justement dans le réseau de neurones artificiels qui est complètement connecté, ça correspond ici à la couche cachée, mais appeler maintenant la couche complètement connectée c'est par ce que dans les réseaux artificiels les couches cachées ne sont pas forcément complètement connectées, donc on l'appelle comme ça justement pour faire cette

distinction.

L'objectif de ce réseau supplémentaire ça va être de combiner les features qu'on a déjà ici dans la couche d'entrée et qu'on est parti de l'étape de Flattening pour encore améliorer la prédiction, dans la couche d'entrée on a déjà features qui sont encodés déjà probablement assez efficaces pour prédire la classification de l'image mais en même temps on a aussi cette structure qui existe et qui est faite justement pour prendre des features en entrée comme des variables et les assembler ensemble pour trouver des combinaisons qui font des meilleures prédictions.

Faire envoyer notre gros vecteur qui contient toutes les features après l'étape de flattening, on a déjà fait tout le travail préparatoire de l'étape de convolution, pooling, Flattening, maintenant tout va se passer dans ce réseau. D'abord dans un premier temps, l'information va passer dans le réseau, elle se propage dans les neurones via les synapses, elle va activer certains neurones mais d'autres non et puis dans tout ça, ça va créer une première prédiction, mais supposons que dans la réalité son erreur est bien à ce moment il calcule les erreurs avec une fonction de coût qu'on aura décidé au préalable, en général on utilise la fonction de coût entropie croisée but évidemment c'est de réduire cette erreur. L'erreur est rétro propagée dans le réseau et donc on va ajuster les poids qui sont assiégés à chaque synapse avec l'algorithme du gradient.

En effet, les feature detector vont être ajustées pour reconnaître une image. Elles vont être ajustés aussi entre eux afin d'obtenir une meilleure prédiction la prochaine fois, ensuite et bien ça recommence par la rétro propagation, on a une nouvelle erreur et ça ca be rétro propage encore jusqu'à ce que le réseau soit complètement entraîné.

3.3.2.5 Les fonctions d'activation

— **Fonction ReLU :** elle a utilisé pour que notre modèle soit d'avantage non linéaire. Donc, on veut accentuer la non linéarité et ça c'est possible grâce à la fonction , pour une image, on a des couleurs qui sont complètement différentes et qui peuvent être changer d'un pixel à l'autre (et cette opération est non linéaire). Le problème c'est que quand on applique seulement la fonction de convolution, on prend le risque de créer quelque chose de linéaire et donc cette fonction elle va retirer tout ce qui est noir elle va retirer tous les valeurs négatives vont être remplacées par du zéro.

— **Leaky ReLU :**

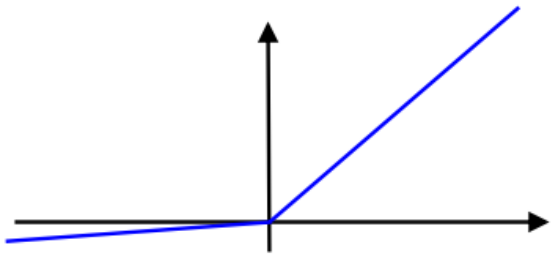


FIGURE 3.12 – Leaky ReLU .[44]

$$f(x) = \begin{cases} \alpha x & \text{si } x \text{ est négatif} \\ x & \text{autrement} \end{cases}$$ [44]

- Gradient = 0 signifie impossibilité d'entraîner.
- Pente très légère dans la partie négative.
- Si un paramètre a (entraînable) par neurone/couche, on obtient la PReLU.
- Donne des distributions de sorties plus centrées à 0 que ReLU.[44]

— **Fonction sigmoïde :** cette fonction donne une valeur entre 0 et 1, une probabilité. Elle est donc très utilisée pour les classifications binaires, lorsqu'un modèle doit déterminer seulement deux étiquettes.

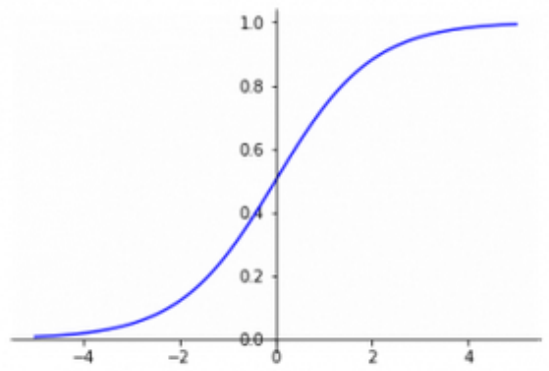


FIGURE 3.13 – Fonction sigmoïde.[44]

3.4 Conclusion

Dans ce chapitre visait à élucider notre objectif à partir d'image LDR, on génère une image HDR finale à base de deep learning et nous avons présenté les détails de la conception du système.

Les environnements et les outils de développement utilisés dans la réalisation du système et les résultats seront décrits dans le chapitre suivant.

Chapitre 4

Implémentation, discussion et résultats

4.1 Introduction

Dans ce chapitre, nous présentons le développement de la conception détaillée qui amène notre méthode de conception d'un système d'apprentissage profond pour la reconstruction HDR avec CNN(HDR-CNN). Cette étape comprend :

— La définition comment l'information de système doit être construite (environnements et outils de développement).

— S'assurer que l'information de système est opérationnel et utile.

— Affichage de résultats obtenus.

4.2 Environnements et outils de développement

4.2.1 Python

Python est un langage de programmation de haut niveau interprété (il n'y a pas d'étape de compilation) et orienté objet avec une sémantique dynamique. Il est très sollicité par une large communauté de développeurs et de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction du coût de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes. Python et ses bibliothèques sont disponibles (en source ou en binaires) sans charges pour la majorité des plateformes et peuvent être redistribués gratuitement [1].

4.2.2 Tensorflow

Est une bibliothèque de logiciels open source pour le calcul numérique haute performance. Son architecture flexible permet un déploiement facile du calcul sur une variété de plates-formes

1. https://docs.python.org/fr/3/tutorial/

(CPU, GPU, TPU), et des ordinateurs de bureau aux clusters de serveurs en passant par les périphériques mobiles et périphériques. Développé à l'origine par des chercheurs et des ingénieurs de l'équipe Google Brain au sein de l'organisation IA de Google, il est livré avec un support solide pour l'apprentissage automatique et l'apprentissage en profondeur et le noyau de calcul numérique flexible est utilisé dans de nombreux autres domaines scientifiques [2].

4.2.3 Keras

Keras est un framework open source d'apprentissage profond pour le Python, capable de s'exécuter sur TensorFlow. Il est écrit par Francis Chollet, membre de l'équipe Google. Keras est utilisé dans un grand nombre de startups, de laboratoires de recherche (dont le CERN, Microsoft Research et la NASA) et de grandes entreprises telles que Netflix, Yelp, Square, Uber, Google, etc. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro Electronic Intelligent Robot Operating System). En 2017, l'équipe TensorFlow de Google a pris la décision de fournir un support pour Keras et de l'intégrer dans la bibliothèque principale de TensorFlow. Il présente un ensemble d'abstractions de plus haut niveau et plus intuitives qui facilitent la configuration des réseaux neuronaux.[45]

4.2.4 Nympy

Est une bibliothèque permettant d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres, des fonctions sophistiquées(diffusion), on peut aussi l'intégrer le code C / C ++ et Fortran. [3]

4.2.5 OpenCV

OpenCV (pour Open Computer Vision) est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel. La société de robotique Willow Garage et la société ItSeez se sont succédé au support de cette bibliothèque. Depuis 2016 et le rachat de ItSeez par Intel, le support est de nouveau assuré par Intel. Cette bibliothèque est distribuée sous licence BSD.[41]

4.2.6 PIL

Pillow est une bibliothèque de traitement d'image, qui est un fork et successeur du projet PIL (Python Imaging Library). Elle est conçue de manière à offrir un accès rapide aux données contenues dans une image, et offre un support pour différents formats de fichiers tels que PPM,

2. https://pypi.org/project/tensorflow-gpu/
3. http://www.numpy.org/

PNG, JPEG, GIF, TIFF et BMP. Pillow dispose de capacités de traitement d'images relativement puissantes, et a pour but d'offrir une solide base à toute application générale de traitement d'images [4].

4.2.7 OpenEXR

OpenEXR fournit la spécification et l'implémentation de référence du format de fichier EXR, le format de stockage d'images de qualité professionnelle de l'industrie cinématographique. L'objectif du format est de représenter avec précision et efficacité les données d'image linéaire de scène à plage dynamique élevée et les métadonnées associées, avec une forte prise en charge des cas d'utilisation multi-parties et multicanaux. La bibliothèque est largement utilisée dans les logiciels d'application hôte où la précision est essentielle, tels que le rendu photoréaliste, l'accès à la texture, la composition d'images, la composition profonde et l'analyse des dépendances [5].

4.2.8 Matplotlib

est une bibliothèque de traçage pour le langage de programmation Python et son extension mathématique numérique NumPy . Il fournit une API orientée objet permettant d'incorporer des graphiques dans des applications à l'aide de kits d'outils d'interface graphique à usage général tels que Tkinter , wxPython , Qt ou GTK +.[46]

4.2.9 Pandas

pandas est une bibliothèque open source sous licence BSD fournissant des structures de données hautes performances et faciles à utiliser, ainsi que des outils d'analyse des données pour le langage de programmation Python . pandas est un projet sponsorisé par NumFOCUS . Cela contribuera au succès du développement de pandas en tant que projet open source de classe mondiale et permettra de faire un don au projet.[46]

4.2.10 Configuration utilisée dans l'implémentation

4.2.10.1 Configuration matérielle à distante (Google Colab)

Google Colaboratory, souvent raccourci en "Colab" est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d'installer quoi que ce soit sur notre ordinateur à

4. https://pypi.org/project/Pillow/
5. https://www.openexr.com/

l'exception d'un navigateur.cet environnement nous permet d'écrire et d'exécuter du code Python dans votre navigateur, avec aucune configuration requise ,accès gratuit aux GPU ,partage facile.[47]

De plus, les documents Colab (Jupyter Notebook) sont enregistrés directement votre compte Google Drive. L'infrastructure distante mise à disposition par Google Colab et utilisée pour l'entraînement possède cette configuration :

— Processeur Intel Core Xeon CPU @2.3Ghz, 45MB Cache.

— Processeur graphique NVIDIA Tesla K80, ayant 2496 cœurs CUDA, Compute 3,7, 12 Go (11.439 Go utilisable) GDDR5 VRAM.

— Mémoire vive de 12.6 Go.

— Disque dur de capacité 320 Go.

— Jupyter Notebook

— Système d'exploitation Linux x64 bits.

4.2.10.2 Configuration matérielle locale

La configuration matérielle locale utilisé dans notre implémentation est :

— processeur Intel(R) Xeon(R) CPU E5-1620 v2 @ 3.70GHz.

— RAM de taille 8 GO.

— Système d'exploitation Ubuntu 16.04.

4.3 Implémentation

Afin d'implémenter notre modèle d'apprentissage profond HDR-CNN, nous avons utilisé le langage Python avec quelques APIs et bibliothèques bien connues travaillant sur l'apprentissage profond (CNN). L'une des bibliothèques Python les plus puissantes et les plus faciles à utiliser pour le développement et l'évaluation de modèles d'apprentissage profond est Keras, elle englobe la bibliothèque de calcul numérique efficace TensorFlow. Les étapes suivantes présentent notre implémentation du modèle d'apprentissage profond HDR-CNN.

4.3.1 Préparation des données

Après avoir rechercher et télécharger le jeu de données, nous avons tout d'abord appliqué des opérations de normalisation et d'extraction de patchs sur chaque image du jeu de données(voir la liste 4.1) pour obtenir le jeu de données des patchs qui est divisé en deux ensembles distincts : apprentissage et test, en utilisant une répartition de 88% d'apprentissage, 11% de test.

Liste 4.1 – Python code img_io.py pour lire jeu de données d'apprentissage et test

```
#Apprentissage
def load_training_pair(name_hdr, name_jpg):

    data = np.fromfile(name_hdr, dtype=np.float32)
    ss = len(data)

    if ss < 3:
        return (False,0,0)

    sz = np.floor(data[0:3]).astype(int)
    npix = sz[0]*sz[1]*sz[2]
    meta_length = ss - npix

    # Lire HDR ground truth
    y = np.reshape(data[meta_length:meta_length+npix], (sz[0], sz[1],
       sz[2]))

    # Lire LDR image
    x = scipy.misc.imread(name_jpg).astype(np.float32)/255.0

    return (True,x,y)
#Test
# Lire et preparer une image 8 bits dans une resolution specifiee
def readLDR(file, sz, clip=True, sc=1.0):
    try:
        x_buffer = scipy.misc.imread(file)

        # Clip image, de sorte que le ratio ne soit pas modifie par
           resize de image.
        if clip:
            sz_in = [float(x) for x in x_buffer.shape]
            sz_out = [float(x) for x in sz]

            r_in = sz_in[1]/sz_in[0]
            r_out = sz_out[1]/sz_out[0]

            if r_out / r_in > 1.0:
                sx = sz_in[1]
                sy = sx/r_out
            else:
```

```
                sy = sz_in[0]
                sx = sy*r_out

            yo = np.maximum(0.0, (sz_in[0]-sy)/2.0)
            xo = np.maximum(0.0, (sz_in[1]-sx)/2.0)

            x_buffer = x_buffer[int(yo):int(yo+sy),int(xo):int(xo+sx)
                ,:]

        # resize des images et conversion en float
        x_buffer = scipy.misc.imresize(x_buffer, sz)
        x_buffer = x_buffer.astype(np.float32)/255.0
```

4.3.2 Construire HDR-CNN

4.3.2.1 Importer des bibliothèques et des modules

Pour construire des réseaux de neurones profonds avec tensorflow et keras, nous importons tout d'abord les différentes bibliothèques et modules présentés dans la liste 4.2 et décrits dans les paragraphes suivantes.

Liste 4.2 – Les bibliothèques

```
from tensorflow.keras.activations import sigmoid, tanh
from tensorflow_addons.layers import AdaptiveAveragePooling2D
from tensorflow.keras.layers import Conv2D, LeakyReLU, ReLU,
   BatchNormalization, Conv2DTranspose, Reshape, Add, UpSampling2D ,
   Concatenate , Input
from tensorflow.keras.applications import vgg19
from tensorflow.keras.utils import multi_gpu_model
from PIL import Image
import OpenEXR
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

4.3.2.2 Initialisation des paramètres d'apprentissage

Avant de pouvoir commencer à entraîner le réseau, nous devons initialiser ses paramètres. Les valeurs des paramètres utilisés sont déterminées en se basant sur des valeurs suggérées précédemment, que l'on peut trouver dans la littérature et par des expériences. Les détails des paramètres, utilisés dans cette étude, sont décrits dans le tableau 4.1 et présentés dans la liste 4.3.

Paramètres	Type	Valeurs	Descriptions
lr	float	1e-3	Valeur à virgule flottante ou planification ,Appelable qui n'accepte aucun argument et retourne la valeur réelle à utiliser, Le taux d'apprentissage. La valeur par défaut est 0,001.
num_epochs	int	1600	Le nombre d'époques est le nombre de passages complets. à travers l'ensemble de données d'entraînement Le nombre d'epochs est un paramètre qui définit le nombre de fois que l'algorithme d'apprentissage travaillera sur l'ensemble des données d'apprentissage.
train_batch_size	int	16	est un hyperparamètre qui définit le nombre d'échantillons à avant de mettre à jour les paramètres du modèle interne
pretrain_dir	str	"weights /best.h5"	Chemin d'accès pour enregistrer ficher de poids

TABLE 4.1 – Description des paramètres

Liste 4.3 – Initialisation des paramètres

```
lr = 1e-3
num_epochs = 1600
train_batch_size = 16
pretrain_dir = "weights/best.h5"
```

4.3.2.3 Création et ajustement du modèle

— **Modèle Keras layer :**

— **Conv2D :** cette couche crée un noyau convolutif qui est convolué avec la couche d'entrée pour produire un tenseur de sortie.

— **AveragePooling2D :** L'échantillonnage descendant de la représentation d'entrée, on calcule la valeur moyenne sur la fenêtre définie par la taille du pool pour chaque dimension le long de l'axe des caractéristiques.

— **Modèle résumé :** à travers l'illustration résumé, nous pouvons reconnaître chaque couche avec ses entrées/sorties qui nous permet de voir l'architecture de notre modèle de manière détaillée (voir les figures 4.1).

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 64)	6976
conv2d_2 (Conv2D)	(None, 256, 256, 64)	6976
conv2d_1 (Conv2D)	(None, 256, 256, 6)	3462
conv2d_3 (Conv2D)	(None, 256, 256, 6)	3462
conv2d_4 (Conv2D)	(None, 256, 256, 64)	1216
conv2d_5 (Conv2D)	(None, 128, 128, 32)	18464
batch_normalization (BatchNorma	(None, 128, 128, 32)	128
re_lu (ReLU)	(None, 128, 128, 32)	0
conv2d_6 (Conv2D)	(None, 64, 64, 64)	18496
batch_normalization_1 (BatchNor	(None, 64, 64, 64)	256
batch_normalization_1 (BatchNor	(None, 64, 64, 64)	256
re_lu_1 (ReLU)	(None, 64, 64, 64)	0
conv2d_7 (Conv2D)	(None, 32, 32, 128)	73856
batch_normalization_2 (BatchNor	(None, 32, 32, 128)	512
re_lu_2 (ReLU)	(None, 32, 32, 128)	0
conv2d_8 (Conv2D)	(None, 16, 16, 256)	295168
batch_normalization_3 (BatchNor	(None, 16, 16, 256)	1024
re_lu_3 (ReLU)	(None, 16, 16, 256)	0

conv2d_9 (Conv2D)	(None, 16, 16, 256)	65792
conv2d_10 (Conv2D)	(None, 16, 16, 256)	590080
conv2d_11 (Conv2D)	(None, 16, 16, 256)	1638656
re_lu_4 (ReLU)	(None, 16, 16, 256)	0
re_lu_5 (ReLU)	(None, 16, 16, 256)	0
re_lu_6 (ReLU)	(None, 16, 16, 256)	0
add (Add)	(None, 16, 16, 256)	0
conv2d_12 (Conv2D)	(None, 16, 16, 256)	590080
add_1 (Add)	(None, 16, 16, 256)	0
conv2d_13 (Conv2D)	(None, 16, 16, 256)	65792
conv2d_14 (Conv2D)	(None, 16, 16, 256)	590080
conv2d_15 (Conv2D)	(None, 16, 16, 256)	1638656
re_lu_7 (ReLU)	(None, 16, 16, 256)	0
re_lu_8 (ReLU)	(None, 16, 16, 256)	0
re_lu_9 (ReLU)	(None, 16, 16, 256)	0
add_2 (Add)	(None, 16, 16, 256)	0

conv2d_16 (Conv2D)	(None, 16, 16, 256)	590080
add_3 (Add)	(None, 16, 16, 256)	0
conv2d_17 (Conv2D)	(None, 16, 16, 256)	65792
conv2d_18 (Conv2D)	(None, 16, 16, 256)	590080
conv2d_19 (Conv2D)	(None, 16, 16, 256)	1638656
re_lu_10 (ReLU)	(None, 16, 16, 256)	0
re_lu_11 (ReLU)	(None, 16, 16, 256)	0
re_lu_12 (ReLU)	(None, 16, 16, 256)	0
add_4 (Add)	(None, 16, 16, 256)	0
conv2d_20 (Conv2D)	(None, 16, 16, 256)	590080
add_5 (Add)	(None, 16, 16, 256)	0
conv2d_21 (Conv2D)	(None, 16, 16, 256)	65792
conv2d_22 (Conv2D)	(None, 16, 16, 256)	590080
conv2d_23 (Conv2D)	(None, 16, 16, 256)	1638656
re_lu_13 (ReLU)	(None, 16, 16, 256)	0
re_lu_14 (ReLU)	(None, 16, 16, 256)	0
re_lu_15 (ReLU)	(None, 16, 16, 256)	0
add_6 (Add)	(None, 16, 16, 256)	0
conv2d_24 (Conv2D)	(None, 16, 16, 256)	590080
add_7 (Add)	(None, 16, 16, 256)	0
conv2d_25 (Conv2D)	(None, 16, 16, 256)	65792
conv2d_26 (Conv2D)	(None, 16, 16, 256)	590080
conv2d_27 (Conv2D)	(None, 16, 16, 256)	1638656
re_lu_16 (ReLU)	(None, 16, 16, 256)	0
re_lu_17 (ReLU)	(None, 16, 16, 256)	0
re_lu_18 (ReLU)	(None, 16, 16, 256)	0
add_8 (Add)	(None, 16, 16, 256)	0

conv2d_32 (Conv2D)	(None, 16, 16, 256)	590080
add_11 (Add)	(None, 16, 16, 256)	0
conv2d_33 (Conv2D)	(None, 16, 16, 256)	65792
conv2d_34 (Conv2D)	(None, 16, 16, 256)	590080
conv2d_35 (Conv2D)	(None, 16, 16, 256)	1638656
re_lu_22 (ReLU)	(None, 16, 16, 256)	0
re_lu_23 (ReLU)	(None, 16, 16, 256)	0
re_lu_24 (ReLU)	(None, 16, 16, 256)	0
add_12 (Add)	(None, 16, 16, 256)	0
conv2d_36 (Conv2D)	(None, 16, 16, 256)	590080
add_13 (Add)	(None, 16, 16, 256)	0
conv2d_37 (Conv2D)	(None, 16, 16, 256)	65792
conv2d_38 (Conv2D)	(None, 16, 16, 256)	590080
conv2d_39 (Conv2D)	(None, 16, 16, 256)	1638656
re_lu_25 (ReLU)	(None, 16, 16, 256)	0
re_lu_26 (ReLU)	(None, 16, 16, 256)	0
re_lu_27 (ReLU)	(None, 16, 16, 256)	0
add_14 (Add)	(None, 16, 16, 256)	0
conv2d_40 (Conv2D)	(None, 16, 16, 256)	590080
add_15 (Add)	(None, 16, 16, 256)	0
conv2d_41 (Conv2D)	(None, 16, 16, 256)	65792
conv2d_42 (Conv2D)	(None, 16, 16, 256)	590080
conv2d_43 (Conv2D)	(None, 16, 16, 256)	1638656
re_lu_28 (ReLU)	(None, 16, 16, 256)	0
re_lu_29 (ReLU)	(None, 16, 16, 256)	0
re_lu_30 (ReLU)	(None, 16, 16, 256)	0
add_16 (Add)	(None, 16, 16, 256)	0

```
conv2d_transpose (Conv2DTranspo (None, 32, 32, 128)  524416
_________________________________________________________________
batch_normalization_4 (BatchNor (None, 32, 32, 128)  512
_________________________________________________________________
leaky_re_lu (LeakyReLU)         (None, 32, 32, 128)  0
_________________________________________________________________
add_21 (Add)                    (None, 32, 32, 128)  0

_________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 64, 64, 64)   131136
_________________________________________________________________
batch_normalization_5 (BatchNor (None, 64, 64, 64)   256
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)       (None, 64, 64, 64)   0
_________________________________________________________________
add_22 (Add)                    (None, 64, 64, 64)   0

_________________________________________________________________
conv2d_transpose_2 (Conv2DTrans (None, 128, 128, 32) 32800
_________________________________________________________________
batch_normalization_6 (BatchNor (None, 128, 128, 32) 128
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)       (None, 128, 128, 32) 0
_________________________________________________________________
add_23 (Add)                    (None, 128, 128, 32) 0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTrans (None, 256, 256, 3)  1539
_________________________________________________________________
batch_normalization_7 (BatchNor (None, 256, 256, 3)  12
_________________________________________________________________
re_lu_34 (ReLU)                 (None, 256, 256, 3)  0
=================================================================
Total params: 29,966,875
Trainable params: 29,965,461
Non-trainable params: 1,414
```

FIGURE 4.1 – l'architecture détaillée

4.3.2.4 Apprentissage de modèle

Une fois le modèle créé, nous avons crée une fonction pour compiler le modèle (voir la liste 4.4).

Liste 4.4 – Compiler le modèle

```
1 def train(config):
```

```
    SDR = generate_HDR_dataset.DataGenerator(config.images_path,
       config.train_batch_size)
    lr = config.lr

    model_x = HDR.NHDRRNet(config)
    x = Input(shape=(3, 256, 256, 6))
    out = model_x.main_model(x)
    model = Model(inputs=x, outputs=out)
    model.summary()

    if config.load_pretrain:
        model.load_weights(config.pretrain_dir)
        print('pretraitement charge')

    min_loss = 10000100
    print("Commencer l'apprentissage ...")
    loss_data = []
    for epoch in range(config.num_epochs):
        total_loss = 0
        if epoch + 1 > 80000:
            if epoch + 1 % 20000 == 0:
                lr = lr * 0.9
        optimizer = tf.keras.optimizers.Adam(learning_rate=lr, epsilon
           =1e-8)
        for iteration in range(len(SDR)):
            with tf.GradientTape() as tape:
                img_lowlight = SDR[iteration]

                imgs = img_lowlight[:, :3, :, :, :]
                imgs = tf.dtypes.cast(imgs, tf.float32)
                gt = img_lowlight[:, 3, :, :, :3]
                gt = tf.dtypes.cast(gt, tf.float32)
                out = model(imgs)

                gt = tf.math.log(1 + MU * gt) / tf.math.log(1 + MU)
                out = tf.math.log(1 + MU * out) / tf.math.log(1 + MU)
                mse = tf.keras.losses.MeanSquaredError()
                loss = mse(gt, out)

                loss_data.append(loss)

```

```
            grads = tape.gradient(loss, model.trainable_weights)

            optimizer.apply_gradients(zip(grads, model.
               trainable_weights))

            if (iteration + 1) % config.checkpoint_ep == 0:
                message = ''
                if loss < min_loss:
                    min_loss = loss.numpy()
                    model.save_weights(os.path.join(config.
                       checkpoints_folder, "best.h5"))
                    print(' min loss: %.5f' % min_loss)
```

4.3.2.5 Test

Cette phase est utilisée pour faire des prédictions sur des nouvelles images de test, afin d'estimer et d'évaluer si le modèle fonctionne et a bien appris. D'estimer et d'évaluer le modèle s'il est performant et bien appris. Les fonctions utilisées vers la phase de prédiction sont présentées comme suit :

— Chargez le modèle : nous chargeons le meilleur modèle entraîné avec les poids sauvegardés de l'étape précédente.

Liste 4.5 – Chargez le modèle

```
tf.flags.DEFINE_string("params", "best.h5", "Chemin vers les
   poids du modale")
```

— Prédicat : crée la fonction get_final (voir la liste 4.6).

Liste 4.6 – Prédicat

```
#Prediction finale du modele
def get_final(network, x_in):
    sb, sy, sx, sf = x_in.get_shape().as_list()
    y_predict = network.outputs

    # Highlight mask
    thr = 0.05
    alpha = tf.reduce_max(x_in, reduction_indices=[3])
    alpha = tf.minimum(1.0, tf.maximum(0.0, alpha-1.0+thr)/thr)
    alpha = tf.reshape(alpha, [-1, sy, sx, 1])
    alpha = tf.tile(alpha, [1,1,1,3])

    # linearisees entree et prediction
```

```
    x_lin = tf.pow(x_in, 2.0)
    y_predict = tf.exp(y_predict)-1.0/255.0

    # Melange alpha
    y_final = (1-alpha)*x_lin + alpha*y_predict

    return y_final
```

— Visualisation des résultats : nous avons deux types de résultats à être démontrés. Le premier est le tracé de courbe (voir la liste 4.7) qui représentent le développement de métrique au cours du processus d'apprentissage (Loss et nombre de Epoch). Deuxième résultat des données prédites(voir la liste 4.8), que nous générons à partir d'image LDR.

Liste 4.7 – Code pour tracé le courbe de Loss par à pour nombre des epochs

```
def print_loss(y,arr):
    x =np.arange(0,len(arr),1).tolist()
    y =arr
# plotting
    plt.xlabel("Epochs")
  plt.xlim([0,100])
    plt.ylabel("Loss")
  plt.xlim([0,1])
    plt.plot(x, y, color="red")
    plt.savefig("loss.png",format="png")
    plt.show()
```

Liste 4.8 – Visualisation des résultats

```
   # Ecrire une image HDR en utilisant OpenEXR
def writeEXR(img, file):
    try:
        img = np.squeeze(img)
        sz = img.shape
        header = OpenEXR.Header(sz[1], sz[0])
        half_chan = Imath.Channel(Imath.PixelType(Imath.PixelType.HALF
           ))
        header['channels'] = dict([(c, half_chan) for c in "RGB"])
        out = OpenEXR.OutputFile(file, header)
        R = (img[:,:,0]).astype(np.float16).tostring()
        G = (img[:,:,1]).astype(np.float16).tostring()
        B = (img[:,:,2]).astype(np.float16).tostring()
        out.writePixels({'R' : R, 'G' : G, 'B' : B})
        out.close()
```

```
15 # tone mapping
16 def writehdr(img, file, exposure=0):
17
18     sc = np.power(np.power(2.0, exposure), 0.5)
19
20     try:
21         scipy.misc.toimage(sc*np.squeeze(img), cmin=0.0, cmax=1.0).
               save(file)
```

4.4 Résultats

Dans cette section, nous présentons les résultats obtenus pour le modèle de reconstruction HDR en utilisant une seule exposition basée.

Le premier résultat, après lancer l'apprentissage, on a tracé le courbe de fonction loss en fonction de nombre des epochs, On a choisi 400 epochs(voir la figure 4.2). Nous remarquons dans le diagramme que la valeur de Loss est proche de zéro et la valeur finale est Loss= 0.0014.

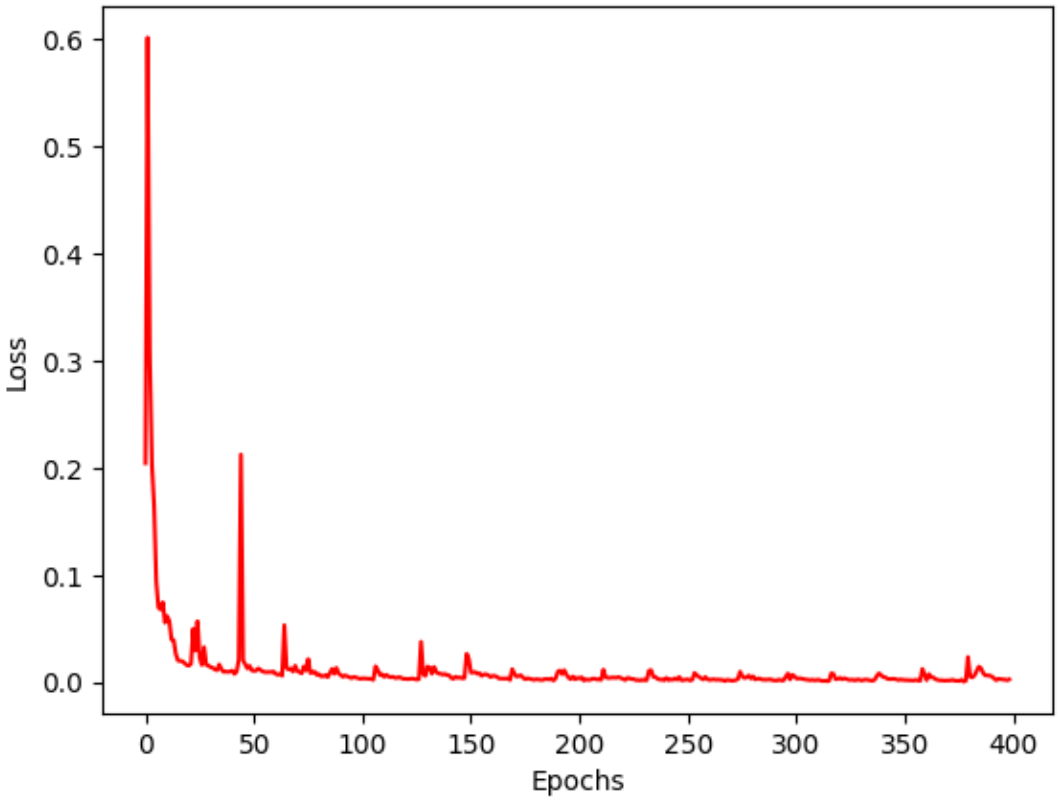


FIGURE 4.2 – Illustration des coefficients Loss de notre apprentissage.

Deuxième résultat, après lancer les tests de prédiction des images LDRs, on a obtenu les résultats HDR et on a utilisé une seule exposition :

(a)

(b)

FIGURE 4.3 – (a) Image LDR d'entrée, (b) Le résultat HDR

La figure suivante 4.4 montre une comparaison entre quelques objets de l'image LDR et de résultat HDR. On observe que la qualité des images HDRs est plus élevé que les images LDRs.

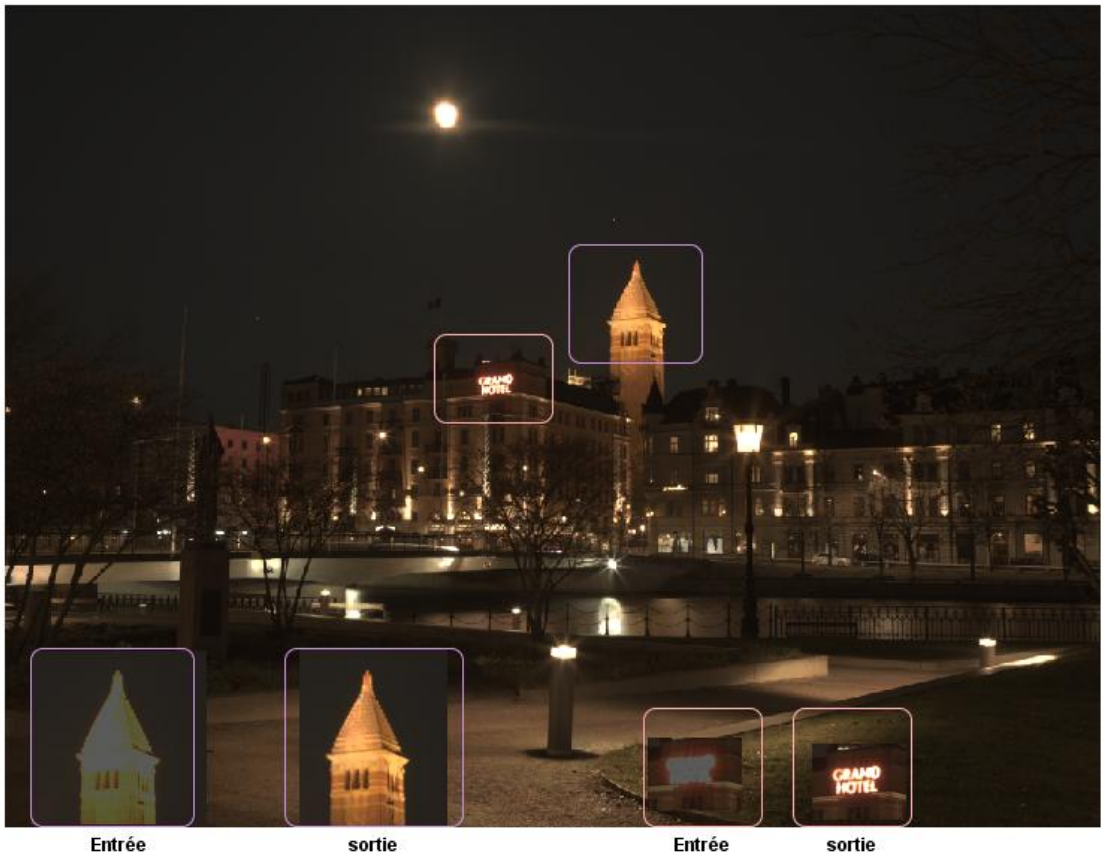


FIGURE 4.4 – Comparaison entre les objets d'image LDR et HDR

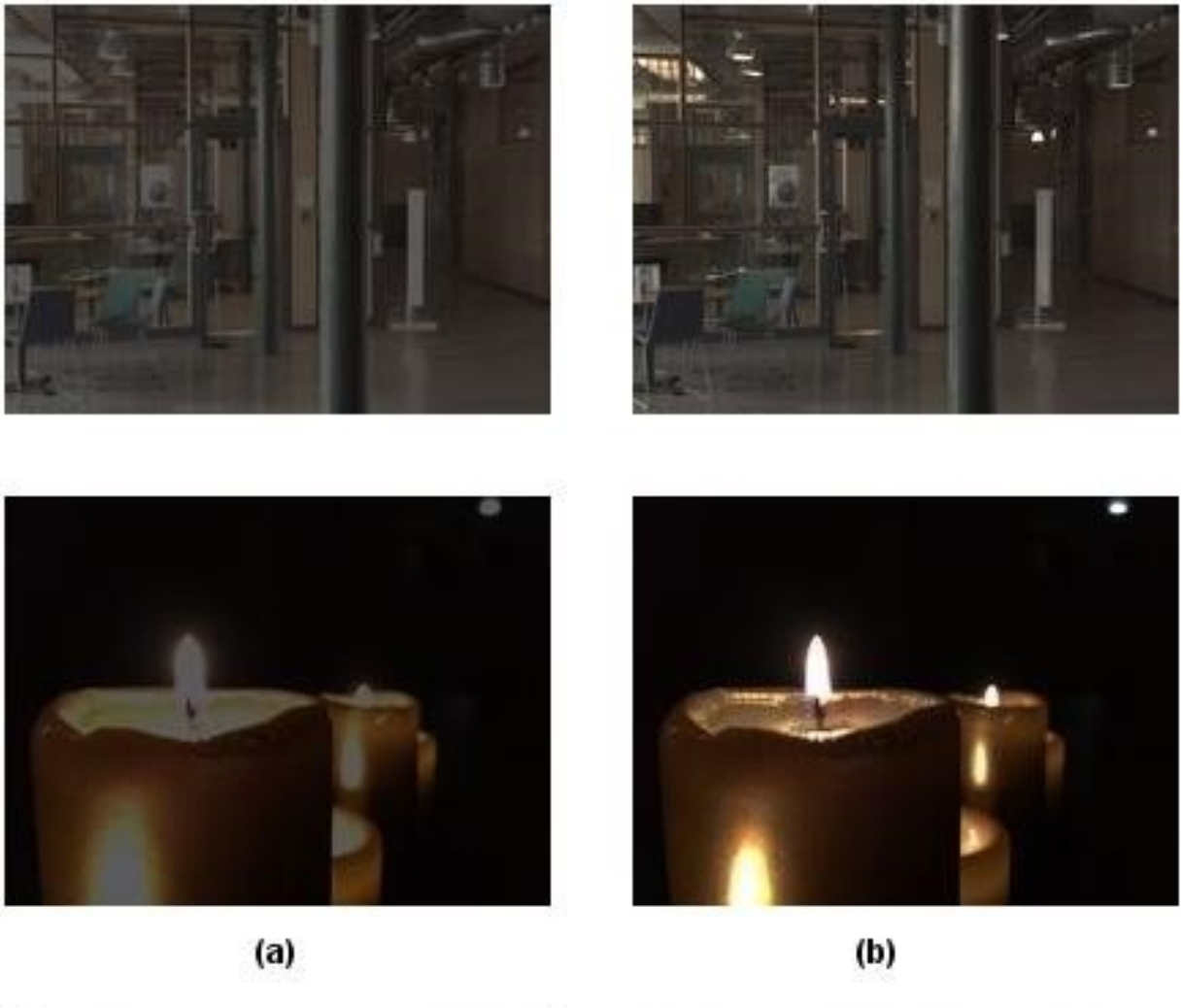


FIGURE 4.5 – Comparaison entre (a) image LDR et (b) résultat HDR

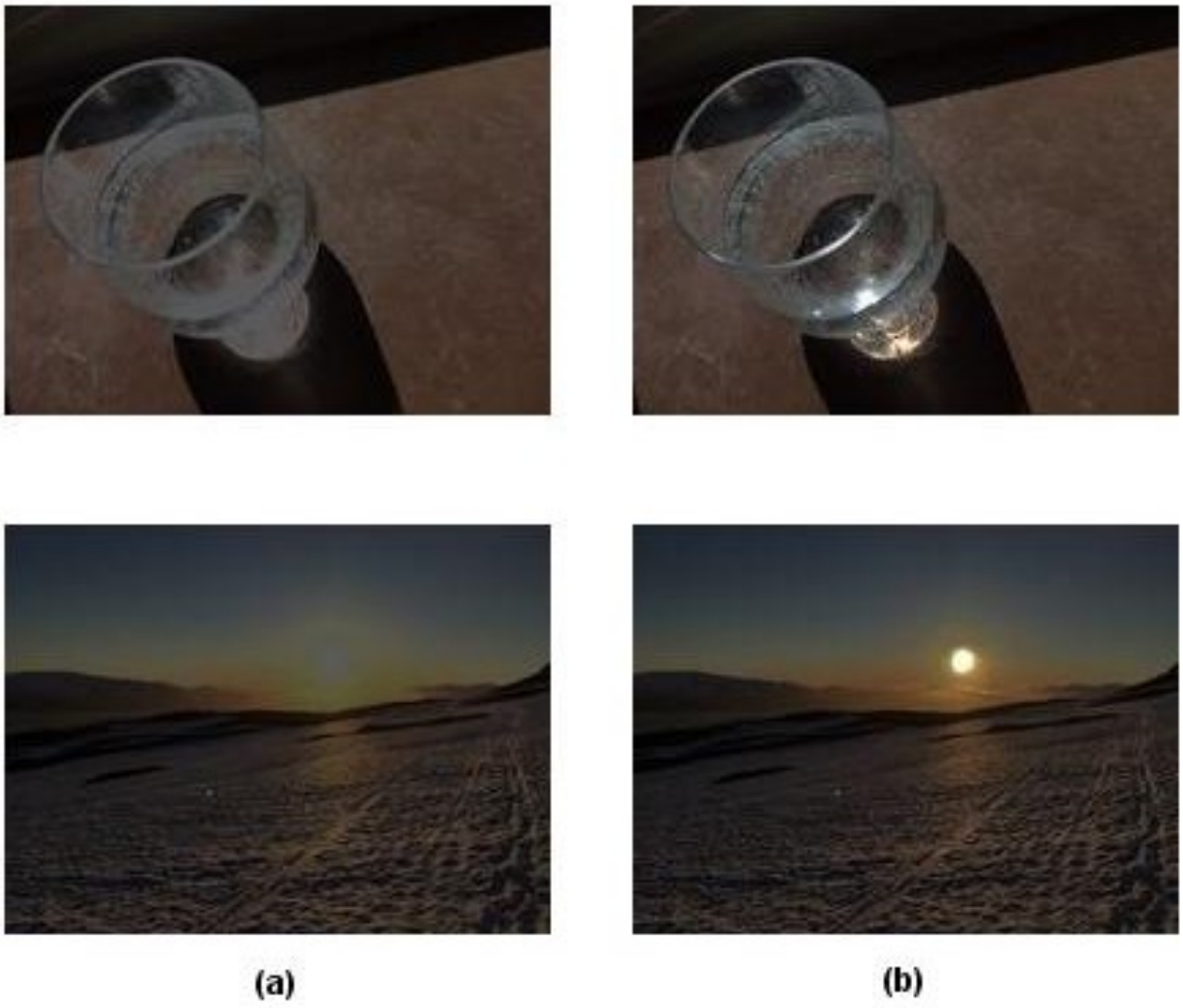


Figure 4.6 – Comparaison entre (a) image LDR et (b) resultat HDŕ

4.5 Conclusion

Dans ce chapitre, nous avons décrit l'implémentation détaillée de notre système HDR-CNN avec les outils utilisés pour développer de ce système, à partir de préparation les donnes et construire modale HDR-CNN , nous avons présenté les résultats des images HDRs après l'apprentissage.

Conclusion Générale

Les images HDRs(HDR images, High Dynamic Range) ouvrant un nouveau domaine de concurrence pour les fabricants des systèmes pour générer des images HDRs à partir des images basse gamme dynamique (LDRs, Low Dynamic Range). Dans ce mémoire, Nous avons présenté un système de reconstruction HDR basé sur le deep learning.

Premièrement, Notre système de reconstruction HDR génère des images HDR à partir des images LDR basé sur le réseaux de neurones convolutifs (CNN, Convolutional Neural Networks), il utilise l'architecture VGG-19, ce système est devisé en deux parties, partie d'apprentissage qui possède comme entrée la base de données composé des images LDRs et des images HDR correspondant au système entraîner, la partie test ou prédiction qui à partir des images LDRs, on génère les images HDR correspondantes.

Notre système a donné des bons résultats au niveau de la reconstruction HDR et d'illustration des coefficients Loss de notre apprentissage. D'après les résultats obtenu, nous remarquons que la valeur Loss est très proche de zéro, la valeur finale est Loss : 0.0014541384298354387.

L'avantage de la méthode programmée est que nous avons utilisé une seule exposition d'image LDR. et elle donne des bons résultats en terme de qualité.

Nous espérons dans l'avenir de continuer notre recherche dans ce domaine pour générer des vidéos HDRs.

Bibliographie

[1] Josselin Petit. “Génération, visualisation et évaluation d’images HDR: application à la simulation de conduite nocturne”. Thèse de doct. Université Claude Bernard-Lyon I, 2010.

[2] J Bullier. “Architecture fonctionnelle du système visuel”. In : *Vision: aspects perceptifs et cognitifs* (1998), p. 11-42.

[3] Alexandre Benoit. “Le système visuel humain au secours de la vision par ordinateur”. Thèse de doct. Institut National Polytechnique de Grenoble-INPG, 2007.

[4] H Kolb et al. *Webvision: The Organization of the Retina and Visual System-Photoreceptors*. 1996.

[5] Ana Radonjić et al. “The dynamic range of human lightness perception”. In : *Current Biology* 21.22 (2011), p. 1931-1936.

[6] William JK Beaudot. “Le traitement neuronal de l’information dans la rétine des vertébrés: Un creuset d’idées pour la vision artificielle”. Thèse de doct. Grenoble INPG, 1994.

[7] John Simon Werner et Lothar Spillmann. *Visual perception: The neurophysiological foundations*. Academic press, 1990.

[8] Hannah E Smithson. “Sensory, computational and cognitive components of human colour constancy”. In : *Philosophical Transactions of the Royal Society B: Biological Sciences* 360.1458 (2005), p. 1329-1346.

[9] H-W Bodmann et M La Toison. “Predicted brightness-luminance phenomena”. In : *International Journal of Lighting Research and Technology* 26.3 (1994), p. 135-143.

[10] NORMAND BALLARD. “La photo HDR”. In : ().

[11] Cambodge Bist. “Combining aesthetics and perception for display retargeting”. Thèse de doct. Rennes 1, 2017.

[12] Benoit Paissard et Olivier Ponsonnet. “Images HDR et rendu HDR”. In : ().

[13] MUSTAPHA BOUDERBANE. “Systèmes de vision á grande dynamique auto-adaptables”. In : (2020).

[14] Erik Reinhard et al. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010.

[15] Alain Simac-Lejeune. “Points d’intérêt dans les vidéos HDR”. In : *COnférence en Recherche d’Information et Applications*. 2012.

[16] Adrien Gruson. *Tone Mapping and white balancing for walkthrough*. 2011.

[17] Y. ni, Y. Z. Bogdan Arion et Potet. “120 db wdr cmos intensified camera for night vision. OPTRO21012.” In : (2012).

[18] D. J. Griffiths et A Wicks. *High speed high dynamic range video.* (2017).

[19] Shanmuganathan Raman et Subhasis Chaudhuri. “A matte-less, variational approach to automatic scene compositing”. In : *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, p. 1-6.

[20] David J Griffiths et Alfred Wicks. “High speed high dynamic range video”. In : *IEEE Sensors Journal* 17.8 (2017), p. 2472-2480.

[21] S Mann et RW Picard. *MI of TPC Section, On, being’undigital’with digital cameras: Extending dynamic range by, combining differently exposed pictures*. 1995.

[22] Paul E Debevec et Jitendra Malik. “Recovering high dynamic range radiance maps from photographs”. In : *ACM SIGGRAPH 2008 classes*. 2008, p. 1-10.

[23] Ruben Mayer et Hans-Arno Jacobsen. “Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques and Tools”. In : *CoRR* abs/1903.11314 (2019). arXiv : 1903.11314. url : http://arxiv.org/abs/1903.11314.

[24] Franck Martins. *AI Edge Computing – New Paradigm for IoT*.

[25] Franck Martins. *Obstacle Avoidance in Self driving Cars*.

[26] Khelil Hathem. *Digital pathology image analysis with deep learning*. 2020.

[27] Nour el houda MAHDJOUB. *Deep learning for the segmentation of MRIbrain tissues*. 2020.

[28] Olivier Petit. *Segmentation sémantique d’images médicales 3D pardeep learning*. 2018.

[29] Sindhuja Kotala et al. *Automatic Colorization of Black and White Images using Deep Learning*. April 2019.

[30] Bart Mesman et Henk Corporaal Maurice Peemen. *Efficiency Optimi-zation of Trainable Feature Extractors for a Consumer Platform*. August 2011.

[31] Yoshua Bengio Razvan Pascanu Tomas Mikolov. *On the difficulty oftraining recurrent neural networks*. 2013.

[32] Ignacio Hernandez et Christian García Ricardo Crespo Claudio Alvarez. *Aspatially explicit analysis of chronic diseases insmall areas : acase study ofdiabetes in Santiago*. Chile 2020.

[33] Patrick GARDA acques-Olivier KLEIN Hubert PUJOL. *Simulation dela machine de Boltzmann en temps réel*. 1993.

[34] Kirill Eremenko. *Deep Learning A-Z™: Autoencoders - Stacked Autoen-coders*. Aug. 22, 2018.

[35] Gabriel Eilertsen et al. “HDR image reconstruction from a single exposure using deep CNNs”. In : *ACM transactions on graphics (TOG)* 36.6 (2017), p. 1-15.

[36] Nima Khademi Kalantari et Ravi Ramamoorthi. “Deep high dynamic range imaging of dynamic scenes.” In : *ACM Trans. Graph.* 36.4 (2017), p. 144-1.

[37] Ce Liu et al. “Beyond pixels: exploring new representations and applications for motion analysis”. Thèse de doct. Massachusetts Institute of Technology, 2009.

[38] Kenta Moriwaki et al. “Hybrid loss for learning single-image-based HDR reconstruction”. In : *arXiv preprint arXiv:1812.07134* (2018).

[39] Zeeshan Khan, Mukul Khanna et Shanmuganathan Raman. “FHDR: HDR Image Reconstruction from a Single LDR Image using Feedback Network”. In : *arXiv preprint arXiv:1912.11463* (2019).

[40] Yu-Lun Liu et al. “Single-image hdr reconstruction by learning to reverse the camera pipeline”. In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, p. 1651-1660.

[41] SALMI MOHAMED AMINE BOUZID. *Reconnaissance de caractères de sous-titres vidéo en utilisant l’apprentissage automatique*. 2020.

[42] Muhammad Mateen et al. “Fundus image classification using VGG-19 architecture with PCA and SVD”. In : *Symmetry* 11.1 (2019), p. 1.

[43] A. A. e. S. Amidi. “CS 230 – Apprentissage profond : Réseaux de neurones Convolutionnels”. In : *Université de stanford* (2019).

[44] Philippe Giguère. “Apprentissage par Réseau de neurones profonde,cours Introduction Fonctions d’activation”. In : (2020).

[45] Medjdoubi Abdelkader. *L’ANALYSE DU SENTIMENT UTILISANT LE DEEP LEARNING*. 2019.

[46] DIALLO Nene Adama Dian. *La reconnaissance des expressions faciales*. Juillet 2019.

[47] Chaima KIHEL. *Analyse des sentiments en utilisant l’apprentissage Profond: Cas de la langue Arabe*. 2020.