Estimation de poses humaines par apprentissage profond : application aux passagers des véhicules autonomes

Romain Guesdon

Université Lumière Lyon 2
Ecole Doctorale : ED 512 Informatique et Mathématiques
LIRIS

Estimation de poses humaines par apprentissage profond : application aux passagers des véhicules autonomes

Romain GUESDON

Informatique

Directrice de thèse : Laure TOUGNE RODET
Co-encadrant de thèse : Carlos CRISPIM-JUNIOR

Soutenue publiquement le 16/02/2024

Composition du jury :
Vincent FREMONT, Professeur des Universités, École Central Nantes, *Rapporteur*
François BREMOND, Directeur de recherches, INRIA Sophia Antipolis, *Rapporteur*
Alice CAPLIER, Professeur des Universités, Grenoble INP, *Examinatrice*
Renaud MARLET, Directeur de recherches, École des Ponts ParisTech, *Examinateur*
Xuguang WANG, Directeur de recherches, Université Gustave Eiffel, *Examinateur*
Laure TOUGNE RODET, Professeur des Universités, Université Lyon 2, *Directrice de thèse*
Carlos CRISPIM-JUNIOR, Maître de conférences, Université Lyon 2, *Co-encadrant*

Remerciements

Je remercie tout d'abord mes encadrants Laure et Carlos pour m'avoir accompagné tout au long de cette thèse. Leur aide, leur expertise scientifique et leur patience ont été des éléments précieux pour la réalisation de ces travaux et de ce manuscrit.

Je remercie également les permanents du LIRIS Lyon 2 et les doctorants présents et passés. Les différents moments et échanges, scientifiques ou non, partagés lors des déjeuners, groupes de travail et autres *afterworks* ont grandement contribué à la bonne ambiance quotidienne. Une pensée particulière pour Dev, Jonas et Mehdi avec qui j'ai partagé un bureau pour les *brainstormings* et autres discussions.

Je souhaite également remercier ma famille et mes proches pour leur soutien sans faille tout au long de cette aventure. Ils ont été pour moi une grande source de motivation qui m'a permis de mener à bien cette thèse.

Enfin, merci à la région Auvergne-Rhône-Alpes pour avoir financé ces travaux de recherche.

Résumé

La recherche concernant les voitures autonomes a fortement progressé ces dernières décennies, en se concentrant particulièrement sur l'analyse de l'environnement extérieur et sur les tâches liées à la conduite. Cela a permis une importante croissance de l'autonomie des véhicules particuliers. Dans ce nouveau contexte, il peut être pertinent de s'intéresser aux passagers de ces véhicules autonomes afin d'étudier le comportement de ces derniers face à cette révolution du moyen de transport. C'est pour approfondir ces thématiques que le projet région AURA AutoBehave a été mis en place. Ce projet réunit plusieurs laboratoires menant des recherches dans différentes disciplines scientifiques liées à cette thématique telles que la vision par ordinateur, la biomécanique, les émotions ou encore l'économie des transports. Cette thèse menée au laboratoire LIRIS s'inscrit donc dans ce projet, dans laquelle nous nous intéressons aux méthodes d'estimation de poses humaines des passagers par apprentissage profond. Nous avons d'abord étudié les solutions de l'état de l'art, et avons développé un jeu de données ainsi qu'une métrique plus adaptée aux contraintes de notre contexte. Nous nous sommes également intéressés à la visibilité des points afin d'aider l'estimation de la pose. Par la suite, nous nous sommes attaqués à la problématique de généralisation de domaine pour l'estimation de poses dans le but de proposer une solution efficace dans des conditions inconnues. Ainsi, nous nous sommes intéressés à la génération de données synthétiques de passagers pour l'estimation de poses afin de combler le manque de jeux de données annotés disponibles dans notre contexte. Nous avons étudié l'application de réseaux génératifs ainsi que de méthodes modélisation 3D à notre problématique. Nous nous sommes appuyés sur ces données pour proposer différentes stratégies d'entraînement et deux nouvelles architectures. L'approche par fusion proposée associée aux stratégies d'entraînement permet de tirer profit de jeux de données génériques et de jeux de données spécifiques, afin d'améliorer les capacités de généralisation des méthodes d'estimation de poses à l'intérieur d'une voiture, en particulier sur le bas du corps.

Abstract

Research into autonomous cars has made great strides in recent decades, focusing particularly on analysis of the external environment and driving-related tasks. This has led to a significant increase in the autonomy of private vehicles. In this new context, it may be relevant to take an interest in the passengers of these autonomous vehicles, to study their behavior in the face of this revolution in the means of transport. The AURA AutoBehave project has been set up to explore these issues in greater depth. This project brings together several laboratories conducting research in different scientific disciplines linked to this theme, such as computer vision, biomechanics, emotions, and transport economics. This thesis carried out at the LIRIS laboratory is part of this project, in which we focus on methods for estimating the human poses of passengers using deep learning. We first looked at state-of-the-art solutions and developed both a dataset and a metric better suited to the constraints of our context. We also studied the visibility of the keypoints to help estimate the pose. We then tackled the problem of domain generalisation for pose estimation to propose an efficient solution under unknown conditions. Thus, we focused on the generation of synthetic passenger data for pose estimation. Among other things, we studied the application of generative networks and 3D modeling methods to our problem. We have used this data to propose different training strategies and two new network architectures. The proposed fusion approach associated with the training strategies makes it possible to take advantage of both generic and specific datasets, to improve the generalisation capabilities of pose estimation methods inside a car, particularly on the lower body.

Table des matières

Chapitre 1

Introduction

Dans ce premier chapitre, nous décrivons le contexte dans lequel se place cette thèse. Nous introduisons d'abord les problématiques liées aux véhicules autonomes. Nous présentons ensuite le projet AutoBehave ainsi que les différents laboratoires membres. Enfin, nous décrivons les contraintes associées à cette thèse et le cadre de recherche dans lequel nous nous plaçons.

Sommaire

1.1 Contexte

Le monde automobile connaît depuis ces dernières décennies un important développement de l'automatisation des véhicules. Afin de mesurer ce niveau d'automatisation, une classification définie par la *SAE International* (*Society of Automotive Engineers*) est généralement utilisée (SAE International, 2023). Cette taxonomie définit six niveaux d'autonomie de conduite, 0 étant une conduite entièrement manuelle et 5 une gestion du véhicule complètement autonome (Figure 1.1). Ainsi, dès les années 2000, des systèmes comme le régulateur de vitesse ou l'antiblocage des roues (ABS) ont été implantés dans la majorité des véhicules particuliers, permettant d'atteindre le niveau 1 d'autonomie. Beaucoup de voitures moyennes et haut de gammes possèdent aujourd'hui une autonomie de niveau 2. Ce niveau permet une conduite partiellement autonome sur des grands axes, rendue possible par des outils de respect des distances de sécurité, de maintien de cap ou encore de lecture des panneaux.

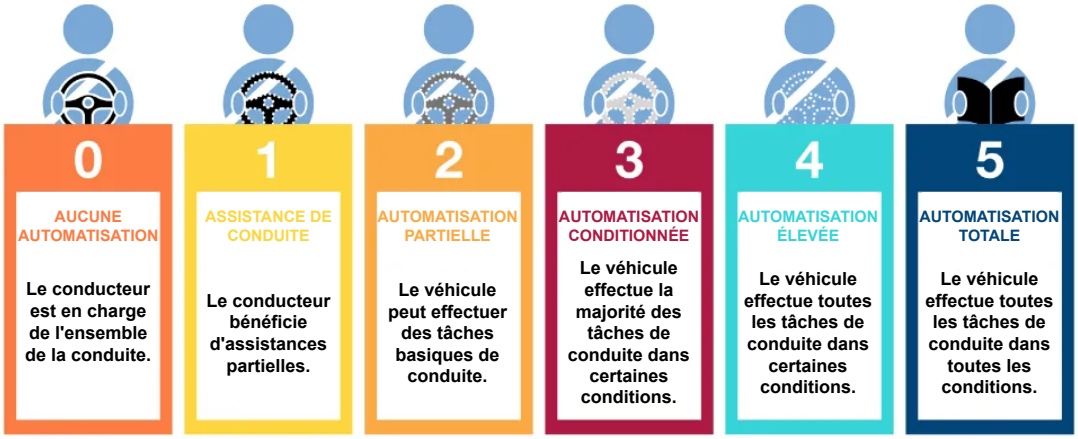


FIGURE 1.1 – Les différents niveaux d'autonomie définis par la SAE. [1]

Cependant, un grand nombre de constructeurs automobiles se sont lancés dans le développement de véhicules avec une autonomie de niveau 4 ou 5, c'est-à-dire des véhicules capables d'effectuer toutes les tâches de conduite. On peut par exemple citer Navly, une navette électrique proposée par l'entreprise Navya (Navya, 2023), (Figure 1.2-gauche). Ce véhicule entièrement autonome a circulé dans le quartier de Confluence à Lyon entre 2016 et 2020, permettant à des utilisateurs d'effectuer des trajets sans intervention d'un conducteur. D'autres exemples d'applications pour les véhicules individuels sont les systèmes de conduite autonome développés par Waymo (Waymo, 2023) ou Tesla (Tesla, 2023) (Figure 1.2-droite). Ces systèmes donnent au véhicule la capacité de naviguer sans intervention du conducteur sur des autoroutes, voire en ville aux États-Unis (cette fonctionnalité est actuellement limitée en dehors des États-Unis pour des raisons techniques et légales).

Dans ce contexte, la recherche concernant les véhicules autonomes s'est principalement portée sur le développement des méthodes de résolution de la tâche de conduite autonome. Entre autres, dans le domaine de la vision par ordinateur, une attention particulière a été

1. Illustration : `https://ackodrive.com/car-guide/autonomous-cars-and-levels-of-autonomous-driving/`

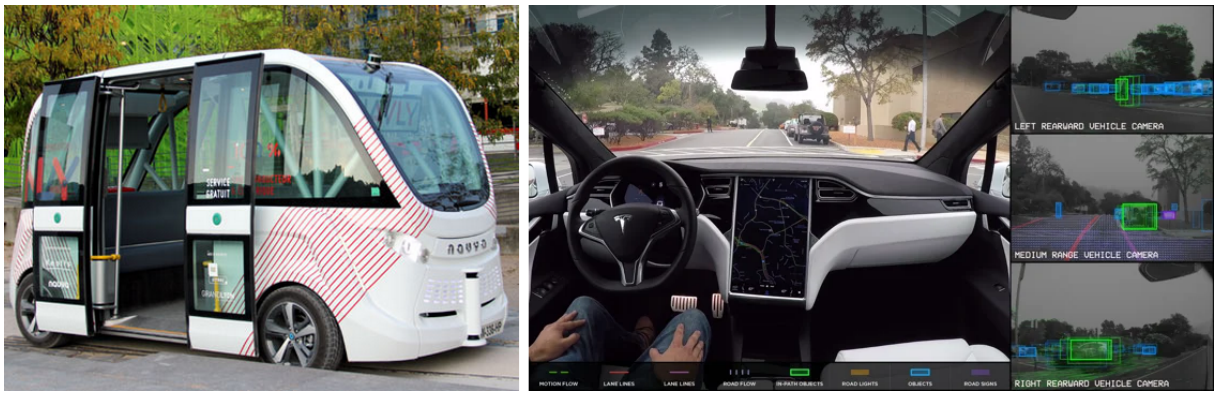


FIGURE 1.2 – Illustrations de la navette Navly (à gauche) et de l'*Autopilot* Tesla (à droite). [2]

portée sur l'analyse de l'environnement extérieur du véhicule, avec des tâches comme la détection d'obstacles, la reconnaissance des voies ou de la signalétique, ou encore la détection des autres véhicules ou des piétons (Janai et al., 2020; Parekh et al., 2022). L'objectif de ces recherches est d'améliorer l'autonomie et la fiabilité du véhicule pour aboutir à la conduite dans tout environnement sans intervention humaine, tout en garantissant la sécurité aussi bien des passagers que des personnes extérieures.

Pour autant, il peut être pertinent de s'intéresser aux passagers de ces véhicules autonomes. En effet, la nouvelle approche du transport individuel apportée par les voitures autonomes pose un certain nombre de questions sur la perception et la réaction des usagers. Comment réagirons-nous à la conduite d'un tel véhicule à 50, voire 130 km/h ? Qu'effectuera le "conducteur" à la place de la conduite ? Dans quelles activités sera réinvesti ce temps libéré ? Quelles postures prendront les passagers de ces nouveaux types de véhicules ? Quels seront les impacts économiques de ces nouvelles activités ? Les aménagements intérieurs pourraient-ils alors être adaptés pour un meilleur confort de vie à bord ?

1.2 Projet AutoBehave

Le projet AutoBehave, pour "*AUTOmatic analysis of BEHAVEviors in autonomous vehicles*" (AutoBehave, 2023), cherche à répondre à ces questions soulevées dans la section précédente. AutoBehave est un projet de recherche visant à étudier les comportements et les émotions à l'intérieur des véhicules autonomes individuels. Ce projet financé par la région Auvergne-Rhône-Alpes regroupe plusieurs laboratoires couvrant différents domaines scientifiques entourant cette thématique :

- LIRIS (Laboratoire d'InfoRmatique en Images et Systèmes d'information) : ce laboratoire en informatique s'intéresse à différentes thématiques telles que l'intelligence artificielle, l'analyse de données volumineuses, la vision par ordinateur, la cybersécurité, la transformation digitale ou l'apprentissage humain. En particulier, l'équipe participant à ce projet est l'équipe Imagine spécialisée dans la vision par ordinateur et les l'apprentissage automatique. Dans le projet AutoBehave, le LIRIS cherche

2. Illustrations : `https://www.transbus.org/dossiers/navettes-autonomes.html` ; `https://www.tesla.com/autopilot`

à étudier les méthodes de traitement de l'image et d'apprentissage profond pour l'analyse du comportement des passagers tels que leur pose ou leurs activités.

- LBMC (Laboratoire de Biomécanique et Mécanique des Chocs) : le LBMC étudie des thématiques de recherche autour de la biomécanique humaine telles que la biomécanique des chocs, la mécanique des structures, la biomécanique des tissus, l'ergonomie physique, la biomécanique du mouvement, etc. En particulier, le LBMC s'intéresse dans ce projet à l'analyse des mouvements, de l'assise et de la posture des passagers, ainsi que leur modélisation.
- LAET (Laboratoire Aménagement Économie Transports) : ce laboratoire est spécialisé sur les questions des transports, de la mobilité et des territoires, et en particulier sur les aspects économiques et d'aménagement.
- LESCOT (Laboratoire Ergonomie et Sciences Cognitives pour les Transports) : le LESCOT étudie des domaines tels que l'analyse de l'état émotionnel et des activités des passagers dans un véhicule, afin de mieux appréhender leurs comportements et interactions.

L'implication de ces différents laboratoires dans ce projet permet d'étudier différents verrous scientifiques concernant les passagers de véhicules autonomes. Tout d'abord, l'enregistrement de la posture à l'intérieur du véhicule grâce à des caméras, mais aussi d'autres capteurs tels que des marqueurs infrarouges, ainsi que son analyse (LIRIS / LBMC). Ensuite, l'étude des actions et activités réalisées par les passagers ainsi (LIRIS / LBMC / LESCOT), ainsi que l'analyse des impacts économiques qu'ils pourraient induire sur les moyens de transports à venir (LAET). À ces laboratoires s'associe également DEMS, un bureau d'étude industriel spécialisé dans le transport routier. L'objectif de ce dernier est de s'appuyer sur les observations et analyses des différents acteurs d'AutoBehave afin de proposer le cahier des charges de l'hypothétique habitacle d'un véhicule autonome de demain.

Étant donné que ces problématiques ont été peu explorées dans la littérature, il est nécessaire de rassembler une grande quantité de données sous différentes modalités. Ainsi, un des objectifs du projet AutoBehave est de constituer plusieurs jeux de données, dont un jeu de données multimodal composé de vidéos enregistrées dans l'habitacle de véhicules autonomes. Ce jeu de données rassemblera des informations sur la posture et les activités des participants, mais aussi des informations sur leurs ressentis et émotions résultants de questionnaires. Une grande partie de ces données a été collectée au cours d'une expérimentation en conditions réelles sur le campus de Nantes, décrite dans l'Annexe A. L'ensemble des données collectées requiert cependant un traitement afin d'en extraire les données utiles, en particulier pour les enregistrements vidéo. Ainsi, un autre objectif du projet AutoBehave est le développement d'outils pour l'analyse automatique du comportement des passagers. Ces outils pourront permettre d'extraire des données telles que la posture, l'activité, l'attention ou encore l'émotion des passagers du véhicule à travers l'analyse d'images et de vidéos.

1.3 Contraintes et verrous scientifiques de la thèse

C'est dans le contexte du projet AutoBehave que se place cette thèse. Elle s'est déroulée au sein du LIRIS, plus précisément dans l'équipe Imagine. L'objectif de cette thèse est de proposer une solution permettant l'analyse automatique de la posture, ou "pose",

du conducteur d'un véhicule autonome à partir d'images afin d'apporter différentes informations pour les études de comportement du projet. Initialement, une partie de cette étude devait s'appuyer sur le jeu de données constitué suite à l'expérimentation du projet AutoBehave. Cependant, des contraintes organisationnelles causées en partie par la crise de la Covid a entraîné un important retard dans l'acquisition de ces données. Nous avons donc dû nous appuyer sur des données préexistantes pour la réalisation de cette thèse.

Les travaux existants de l'état de l'art concernant l'étude de passagers de voitures s'intéressent principalement à l'étude de comportements liés à la conduite tels que la détection du regard (Khan and Lee, 2019), de l'attention (Halin et al., 2021) ou encore de signes d'endormissements (Ramzan et al., 2019). Ils se basent donc principalement sur l'analyse d'images enregistrées par une caméra placée face au conducteur permettant une bonne vue sur le visage. Cette approche ne convient pas à notre problématique puisque l'objectif est d'obtenir des informations utiles aux autres membres du projet et en particulier au LBMC s'intéressant à l'étude de la biomécanique. Il est donc nécessaire d'obtenir une analyse de la totalité du corps, ce qui requiert un angle de vue de côté.

FIGURE 1.3 – Illustration d'une acquisition couleur (à gauche) et profondeur (à droite) (une colorisation de l'image de profondeur est appliquée pour aider la visualisation).

La spécificité du contexte dans lequel se place ce projet réduit grandement les données à notre disposition. En effet, plusieurs contraintes nous sont imposées. Premièrement, le type de caméras utilisées. Deux modalités de capture d'images sont habituellement utilisées : l'image couleur, ou encore image RGB (*Red Green Blue*), et l'image de profondeur (Figure 1.3). L'image couleur permet d'obtenir des informations sur la luminosité, la couleur et la texture de la scène. À l'inverse, l'image de profondeur permet d'obtenir des indications sur la structure des objets et le placement dans la scène. L'image de profondeur dans notre contexte peut par exemple être utilisée pour faciliter la détection et la segmentation du passager dans l'habitacle, et en particulier des membres de la personne. L'utilisation d'une caméra particulière est cependant nécessaire pour acquérir ces images et nous ne disposions pas de telles données au commencement de notre étude. L'utilisation d'informations de profondeur n'a donc pas été considérée dans ces travaux.

Deuxièmement, l'étude de la pose peut être effectuée à partir d'images ou de clips vidéo. L'utilisation de vidéos présente l'avantage d'apporter une information supplémentaire qui est la cohérence temporelle. Cependant, il n'existait pas à notre connaissance de jeu de données vidéo disponible dans notre contexte avec la pose annotée. En outre, l'acquisition et l'annotation de données vidéo sont plus longues que celles d'images simples, et nous disposons d'une faible capacité d'annotation. Nous avons donc décidé de nous concentrer sur la détection de poses sur des images.

Enfin, il est possible d'exprimer la pose observée en trois dimensions (3D) ou en deux

dimensions (2D), cette dernière représentant la projection dans la perspective de la caméra. L'étude de la posture en 3D est une approche plus naturelle puisqu'elle permet de prendre en compte la structure globale du corps. De plus, la pose 3D est plus facilement interprétable puisqu'elle ne dépend pas du positionnement de la caméra. Cependant, les annotations associées sont souvent difficiles à obtenir. En effet, la capture fiable de modèles 3D nécessite l'utilisation d'équipements spécifiques tels que plusieurs caméras, des marqueurs, des caméras de profondeur, etc. (Figure 1.4). Cela demande la mise en place d'une plate-forme de capture ; ce qui est difficile dans un environnement restreint tel que l'intérieur d'une voiture, à moins d'utiliser un simulateur de conduite ouvert en intérieur.

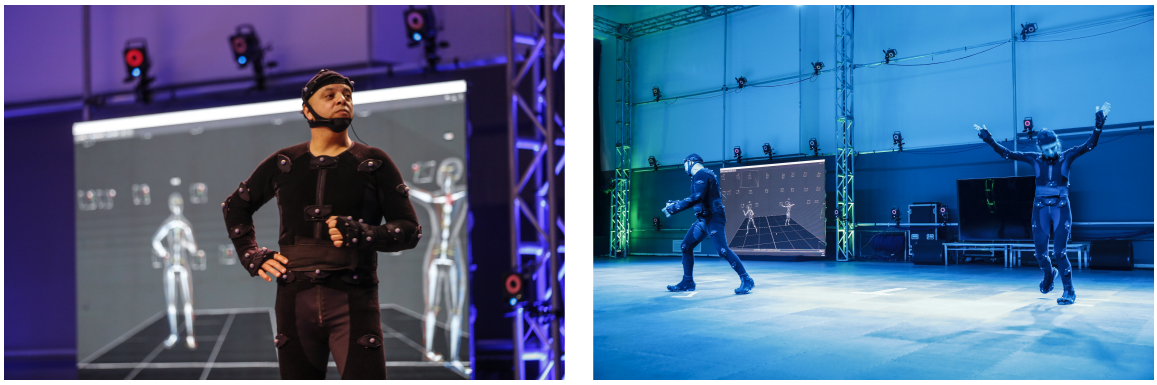


FIGURE 1.4 – Exemple d'installation pour la capture de pose 3D utilisant des marqueurs. [3]

Ainsi, les travaux de cette thèse se portent sur l'étude de la pose humaine 2D sur des images couleur. Nous étudions l'efficacité des solutions de l'état de l'art pour la détection de pose dans le contexte de l'intérieur d'un véhicule autonome. Les difficultés particulières apportées par notre contexte sont l'angle de vue atypique apportant entre autres une importante occultation de certaines parties du corps, le bruit visuel dû à l'environnement extérieur et le faible contraste dans l'habitacle. Nous avons décidé de nous concentrer sur les solutions s'appuyant sur l'apprentissage profond. En effet, bien que des méthodes pour la détection de pose 2D utilisant des approches géométriques classiques existent (Andriluka et al., 2009; Yang and Ramanan, 2011), l'utilisation de l'apprentissage profond depuis ces dernières années a montré de bien meilleures performances et représente aujourd'hui la grande majorité des solutions de l'état de l'art (Munea et al., 2020). L'utilisation d'approches par apprentissage profond est cependant dépendante du nombre et de la qualité des données à disposition. Il est donc nécessaire d'apporter une attention particulière à l'impact du faible nombre de données disponibles dans notre contexte et de trouver des solutions pour combler ce manque. De plus, la solution proposée devra être robuste aux variations de conditions que l'on peut observer entre différentes expérimentations, et ce, sans nécessiter d'annoter et d'entraîner à nouveau. Une attention particulière est donc à prêter aux capacités de généralisation des méthodes utilisées.

Ce manuscrit est organisé en 5 chapitres :

- Le Chapitre 2 consiste en une revue de la littérature concernant les différentes notions scientifiques liées à nos travaux : les réseaux de neurones artificiels et leur optimisation, ainsi que les différents aspects associés à l'estimation de poses hu-

3. Illustrations : `https://flic.kr/p/2nk4e5f` - `https://flic.kr/p/2nk1N3U`

maines tels que les jeux de données, les différentes architectures existantes et les métriques d'évaluation des performances.

- Le Chapitre 3 présente l'étude de l'état de l'art dans notre contexte et plusieurs contributions pour répondre aux lacunes de la littérature. Nous introduisons ainsi notre jeu de données DriPE pour l'estimation de poses de conducteur, une nouvelle métrique d'évaluation mAPK ainsi qu'un métamodèle pour prédire la visibilité des parties du corps.
- Nous étudions dans le Chapitre 4 des méthodes d'augmentation de données pour la généralisation, en particulier la génération d'images synthétiques par réseaux génératifs et par modélisation 3D.
- Enfin, le Chapitre 5 décrit l'application de méthodes de généralisation de domaine afin de tirer profit de différentes sources de données telles que différentes stratégies d'entraînement et architectures.

Chapitre 2

État de l'art

Ce chapitre présente l'état de l'art des différentes notions abordées dans ces travaux. Nous introduisons tout d'abord l'apprentissage profond avec les réseaux de neurones et leur optimisation, ainsi que des architectures de référence. Nous présentons ensuite les différents aspects de l'estimation de poses humaines 2D : les méthodes de modélisation du corps, les jeux de données et les réseaux de l'état de l'art ainsi que les métriques utilisées pour l'évaluation.

Sommaire

2.1 Réseaux de neurones artificiels

2.1.1 Perceptron

Les réseaux de neurones artificiels actuels tirent leur origine d'un modèle mathématique inspiré des neurones biologiques. Une cellule neuronale est composée de trois parties principales : les dendrites, le corps cellulaire et l'axone (Figure 2.1 - gauche) (F.R.C., 2022). Les dendrites jouent le rôle de récepteurs pour le neurone. Chacune d'entre elles reçoit des signaux électriques en provenance d'un ou plusieurs autres neurones, qu'elle transmet alors au corps cellulaire. Si le corps cellulaire reçoit suffisamment de stimulations électriques, il génère un signal électrique. Ce nouveau signal est alors transmis par l'axone à d'autres neurones. La capacité décisionnelle d'un neurone biologique est donc très sommaire. Cependant, l'interconnexion de plusieurs milliards permet la prise de décisions complexes.

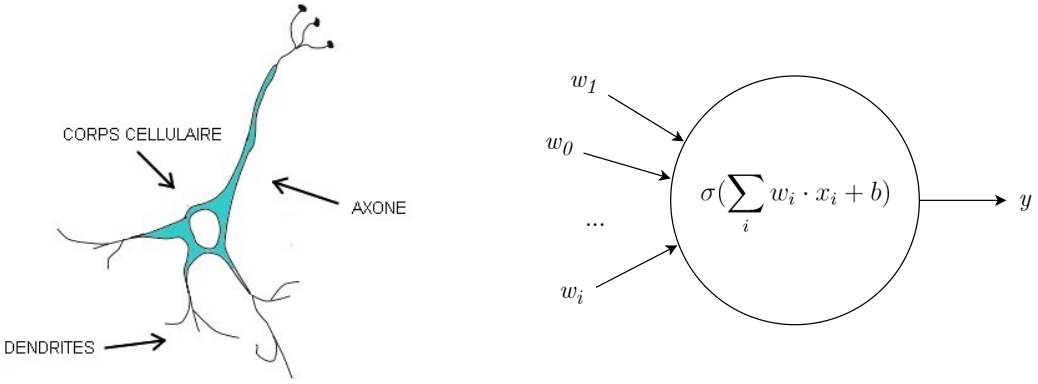


FIGURE 2.1 – Schéma d'un neurone biologique [4](à gauche) et d'un perceptron (à droite).

Suivant une logique similaire, Rosenblatt a proposé en 1958 un modèle mathématique de neurone artificiel (Rosenblatt, 1958) : le perceptron (Figure 2.1 - droite). Cette modélisation est composée d'une ou plusieurs entrées, sur lesquelles sont appliqués des poids. Une valeur fixe (appelée biais) peut aussi être ajoutée. Si la somme de ces entrées dépasse un certain seuil, la sortie du perceptron vaut 1, 0 sinon. Mathématiquement, le perceptron peut être défini par la fonction suivante :

$$y = \sigma(\sum_i w_i \cdot x_i + b) \tag{2.1}$$

où x_i sont les entrées i, w_i les poids associés, b le biais et y la sortie du perceptron. La fonction de seuil σ est plus généralement appelée fonction d'activation.

À l'instar du neurone biologique, le perceptron est un modèle simple qui possède une capacité limitée. En effet, sa composition le cantonne à la modélisation de fonctions linéaires avec une fonction d'activation, et rend impossible la représentation de fonctions non-linéaires.

2.1.2 Réseau de neurones multicouche

En suivant la logique biomimétique, un modèle combinant plusieurs perceptrons a donc été proposé pour permettre de répondre à des problèmes plus complexes (Rosenblatt,

4. Illustration : `https://commons.wikimedia.org/wiki/File:Neurone_biologique.JPG`

1958). Cette architecture est nommée perceptron multicouche (ou MLP pour *MultiLayer Perceptron* en anglais).

Un MLP est constitué de plusieurs rangées de neurones appelées couches. On discerne trois types de couches : la couche d'entrée, la couche de sortie et les couches intermédiaires ou couches cachées. En dehors de la première couche, chaque neurone reçoit en entrée un vecteur contenant l'ensemble des sorties des neurones de la couche précédente, appelé vecteur de caractéristiques. Les couches successives sont donc totalement connectées (*fully connected*, ou FC), comme illustré dans la Figure 2.2. Le nombre de neurones artificiels dans les couches d'entrées et de sorties est défini par la dimension de la variable d'entrée et de sortie du MLP. Cependant, le nombre de couches cachées et de neurones dans chacune de ces couches n'est pas fixe, ce sont ces hyperparamètres qui vont définir l'architecture du réseau. Le réseau le plus simple est composé d'une seule couche intermédiaire, et plus des couches sont ajoutées, plus il est dit "profond".

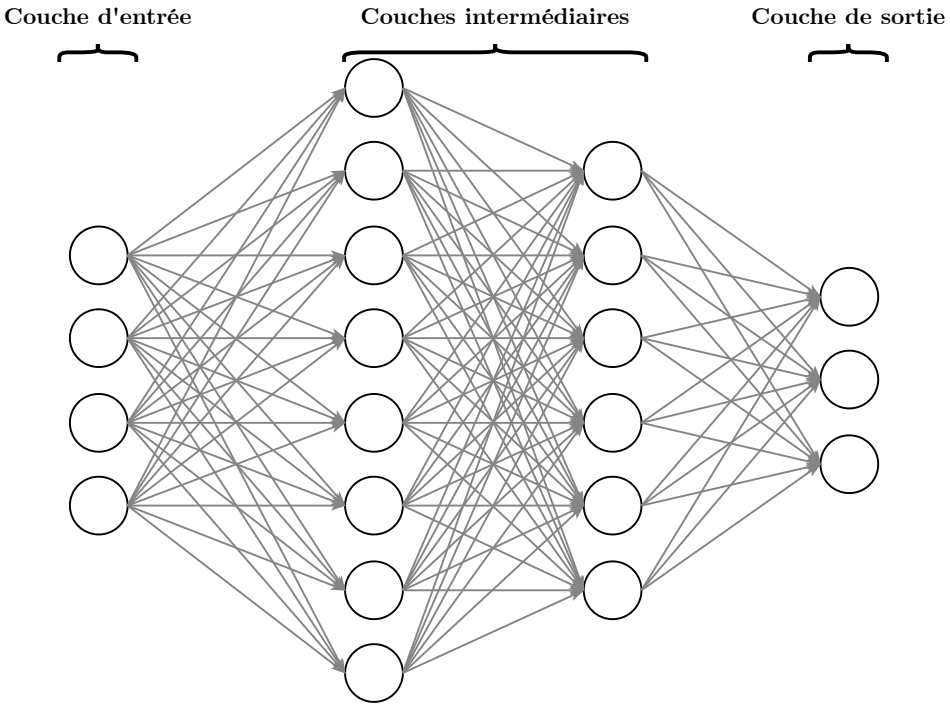


FIGURE 2.2 – Schéma d'un réseau perceptron multicouche (MLP) à deux couches intermédiaires.

Dans cette configuration, la fonction d'activation du perceptron est primordiale puisqu'elle permet de casser la linéarité du réseau (en présupposant que la fonction utilisée n'est pas linéaire). En effet, sans fonction d'activation, l'ensemble d'un réseau de neurones multicouche peut-être factorisé en un seul perceptron, ce qui fait perdre la complexité du réseau. Au cours de l'évolution des réseaux de neurones artificiels, différentes fonctions d'activation ont été utilisées, pour permettre une propagation différente de l'information (Jagtap and Karniadakis, 2022). Les plus présentes dans l'état de l'art récent sont représentées dans la Figure 2.3.

La structure du MLP permet de modéliser des fonctions bien plus complexes qu'un neurone unique. Une application classique est la classification. En considérant un vecteur d'entrée de dimension n noté $X_k = \{x_1, ..., x_n\}$, on cherche à déterminer la classe y_k dans laquelle se trouve ce vecteur, avec $y_k \in Y = \{y_1, ..., y_c\}$. Pour ce faire, on peut utiliser

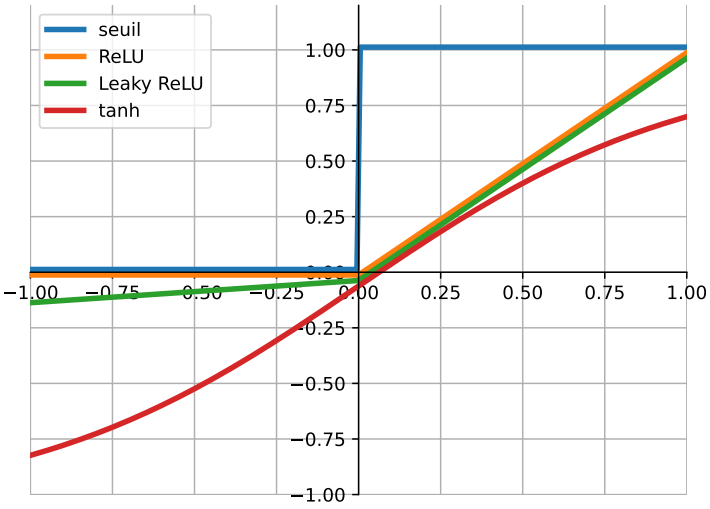


FIGURE 2.3 – Courbes de différentes fonctions d'activation

un MLP à n neurones d'entrées et c neurones de sortie. Chaque neurone de la couche de sortie est associé à une classe et produit un score indiquant la pertinence du vecteur d'entrée pour cette classe. Puisque rien ne force théoriquement le réseau de neurones à produire une distribution de probabilité, une fonction softmax est appliquée à la sortie $\{\hat{y_1}, ..., \hat{y_c}\}$ afin de normaliser les scores pour obtenir une somme égale à 1. Cette fonction est définie comme :

$$\text{Sigmoïde}(\hat{y_i}) = p(y_k = y_i|X) = \frac{e^{\hat{y_i}}}{\sum_{j=1}^{c} e^{\hat{y_j}}} \tag{2.2}$$

Cette méthode de classification peut théoriquement être appliquée sur toute modalité pouvant être représentée par un vecteur de dimension finie (signal sonore, image, texte, etc.) (Druzhkov and Kustikova, 2016; Li et al., 2020; Bhattacharya et al., 2021). En pratique cependant, la dimension de l'entrée est limitée par les capacités de calcul. Par exemple, le jeu de données MNIST (LeCun and Cortes, 2010) contient des images pour la reconnaissance de chiffres manuscrits en niveau de gris de 28 par 28 pixels, ce qui nécessite donc $28 * 28 = 784$ paramètres d'entrée. Un MLP peut être utilisé pour résoudre cette tâche de classification, et permet d'obtenir des performances correctes (Vijendra et al., 2016). Cependant, une image couleur aujourd'hui peut avoir une taille 1280 par 720 pixels, avec trois canaux RGB , soit une couche d'entrée de $1280 * 720 * 3 = 2\,7642$ k paramètres. Or, dans un réseau entièrement connecté avec une entrée de dimension n, chaque neurone de la première couche cachée possède $n + 1$ paramètres. De plus, dans un MLP utilisé pour la classification, la pratique est de graduellement diminuer le nombre de neurones par couche cachée pour passer du nombre d'entrées au nombre de sorties. Ainsi, un MLP dimensionné pour classer des images couleurs de grande taille possédera plusieurs milliards de paramètres. L'optimisation d'un tel réseau nécessite des ressources de calcul très importantes.

Une autre limitation du MLP est la conversion des données d'entrée en vecteur à une dimension. Ainsi, pour une image en niveau de gris possédant deux dimensions (horizon-

tale et verticale), deux pixels quelconques ont théoriquement la même relation que deux pixels voisins vis-à-vis d'un neurone de la première couche puisque tous les neurones sont liés entre eux. On a donc une perte de l'information spatiale de l'image.

Afin de répondre à ces problématiques, des méthodes plus appropriées au traitement de telles données à haute dimensionnalité ont été développées.

2.1.3 Réseau de neurones convolutif

Le premier réseau de neurones développé initialement pour traiter des images et répondre aux problématiques évoquées dans les paragraphes précédents est le Neocognitron (Fukushima, 1980). De manière similaire aux réseaux de neurones, cette architecture s'inspire d'observations faites sur l'anatomie du cerveau, et plus particulièrement sur le cortex visuel. En effet, deux types de couches y sont définis, reprenant le principe de fonctionnement de deux familles de cellules :

- les cellules S (ou *Simple cells*), qui réagissent aux traits et bordures avec une orientation et une localisation particulière,
- les cellules C (ou *Complex cells*), qui sont aussi sensibles à des formes orientées spécifiquement, mais sans contrainte de placement.

Une succession de ces deux types de couches permet d'effectuer des tâches comme de la reconnaissance de caractères, en détectant les formes caractéristiques de chaque lettre.

Cependant, la majorité des méthodes modernes de traitement d'images par apprentissage profond se basent sur un outil mathématique plus simple et généralisable : la convolution. Appliquer une convolution sur une image I revient à calculer la somme pondérée du voisinage de chaque pixel et créer une nouvelle image avec ces valeurs. Les poids utilisés sont stockés dans une matrice N appelée noyau de convolution, ou "filtre", et les mêmes coefficients sont appliqués sur la totalité de l'image. Mathématiquement, le produit de convolution pour un pixel de coordonnées (a, b) s'exprime comme :

$$(I * N)(a, b) = \sum_i \sum_j I(a, b) \cdot N(a - i, b - j) \tag{2.3}$$

Opération	**Identité**	**Flou moyen**	**Flou gaussien (approximation)**	**Filtre laplacien**
Filtre	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
Résultat	 *			

TABLEAU 2.1 – Exemples d'applications de filtres par convolution [5]. Notation : image initiale * (correspondant à la sortie de l'identité).

5. Illustrations : `https://en.wikipedia.org/wiki/Kernel_(image_processing)`

En faisant varier la taille du noyau et les valeurs de ses poids, on peut alors réaliser un certain nombre de filtres, comme illustré dans le Tableau 2.1 : détection de contours, amélioration de netteté, flou, etc.

Les réseaux convolutifs sont basés sur ce principe de filtres de convolution. L'un des premiers réseaux de neurones à utiliser cette méthode est LeNet (LeCun et al., 1998). Ce réseau, développé pour la reconnaissance de caractères, contient trois couches de convolutions. Dans chaque couche, les poids des noyaux de convolutions sont appris, comme les poids d'un neurone de perceptron. Ces couches sont paramétrables de plusieurs manières (Figure 2.4) :

- la taille du noyau de convolution, qui est le plus souvent carré et de largeur impaire K, afin de pouvoir être centré,
- le pas de déplacement (*stride*) S du noyau sur la matrice d'entrée.
- l'ajout de bords (*padding*) P. Plusieurs options sont possibles pour choisir la valeur de ces pixels de *padding* : copie des bords de l'image, symétrie, valeur unique, etc.

Ces paramètres permettent de modifier le comportement de la couche de convolution ou la dimension du vecteur de sortie. Par exemple, pour préserver la taille de l'image d'entrée, il est nécessaire d'utiliser un pas égal à 1. Cependant, si l'on souhaite utiliser un noyau avec une taille supérieure à 1, il est nécessaire d'ajouter un *padding*. Ainsi, pour un noyau de taille $K = 5$, un *padding* $P = (K - 1)/\,2 = 2$ est nécessaire de chaque côté (si K est impaire).

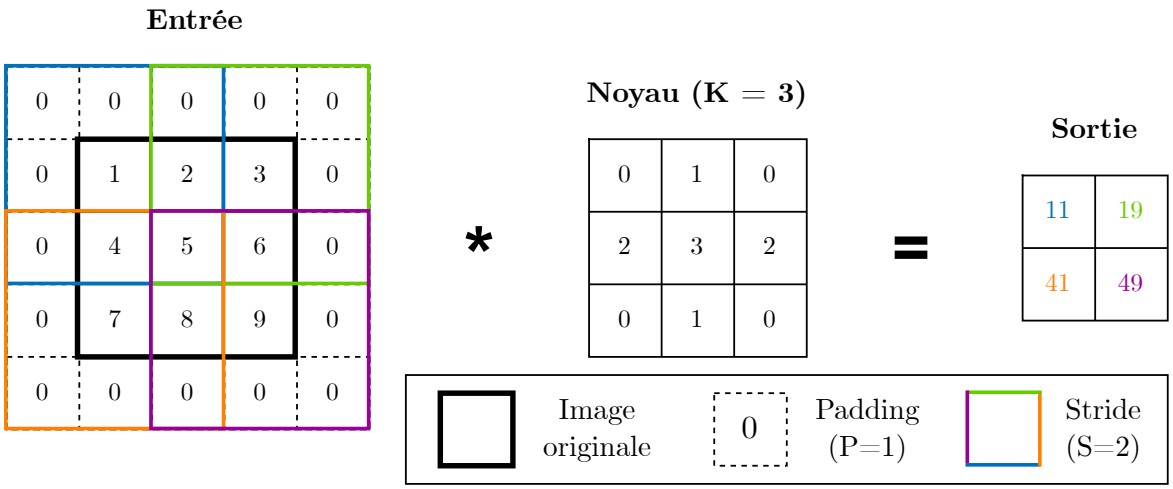


FIGURE 2.4 – Illustration de l'application d'une convolution, avec $K = 3$, $P = 1$ et $S = 2$

Cependant, pour pouvoir extraire le maximum d'informations, C noyaux différents sont appris en parallèle et indépendamment pour chaque couche de convolution. Puisque chaque filtre génère sa propre carte de caractéristiques, on a donc C cartes en sortie de la couche. Ces cartes sont appelées canaux, chaque canal étant une matrice partageant la même taille avec les autres canaux d'une même couche. Par exemple, une couche de convolution avec $C = 3$ peut permettre de générer une image couleur RGB puisqu'elle est composée de 3 canaux. Les trois couches de convolution du réseau LeNet possèdent respectivement 6, 16 et 120 canaux.

En plus des couches de convolution, des couches intermédiaires appliquant des fonctions non linéaires comme une fonction maximum (*Max Pooling*) ou une fonction moyenne (*Average Pooling*) sont ajoutées. Utiliser un pas supérieur à 1 a pour effet de réduire la

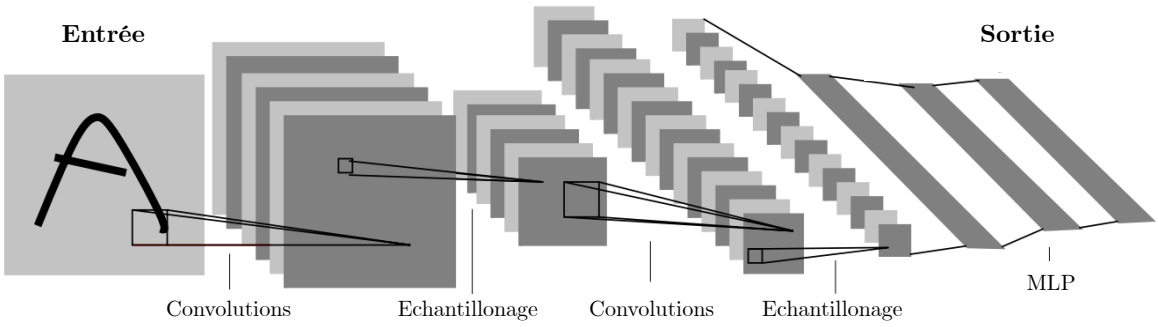


FIGURE 2.5 – Architecture du réseau convolutif LeNet (LeCun et al., 1998).

taille de la matrice, autrement dit de l'échantillonner. L'utilisation de ces couches permet de concentrer l'information utile en éliminant les coefficients des valeurs de caractéristiques à faibles valeurs, et de traiter l'information spatiale à différentes échelles grâce à la réduction de la taille des matrices. Cette diminution de la dimension des vecteurs de caractéristiques permet également d'économiser des ressources de calcul, ou de pouvoir augmenter le nombre de canaux des couches suivantes tout en limitant l'augmentation de la taille des vecteurs de caractéristiques.

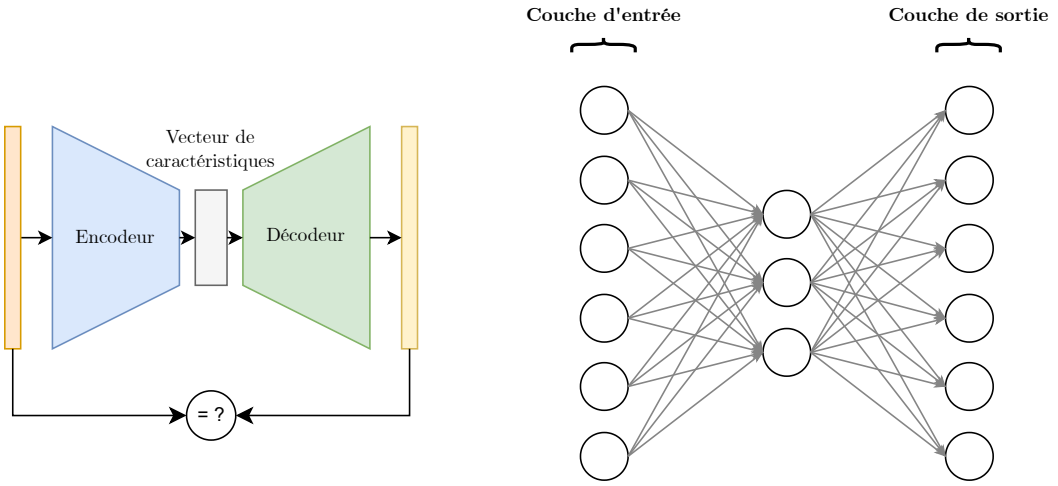
Par ailleurs, on peut observer que l'architecture de LeNet (Figure 2.5) se divise en deux parties distinctes. La première partie composée des couches de convolutions et de leurs couches annexes est appelé "extracteur de caractéristiques". Comme son nom l'indique, son rôle est d'extraire un vecteur de caractéristiques, qui sera utilisé dans le deuxième bloc pour résoudre une tâche précise. Dans le cas de LeNet, ce deuxième bloc est composé d'une couche totalement connectée suivie d'une couche sigmoïde, puis une couche de "fonction de base radiale" qui calcule la distance entre le vecteur d'entrée et des vecteurs prédéfinis représentant chaque caractère à détecter. Dans des implémentations plus modernes de LeNet, ce deuxième bloc est remplacé par trois couches totalement connectées utilisées pour estimer les probabilités de classification. Cette structure en deux parties permet de réutiliser une architecture d'extracteur de caractéristiques pour une tâche différente en modifiant simplement le bloc de sortie.

Enfin, d'une manière similaire aux perceptrons, une couche d'activation est ajoutée en complément de chaque couche de convolution. Cette couche joue le même rôle que dans un MLP en appliquant sur chaque valeur du vecteur de caractéristiques issu de la couche de convolution la fonction d'activation, afin d'éviter que l'ensemble du réseau soit linéaire. Au total, le réseau LeNet utilise trois couches de convolution + ReLU, intercalées de couches *MaxPooling* (Figure 2.5). L'avantage principal de cette architecture pour des tâches de classification d'images est son rapport nombre de poids / performances. En effet, le réseau est constitué de 60k paramètres, mais atteint une justesse supérieure à 95 % sur de la reconnaissance de caractères manuscrits du jeu MNIST (les méthodes d'évaluation des réseaux seront présentées dans la Section 2.2.1.3). En comparaison, un MLP composé d'une couche cachée de taille 100 permet d'obtenir des performances semblables, mais possède alors près de 80k paramètres. Dans ce cas, l'écart est assez faible, puisque la tâche est simple et la dimension des images d'entrée est petite (28 par 28 pixels). Cependant, le nombre de paramètres d'un réseau convolutif n'augmente pas avec la définition des images d'entrée, puisque les mêmes noyaux de convolution sont simplement appliqués sur

plus de pixels. Cette observation n'est pas vraie pour un MLP ; ce qui rend l'utilisation de réseaux convolutifs pour des images de tailles plus conséquentes tout indiqué.

2.1.4 Auto-Encodeur

Les applications pour les réseaux de neurones présentées jusque-là sont des tâches dites supervisées, c'est-à-dire que le réseau est entraîné pour produire une réponse précise et connue pour chaque entrée. Cela nécessite deux choses : une tâche pouvant être modélisée par un système dont chaque entrée attend une sortie unique, et des données annotées. On peut cependant chercher à effectuer des tâches dont la réponse n'est pas unique ou est inconnue. Un exemple d'une telle tâche est le partitionnement de données où l'on cherche regrouper des données par similarité. Dans cette situation, la méthode d'entraînement supervisé présentée précédemment n'est pas applicable. Il existe alors des méthodes dites "non supervisées", dont l'une des plus populaires est l'utilisation de réseaux appelés "auto-encodeurs".



(a) Schéma de l'architecture globale d'un auto-encodeur.

(b) Exemple d'auto-encodeur totalement connecté.

FIGURE 2.6 – Schémas d'auto-encodeurs.

Un réseau de neurones auto-encodeur est un modèle entraîné pour reproduire la donnée en entrée (Rumelhart et al., 1985). Son architecture se divise en deux parties symétriques : l'encodeur et le décodeur (Figure 2.6-a). Grâce à sa structure de taille décroissante, l'encodeur f calcule une représentation de dimension inférieure de l'entrée sous la forme d'un vecteur de caractéristiques $v = f(x)$. Le décodeur g à l'inverse ré-augmente la taille pour revenir à la dimension de départ à partir du vecteur de caractéristiques. On cherche donc à résoudre l'équation :

$$g(f(x)) = x \tag{2.4}$$

L'architecture la plus simple d'auto-encodeur est un réseau totalement connecté avec les couches d'entrée et de sortie de taille identique et une seule couche intermédiaire plus petite (Figure 2.6-b). Cette structure en goulot d'étranglement (*bottleneck*) permet de forcer le réseau à identifier et encoder les informations structurantes de l'entrée nécessaires à la reconstruction de l'image. Une approche similaire est utilisée avec les réseaux convolutifs. Ainsi, une architecture d'auto-encodeur convolutif couramment utilisée dans

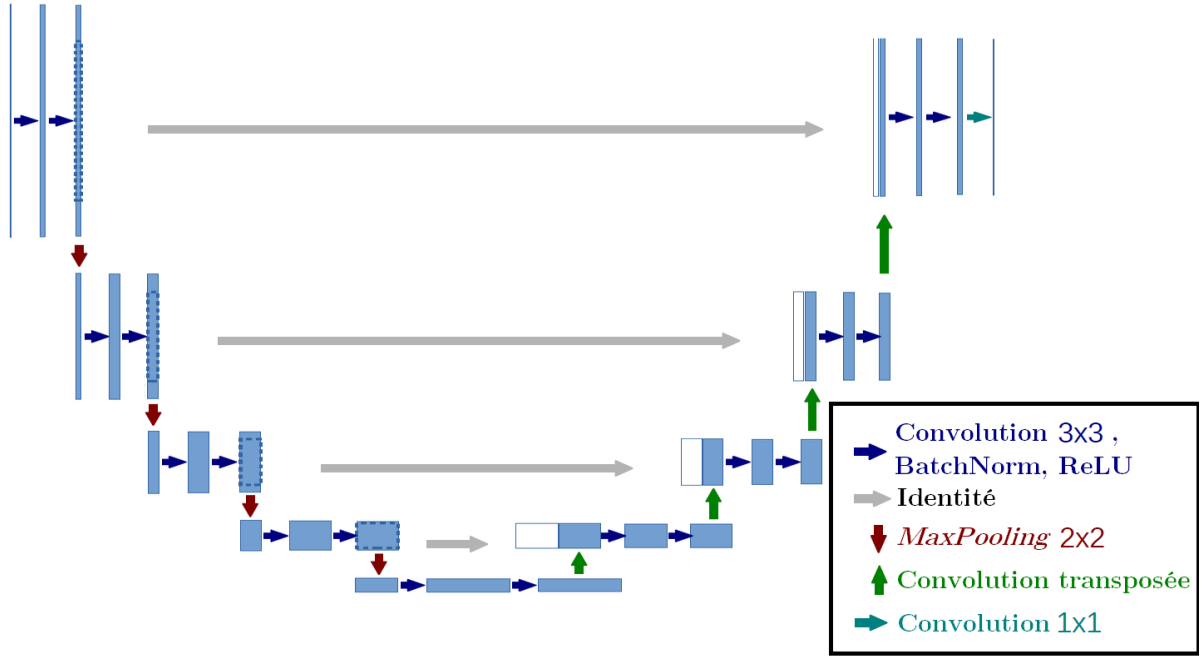


FIGURE 2.7 – Architecture de l'auto-encodeur convolutif U-Net (Ronneberger et al., 2015).

la littérature est le réseau U-Net (Ronneberger et al., 2015). Cette architecture prend une forme de "U" et est composée de plusieurs niveaux (Figure 2.7). À chaque niveau de l'encodeur, deux couches de convolution combinées chacune à une couche de *BatchNorm* et de ReLU extraient un vecteur de caractéristiques, qui passe ensuite dans une couche de *MaxPooling* pour réduire sa taille. Ce vecteur est alors donné en entrée du niveau suivant, ce qui permet d'extraire successivement les caractéristiques à différentes échelles. Le décodeur est constitué d'une structure similaire. Ainsi, à chaque niveau, deux couches convolutives suivies d'une couche de convolution transposée traitent le vecteur de caractéristiques qui est donc agrandit progressivement. En outre, pour faciliter l'entraînement et la conservation des caractéristiques entre les différentes échelles, la sortie de chaque niveau de l'encodeur est concaténée à l'entrée de chaque niveau équivalent du décodeur. Ce réseau à l'avantage d'adopter une structure permettant de calculer des caractéristiques à différentes échelles ; ce qui peut être utile pour la compréhension de la structure du corps par le réseau, tout en restant relativement simple et avec un nombre restreint de paramètres.

Entraîner un auto-encodeur à reconstruire les données d'entrée permet d'entraîner le réseau sans nécessiter de vérités terrain associées aux données, puisque l'erreur s'exprime comme la différence entre l'entrée et la sortie. L'intérêt est de pouvoir réutiliser cet encodeur entraîné comme extracteur de caractéristiques pour d'autres tâches. Par ailleurs, on retrouve couramment cette structure de *bottleneck* dans l'état de l'art dans l'optique d'extraire l'information utile (He et al., 2016; Newell et al., 2016; Zhu et al., 2017).

2.2 Optimisation des réseaux

Quelle que soit son architecture (par exemple MLP, réseau convolutif), un réseau de neurones est composé de paramètres, ou "poids", et a pour but de réaliser une ou plusieurs tâches définies. Bien que l'on pourrait en théorie définir manuellement ces poids, cela est en réalité difficile puisqu'ils dépendent généralement de la tâche, des données cibles, de l'architecture du réseau, etc. Nous décrivons dans cette partie les procédés et les outils utilisés pour la recherche de ces valeurs.

2.2.1 Étapes d'optimisation

L'optimisation d'un réseau de neurones se divise généralement en trois étapes : l'entraînement, la validation et l'évaluation. Nous décrivons dans cette section les spécificités de chaque phase.

2.2.1.1 Entraînement

La phase d'entraînement, ou d'apprentissage, est la phase visant à mettre à jour les poids du réseau dans le but d'effectuer une ou plusieurs tâches ciblées. Cette mise à jour s'appuie sur les données fournies et sur l'erreur effectuée par le réseau. Le but recherché est de converger vers les paramètres permettant de minimiser cette erreur. Plus précisément, l'entraînement d'un réseau de neurones se divise en quatre parties :

- propagation avant : les données d'entraînement sont passées une à une à travers le réseau afin d'obtenir une prédiction,
- calcul de l'erreur : une fonction de coût est utilisée pour mesurer l'erreur réalisée par le réseau,
- rétropropagation de l'erreur : en partant de l'erreur du réseau, les gradients de la fonction de coût au voisinage de chaque poids sont calculés,
- mise à jour des poids : les poids de chaque couche sont mis à jour en fonction du gradient calculé.

Tout d'abord, une inférence des données est réalisée en utilisant les poids actuels du réseau. Ces poids sont généralement initialisés aléatoirement en suivant une distribution spécifique avec des valeurs proches de 0 (Glorot and Bengio, 2010). En effet, initialiser tous les poids avec une constante telle que 0 ou 1 entraîne un gradient identique pour chaque poids sur une même couche, ce qui rend la mise à jour pour l'apprentissage moins efficace.

Une fois une prédiction $\hat{y}$ donnée par le réseau, une fonction de coût est utilisée pour mesurer l'erreur faite par le réseau. La fonction de coût utilisée dépend de la tâche recherchée et de la méthode d'entraînement (supervisée, non-supervisée, etc.). Les fonctions les plus couramment utilisées sont la distance arithmétique, la distance quadratique ou l'entropie croisée (Goodfellow et al., 2016).

À partir de l'erreur calculée, on cherche alors à estimer de combien faire varier chaque poids du réseau pour tendre vers la sortie attendue. Pour ce faire, l'algorithme de descente de gradient est utilisé. Cet algorithme calcule les dérivées partielles, ou "gradient", de la fonction de coût par rapport à chaque poids du réseau. Ce procédé nécessite donc que chaque fonction utilisée dans le réseau à entraîner soit différentiable (couches, fonctions d'activation, fonctions de coût, etc.). Cependant, plus les poids appartiennent à des

couches éloignées de la sortie du réseau, plus il est difficile de calculer directement la fonction de dérivée partielle puisqu'elle dépend de toutes les autres couches intermédiaires. Pour cette raison, les gradients sont calculés couche par couche en partant de la dernière, et le théorème de dérivation des fonctions composées est utilisé pour calculer le gradient dans une couche en fonction de la couche suivante, selon l'équation suivante :

$$\delta_j^{(k)} = \frac{\partial L}{\partial w_j^{(k)}} = \sigma'(z_j^{(k)}) * \sum_i w_i^{(k+1)} \delta_i^{(l+1)} \tag{2.5}$$

où L est la fonction de coût, $\delta_j^{(k)}$ le gradient du neurone j de la couche k, z_j^k la sortie de ce même neurone avant l'application de sa fonction de coût σ, et ∂ représente la dérivée partielle.

Finalement, les gradients calculés sont utilisés pour mettre à jour les poids du réseau, selon un facteur λ appelé taux d'apprentissage :

$$\hat{w}_j^{(k)} = w_j^{(k)} - \lambda \frac{\partial L}{\partial w_j^{(k)}} \tag{2.6}$$

Ce taux d'apprentissage permet de contrôler à quelle vitesse est mis à jour le poids. En effet, un gradient est calculé après chaque inférence. Utiliser un taux d'apprentissage à 1 dans l'équation 2.6 permet d'obtenir une erreur minimale pour la donnée d'entrée, mais n'assure pas de s'approcher du minimum sur l'ensemble des données. Il est donc important de choisir un taux d'apprentissage suffisamment grand pour converger rapidement vers le minimum de la fonction de perte, tout en s'assurant de ne pas tourner autour sans jamais s'en approcher, comme illustré dans la Figure 2.8.

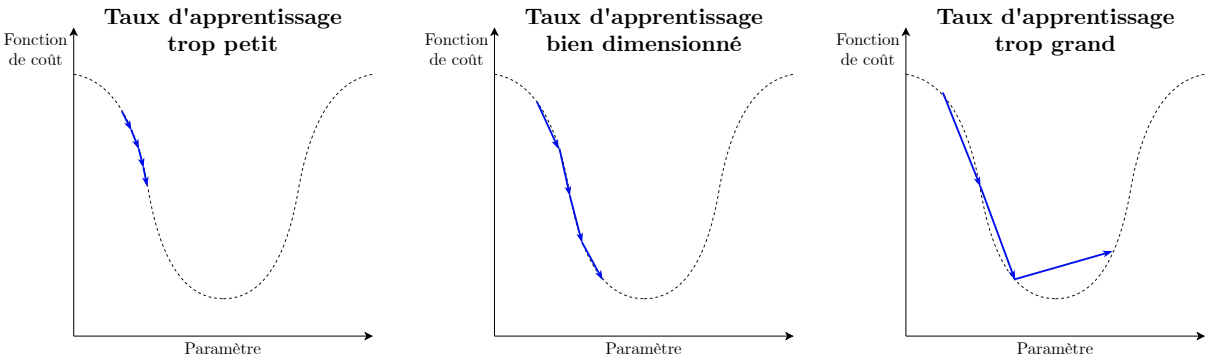


FIGURE 2.8 – Impact de différents taux d'apprentissage sur la descente de gradient.

Plusieurs travaux se sont intéressés à la recherche des minimaux locaux pour l'entraînement des réseaux de neurones (Choromanska et al., 2015; Kawaguchi, 2016). D'autres algorithmes ont aussi été proposés pour mettre à jour les poids. Parmi eux, on peut citer l'utilisation d'inertie (*momentum* (Rumelhart et al., 1985)) ou différentes méthodes basées sur le principe de taux d'apprentissage adaptatifs telles que AdaGrad (Duchi et al., 2011), RMSProp (Hinton et al., 2012) ou Adam (Kingma and Ba, 2014).

Comme montré dans l'équation 2.5, le calcul des gradients utilise la sortie z de chaque neurone. Il est donc nécessaire de conserver en mémoire tous les résultats intermédiaires lors de l'inférence pour effectuer la rétropropagation de gradient ; ce qui requiert une importante place en mémoire. Pour cette raison, il est difficile, voire impossible selon la taille du réseau et la taille du jeu de données, de calculer les gradients pour l'entièreté

du jeu de données en une fois. À la place, les images peuvent passer dans le réseau individuellement, ou par paquets appelés "*batchs*" ou "*mini-batchs*" (d'où l'appellation d'apprentissage stochastique, *Stochastic Gradient Descent* ou SGD). Les éléments sont généralement choisis aléatoirement et leur nombre par *batch* est choisi en fonction de la taille du réseau, des données d'entrée et des ressources matérielles à disposition. Une taille de *batch* plus importante permet un apprentissage plus rapide, mais il a été montré qu'une taille trop grande nuisait à la généralisation du réseau (Keskar et al., 2016).

Une passe de données est appelée "itération", et cette itération est répétée jusqu'à ce que toutes les données d'entraînement aient été proposées au réseau, ce qui correspond à une "*epoch*". Il est généralement nécessaire de faire voir plusieurs fois l'ensemble des données d'apprentissage au réseau, autrement dit l'entraîner sur plusieurs *epochs*, pour pouvoir converger vers l'erreur minimum. Le temps d'entraînement et les performances finales d'un réseau dépendent donc du choix du taux d'entraînement, du nombre d'*epochs*, etc. Tous ces paramètres liés à la stratégie d'entraînement sont appelés "hyperparamètres".

2.2.1.2 Validation

L'entraînement d'un réseau de neurones dépendant d'un grand nombre de facteurs, il est nécessaire de contrôler sa progression. Un premier indicateur est l'erreur calculée par la fonction de coût sur les images d'entraînement. Cependant, l'objectif de l'entraînement est d'obtenir un réseau capable de généraliser sur des données inconnues, bien que généralement proches des données d'entraînement. L'erreur sur ces données d'entraînement ne permet pas totalement de mesurer cette généralisation et peut même dissimuler un défaut appelé sur-apprentissage (ou *overfitting*) (Hawkins, 2004). En effet, la taille importante d'un réseau peut lui permettre d'intégrer trop d'informations sur les données d'entraînement (Figure 2.9). Le réseau est alors capable de restituer la réponse attendue pour chaque donnée ; ce qui lui permet d'obtenir une erreur nulle lors de l'entraînement, mais empêche toute généralisation sur des données inconnues. Ce comportement est particulièrement observé si le réseau est entraîné trop longtemps, avec un taux d'apprentissage trop fort ou avec trop peu de données vis-à-vis de la taille du réseau.

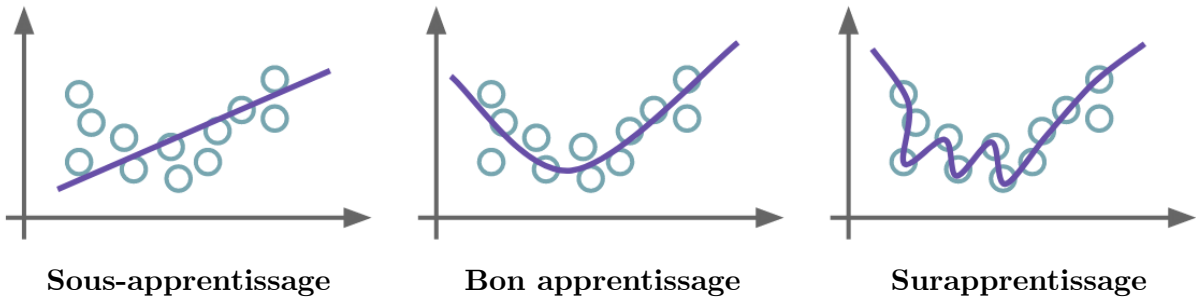


FIGURE 2.9 – Illustration de surapprentissage (**O** : données d'entraînement, — : fonction apprise). [6].

Pour contrôler les capacités de généralisation du réseau, il est donc nécessaire de l'évaluer sur un nouveau jeu de données, appelé "jeu de validation". Cet ensemble contient de nouvelles données que le réseau n'a pas vu pendant l'entraînement ; ce qui permet d'estimer l'erreur que le réseau effectuerait dans une mise en application. De ce fait, une

6. Illustrations : `https://commons.wikimedia.org/wiki/File:Underfitting_e_overfitting.png`

erreur d'entraînement qui continue de diminuer tandis que l'erreur de validation stagne ou augmente peut permettre de détecter un sur-apprentissage.

De plus, la validation peut servir pour déterminer les valeurs optimales des hyperparamètres pour l'entraînement. Plusieurs stratégies peuvent-être utilisées pour déterminer le taux d'apprentissage, la taille des *batchs*, le nombre d'*epochs*, l'inertie, etc. Parmi elles, la recherche par grille (*grid search*) consiste à tester exhaustivement l'ensemble des combinaisons de valeurs possibles. La recherche aléatoire (*random search*) commence par tester un échantillon de combinaisons choisies aléatoirement pour réduire la taille de l'espace de recherche. En pratique, ces recherches peuvent être faites sur des entraînements réduits pour accélérer la recherche. Les hyperparamètres tels que le nombre d'*epochs* ou les taux d'apprentissage peuvent aussi être adaptés au cours d'un entraînement lorsque l'erreur de validation commence à stagner.

2.2.1.3 Évaluation

Une fois le modèle entraîné, on cherche à évaluer le réseau pour estimer ses performances sur la tâche cible. En effet, l'erreur mesurée lors de la phase de validation ne reflète pas complètement les performances sur de nouvelles données pour deux raisons. Premièrement, bien que les données du jeu de validation ne soient pas directement utilisées lors de la phase d'apprentissage, elles peuvent-être utilisées pour choisir les hyperparamètres de l'entraînement. Le réseau est donc indirectement optimisé pour les données de validation. Afin d'éviter ce biais, l'évaluation finale se fait sur un troisième jeu de données contenant des données inédites appelé "jeu de test" ou "jeu d'évaluation". Dans certains *benchmarks*, les annotations, voire les données d'évaluation ne sont pas fournies afin d'éviter que les réseaux soient optimisés pour, ce qui fausserait les mesures de performance.

Deuxièmement, les fonctions de coût utilisées lors de l'entraînement ou de la validation permettent de quantifier l'erreur effectuée par le réseau lors de la phase d'entraînement, mais ne sont pas forcément représentatives de la tâche ciblée. Par exemple, l'erreur d'entraînement d'un réseau pour la classification est généralement calculée comme la distance les probabilités produites pour chaque classe et la distribution attendue. Cependant, lors de la mise en application de ce réseau, un choix final sera effectué (par exemple en prenant la classe ayant la plus haute probabilité prédite). Ce qui importe est alors de mesurer si cette prédiction est correcte ou non. Nous décrivons dans la section suivante des métriques de base couramment utilisée pour l'évaluation des performances de réseaux de neurones.

2.2.2 Métriques d'évaluation

Deux métriques de base pour l'évaluation sont la précision et le rappel, qui s'appuient sur une catégorisation des prédictions. Pour une tâche de classification par exemple, on définit :

- les vrais positifs (*True Positives*, ou TP) : la classe prédite correspond à celle de la vérité terrain,
- les faux positifs (*False Positives*, ou FP) : la classe prédite ne correspond pas à celle de la vérité terrain,
- les vrais négatifs (*True Negatives*, ou TN) : aucune prédiction n'est donnée pour un élément de la vérité terrain, à juste titre,
- les faux négatifs (*False Negatives*, ou FN) : aucune prédiction n'est donnée pour un élément qui possède une classe dans la vérité terrain.

De cette catégorisation, on peut alors calculer la précision P et le rappel R, définis comme :

$$\mathrm{P} = \frac{TP}{TP + FP} \qquad \mathrm{R} = \frac{TP}{TP + FN} \tag{2.7}$$

Ces deux métriques peuvent-être associées pour calculer le score F1 qui représente la moyenne harmonique de la précision et du rappel, défini comme :

$$\mathrm{F1} = \frac{2}{\frac{1}{\mathrm{P}} + \frac{1}{\mathrm{R}}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.8}$$

La précision mesure combien de prédictions parmi celles données par le réseau sont pertinentes. Le rappel quant à lui évalue combien d'éléments parmi ceux annotés ont été correctement identifiés par le réseau. Ces deux métriques sont donc complémentaires. Reprenons l'exemple de la tâche de classification. Considérons que l'on prend la décision finale en utilisant un seuil tel que, si aucune classe ne possède une probabilité prédite supérieure à ce seuil, alors on considère que le réseau n'a pas détecté d'objet. Plus ce seuil est placé haut, plus les prédictions finales conservées auront des scores de confiance élevés, mais beaucoup de données n'auront aucune prédiction. Cela entraîne donc un nombre de faux positifs théoriquement faible, mais un nombre de faux négatifs très élevé, ce qui donne une bonne précision, mais un rappel bas. En suivant la même logique, un seuil d'acceptation faible résulte en un rappel haut, mais une précision basse.

Pour prendre en compte ce phénomène, on mesure généralement différentes paires de valeurs précision / rappel, puis on utilise ces valeurs pour calculer une nouvelle métrique. L'une d'elle est la précision moyenne (*Average Precision*, ou AP). Pour un seuil fixé, les prédictions sont classées par ordre décroissant de confiance. On parcourt alors la liste du premier au dernier rang. Pour chaque rang, la précision est calculée comme le rapport entre le nombre de prédictions correctes et le nombre de prédictions total au-dessus de ce rang. On a donc :

$$\mathrm{P}(i) = \frac{\sum_{k=0}^{i} \delta(\|\hat{y}_k - y_k\| > seuil)}{\sum_{k=0}^{i} \delta(\hat{y}_k > 0)} = \frac{\sum_{k=0}^{i} TP_k}{\sum_{k=0}^{i} TP_k + FP_k} \tag{2.9}$$

où $\mathrm{P}(i)$ est la précision au rang i, y_k et $\hat{y}_k$ la vérité terrain et la prédiction respectivement au rang k, δ la fonction échelon unité et *seuil* le seuil d'acceptation des prédictions. Le rappel $\mathrm{R}(i)$ est quant à lui calculé comme le rapport entre le nombre d'éléments correctement détectés au-dessus de ce rang et le nombre d'éléments annotés total dans le jeu d'évaluation :

$$\mathrm{R}(i) = \frac{\sum_{k=0}^{i} \delta(\|\hat{y}_k - y_k\| > seuil)}{\sum_{k} \delta(y_k > 0)} = \frac{\sum_{k=0}^{i} TP_k}{\sum_{k} TP_k + FN_k} \tag{2.10}$$

À partir de ces valeurs calculées, on peut tracer par interpolation une courbe de la précision en fonction du rappel. Comme expliqué précédemment, la précision décroît globalement en fonction du rappel. L'interpolation est donc calculée comme la précision maximum mesurée pour un rappel supérieur ou égal au rappel actuel (Figure 2.10). On a donc :

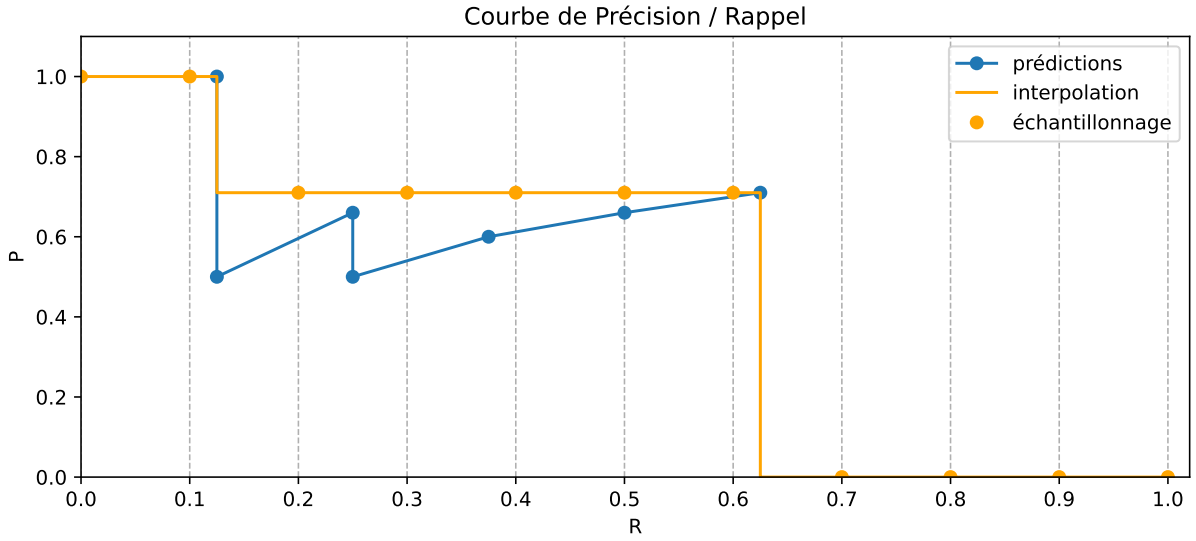


FIGURE 2.10 – Illustration de l'interpolation d'une courbe Précision / Rappel pour le calcul de l'*Average Precision*.

$$\mathrm{PI}(r) = \max_{k>=r}(\mathrm{P}(k)) \tag{2.11}$$

où $\mathrm{PI}(r)$ est la précision interpolée pour la valeur de rappel r. La précision interpolée est alors échantillonnée avec un intervalle de rappel fixe, avec, par exemple, 11 valeurs pour (Everingham et al., 2010) et 101 valeurs pour (Ruggero Ronchi and Perona, 2017)). Enfin, la précision moyenne est calculée comme la moyenne de ces valeurs.

2.3 Apprentissage sur différents domaines

Comme mentionné dans la section précédente, les réseaux de neurones sont entraînés et évalués sur des ensembles respectifs. La pratique la plus commune est d'utiliser pour ces deux phases des données différentes, mais suffisamment proches aussi bien dans l'aspect visuel que sémantique, afin que les caractéristiques apprises puissent être efficacement transférées à l'application finale. On choisit donc généralement des données appartenant à la même distribution (ou même "domaine"). Cependant, cette approche, bien qu'idéale pour une application définie, n'est pas toujours applicable en conditions réelles. En effet, un grand nombre de données suffisamment variées et annotées sont théoriquement requises pour pouvoir entraîner un réseau. De telles données dans le même domaine que l'application finale peuvent-être difficiles à obtenir. En outre, le domaine d'application peut être inconnu au moment de l'entraînement du réseau. Enfin, on peut souhaiter que le réseau entraîné soit capable de généraliser sur un grand nombre de domaines, connus ou non. Pour toutes ces raisons, différentes méthodes de "transfert d'apprentissage" (*transfer learning*, (Zhuang et al., 2020)) ont été proposées pour tenter de répondre à ces problématiques. Nous présentons dans cette section trois catégories abordées par la suite dans nos travaux : le "réglage fin" (généralement appelé "*fine-tuning*"), "l'adaptation de domaine" et la "généralisation de domaine".

2.3.1 *Fine-tuning*

L'une des méthodes les plus généralement utilisées pour le transfert d'apprentissage est la méthode de "réglage fin", ou "*fine-tuning*". L'approche par *fine-tuning*, consiste à utiliser un réseau pré-entraîné sur un grand jeu de données générique, puis à effectuer une deuxième phase d'entraînement plus courte sur des données du domaine cible. Cette stratégie s'appuie sur la capacité des réseaux de neurones profonds à apprendre des caractéristiques génériques sur un grand nombre de données génériques qui sont applicables ensuite à un domaine plus spécifique. Cette méthode peut être mise en place de différentes manières (Cetinic et al., 2018). Le deuxième entraînement est généralement effectué sur un jeu de données plus petit, durant moins d'*epochs* et en réduisant le taux d'apprentissage. Pour des tâches comme la classification, les poids entraînés de l'extracteur de caractéristiques sont généralement conservés tandis que la dernière couche entièrement connectée est réinitialisée aléatoirement. Changer la dernière couche permet également d'appliquer un réseau sur une tâche cible avec des paramètres expérimentaux différents de ceux d'entraînement, par exemple, la classification avec un nombre ou des classes différentes. Les poids de l'extracteur de caractéristiques peuvent également être gelés, ce qui permet de fortement diminuer le temps d'entraînement. L'approche par *fine-tuning* est aujourd'hui très employée dans la littérature puisqu'elle permet d'entraîner facilement et rapidement un réseau sur un domaine spécifique, en profitant des connaissances générales apprises sur des grands jeux de données par ce réseau, et dont les poids d'entraînement sont souvent disponibles publiquement. Cette approche requiert cependant d'avoir des données annotées pour effectuer un apprentissage supervisé, et les performances du réseau ainsi entraîné dépendent de la qualité des données utilisées lors du *fine-tuning* et de l'écart de domaine entre l'application originale et celle visée.

2.3.2 Adaptation de domaine

L'adaptation de domaine est une discipline cherchant à réduire la distance entre la distribution des données d'entraînement et celles du domaine cible (Farahani et al., 2021). À la différence du *fine-tuning*, on ne dispose généralement pas directement des annotations des données cibles. Cela présuppose donc d'avoir accès à des données appartenant à la distribution cible ou au moins des informations sur cette dernière.

Une première approche consiste à aligner les domaines sources et cibles. Pour ce faire, les autres méthodes d'adaptation de domaine, on peut chercher à minimiser la distance entre les deux distributions à l'aide de métriques telles que la métrique de Wasserstein, la divergence Kullback-Leibler (KL) (Kullback and Leibler, 1951), ou encore la *Maximum Mean Discrepancy* (MMD) (Gretton et al., 2006). Une autre approche utilisée est l'approche discriminative. Ces méthodes s'appuient sur un réseau discriminateur entraîné à différencier les vecteurs de caractéristiques issus des données sources de ceux encodés à partir des données cibles. Ce discriminateur peut alors être utilisé pour calculer une fonction de coût discriminative appliquée lors de l'entraînement du modèle étudié. Une telle méthode est par exemple utilisée dans l'*Adversarial Discriminative Domain Adaptation* (Tzeng et al., 2017). Cette approche nécessite cependant une importante quantité de données sources et cibles afin d'entraîner efficacement le discriminateur. Des travaux s'intéressent également à la problématique du manque d'accès à une partie des données sources, en adaptant le discriminateur aux données à disposition (Xia et al., 2021).

Les méthodes d'adaptation de domaines sont intéressantes lorsque le domaine cible est connu, mais que les données ou annotations disponibles sont insuffisantes pour appliquer

des méthodes d'entraînement directes telles que le *fine-tuning*. Cependant, cette approche ne permet pas d'améliorer les performances d'un réseau sur un domaine à propos duquel on ne dispose pas d'informations.

2.3.3 Généralisation de domaine

La généralisation de domaine poursuit un objectif similaire à l'adaptation de domaine : entraîner un réseau pour un domaine cible en s'appuyant sur des données avec une distribution différente. Cependant, à la différence de l'adaptation de domaine, nous n'avons pas accès aux données cibles ou à leur distribution. Il n'est donc pas possible de directement diminuer la distance entre les domaines puisque cette dernière n'est pas calculable. Les approches pour la généralisation de domaine sont variées et s'appuient généralement sur l'utilisation de plusieurs jeux de données sources (Zhou et al., 2022). On cherche alors à utiliser ces sources afin de faire apprendre au réseau une représentation la plus générique possible. On peut réunir les approches de généralisation de domaine en trois grandes catégories :

Enrichissement des données

Cette première catégorie de méthodes s'intéresse aux données d'entraînement dans le but d'enrichir la distribution sur lequel le réseau est entraîné. Pour ce faire, deux approches sont possibles. Premièrement, l'augmentation de données vise à appliquer des modifications sur les données d'entraînement afin d'augmenter partiellement la diversité de ces dernières. Un grand nombre d'augmentations sont possibles selon le type de données et la tâche ciblée. Pour des images, on peut appliquer des transformations aléatoires classiques telles que des transformations géométriques, de couleurs, des suppressions de pixel, ou encore combiner plusieurs images (Figure 2.11) (Shorten and Khoshgoftaar, 2019) . On peut également enrichir les données en appliquant des méthodes génératives par apprentissage profond afin de modifier aléatoirement le style des images (Jackson et al., 2019; Wang et al., 2022).

FIGURE 2.11 – Illustration de transformations pour l'enrichissement de données, avec de gauche à droite : l'image d'origine, rotation, inversion des canaux de couleur, suppression de *patchs*, combinaison linéaire de deux images. [7]

Lorsque la quantité de données à disposition n'est pas suffisante pour appliquer efficacement des méthodes d'augmentation de données, on peut alors s'intéresser aux méthodes de génération de données. Comme le nom l'indique, ces méthodes consistent à générer des nouvelles données de manière synthétiques accompagnées de leurs annotations afin d'augmenter la quantité et la diversité des données d'entraînement. Il est possible de générer

7. Illustrations : `https://pytorch.org/vision/stable/auto_examples/plot_transforms.html#sphx-glr-auto-examples-plot-transforms-py`

ces données de manière procédurale à l'aide de logiciels de modélisation (Cruz et al., 2020; Katrolia et al., 2021), ou par l'utilisation de réseaux génératifs profonds (Goodfellow et al., 2014; Isola et al., 2017; Huang et al., 2020). La génération de données permet à moindre coût de créer un grand nombre de données annotées avec une importante diversité. Cependant, la différence de domaine entre les images synthétiques et les images réelles (ou *domain gap*) peut rendre difficile la généralisation du réseau entraîné sur les données cible (Tremblay et al., 2018).

Apprentissage de représentation

L'apprentissage de représentation s'intéresse à la représentation apprise par le réseau, en particulier par l'encodeur, en cherchant à maximiser l'invariance de cette représentation vis-à-vis du domaine des données d'entrée. Afin d'apprendre une représentation invariante au domaine, un grand nombre d'approches sont possibles selon la tâche cible. Par exemple, plusieurs travaux utilisent différents encodeurs spécifiques à chaque jeu d'entraînement pour de la classification (Ding and Fu, 2017; Qin et al., 2022). Il est également possible de séparer les caractéristiques inhérentes au domaine d'entraînement de celles utiles pour la tâche cible en appliquant sur les données d'entrée différents filtres spécifiques au contexte, par exemple, pour de la réidentification de personnes (Jin et al., 2020). D'autres travaux utilisent plusieurs branches pour normaliser les données provenant de différentes sources, comme les images IRM de différentes parties du corps pour de la segmentation (Liu et al., 2020). Toutes ces approches restent dépendantes de la tâche et du contexte ciblés, et leur efficacité pour la généralisation est conditionnée par le nombre et la diversité des données d'entraînement à disposition.

Stratégie d'apprentissage

Enfin, un certain nombre de méthodes pour la généralisation de domaine s'intéressent plus largement aux stratégies d'apprentissage. On peut par exemple mentionner l'*ensemble learning* qui utilisent les prédictions de plusieurs réseaux pour déduire la prédiction finale (Dubey et al., 2021; Zhou et al., 2021). Le *meta learning* quant à lui cherche à généraliser en apprenant sur plusieurs tâches différentes (Finn et al., 2017; Chen et al., 2022a).

L'application de stratégies de généralisation de domaine est nécessaire pour entraîner un réseau capable de généraliser sur des données inconnues, en particulier lorsque l'on possède un nombre restreint de jeux de données différents. Les méthodes d'augmentation et de génération sont souvent les plus efficaces pour combler un manque de données. Cependant, elles peuvent ne pas être suffisantes si la quantité de données initiales est trop faible ou que la problématique de *domain gap* est trop difficile à résoudre. Il est alors nécessaire de s'intéresser à des méthodes touchant à l'architecture, ou à la stratégie d'apprentissage du réseau. Toutefois, il est impératif d'adapter ces méthodes à la tache cible, dans notre cas l'estimation de poses humaines, et d'évaluer leur pertinence dans ce contexte.

2.4 Architectures de référence pour ces travaux

Cette section présente quelques architectures de la littérature qui font office de références pour le traitement d'images par réseau de neurones profond, ou qui sont importantes dans le déroulé de nos travaux.

2.4.1 Extracteurs de caractéristiques convolutifs

Nous commençons par décrire l'architecture de quelques extracteurs de caractéristiques à convolutions de l'état de l'art utilisés pour le traitement d'images. Ces modèles convolutifs sont souvent introduits pour de la classification d'images, en les associant avec des couches totalement connectées pour calculer les prédictions finales. Ces extracteurs peuvent cependant être utilisés pour un grand nombre de tâches en adaptant les couches de sortie.

2.4.1.1 AlexNet

Une des premières architectures convolutives utilisées pour l'extraction de caractéristiques est le réseau AlexNet (Krizhevsky et al., 2012). Cette architecture reprend le principe général de LeNet mais est conçu pour traiter des images plus grandes ($224 \times 224 \times 3$ pixels au lieu de 28×28 pixels initialement). Pour ce faire, trois couches convolutives et une couche de *MaxPooling* sont ajoutées avant le réseau totalement connecté. De plus, le nombre de neurones des couches connectées est grandement augmenté, passant d'une centaine par couche à 4096. Cette architecture fut le premier réseau convolutif à remporter le défi de classification Imagenet (Deng et al., 2009) nécessitant de classer 150 k images dans 1000 classes d'objets.

2.4.1.2 VGG

Le réseau VGG (pour *Visual Geometry Group*) (Simonyan and Zisserman, 2015) est une architecture cherchant à extraire les caractéristiques d'une image à différentes échelles. Ainsi, une succession de blocs convolutifs composés de quelques couches convolutives suivie d'une couche de *MaxPooling* est utilisée (Figure 2.12). Cette structure réduit successivement la dimension du vecteur de caractéristiques, tout en augmentant sa profondeur. Par exemple, une image d'entrée de dimension $W \times H \times 3$ donne en sortie du premier bloc un vecteur de dimension $W/2 \times H/2 \times 64$, et de dimension $W/16 \times H/16 \times 512$ en sortie du quatrième bloc. Finalement, le vecteur de caractéristiques passe dans 3 couches FC pour obtenir la prédiction finale. Les couches de convolution utilisées dans VGG ont toutes un noyau de taille $K = 3$, ce qui permet de limiter le nombre de poids dans le réseau.

La diminution successive de la dimension du vecteur permet de travailler à plusieurs échelles et ainsi de prendre en compte les détails et la structure globale de l'image. De plus, cette architecture présente l'avantage d'être facilement modulable en augmentant le nombre de blocs et de couches par bloc. Les auteurs proposent dans leurs travaux plusieurs arrangements de couches de neurones, faisant varier leur nombre de 11 à 19, les performances augmentant avec le nombre de couches utilisées. Les variantes 16 et 19 sont cependant les plus utilisées (Pishchulin et al., 2016; Lifshitz et al., 2016; Hu and Ramanan, 2016)

2.4.1.3 ResNet

La recherche de meilleures performances et de solutions à des tâches plus complexes a entraîné une rapide croissance du nombre de couches composant les architectures convolutives. Ces réseaux plus profonds ont cependant fait face à une problématique : à partir d'une certaine taille, la performance des réseaux stagne puis décroît au fur et à mesure que leur profondeur augmente (He et al., 2016). Cet effet est en partie expliqué par la perte de gradient (*vanishing gradient*) (Bengio et al., 1994). En effet, plus un réseau est grand,

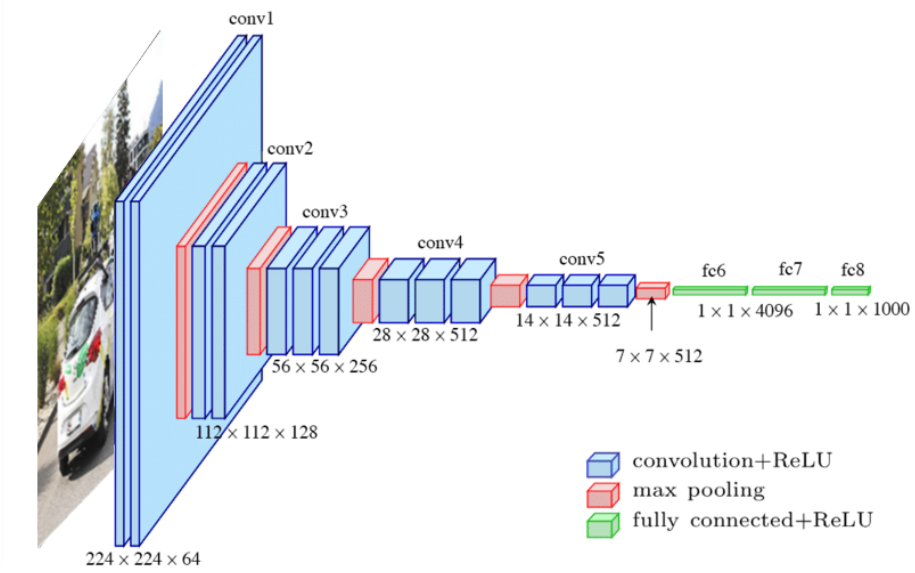


FIGURE 2.12 – Schéma de l'architecture VGG 16 (Ferguson et al., 2017).

moins le gradient de chaque couche est important, en particulier des premières couches du réseau qui sont les plus éloignées de la fonction de coût. Cela rend l'entraînement de réseaux profonds difficile. La solution proposée par He *et. al.* (He et al., 2016) consiste en l'ajout de connexions entre différentes couches jouant le rôle de "raccourcis". L'intérêt est double : permettre aux réseaux d'apprendre facilement une fonction d'identité, et permettre au gradient de "remonter" plus facilement vers les premières couches.

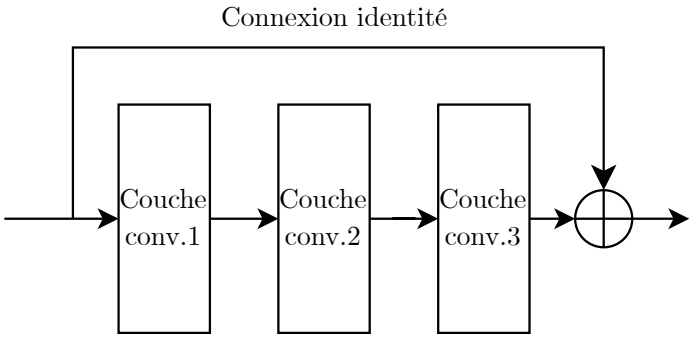


FIGURE 2.13 – Schéma d'un bloc résiduel du réseau ResNet.

Dans les architectures ResNet, ce procédé est modélisé par le biais de blocs appelés "blocs résiduels", illustré dans la Figure 2.13. Chaque bloc est composé de deux voies : deux ou trois couches de convolution d'une part, et une connexion directe pour l'identité d'autre part. Pour finir, une concaténation est effectuée entre la sortie de ces deux voies. Ces blocs sont utilisés pour construire différentes architectures ResNet avec un nombre de couches allant de 18 à 151 couches pour les réseaux les plus profonds. La flexibilité et les performances de cette structure ont fait qu'elle s'est rapidement imposée dans un grand nombre d'architectures pour l'extraction de caractéristiques (Newell et al., 2016; Redmon and Farhadi, 2018; Sandler et al., 2018).

2.4.2 Réseaux antagonistes génératifs

Les réseaux de neurones sont utilisés pour beaucoup de tâches de prédiction (prédiction de classes, de boîtes englobantes, de coordonnées etc.). Ils peuvent aussi être utilisés pour générer des nouvelles données artificielles. Nous présentons dans cette sous-section la méthode par apprentissage profond la plus utilisée pour cette tâche : les réseaux antagonistes génératifs (ou *Generative Adversarial Networks*, GAN).

2.4.2.1 GAN

Le principe des architectures antagonistes (Goodfellow et al., 2014) est d'utiliser deux réseaux et de les confronter (ou de les faire collaborer selon le point de vue) durant l'entraînement. D'une part, un réseau G appelé générateur cherche à apprendre à projeter une entrée $z \in Z$, généralement un vecteur de bruit, dans un espace cible X. D'autre part, un deuxième réseau D appelé discriminateur est entraîné à prédire la probabilité que l'entrée provienne de X, et non du générateur. Il est donc entraîné comme un classifieur avec deux classes : les données $x \in X$ et les données $G(z)$. Le générateur quant à lui est entraîné en utilisant comme erreur la prédiction du discriminateur $D(G(z))$. L'objectif de l'entraînement est d'atteindre l'équilibre dans lequel le générateur produit des données parfaitement réalistes ce qui pousse le discriminateur à classer aléatoirement les données produites. On cherche donc à trouver les valeurs optimales de G et de D :

$$\arg\min_G \max_D \mathscr{L}_{GAN} = \mathbb{E}_{x\in X}[\log D(x)] + \mathbb{E}_{z\in Z}[\log(1 - D(G(z)))] \tag{2.12}$$

Cependant, comme indiqué dans l'article original, l'équilibre entre le générateur est le discriminateur peut être difficile à atteindre. En effet, si le discriminateur prend trop de retard sur le générateur, le discriminateur peut rencontrer des difficultés à différencier les images réelles des générées, induisant des gradients pour le générateur rendant difficile la convergence de ce dernier. À l'inverse, si le discriminateur est trop performant par rapport au générateur, cela peut bloquer l'apprentissage du générateur qui va alors totalement diverger. Pour répondre à ces problématiques et stabiliser l'entraînement, plusieurs variantes de stratégies d'entraînement ont été proposées (Nowozin et al., 2016; Arjovsky et al., 2017; Gulrajani et al., 2017).

L'intérêt d'une telle approche est de permettre l'apprentissage par un réseau générateur de la distribution d'un ensemble de données sans nécessiter de données étiquetées ou de supervision supplémentaire. Si les données d'entrées sont suffisamment nombreuses et diversifiées, on peut alors utiliser ce générateur pour produire de nouvelles données appartenant à l'ensemble cible (Goodfellow et al., 2014; Radford et al., 2015). Cependant, si les données d'entrées ne sont pas suffisamment représentatives de l'ensemble que l'on cherche à modéliser, le réseau ne sera pas en mesure de généraliser et produira des résultats très proches des données d'entraînement (Kim and Park, 2023). Nous aurons l'occasion d'étudier cette problématique plus en détail par la suite.

2.4.2.2 GAN convolutif

L'utilisation des GAN pour la génération d'images a logiquement mené à l'utilisation de couches convolutives dans l'architecture du générateur et du discriminateur (Radford et al., 2015) (appelés DCGAN pour *Deep Convolutional GAN*).

8. Illustration : `https://github.com/vdumoulin/conv_arithmetic`

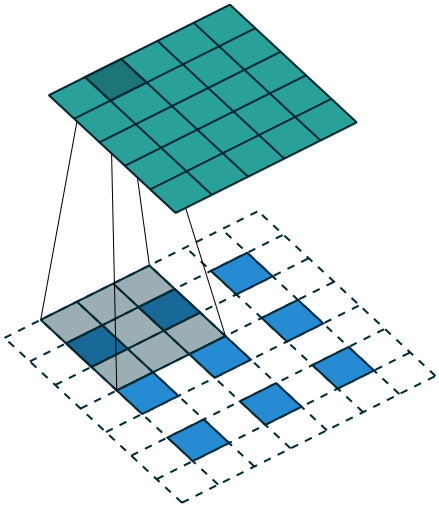


FIGURE 2.14 – Schéma d'une *fractionally-strided convolution* avec l'entrée fractionnée en bas et la sortie en haut. [8]

Dans cette version de GAN convolutif, une image est générée à partir d'un vecteur aléatoire de taille 100 redimensionné sous forme de matrice. Ce dernier est alors donné en entrée d'un décodeur composé de quatre couches convolutives. À chaque étape de convolution, la taille du vecteur de caractéristiques est agrandie pour correspondre à la taille finale de l'image. Pour ce faire, des convolutions à pas fractionné (*fractionally-strided convolutions* ou *transposed convolution* sont utilisées (Figure 2.14). Cette couche reprend le fonctionnement d'une couche de convolution classique, mais insère entre chaque pixel de l'image d'entrée des pixels mis à zéro. L'opération de convolution est alors appliquée sur ces pixels ; ce qui permet d'augmenter artificiellement la taille de l'image.

Bien que cette première architecture soit assez sommaire, elle a ouvert la voie à un grand nombre de réseaux GAN convolutifs pour la génération d'images (Liu and Tuzel, 2016; Karras et al., 2017; Brock et al., 2018; Zhang et al., 2019; Karras et al., 2019).

2.5 Estimation de poses humaines

Nous allons maintenant présenter dans cette section les différents éléments en lien avec nos travaux sur l'estimation de poses. La tâche d'estimation de poses humaines (*Human Pose Estimation* ou HPE) cherche à détecter la pose d'une ou plusieurs personnes à partir d'une ou plusieurs images. La pose peut être modélisée en 3D si l'on désire la représentation de la pose dans l'espace, ou en 2D en utilisant la projection de la pose observée sur l'image. Nos travaux sur l'estimation de poses étant exclusivement portés sur la détection à partir d'une seule image, dite détection monoculaire, nous ne traiterons pas de détection depuis plusieurs angles de vue. De plus, comme expliqué dans le Chapitre 1, nos travaux se concentrent sur l'estimation de poses 2D en raison des contraintes techniques imposées par l'acquisition de données 3D. Nous présenterons donc dans cette section principalement les travaux de la littérature sur l'estimation 2D.

2.5.1 Modélisation du corps

Le corps humain est un objet complexe composé d'un grand nombre d'aspects : forme, couleurs, textures, etc. Le nombre d'aspects nécessitant d'être modélisés dépend cependant de la tâche à effectuer et de l'application visée. Dans le cas de l'estimation de poses, la tâche consiste en deux aspects :

- la détection globale du corps dans l'image,
- l'estimation de la déformation des parties du corps causée par le mouvement des articulations.

Il convient donc de trouver une modélisation paramétrable permettant de représenter cette déformation pour pouvoir ensuite placer ce modèle dans l'image et estimer ses paramètres. Trois catégories de modèles humains sont généralement utilisées : le modèle basé squelette, le modèle basé contours et le modèle basé volumes (Figure 2.15).

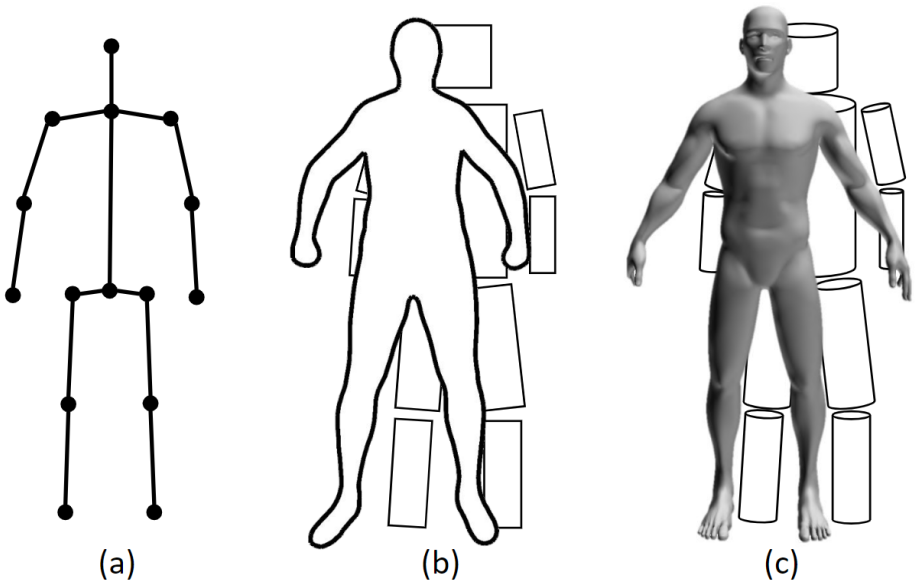


FIGURE 2.15 – Trois catégories de modèles humains : basés squelette (a), basés contours (b) et basés volumes (c). (Chen et al., 2020)

2.5.1.1 Modèle basé contours

La modélisation basée contours consiste à représenter chaque partie du corps par une forme plane et n'est donc applicable qu'à de l'estimation de poses 2D. Les premiers travaux pour l'estimation de poses utilisaient des boites englobantes sous forme de rectangle pour chaque membre du corps (Cootes et al., 1995; Ju et al., 1996). Cette représentation est relativement complexe sans pour autant apporter beaucoup d'informations sur la forme. Pour cette raison, elle est peu utilisée aujourd'hui.

2.5.1.2 Modèle basé volumes

La modélisation basée volumes reprend à l'origine la même idée que la modélisation de contours en l'appliquant à la 3D. Dans les premières versions, des formes simples telles que des cylindres sont utilisées pour modéliser les parties du corps (Sidenbladh et al.,

2000). Ces modèles possèdent donc les mêmes défauts que les modèles de contours, et sont donc rapidement remplacés par des modélisations plus réalistes de la surface du corps humain composées d'un maillage géométrique (Anguelov et al., 2005; Loper et al., 2015; Joo et al., 2018). Ces modèles, semblables à ceux utilisés dans l'animation 3D, sont totalement déformables et permettent une représentation très réaliste du corps observé et de sa pose. Ils ont cependant l'inconvénient de comporter un grand nombre de paramètres.

2.5.1.3 Modèle basé squelette

Le modèle basé squelette, ou modèle en bâtons, est composé d'un ensemble de points clés, généralement entre 10 et 30, reliés entre eux par des segments (Morris and Rehg, 1998). Comme le nom de ce modèle l'indique, la plupart de ces points représentent les articulations principales du corps humain, et les segments les os qui les relient. Cependant, des points peuvent aussi être ajoutés pour les yeux, le nez, etc., afin de permettre la modélisation du visage. Dans ce cas, les segments ne représentent plus un os précisément, mais plutôt une structure osseuse. Le nombre de points peut fortement augmenter si l'on cherche à modéliser plus en détail les mains, les pieds ou le visage, bien que la détection précise de ces parties du corps fasse l'objet d'un pan spécifique de la littérature (Zhao et al., 2003; Erol et al., 2007).

La paramétrisation de cette structure est généralement réalisée en utilisant les coordonnées de chaque point, que ce soit en 2D ou en 3D (Newell et al., 2016; Martinez et al., 2017; Cao et al., 2019; Zheng et al., 2021). Dans ce cas, seuls les points sont utilisés, les segments servant simplement d'aide pour la visualisation des résultats. On peut aussi représenter ce squelette comme un modèle cinétique en considérant la rotation entre chaque articulation. Cette approche peut être utilisée en 3D pour appliquer des contraintes réalistes au modèle (Pons-Moll et al., 2014; Akhter and Black, 2015).

L'intérêt de cette modélisation basée squelette est sa simplicité, puisqu'elle n'est composée que du strict minimum pour représenter la pose humaine et ne comporte donc que peu de paramètres. L'inconvénient est qu'elle apporte très peu d'informations sur la structure globale du corps, en particulier sur la forme. Cette modélisation est cependant majoritairement utilisée dans la littérature, en particulier pour l'estimation de poses en deux dimensions où l'on cherche à estimer la projection du squelette sur l'image. C'est donc ce modèle que nous utilisons dans la suite de nos travaux.

2.5.2 Jeux de données

Les jeux de données jouent un rôle important dans la performance des méthodes par apprentissage profond. L'amélioration des performances des réseaux d'estimation de poses est en partie due à la publication de nouveaux jeux de données avec plus d'images, de personnes et de diversité dans les poses, les angles de vue, les fonds, etc. Nous présentons dans cette section les principaux jeux de données dans la littérature pour l'estimation de poses. Cette présentation est résumée dans le Tableau 2.2. De plus, nous reprenons dans l'Annexe B les différents jeux de données pour l'estimation de poses utilisés dans ces travaux.

2.5.2.1 Contexte générique

Les jeux de données les plus utilisés pour la tâche d'estimation de poses sont les jeux de données génériques. Les images contenues dans ces ensembles sont généralement extraites

de banques de données publiques d'images ou de vidéos ce qui permet d'obtenir un grand nombre d'images dans des contextes et environnements variés.

LSP Le premier jeu de données conséquent publié pour l'estimation de poses est le jeu *Leeds Sports Pose* (LSP) (Johnson and Everingham, 2010), avec originellement 1 k images, et étendu plus tard à 11k images. Il contient des images de personnes pratiquant différentes activités sportives extraites de la plateforme Flickr. Le squelette est annoté en utilisant 14 points, dont un seul pour la tête posée au sommet du crâne.

FLIC Le jeu de données *Frames Labeled In Cinema* (FLIC) (Sapp and Taskar, 2013) est composé d'environ 5 k images extraites de différents films hollywoodiens. Plus précisément, des images sont sélectionnées dans 30 films par un détecteur de personnes, puis une sélection plus précise est effectuée et la partie haute du corps est annotée avec 7 points clé.

MPII L'institut en informatique Max Planck (*Max Plank Institute for Informatics* ou MPII) a publié en 2014 un jeu de données de 25 k images (Andriluka et al., 2014). Ces images sont manuellement extraites de 4k clips vidéo provenant de YouTube. Plusieurs annotations sont disponibles, parmi lesquelles un squelette de 14 points identique à celui de LSP, une boite englobante pour la tête et l'orientation 3D de la tête est du torse.

COCO Le jeu de données *Common Objects in Context* (COCO) (Lin et al., 2014) publié par Microsoft en 2014 est originellement un jeu de données pour la détection d'objets étendu plus tard à l'estimation de poses. Le jeu pour l'estimation de poses est composé de 120 k images extraites de Bing, Google et Flickr, annotées avec un squelette de 17 points, mais aussi avec des boîtes englobantes et des masques de segmentation. En raison de son grand nombre d'images, il reste aujourd'hui le jeu de données le plus utilisé pour l'entraînement et l'évaluation des réseaux d'estimation de poses dans un contexte général.

CrowdPose Le jeu de données *CrowdPose* (Li et al., 2019a) est un jeu se concentrant sur l'estimation de poses dans des scènes de foules où un grand nombre de personnes sont présentes. Pour sa construction, un indice de foule est calculé sur les images de jeux de données publiques tels que COCO et MPII. Cet indice est calculé pour chaque personne en fonction du nombre de points clés présents dans la boîte englobante appartenant à d'autres personnes de l'image. Au total, 30 000 images sont sélectionnées pour obtenir une répartition équilibrée de cet index de foule.

En plus des annotations de coordonnées des points du corps, les jeux de données peuvent contenir d'autres vérités terrain telles que des boîtes englobantes du corps, de la tête, des masques de segmentation, etc. Une autre information complémentaire est la visibilité des points. À chaque point du corps est associée une étiquette selon que le point est totalement visible, non visible mais annoté car le reste du corps est suffisamment visible pour permettre la localisation du point, ou non annoté si le point est trop fortement occulté ou en dehors du champ de l'image. Il est important de noter que toutes les annotations de ces jeux de données ont été effectuées par le service Amazon Mechanical Turk (Amazonmturk, 2022) qui propose une annotation manuelle des images, mais qui ne permet pas de garantir la qualité ou l'homogénéité des annotations.

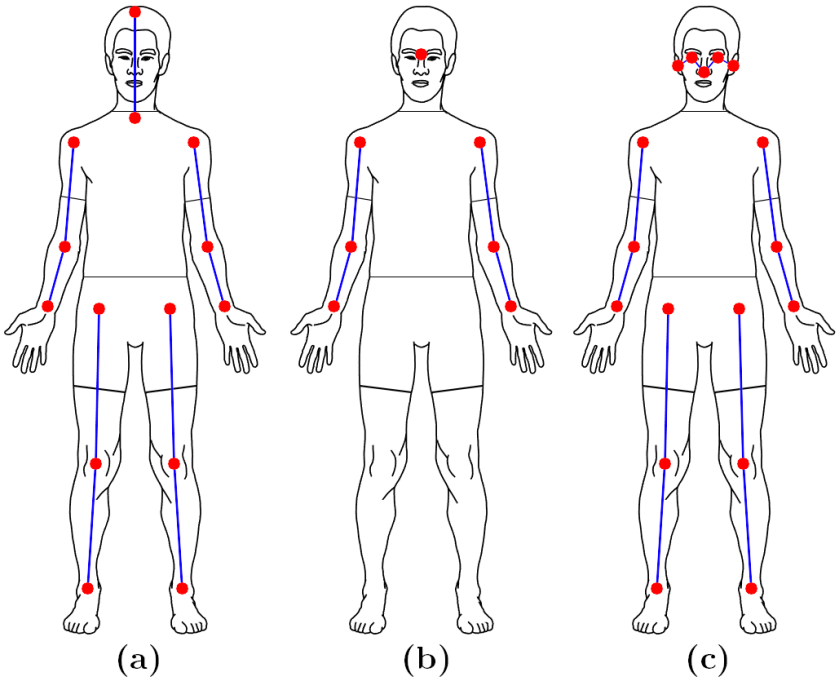


FIGURE 2.16 – Comparaison des modélisations basées squelette utilisés dans : (a) LSP et MPII, (b) FLIC et (c) COCO et CrowdPose. [9]

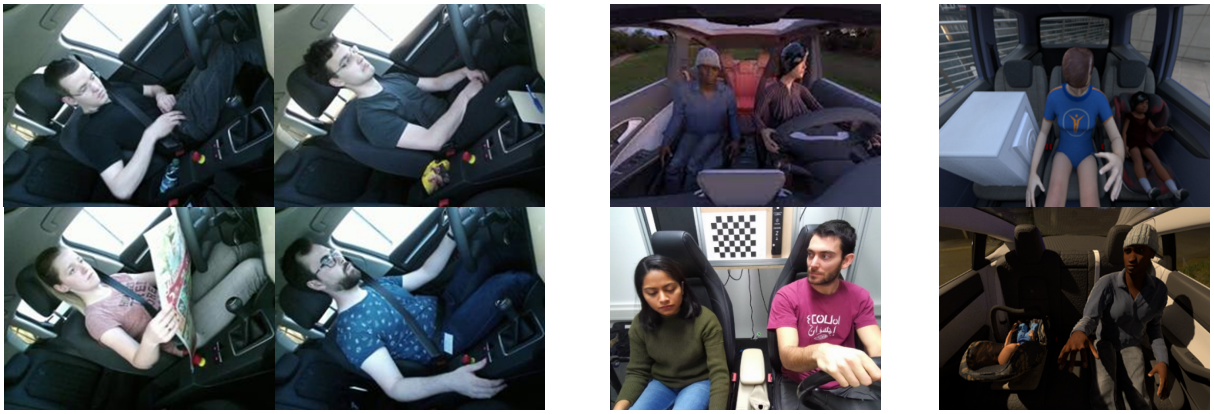
Le jeu COCO reste aujourd'hui très majoritairement utilisé comme jeu de données pour l'estimation de poses dans un contexte générique, du fait de son grand nombre d'images et de personnes. Bien que MPII ait longtemps été utilisé, il l'est de moins en moins car les performances plafonnent depuis plusieurs années à plus de 90 % de précision (Newell et al., 2016; Chen et al., 2017; Li et al., 2019b). Le jeu CrowdPose est quant à lui principalement utilisé pour l'entraînement et l'évaluation de réseaux se concentrant sur l'estimation de poses dans des foules (Ding et al., 2022; Yang et al., 2023).

2.5.2.2 Contexte de voiture

Bien que les images génériques des grands jeux de données mentionnés précédemment peuvent être utiles pour entraîner ou évaluer un réseau, elles n'illustrent pas toujours certaines situations difficiles provenant de contextes spécifiques. Pour cette raison, plusieurs jeux de données centrés sur la surveillance de passagers de voitures ont été publiés (Borghi et al., 2017; Jegham et al., 2019; Martin et al., 2019; Borghi et al., 2020; Feld et al., 2020; Ortega et al., 2020; Kopuklu et al., 2021) (Figure 2.17). Cependant, la plupart de ces jeux de données sont construits pour de la reconnaissance d'actions, ou pour des tâches spécifiques liées à la sécurité du conducteur telles que l'estimation de l'orientation de la tête, la détection d'attention, etc.

Ainsi, le seul jeu de données public d'images réelles de conducteurs fournissant des pseudo-annotations de poses humaines est le jeu Drive&Act (Martin et al., 2019). Ces annotations sont cependant incomplètes puisqu'elles ont été obtenues à l'aide d'un réseau de neurones pré-entraîné pour l'estimation de poses et ne peuvent donc pas être considérées comme des vérités terrain. Drive&Act contient 12 heures de vidéo de clips enregistrés

9. Illustration : `https://commons.wikimedia.org/wiki/File:Silhouette_humain_asexue_anterieur_posterieur.svg`



(a) (b) (c)

FIGURE 2.17 – Exemples d'images de jeu de données de passagers : (a) : Drive&Act (Ortega et al., 2020), (b) : TICaM (Katrolia et al., 2021), et (c) : SVIRO (Cruz et al., 2020).

dans un simulateur, c'est-à-dire une voiture fixe placée dans un studio (Figure 2.17-a). Ces images représentent 15 acteurs effectuant 83 tâches parmi lesquelles des tâches de conduite, mais aussi des interactions avec des objets du quotidien comme un ordinateur, un journal, etc. On peut également mentionner le jeu de données DMD (*Driver Monitoring Dataset*) (Ortega et al., 2020). Ce jeu contient 41 heures d'enregistrement dans un simulateur et en extérieurs avec 37 participants réalisant différentes activités de conduites. Trois angles de vues sont utilisés pour enregistrer des vidéos couleur, profondeur et infrarouge. Différentes annotations sont proposées telles que les actions, les mains sur le volant, ou encore les objets dans la scène. Les annotations de la pose du visage et du corps ont également été annoncées, mais ces dernières ne sont pas encore publiquement disponibles à notre connaissance.

Le faible nombre de données disponibles pour l'estimation de poses dans un véhicule est entre autres explicable par la difficulté de mettre en place des expérimentations. Pour pallier ce problème, des jeux de données synthétiques ont aussi été publiés. Les images synthétiques sont générées par ordinateur, généralement en utilisant des logiciels de modélisation 3D. L'avantage de cette méthode est qu'elle permet de générer à moindre coût un grand nombre d'images avec une diversité théoriquement infinie. SVIRO (Cruz et al., 2020) est un jeu de données pour les scénarios dans l'habitacle du véhicule. Il met en

	MPII	COCO	Drive&Act	SVIRO	TICaM
Année	2014	2015	2019	2020	2021
# Images	25 k	120 k	9,6 M	25 k	126k
# Personnes	38 k	270 k	15	22	13
Réel / Synthétique	Réel	Réel	Réel	Synthétique	Synthétique et réel
Annotation	Manuelle	Manuelle	-	Générée	Générée

TABLEAU 2.2 – Tableau comparatif des principaux jeux de données utilisés dans nos travaux pour l'estimation de poses générique (à gauche) et dans un habitacle (à droite).

scène des personnes et des objets sur la banquette arrière dans différentes configurations, et fournit plusieurs modalités d'images telles que RGB ou des images de profondeur, en plus d'annotations de poses et des boîtes englobantes (Figure 2.17-c). Le jeu de données TICaM (Katrolia et al., 2021) combine des images réelles enregistrées dans un simulateur et des images synthétiques pour la surveillance d'actions dans la voiture (Figure 2.17-b). Ces deux jeux de données restent cependant principalement ciblés pour la surveillance de passagers et présentent donc peu de diversité pour l'estimation de poses, en particulier concernant le nombre de personnes et dans les poses présentées.

2.5.3 Méthodes de l'état de l'art pour l'estimation de poses

Les méthodes d'estimation de poses se divisent en deux catégories : les méthodes mono-personnes qui se concentrent sur des images présentant une seule personne, et les méthodes multi-personnes.

2.5.3.1 Estimation mono-personne

L'estimation de la pose à l'aide d'un réseau de neurone peut être faite par deux méthodes : la méthode par régression cherchant à prédire directement les coordonnées des points clés, et la méthode par détection qui utilise des cartes de chaleur pour indiquer la probabilité de la localisation de chaque point clé.

Méthodes par régression

Les premières méthodes utilisant des réseaux de neurones convolutifs pour de l'estimation de poses humaines utilisent des extracteurs de caractéristiques de la littérature pour prédire les coordonnées x et y de chaque point du squelette. Pour un squelette de 17 points, la couche de sortie est donc de taille 34. Cette approche est appelée méthode par régression. Par exemple, DeepPose (Toshev and Szegedy, 2014) utilise AlexNet dans une méthode récursive pour estimer puis affiner les coordonnées. Carreira *et. al.* (Carreira et al., 2016) proposent un réseau itératif avec une estimation de l'erreur, basé sur l'architecture GoogleNet (Szegedy et al., 2015). Finalement, Sun *et. al.* (Xiao et al., 2018) utilisent ResNet combiné à une représentation de la pose en utilisant les jonctions entre les points pour prédire la pose. Les méthodes par régression manquent cependant de robustesse à cause de la forte non-linéarité entre l'image et les coordonnées des points.

Méthodes par détection

Pour remédier aux faiblesses des méthodes par régression, la majorité des solutions de ces dernières années proposent une approche par détection. Ces méthodes visent généralement à prédire des cartes de chaleur (*heatmaps*) qui sont des images où la valeur de chaque pixel est donnée entre 0 et 1 et est assimilable à la probabilité qu'un point clé soit situé à ce pixel. On cherche donc à générer une carte de chaleur par point clé. Les coordonnées finales d'un point sont alors les coordonnées du pixel maximum Figure 2.18. L'intérêt de cette approche est de ne pas entraîner le réseau à prédire une carte avec un seul pixel à 1 et tous les autres à 0, mais plutôt une distribution gaussienne centrée autour des points clés, ce qui permet de guider l'entraînement du réseau.

ConvNet Pose Une des premières méthodes utilisant les cartes de chaleur pour la prédiction de poses est une application d'un réseau de neurone convolutif (*ConvNet*)

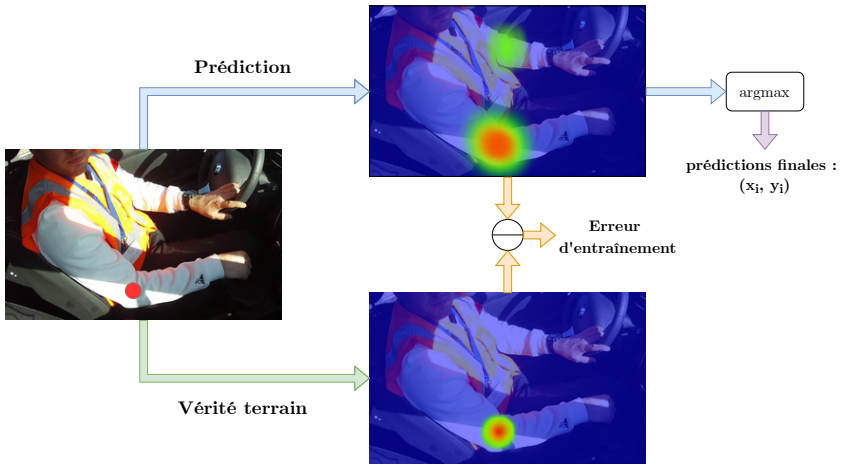


FIGURE 2.18 – Exemple de carte de chaleur pour l'estimation de poses humaines. L'image en transparence ainsi que la couleur des cartes de chaleur sont seulement là pour aider la visualisation.

(Tompson et al., 2014). Cette architecture utilise plusieurs branches en parallèle qui calculent des caractéristiques à différentes échelles de l'image (Figure 2.19). En effet, l'image d'entrée est dupliquée et sa taille est réduite à différentes échelles avant d'être donnée en entrée de chaque branche. L'intuition derrière cette approche est la même que celle derrière VGG, c'est-à-dire de capturer des informations spatiales à différentes échelles, aussi bien sur la structure globale que sur les détails. Ainsi, une suite de convolution 5×5, ReLU et *MaxPooling* traitent les images indépendamment. Puisque les images d'entrée sont de différentes tailles, il en est de même pour les vecteurs de caractéristiques calculés. Ces vecteurs sont donc redimensionnés à la même taille avant d'être additionnés. Enfin, deux dernières couches de convolutions 9×9 calculent les cartes de chaleur finales.

Les auteurs associent à ce détecteur de points un module basé sur un graphe entièrement connecté. Ce graphe est utilisé pour modéliser les contraintes de connectivités entre les différents points du corps. Ainsi, pour un point A du corps connecté à un point B, le module apprend la carte de probabilité $P_{A|B}(i, j)$ qui représente la probabilité que le point A soit présent en coordonnées (i, j) en considérant que le point B est centré. Ces cartes sont appliquées par convolution aux cartes de chaleur générées par le détecteur de

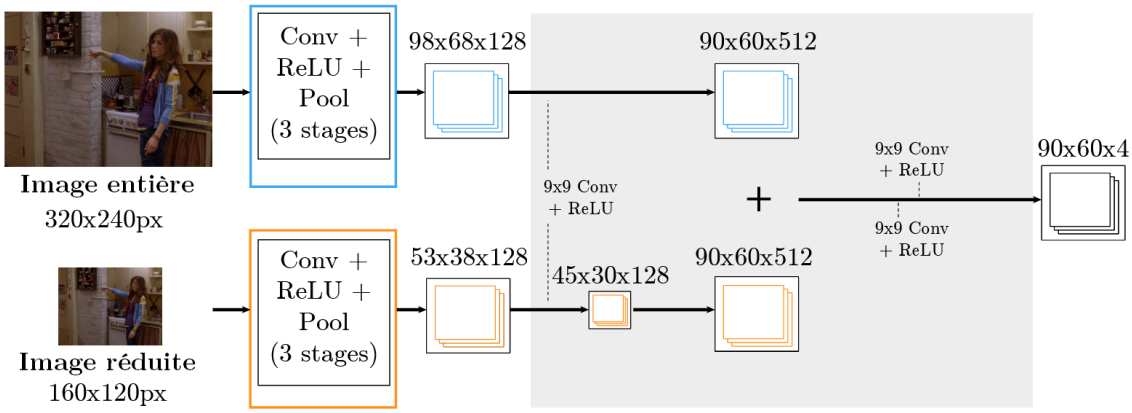


FIGURE 2.19 – Schéma de l'architecture *ConvNet Pose*(Tompson et al., 2014).

points. Le but de ce deuxième module est d'éliminer des faux positifs incohérents avec la structure du corps.

Une version améliorée de ConvNet Pose (Tompson et al., 2015) ajoute un module de raffinement entraîné pour estimer le décalage de chaque point sur les imagettes découpées autour des prédictions. Ce module utilise une architecture convolutive en cascade, et les décalages calculés sont ajoutés aux prédictions de l'architecture principales pour déterminer les prédictions finales. Les avantages de ConvNet Pose sont l'utilisation des cartes de chaleur qui apportent plus d'informations et convergent plus facilement qu'une approche directe par régression. La modélisation de la structure du corps permet d'intégrer des *a priori* réels sur la structure du corps à la décision finale. L'architecture du réseau, et en particulier du module de prédiction des cartes de chaleur reste cependant très sommaire.

Convolutional Pose Machine La *Convolutional Pose Machine* (CPM) (Wei et al., 2016) est une architecture séquentielle constituée d'une répétition de modules convolutifs appliqués successivement (Figure 2.20). Dans le premier module, l'image d'entrée passe à travers une succession de convolution 9x9 et de *MaxPooling* d'une manière similaire à un VGG afin de calculer les caractéristiques à plusieurs échelles (Figure 2.20 - c). Enfin, deux couches de convolution 1x1 sont utilisées pour calculer les cartes de chaleur. Le premier module donne donc une première estimation des cartes de chaleur. Les modules suivants sont alors utilisés pour affiner cette prédiction. Ainsi, en plus de recalculer un vecteur de caractéristiques à partir de l'image d'entrée de manière identique au premier module, les modules suivants prennent aussi en entrée les cartes générées par le module précédent (Figure 2.20 - d). Les cartes sont alors concaténées avec le vecteur de caractéristiques puis passent à travers deux couches de convolutions 11x11, avant de passer à travers les deux couches finales qui calculent une nouvelle version des cartes de chaleur.

Lors de l'entraînement, la fonction de coût (MSE) est appliquée sur les cartes de chaleur à la sortie de chaque module, ce qui permet une bonne propagation du gradient à travers le réseau. Ce procédé peut donc en pratique être répété autant de fois que voulu en ajoutant des modules à la suite pour affiner les prédictions. Cependant, les auteurs montrent que l'augmentation des performances stagnent au-dessus de trois modules.

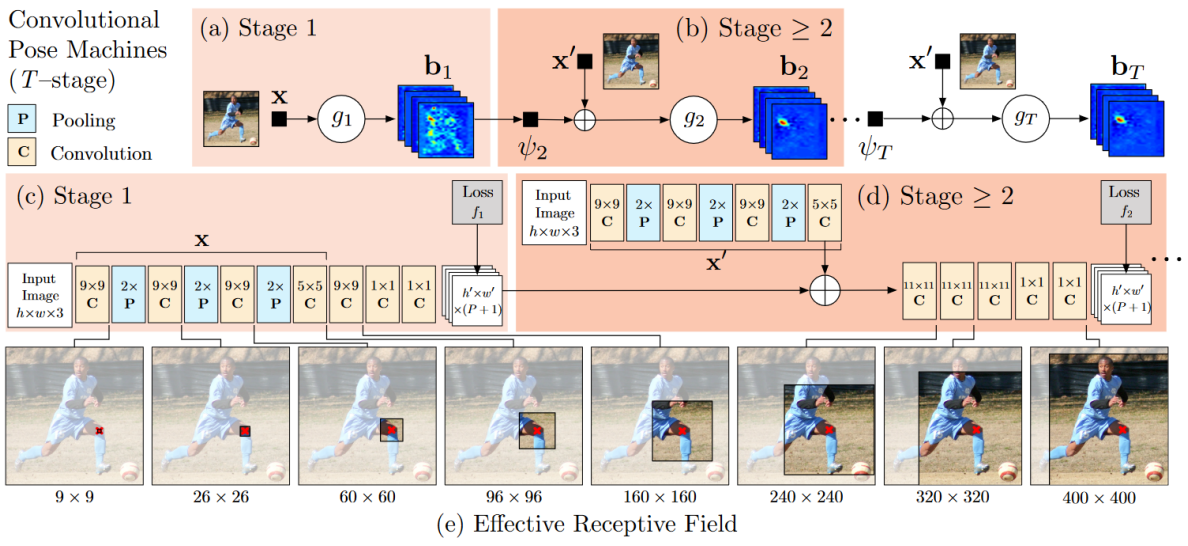


FIGURE 2.20 – Schéma de l'architecture *Convolutional Pose Machine*(Wei et al., 2016).

Cette approche introduit dans l'état de l'art l'approche séquentielle pour effectuer l'estimation de poses mono-personne, en résolvant le problème de *vanishing gradient* par l'utilisation des fonctions de coût intermédiaires.

Stacked Hourglass Une autre architecture majeure par détection est *Stacked Hourglass* (sablier empilé) (Newell et al., 2016). Cette architecture utilise également une approche séquentielle en alignant successivement des modules appelés *Hourglass* (Figure 2.21). Ces modules prennent une structure d'encodeur-décodeur et ont pour but d'extraire les caractéristiques à différentes échelles. Ainsi, l'encodeur extrait des caractéristiques sur quatre niveaux d'échelles différents en réduisant jusqu'à $1/16^e$ de la taille de l'image d'entrée. Cependant, alors que ce vecteur de caractéristiques était directement utilisé dans les CPM pour générer les cartes de chaleur, ici le décodeur ramène successivement le vecteur de caractéristiques à la dimension de départ. Le calcul à chaque niveau est effectué par un bloc résiduel, suivi d'une couche de *MaxPooling* ou de convolution transposée selon l'étape. En plus de cela, une connexion résiduelle est ajoutée entre chaque niveau d'encodage et son équivalent à l'étape de décodage. En effet, chaque vecteur en sortie d'un bloc résiduel d'encodage passe à travers un autre bloc résiduel avant d'être additionné avec l'entrée du bloc correspondant de la convolution transposée.

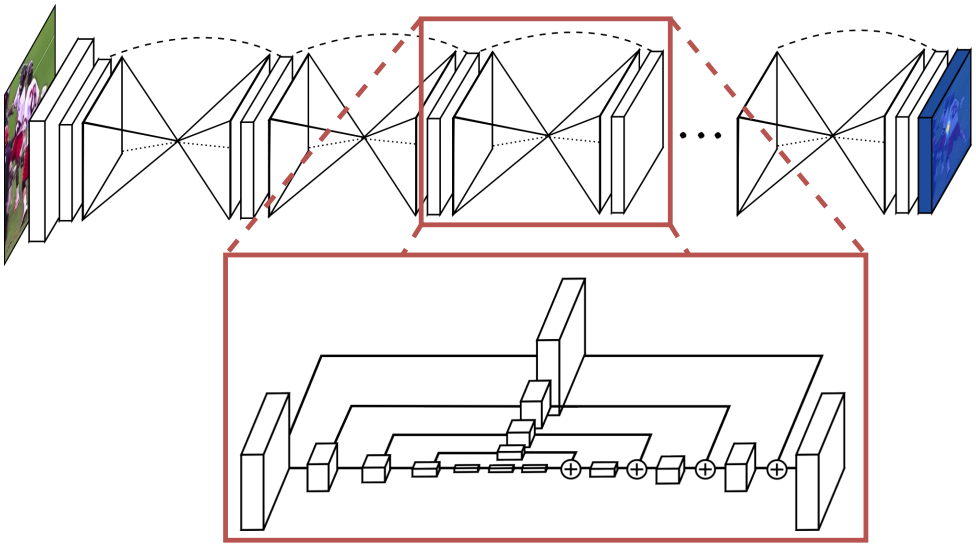


FIGURE 2.21 – Schéma de l'architecture *Stacked Hourglass*(Newell et al., 2016).

Le premier module prend en entrée l'image et produit un vecteur de caractéristiques utilisé pour calculer une première estimation de cartes de chaleur. Les modules suivants prennent alors en entrée une concaténation du vecteur de caractéristiques précédent, les cartes de chaleur ainsi que l'entrée du bloc précédent pour produire une nouvelle estimation. Le nombre de modules utilisés par les auteurs varie généralement de 2 à 8. L'utilisation des connexions résiduelles dans chaque module permet d'améliorer la propagation du gradient à travers tout le réseau. Dans le même but, la supervision intermédiaire est utilisée sur la sortie de chaque module pour faciliter l'entraînement.

L'utilisation d'une architecture en auto-encodeur couplée aux blocs résiduels a permis d'améliorer grandement les performances de l'état de l'art. Cette architecture *Hourglass* a donc inspiré plusieurs autres travaux (Chu et al., 2017; Ke et al., 2018; Tang and Wu, 2019; Tang et al., 2018). Les blocs résiduels ainsi que les modules successifs augmentent cependant fortement la taille du réseau.

Simple Baseline L'architecture Simple Baseline (SBl) (Xiao et al., 2018) pour l'estimation de poses s'appuie sur une structure d'auto-encodeur convolutif classique afin de prédire les cartes de chaleur. Il utilise le réseau ResNet-50 comme extracteur de caractéristiques, mais remplace le dernier bloc totalement connecté par quatre étages de convolution qui ré-augmente progressivement la dimension du vecteur de caractéristiques avant de prédire les cartes de chaleur finales. Chaque étage convolutif est simplement constitué d'une convolution fractionnée suivie d'une couche de *BatchNorm* et de ReLU. L'avantage d'une telle approche est qu'elle est facilement modifiable comme le montre les auteurs dans l'article en utilisant l'architecture avec deux autres extracteurs de caractéristiques, ResNet-101 et 152 respectivement ; ce qui permet d'augmenter légèrement les performances au prix d'un réseau avec beaucoup plus de poids.

Autres architectures En plus de ces méthodes, d'autres architectures basées sur différentes approches ont été mises au point. *Adversarial posenet* (Chen et al., 2017) propose une architecture par apprentissage antagoniste qui combine un générateur de cartes de chaleur avec deux discriminateurs, un pour la pose et un pour la confiance des prédictions. Unipose (Artacho and Savakis, 2020a) reprend la même approche, mais utilise pour la génération de cartes un module en cascade. L'architecture HRNet (Sun et al., 2019) utilise une approche parallèle multi-échelle, comparable à celle utilisée dans le réseau *Stacked Hourglass*, mais ajoute des modules d'échange pour augmenter le partage d'informations entre les différents niveaux.

Finalement, tous les réseaux mentionnés précédemment permettent d'atteindre des performances proches de l'état de l'art sur les *benchmarks* de référence pour l'estimation de poses humaines mono-personne tels que LSP ou MPII. Cependant, les améliorations récentes de l'état de l'art ont été apportées dans le domaine de l'estimation de poses multi-personnes étant donné les défis supplémentaires apportés par la tâche et la grande popularité du jeu de données COCO, jeu principalement multi-personne.

2.5.3.2 Estimation multi-personne

L'estimation de poses humaines multi-personne relève de deux difficultés : trouver l'emplacement des points clés sur l'image, et les associer aux différentes personnes. Pour résoudre ce problème, deux approches peuvent-être utilisées : l'approche ascendante (*bottom-up*) et l'approche descendante (*top-down*).

Approches ascendantes

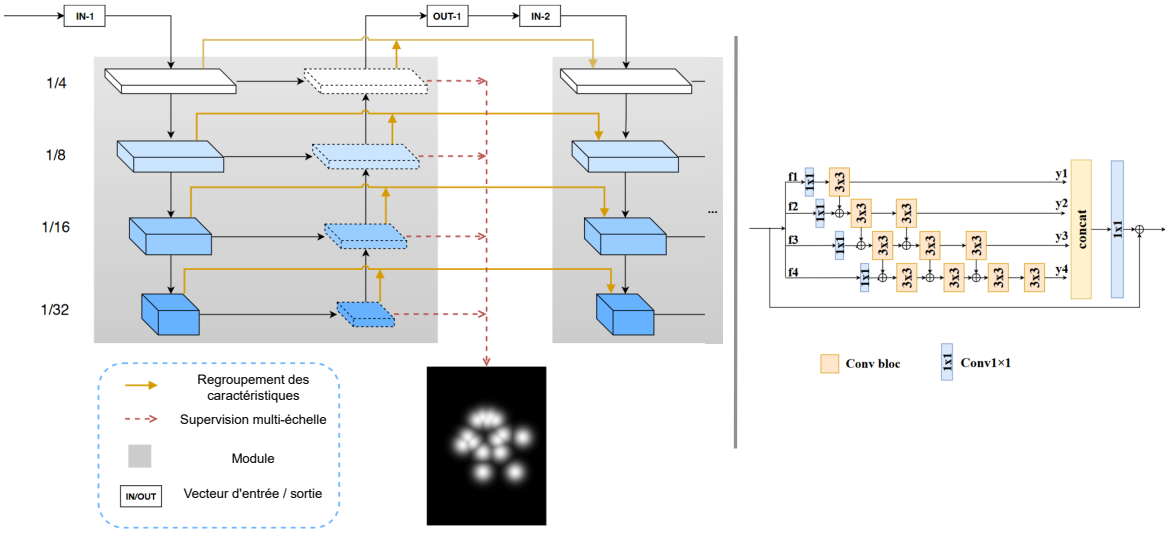
La méthode ascendante cherche à d'abord détecter tous les points clés présents sur une image, puis à en déduire les différentes personnes qui contiennent ces points. Newell *et al.* (Newell et al., 2017) réutilisent leur réseau Stacked Hourglass utilisé pour de l'estimation mono-personne, et l'adapte pour la tâche multi-personne en y ajoutant la prédiction d'une carte d'association pour chaque classe de point. OpenPose (Cao et al., 2017) intègre au réseau CPM la prédiction de champs d'affinité, une carte modélisant les relations des points du squelette entre eux, pour regrouper les points aux personnes correspondantes. Enfin, DEKR Posenet (Geng et al., 2021) utilise un réseau multi-branche afin de prédire par régression les coordonnées des points par rapport au centre. La méthode ascendante se prête particulièrement à l'estimation de la pose dans les foules puisque le nombre d'opérations augmente peu avec le nombre de personnes à détecter. Cependant, il est plus difficile

d'avoir une bonne précision sur la détection des points puisque l'image est vue dans sa globalité.

Approches descendantes

À l'inverse, l'approche descendante consiste à d'abord détecter les personnes présentes dans l'image, puis à trouver les points clés appartenant à chacun. La grande majorité des méthodes descendantes utilisent une architecture pour l'estimation de poses mono-personne précédée d'un réseau de détection de personnes. Par exemple, les auteurs de Simple Baseline (Xiao et al., 2018) ou de HRNet (Sun et al., 2019) utilisent tous deux Faster R-CNN (Ren et al., 2015), tandis que GlobalNet (Chen et al., 2018) utilise une architecture pyramidale (Lin et al., 2017) en forme de U pour extraire les caractéristiques à différentes échelles.

Plus récemment, le réseau MSPN (Li et al., 2019b) utilise pour estimer la pose une architecture à plusieurs étages et une méthode de regroupement des caractéristiques entre ces étages couplée avec le détecteur MegDet (Peng et al., 2018). Plus précisément, MSPN est un réseau à plusieurs modules (les auteurs proposent jusqu'à huit modules), où chaque module est une instance de GlobalNet chargée de prédire les cartes de chaleur(Figure 2.22-a). À l'entrée de chaque niveau de l'encodeur sont ajoutés les vecteurs de caractéristiques provenant du même niveau de l'encodeur et du décodeur du module précédent. Enfin, la fonction de coût est une erreur quadratique appliquée à la sortie de chaque étape des décodeurs, et non uniquement aux prédictions finales de chaque module comme cela peut être fait dans *Stacked Hourglass* par exemple.



(a) - Module MSPN **(b) - Bloc résiduel RSN**

FIGURE 2.22 – Schémas d'un module de MSPN (a) (Li et al., 2019b) et d'un bloc résiduel de RSN (b) (Cai et al., 2020).

RSN (Cai et al., 2020) réutilise une structure similaire à MSPN en la combinant avec des blocs résiduels (Figure 2.22-b). Contrairement au bloc résiduel classique utilisé dans ResNet, ce bloc cherche à obtenir quatre versions différentes du vecteur de caractéristiques d'entrée en le faisant passer dans des branches parallèles composées de convolutions 1×1 et 3×3. Ces différents vecteurs sont fusionnés progressivement en cascade dans chaque branche inférieure. Enfin, les quatre vecteurs sont concaténés avant de passer dans une

convolution 1×1 pour obtenir le vecteur auquel sera ajouté le vecteur d'entrée. Le réseau RSN utilise également en sortie du réseau un module calculant une carte d'attention ainsi qu'un vecteur de pondération des canaux permettant d'affiner les prédictions finales. Les architectures MSPN et RSN ont remporté le *benchmark* COCO dans leur année respective de publication (2018 / 2019).

Les avancées les plus récentes ont été apportées par l'adaptation des *transformers* (Vaswani et al., 2017), et plus précisément des *transformers* visuels (Dosovitskiy et al., 2020), à l'estimation de poses humaines. Les *transformers* sont des réseaux se basant sur les relations entre différentes parties de l'entrée par un mécanisme dit "d'attention". On peut ainsi citer comme exemple VitPose (Xu et al., 2022a) qui découpe l'image d'entrée en *patchs* avant d'appliquer un *transformers* pour extraire les caractéristiques. RTMPose (Jiang et al., 2023) réutilise un principe similaire mais effectue les prédictions horizontales et verticales séparément à la place de l'approche traditionnelles par cartes de chaleur.

Parmi toutes les architectures décrites pour l'estimation de poses multi-personne, les méthodes descendantes restent aujourd'hui les plus performantes et les plus utilisées. Cela peut être en partie expliqué par le fait qu'elles tirent parti des progrès fait dans le domaine de l'estimation de poses mono-personne et de la détection de personne. On peut cependant mentionner OpenPose qui reste une architecture très populaire grâce aux auteurs qui tiennent à jour des dépôts de code source avec plusieurs démonstrations clés en main, les codes d'entraînement et une documentation très complète.

2.5.4 Métriques d'évaluation

Les performances des méthodes d'estimation de poses humaines 2D peuvent-être difficiles à évaluer puisqu'elles dépendent de beaucoup de critères : nombre de points clés visibles, nombre de personnes visibles, taille des sujets, etc.

Une des premières métriques utilisées est le *Percentage of Correct Parts* (PCP) (Eichner et al., 2012). Chaque point clé prédit est considéré correct si sa distance est inférieure à une fraction de la taille du membre auquel il appartient (par exemple 0,5). Le défaut de cette approche est qu'elle pénalise plus sévèrement les erreurs faites sur des points appartenant aux plus petites parties du corps, ces points étant déjà les plus difficiles à prédire étant donné la petite taille des membres. Pour atténuer ce problème, le *Percentage of Correct Keypoints* (PCK) (Yang and Ramanan, 2012) utilise le même seuil d'erreur pour tous les points de l'image. Les deux seuils classiquement utilisés sont 0,2 fois la taille du torse, et 0,5 fois la taille de la tête (cette variante est alors appelée PCKh, pour *head*). Ces métriques sont généralement utilisées pour évaluer des algorithmes sur des jeux de données mono-personnes, comme MPII ou LSP.

Une autre métrique est la précision moyenne (*Average Precision*, ou AP). Pour une détection mono-personne, APK (Yang and Ramanan, 2012) est calculée sur la détection de points clés du corps. Une détection est considérée comme un vrai positif si sa distance à la vérité terrain est inférieure à un seuil fixé, de la même manière que pour les métriques PCP et PCK, et faux positif sinon. Les équations 2.13, 2.14 et 2.15 résument le calcul des trois métriques avec $\hat{kp}_i$ le point i prédit, kp_i la vérité terrain associée, δ la fonction échelon unité, $nb_{points\_annotés}$ le nombre de points annotés de la vérité terrain et $nb_{points\_détectés}$ le nombre de points détectés.

$$\text{PCP} = \frac{\sum_i \delta(\|\hat{kp}_i - kp_i\| > \text{seuil}(kp_i))}{nb_{points\_annotés}} \tag{2.13}$$

$$\text{PCK} = \frac{\sum_i \delta(\|\hat{kp}_i - kp_i\| > \text{seuil}(i))}{nb_{points\_annotés}} \tag{2.14}$$

$$\text{APK} = \frac{\sum_i \delta(\|\hat{kp}_i - kp_i\| > \text{seuil}(kp_i))}{nb_{points\_détectés}} \tag{2.15}$$

Pour évaluer les méthodes d'estimation multi-personne, la plupart des métriques calculent la performance au niveau des personnes détectées et non plus des points individuellement. Ainsi, la métrique mAP (*mean Average Precision*) (Andriluka et al., 2014) appaire d'abord chaque individu détecté avec les personnes présentent dans la vérité terrain en calculant le PCK théorique pour chaque paire possible de détection / annotation. Ensuite, les personnes appairées et non appairées sont utilisées pour calculer la précision et le rappel moyen. Le jeu de données COCO propose une autre métrique pour que nous nommerons cocoAP. Cette métrique utilise l'*Object Keypoint Similarity* (OKS) score (Ruggero Ronchi and Perona, 2017), qui d'une manière comparable à l'*Intersection over Union* (IoU), mesure en se basant sur les points clés la distance entre les personnes détectées et la vérité terrain. Le score OKS est défini comme :

$$\text{OKS} = \frac{\sum_i \exp -\frac{d_i^2}{2.s^2.k_i^2}}{nb_{points\_annotés}} \tag{2.16}$$

où i itère sur chaque point annoté, d_i est la distance euclidienne entre les coordonnées prédites et la vérité terrain, s représente l'échelle de la personne par rapport à l'image et k_i est une constante pour chaque classe de point utilisée pour homogénéiser l'écart-type entre les différentes parties du corps. Une personne est alors considérée comme correctement prédite si son score OKS est supérieur à un seuil fixé entre 0 et 1. La précision AP est calculée comme présenté dans la Section 2.2.1.3, et le rappel AR comme le nombre total de personnes correctement prédites sur le nombre total de personnes annotées. Les valeurs de cocoAP (respectivement cocoAR) sont alors calculées comme la moyenne des différentes valeurs de AP (respectivement AR) obtenues en faisant varier le seuil d'acceptation d'OKS entre 0,5 et 0,95 avec un pas de 0,05.

Les métriques utilisées dans l'état de l'art pour évaluer une méthode d'estimation de poses dépendent généralement du jeu de données utilisé. En effet, chaque jeu de données est souvent associé à une métrique spécifique afin de pouvoir comparer les performances des méthodes d'estimation de poses sur un même jeu (MPII avec PCKh, COCO avec cocoAP). Bien que cette approche puisse convenir dans un contexte générique, les métriques classiques ne sont pas toujours les plus pertinentes dans des contextes particuliers. Par exemple, parce que COCO est initialement un jeu de données pour l'estimation de poses multi-personne, cocoAP mesure la précision des prédictions au niveau des personnes et non des points. Cette approche est moins appropriée pour une application mono-personne où l'on souhaite plutôt mesurer la précision de la détection des points. Il est donc nécessaire de s'interroger sur la pertinence des métriques utilisées selon le contexte d'étude.

2.6 Conclusion

Ce chapitre présente les différents travaux de l'état de l'art sur lesquels se basent cette thèse. Nous nous sommes concentrés sur les réseaux de neurones pour l'apprentissage profond et leur application pour l'estimation de poses. En effet, les réseaux de neurones, principalement convolutifs, permettent d'estimer efficacement la pose d'une ou plusieurs personnes sur une image couleur. Pour ce faire, la majorité des solutions de l'état de l'art utilisent une approche par détection qui génère des cartes de chaleur pour chaque point. Ainsi, plusieurs architectures ont été proposées pour effectuer cette tâche, en s'inspirant de réseaux pour d'autres tâches comme la classification d'objets. Ces solutions pour l'estimation de poses humaines 2D sur des images RGB permettent d'obtenir de bonnes performances dans un contexte général, avec des performances de détection à 94 % de précision sur MPII (Li et al., 2019b; Bulat et al., 2020) et avoisinant les 80 % de précision sur le jeu de validation COCO (Xiao et al., 2018; Cao et al., 2019; Li et al., 2019b; Cai et al., 2020). On observe cependant une saturation des performances sur ces jeux de données avec une augmentation très faible des performances au cours de ces dernières années.

Pour autant, les performances restent à évaluer dans un contexte plus spécifique comme un habitacle de voiture. Dans un tel contexte, des difficultés supplémentaires peuvent être attendues. Les réseaux pour l'estimation de poses entraînés sur des jeux de données génériques pourraient ne pas atteindre des performances aussi hautes sur des images de passager de voitures, puisque des difficultés comme les angles de vues ou le contraste induits par l'habitacle sont peu représentés dans les jeux d'apprentissages génériques. Pour autant, il existe peu de données disponibles, avec peu de diversité, pour entraîner un réseau de neurones dans ce contexte. Cela s'explique entre autres par la difficulté et le coût des expérimentations, et par la spécificité du domaine de recherche.

Un autre aspect à prendre en compte est la métrique d'évaluation. En effet, comme souligné dans ce chapitre, les métriques récentes de l'état de l'art utilisées dans les *benchmarks* récents tendent à calculer les scores tels que la précision ou le rappel au niveau des personnes et non des points. Dans un contexte comme le nôtre où la détection de la personne dans l'image représente une faible difficulté, on peut se questionner sur la pertinence de telles métriques pour estimer la qualité de la prédiction des points.

Chapitre 3

Premières contributions pour l'estimation de poses dans un habitacle de voiture

Dans ce chapitre, nous abordons en détail le sujet de l'estimation de poses dans un habitacle de voiture. Nous introduisons d'abord les problématiques soulevées par l'état de l'art et les illustrons par des résultats préliminaires, puis nous présentons les solutions proposées pour répondre à ces défauts. Premièrement, nous présentons DriPE, un jeu de données d'images de conducteurs en conditions réelles que nous avons mis en place pour l'estimation de poses. Nous décrivons ensuite la métrique mAPK que nous proposons pour permettre une évaluation plus précise de l'estimation de la position des points clés. Enfin, nous présentons un métamodèle pour la prédiction de la visibilité des points clés.

Sommaire

3.1 Premiers essais

Nous nous intéressons dans ces travaux à l'estimation de poses dans le contexte de l'habitacle de voiture. Comme décrit dans le Chapitre 2, l'état de l'art contient un grand nombre d'architectures pour estimer la pose d'une personne sur des grands jeux d'images génériques. Ainsi, le réseau *Stacked Hourglass* (Newell et al., 2016) atteint 91,8 % de points correctement estimés (PCKh@0.5) sur le jeu de test MPII, et RSN (Cai et al., 2020) atteint 79,2 et 84,1 % de précision et rappel moyens (cocoAP) respectivement sur le *benchmark* COCO. Cependant, plusieurs facteurs avantagent l'estimation de poses sur ces jeux de données génériques. Premièrement, les images qui les composent proviennent de sources variées et mettent souvent en scène des sujets vus de face ou légèrement de profil, dans de bonnes conditions photographiques et avec peu d'occultations. Cela facilite d'une part la détection des personnes et de leur pose sur les images, et habitue d'autre part le réseau à ces conditions. Cependant, ces caractéristiques ne correspondent pas à ce que l'on peut observer dans l'habitacle d'une voiture, en particulier lorsque l'on enregistre la personne de profil. Cet angle de capture est nécessaire pour permettre d'observer la majorité du corps dans le but d'extraire le maximum d'informations biomécaniques du haut (tête, épaules, coudes et poignets) comme du bas (hanches, genoux et chevilles) du corps. Les conséquences d'un enregistrement dans ces conditions est un angle de vue inhabituel de la personne et une forte auto-occultation (le côté gauche de la personne est caché par le reste du corps). Deuxièmement, le grand nombre d'images dans ces jeux de données entraîne un lissage des mesures de performance. Il est donc difficile d'estimer le comportement d'un réseau de neurones entraîné avec de tels jeux de données sur des images dans ce contexte particulier.

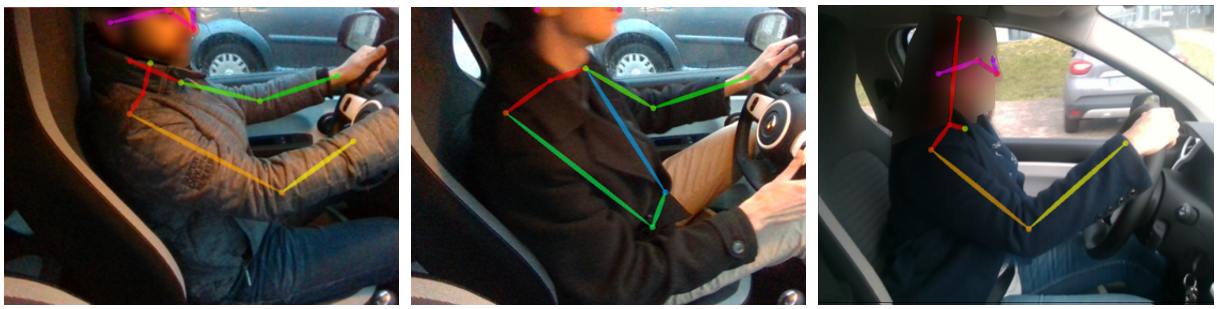
FIGURE 3.1 – Exemples d'images d'un jeu de données générique (MPII, Andriluka et al., 2014) à gauche, et dans le contexte d'un habitacle de voiture à droite. Les images de droite sont floutées sur le visage pour ce manuscrit.

Nous cherchons donc maintenant à évaluer comment des méthodes de l'état de l'art entraînées sur les bases génériques se comportent sur des images prises à l'intérieur d'une voiture. Pour ce faire, nous réalisons l'inférence d'un réseau de neurones avec une vingtaine d'images que nous capturons et annotons préalablement. Cet enregistrement est

réalisé avec un capteur Intel RealSense D435i (Intel RealSense, 2022) dans un véhicule personnel roulant sur le campus de l'Université Lyon 2. La caméra est placée au-dessus de la fenêtre du côté passager en légère contre-plongée pour permettre d'observer au mieux la personne assise côté conducteur. Le but de cette installation est de réaliser une première simulation des conditions pour l'expérimentation du projet AutoBehave. Nous extrayons de ces enregistrements 20 images couleur présentant trois conducteurs différents.

Nous choisissons comme réseau pour notre première évaluation le réseau OpenPose (Cao et al., 2019). Bien que ce réseau ne présente pas les meilleures performances de l'état de l'art, il reste néanmoins très performant et très populaire, avec l'avantage de proposer une implémentation complète et des poids entraînés fournis par les auteurs. Nous utilisons les poids du réseau entraîné à la fois sur COCO et sur MPII fournis par les auteurs.

Nous réalisons donc une inférence simple des images collectées avec OpenPose. Des illustrations des résultats sont visibles dans la Figure 3.2. Comme on peut l'observer, seul les bras et le visage sont correctement détectés lorsqu'ils sont totalement visibles. Les hanches et genoux quant à eux sont totalement ignorés. Cela peut s'expliquer par la position assise vue de côté qui est inhabituelle et rend difficile la reconnaissance de la forme globale du corps, en particulier du bas. De plus, le bras droit non détecté dans l'image (b) semble indiquer que des habits trop amples ou foncés compliquent la détection, puisque le haut du corps contraste moins avec le reste de l'image.



(a) **(b)** **(c)**

FIGURE 3.2 – Premiers résultats d'inférence sur des images de conducteurs avec OpenPose. Les images sont floutées sur le visage pour ce manuscrit.

Ces résultats semblent indiquer que les réseaux de neurones entraînés sur les jeux de données de l'état de l'art sont mis en difficulté par le contexte de l'intérieur de voiture. Cependant, nous ne pouvons pas tirer de conclusions précises en se basant sur si peu d'images, et avec peu de diversité dans les conditions d'expérimentations (véhicule et angle de vue identique, peu de conducteurs). Comme indiqué dans le Chapitre 2, il n'existe pas de jeu de données disponible publiquement contenant des images annotées pour l'estimation de poses, enregistrées dans un environnement réel de véhicule et dans des conditions variées. Il est donc nécessaire de combler ce manque de l'état de l'art pour pouvoir valider ces observations préliminaires.

3.2 Jeu de données DriPE

Nous décidons donc de créer un jeu de données pour répondre à nos besoins contenant des images enregistrées en conditions de roulage réelles en extérieur, ce qui n'est actuellement pas disponible dans l'état de l'art pour l'estimation de poses. Nous utilisons pour

cela des vidéos de plusieurs conducteurs dans différents véhicules, enregistrées principalement au cours d'une expérimentation précédant le début de cette thèse. Ces vidéos ont été enregistrées sur le centre d'essais Transpolis [10], un centre d'expérimentation pour les transports terrestres disposant d'infrastructures routières. Cette section décrit la procédure suivie pour créer le jeu de données DriPE (*Driver Pose Estimation*) à partir de ces enregistrements.

3.2.1 Extraction des images

Nous disposons initialement de 500 clips vidéo d'environ 5 minutes chacun enregistré sur Transpolis. À cela s'ajoute 200 clips d'une à deux minutes enregistrés avec le dispositif utilisé pour acquérir les images présentées dans la section précédentes. Nous disposons au total d'environ 41 heures d'enregistrement vidéo illustrant 19 personnes assises dans différentes voitures, en situation de conduite ou à l'arrêt. Un premier nettoyage est réalisé manuellement pour retirer les segments où les véhicules sont vides (lors des débuts et fins de séquences d'enregistrement principalement) et les clips inutilisables dû à des dysfonctionnements du système de captation. Ce premier traitement réduit la durée totale à 32 heures.

Il est ensuite nécessaire d'extraire les images des vidéos pour constituer notre jeu de données. En prenant en compte le taux d'images par seconde différent selon les contextes d'enregistrement, le nombre d'images total est de 3.3 millions. Cependant, 19 conducteurs sont représentés dans ces images, avec des conditions d'enregistrement qui varient peu pour un même sujet. Ainsi, effectuer un entraînement ou une évaluation sur la totalité de ces données apporterait peu d'informations supplémentaires tout en allongeant excessivement les temps de calcul. De plus, cela rendrait l'annotation manuelle du jeu de données impossible. Plusieurs méthodes sont envisageables pour réduire le nombre d'images. Par exemple, il est possible de choisir aléatoirement parmi l'ensemble des clips, ou de choisir les images selon un pas de parcours fixe. À la place, nous optons pour une méthode cherchant à sélectionner les images les plus intéressantes vis-à-vis des variations de postures, de luminosité, d'arrière-plans, etc. Nous utilisons pour cela deux métriques :

- La similarité structurelle (*Structural SIMilarity*, ou SSIM) (Wang et al., 2004). Cette métrique a été créée dans l'optique de donner une indication sur la similarité visuelle perçue entre deux images. Elle se base sur trois composants qui sont la comparaison de la luminosité, du contraste et de la structure :

$$\mathrm{SSIM}(x, y) = l(x, y) \cdot c(x, y) \cdot s(x, y) \tag{3.1}$$

avec

$$\text{luminosité} : l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{3.2}$$

$$\text{contraste} : c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{3.3}$$

$$\text{structure} : s(x, y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{3.4}$$

avec x et y les deux images comparées, et μ et σ les moyennes et écarts-types respectifs de la valeur des pixels. Trois constantes c_1, c_2, c_3 sont utilisées pour stabiliser

10. Transpolis (69, Rhône) : `https://transpolis.fr/fr/`

les dénominateurs. En pratique, l'image est découpée en *patchs* obtenus avec une fenêtre spatiale glissante, et la métrique est calculée sur chaque *patch*. La similarité totale entre les deux images est alors calculée comme la moyenne de toutes les valeurs.

- La différence de luminosité, exprimée comme :

$$\Delta \mathrm{V}(x, y) = |V_x - V_y| \tag{3.5}$$

$$\text{avec} \quad V_x = \max(R_x, G_x, B_x) \tag{3.6}$$

où R_x, G_x, B_x sont les trois canaux de couleur de l'image x.

Le choix de ces métriques est motivé par deux aspects. Premièrement, nous recherchons des images avec des différences visuelles fortes afin d'assurer que l'échantillon des images sélectionnées soit bien représentatif de l'ensemble des caractéristiques visuelles présentes dans les clips vidéo. Ainsi, en calculant la similarité structurelle entre deux images successives, nous estimons qu'une différence plus importante révèle un fort changement visuel, comme une posture prise par la personne ou un arrière-plan inhabituel. En outre, bien que la similarité structurelle prenne en compte la luminosité dans son calcul, nos expériences ont montré que cette dernière jouait un rôle important dans le succès, ou non, de la détection des points clés. En effet, comme montré dans la Figure 3.2, les conditions inhabituelles de luminosité peuvent modifier la couleur des textures comme la peau ou les vêtements ; ce qui peut perturber la performance des réseaux. En mesurant la variation de luminosité entre deux images successives, nous cherchons à détecter les forts changements qui pourraient représenter des difficultés pour les réseaux comme d'importantes ombres portées, ou un fort ensoleillement soudain.

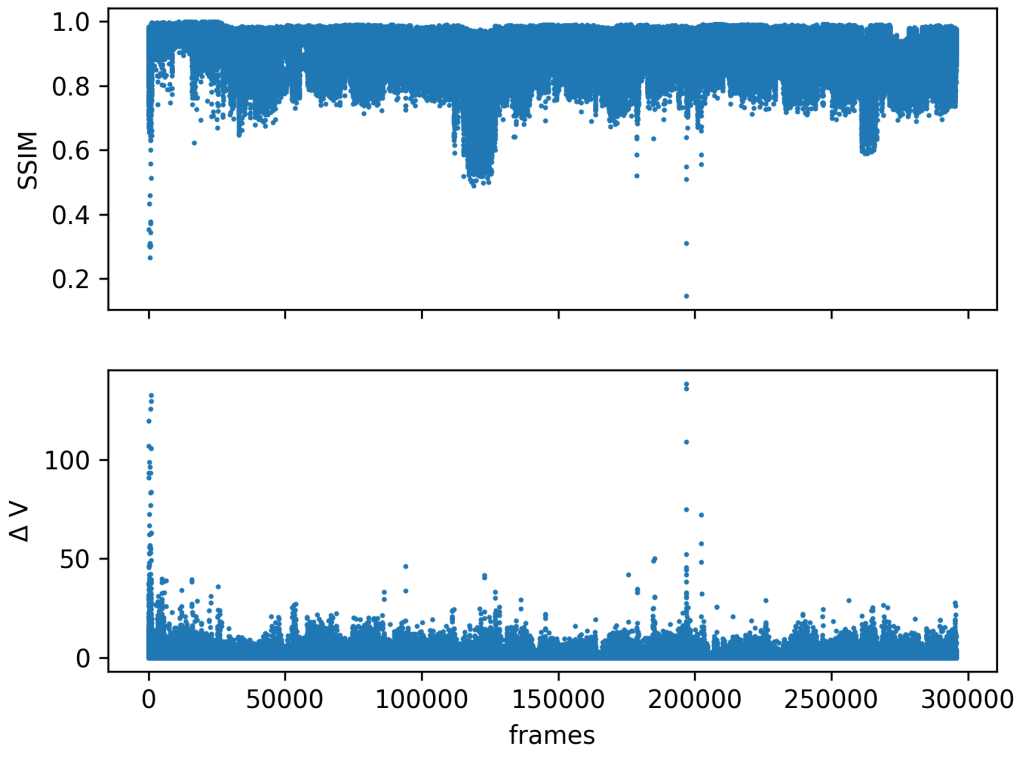


FIGURE 3.3 – Métriques calculées sur les images des clips vidéo de conduite.

FIGURE 3.4 – Exemples d'images présentant différentes valeurs pour les métriques SSIM et ΔV calculées. Les images sont floutées sur le visage pour ce manuscrit.

En pratique, les clips vidéo sont échantillonnés avec une fréquence de 1 image sur 10 sans quoi la différence entre deux images successives est trop faible, et les métriques sont calculées entre deux échantillons successifs de la même vidéo. Une visualisation des résultats est visible dans la Figure 3.3. Il est alors nécessaire de définir un seuil pour chaque métrique afin de sélectionner les images. Nous cherchons les images présentant les plus petites valeurs pour la SSIM et les plus grandes pour la différence de luminosité, ce qui indiquerait des images inhabituelles et donc intéressantes pour le jeu de données. Nous tentons d'abord de définir ces seuils en nous basant sur des observations faites sur des paires d'images avec différents scores. Cependant, comme on peut le voir dans la Figure 3.4, les observations visuelles ne permettent pas de définir des seuils au-dessus desquels les images seraient considérées comme plus intéressantes.

À la place, nous décidons de fixer arbitrairement le nombre d'images à extraire d'après chaque métrique à 5 000, soit un jeu de données total de 10 000 images. De plus, étant donné l'inégale répartition du temps d'enregistrement entre les conducteurs (2 h 20 pour les plus représentés, 20 minutes pour les moins), nous définissons un nombre minimum

FIGURE 3.5 – Échantillon d'images sélectionnées pour DriPE. Les images sont floutées sur le visage pour ce manuscrit.

et maximum d'images à extraire par clip vidéo. Ces quotas ont pour but de rééquilibrer l'échantillonnage afin d'obtenir une meilleure représentation de la diversité des sujets. Enfin, nous imposons un écart minimum de 2 secondes entre deux images sélectionnées afin de limiter une trop forte ressemblance entre les images. La Figure 3.5 présente un échantillon des images sélectionnées.

Nous formons ainsi un jeu de données pour l'estimation de poses humaines de conducteurs en conditions réelles présentant des changements d'illuminations et d'arrière-plans, des ombres occultantes, etc. Le jeu de données est composé de 10 000 images avec 19 sujets dont 7 femmes et 12 hommes. Le Tableau 3.1 résume les statistiques et compare notre jeu de données à Drive&Act (Martin et al., 2019), le seul autre jeu d'images réelles de conduite proposant des pseudo-annotations de la pose. Cependant, ces images ont été enregistrées dans un simulateur et la pose n'a pas été annotée manuellement, contrairement à celles de notre jeu de données. Ainsi, le jeu DriPE illustre plus de sujets avec une meilleure répartition femme / homme. À l'inverse, Drive&Act possède théoriquement plus d'images puisque le jeu de données est constitué de clips vidéo qui n'ont pas été échantillonnés comme nous l'avons fait pour DriPE.

	Drive&Act	DriPE
N° sujets	15	**19**
Femme / Homme	4 / 11	**7 / 12**
Annotation	Réseau de neurones	**Manuelle**
RGB	✓	✓
Profondeur	✓	-
I. R.	✓	-
N° images	**9,6M** (vidéos)	10k
Contexte de conduite	Simulateur de conduite	**Conditions réelles**

TABLEAU 3.1 – Comparaison entre le jeu de données Drive&Act et notre jeu de données DriPE.

3.2.2 Annotation

Pour l'annotation des images, nous souhaitons annoter les points du corps, mais aussi ceux du visage puisque ces informations sont nécessaires pour beaucoup de tâches liées au contexte de la voiture, telles que la détection d'attention ou la reconnaissance d'activités (Borghi et al., 2017; Jegham et al., 2019; Borghi et al., 2020). Nous décidons donc de suivre le système d'annotation du jeu de données COCO qui annote les quatre membres ainsi que les yeux, le nez et les oreilles. Outre les points disponibles, l'utilisation de cette nomenclature facilite l'utilisation de notre jeu de données avec toutes les applications de l'état de l'art utilisant COCO (codes d'entraînement, d'évaluation, etc.). La nomenclature inclut également une boite englobante nécessaire pour le calcul des métriques liées au *benchmark* COCO, ainsi que la classe de visibilité des points.

Nous annotons manuellement toutes les images. Pour ce faire, nous avons développé un logiciel disposant d'une interface simple permettant la synchronisation des annotations sur un serveur de stockage (Figure 3.6). Le développement de ce logiciel d'annotation a permis d'annoter efficacement et de manière collaborative notre jeu de données en organisant des séances d'annotations au sein de notre équipe. Une pré-annotation générée par OpenPose est proposée à l'utilisateur, qui n'a qu'à modifier les points erronés, ajouter les points manquants, la visibilité et la boîte englobante.

FIGURE 3.6 – Interface du logiciel d'annotation manuel.

3.2.3 Évaluation de l'état de l'art sur le jeu de données DriPE

Maintenant que nous avons à disposition un grand nombre d'images annotées de conducteurs dans des conditions variées, nous pouvons évaluer différents réseaux de l'état de l'art pour l'estimation de poses dans ce contexte.

3.2.3.1 Choix des architectures

Nous choisissons pour cette évaluation trois architectures de l'état de l'art : Simple Baseline (SBl) (Xiao et al., 2018), MSPN (Li et al., 2019b) et RSN (Cai et al., 2020). Les architectures de ces réseaux sont présentées dans la Section 2.5.3. Nous décidons d'intégrer dans notre étude un réseau mono-personne (Simple Baseline) et deux réseaux multi-personnes (MSPN et RSN), malgré le fait que le jeu de données DriPE soit seulement mono-personne. Ce choix est motivé par plusieurs points. Premièrement, le jeu de données COCO, dont DriPE reprend la représentation de la pose, est majoritairement utilisé pour de l'évaluation multi-personne. Deuxièmement, puisque l'état de l'art récent se concentre sur l'estimation de poses multi-personne, les réseaux les plus récents et performants font partie de cette catégorie. Troisièmement, comme expliqué dans le Chapitre 2, la majorité des solutions actuelles pour l'estimation de poses multi-personne (comme c'est le cas de MSPN et RSN) sont composées d'un détecteur de personne suivi d'un estimateur de pose mono-personne sur les boites détectées. L'utilisation de ces deux architectures reste donc pertinente dans notre contexte.

3.2.3.2 Entraînement des modèles

L'entraînement des modèles est effectué à l'aide du code fourni par les auteurs respectifs, en suivant leurs recommandations pour les hyperparamètres. Tous les entraînements sont réalisés sur le jeu d'entraînement COCO, avec des *batchs* de 32 images et des opérations d'augmentation des données (retournement horizontal, rotation, changement d'échelle). Ainsi, Simple Baseline est entraîné pendant 140 *epochs* sur COCO avec un taux d'apprentissage de $1E^{-3}$ qui est divisé par 10 à la 90^{e} *epoch*. Son extracteur de caractéristiques (ResNet50) est initialisé avec les poids d'un réseau entraîné pour la classification sur ImageNet. RSN et MSPN sont entraînés pendant 160 *epochs*, avec un taux d'apprentissage de base de $5E^{-4}$ diminuant linéairement jusqu'à 0, et les poids sont

initialisés aléatoirement. Les trois architectures sont entraînées en utilisant l'algorithme d'optimisation ADAM (Kingma and Ba, 2014) avec des taux pour les moyennes mobiles exponentielles de $\beta_1 = 0.9$ et $\beta_2 = 0.999$. Les entraînements sont réalisés sur un ordinateur équipé de deux cartes graphiques Nvidia Titan RTX avec 24 Go de VRAM, un processeur Intel i9900k et 64 Go de RAM. Pour l'évaluation des réseaux, nous prenons les poids de l'*epoch* à laquelle la plus faible erreur sur le jeu de validation a été mesurée. Les prédictions finales pour chaque point clé est le pixel ayant la plus haute valeur sur chaque carte de chaleur correspondante. Les prédictions avec une valeur inférieure à un seuil de confiance ne sont pas conservées afin d'éliminer le bruit sur les cartes de chaleur. Ce seuil est fixé à 0,25.

3.2.3.3 Résultats et discussion

Nous utilisons dans cette étude la métrique cocoAP, qui est la procédure d'évaluation standard pour le *benchmark* HPE COCO (Lin et al., 2014) que nous décrivons plus en détail dans la Section 2.5.4. Nous commençons par évaluer la performance des réseaux sur le jeu de validation COCO (le jeu de test n'étant pas publiquement disponible). Les résultats sont montrés dans le Tableau 3.2. Comme on peut le voir dans ce tableau, les métriques calculées sont réparties en deux catégories : la précision moyenne (AP) et le rappel moyen (AR). De plus, plusieurs sous-valeurs sont calculées. Pour rappel, AP (respectivement AR) est calculée en faisant varier le seuil d'acceptation des prédictions entre 50 et 100 % et en calculant la moyenne obtenue pour ces seuils des valeurs de précision (respectivement rappel). AP^X, où X est un pourcentage, indique la valeur de précision obtenue uniquement avec un seuil d'acceptation de X. Enfin, AP^M et AP^L indique la valeur calculée uniquement sur les personnes dont l'aire sur l'image est comprise entre 32^2 et 96^2 pixels pour les personnes de taille moyenne (M) ou supérieure à 96^2 pixels pour les personnes de taille large (L).

	AP	AP^{50}	AP^{75}	AP^M	AP^L	AR	AR^{50}	AR^{75}	AR^M	AR^L
SBl	73	94	80	73	78	76	92	82	75	80
MSPN	**77**	94	85	75	82	**80**	95	87	77	85
RSN	76	94	84	74	81	79	94	85	76	84

TABLEAU 3.2 – Évaluation de l'estimation de poses sur le jeu de validation de COCO (cocAP en %).

Comme nous pouvons le constater, les réseaux démontrent de bonnes performances, avec une précision et un rappel entre 70 et 80 %. Ainsi, nous validons que les modèles entraînés atteignent une performance proche du travail original (environ 2 % d'écart en moyenne). Nous pouvons noter que Simple Baseline présente des performances légèrement inférieures aux deux autres réseaux, ce qui peut être expliqué par son architecture plus simple et un nombre inférieur de paramètres.

Nous évaluons ensuite la performance sur l'ensemble de test de DriPE en utilisant les modèles entraînés sur COCO. Les résultats sont présentés dans le Tableau 3.3. En raison de l'emplacement de la caméra dans la voiture, DriPE ne contient que des sujets "larges". Par conséquent, il est plus approprié de comparer les performances sur les jeux de données COCO et DriPE en utilisant les valeurs des colonnes AP^L et AR^L puisque les performances sur les sujets "moyen" ne sont pas mesurables (ce qui explique les "-" dans les colonnes AP^M et AR^M).

	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR	AR^{50}	AR^{75}	AR^{M}	AR^{L}
SBl	76	99	90	-	76	82	99	95	-	82
MSPN	81	99	97	-	**81**	85	99	97	-	**85**
RSN	75	99	93	-	75	79	99	95	-	79

TABLEAU 3.3 – Performances des réseaux sur DriPE (cocoAP en %).

Comme le montre les résultats, les réseaux de l'état de l'art permettent d'obtenir des performances comparables sur le jeu de données DriPE et sur COCO (Tableau 3.2 et Tableau 3.3). Bien que ces deux jeux de données n'aient pas la même taille et la même diversité d'images, comparer les performances mesurées sur ces deux jeux permet de mettre en évidence que DriPE propose des situations qui ne sont pas couvertes dans le jeu de données COCO. D'une part, nous constatons qu'en moyenne, la précision et le rappel sont légèrement plus faibles sur DriPE que sur COCO. D'autre part, nous observons des scores de précision et de rappel très élevés sur les trois réseaux lorsque nous utilisons un seuil OKS de 50 % (AP^{50}) ou de 75 % (AP^{75}). Ces résultats semblent suggérer que les réseaux de l'état de l'art atteignent déjà de bonnes performances sur les images de DriPE. De plus, les diminutions de performances sont principalement dues à de faibles erreurs de détection d'après la métrique OKS, puisqu'un seuil d'acceptation à 75 % conserve la majorité des prédictions.

	AP	AP^{50}	AP^{75}	AP^{L}	AR	AR^{50}	AR^{75}	AR^{L}
SBl	97	100	83	**97**	97	100	99	99
MSPN	97	100	99	**97**	98	100	99	**98**
RSN	91	99	98	91	94	100	99	94

TABLEAU 3.4 – Performances des réseaux sur DriPE après *fine-tuning* (cocoAP en %).

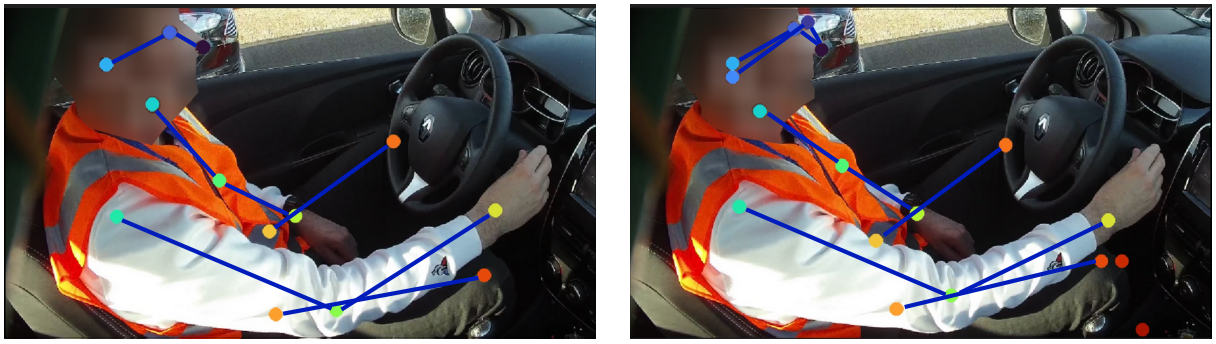
Nous procédons ensuite à un *fine-tuning* des trois réseaux sur l'ensemble d'entraînement DriPE. Cette nouvelle étape d'entraînement est effectuée durant 10 *epochs* avec un taux d'apprentissage égal à $1E^{-4}$. Les résultats dans le Tableau 3.4 indiquent un gain de 20 à 25 % en précision et de 10 à 15 % en rappel après le *fine-tuning* des réseaux. Cette augmentation peut s'expliquer en partie par la variance relativement faible du jeu de données DriPE qui induit une faible différence entre l'ensemble d'entraînement et celui d'évaluation. Malgré cela, l'amélioration des performances suggère que les réseaux ont appris des caractéristiques spécifiques sur DriPE qu'ils n'avaient pas apprises sur un jeu de données générique comme COCO ; ce qui souligne la pertinence du jeu de données DriPE dans ce domaine. Finalement, les résultats mesurés par cocoAP peuvent suggérer que l'estimation de poses à l'intérieur d'une voiture serait un problème presque résolu en utilisant du *fine-tuning*.

3.3 Métrique d'évaluation mAPK

3.3.1 Problématique

Comme montré dans les résultats de la section précédente, les performances des réseaux de l'état de l'art mesurées avec la métrique cocoAP semblent indiquer que ces solutions permettent de résoudre correctement la tâche d'estimation de poses. En effet, les performances mesurées sont bonnes sur un jeu de données générique comme COCO

et ce constat est d'autant plus vrai dans le contexte de l'intérieur d'une voiture illustré par DriPE, moyennant un léger *fine-tuning*. Cependant, l'analyse de résultats qualitatifs ne semble pas refléter les 97 % de précision et de rappel mesurés, comme montré dans la Figure 3.7. En particulier, plusieurs points sont détectés par le réseau, alors qu'ils ne sont pas annotés ; soit parce qu'ils sont en dehors du cadre de l'image, soit parce qu'ils sont considérés comme trop fortement occultés pour être localisés avec précision. Ce phénomène est observé en particulier sur les points du visage, et sur les chevilles.



Vérité terrain Prédiction

FIGURE 3.7 – Exemples de prédictions réalisées par le réseau Simple Baseline après *fine-tuning* sur DriPE. Les images sont floutées sur le visage pour ce manuscrit.

Bien que ces erreurs de détection ne soient pas majoritaires, elles devraient faire baisser les performances mesurées par les métriques, et en particulier la précision puisqu'il semblerait que le réseau soit surconfiant dans ses prédictions. Cela nous amène donc à nous interroger sur la méthode de calcul de cocoAP et sur la pertinence de son utilisation dans notre contexte. Pour rappel, le calcul de la métrique cocoAP utilisée dans le *benchmark* COCO se base sur le score OKS défini dans l'Équation 2.16 et est calculé d'après l'Algorithme 1.

Plusieurs reproches peuvent être faits à cet algorithme. Premièrement, le score OKS prend en compte dans son calcul uniquement les points avec une visibilité positive dans la vérité terrain, autrement dit les points annotés. Ainsi, quelle que soit la confiance avec laquelle le réseau prédit un point non annoté, la mesure de performance finale ne sera pas impactée. Cela tend à favoriser un réseau surconfiant qui chercherait au maximum à prédire des coordonnées pour chaque point sans prendre en compte leur présence dans l'image. Ce comportement peut être problématique dans un contexte où beaucoup de points ne seraient pas annotés à cause de fortes occultations, comme par exemple à l'intérieur d'une voiture avec une caméra placée sur le côté. Secondement, la précision et le rappel sont calculés en termes de personnes correctement détectées et appairées. Cette approche est utilisée pour répondre à une problématique multi-personne où il est question de détecter les personnes dans l'image et de leur attribuer leurs points clés respectifs. Cependant, cela empêche d'obtenir une mesure détaillée de la précision et du rappel de l'estimation de la pose des points du corps et ne semble donc pas pertinent dans un contexte où l'on cherche à connaître la performance au niveau des points du squelette. En se basant sur ces observations, une métrique alternative semble nécessaire pour pouvoir caractériser au mieux les performances des réseaux de l'état de l'art dans un contexte mono-personne avec beaucoup d'occultations.

Algorithme 1 : Calcul de cocoAP

Entrées :
$personnes\_appairées$: paires (gt, dt) des personnes appairées de la vérité terrain et de la détection
$dts\_non\_appairées$: personnes détectées non appairées
$gts\_non\_appairées$: personnes de la vérité terrain non appairées
$seuil\_score$: seuil d'acceptation pour le score OKS
Sorties : AP, AR

$vrais\_positifs = 0$, $faux\_positifs = 0$, $faux\_négatifs = 0$
pour chaque (gt, dt) dans $personnes\_appairées$ **faire**
  si OKS$(gt, dt) > seuil_score$ **alors**
    $vrais\_positifs \mathrel{+}= 1$
  sinon
    $faux\_positifs \mathrel{+}= 1$
    $faux\_négatifs \mathrel{+}= 1$

pour chaque $personne$ dans $gts\_non\_appairées$ **faire**
  $faux\_négatifs \mathrel{+}= 1$
pour chaque $personne$ dans $dts\_non\_appairées$ **faire**
  $faux\_positifs \mathrel{+}= 1$

AP = calculer_AP$(vrais_positifs, faux_positifs)$
AR = calculer_AR$(vrais_positifs, faux_négatifs)$

3.3.2 Définition de mAPK

Pour répondre aux insuffisances de la métrique cocoAP dans notre contexte, nous souhaitons utiliser une métrique qui se calcule à l'échelle des points et qui prend en compte les points faussement détectés. La métrique APK (Yang and Ramanan, 2012) présentée dans la Section 2.5.4 répond à ces prérequis puisqu'elle évalue la bonne estimation de chaque point et l'utilise pour calculer la précision moyenne. Nous souhaitons cependant apporter deux modifications à cette métrique. Premièrement, le seuil d'acceptation dans APK pour la distance entre un point et sa vérité terrain est fixé comme un pourcentage de la taille d'une partie du corps, généralement 50 % de la taille de la tête ou 20 % de la taille du torse. Ces mensurations sont difficiles à calculer dans notre contexte puisque, d'une part, le sommet du crâne n'est pas annoté dans le standard ; COCO ce qui empêche le calcul de la taille de la tête, et d'autre part, le torse n'est pas toujours entièrement visible par la caméra. Pour remédier à ce problème, il serait possible d'ajouter l'une de ces informations dans les vérités terrain de DriPE. Cependant, cela rendrait la métrique non compatible avec les autres méthodes basées sur COCO qui est aujourd'hui le jeu de données majoritairement utilisé dans l'état de l'art pour l'estimation de poses. Nous décidons donc à la place d'utiliser une méthode analogue au calcul du score OKS en réutilisant le *keypoint score* (ks) défini comme :

$$\text{ks}(i) = \exp -\frac{d_i^2}{2.s^2.k_i^2} \tag{3.7}$$

où d_i est la distance euclidienne entre les coordonnées prédites du point i et la vérité terrain associée, s représente l'échelle de la personne par rapport à l'image et k_i est une constante

pour chaque classe de point visant à homogénéiser l'écart-type entre les différentes parties du corps. Cette approche permet de normaliser l'erreur sur chaque point et d'appliquer un seuil entre 0 et 1 pour valider ou non la détection.

Algorithme 2 : Calcul de mAPK

Entrées :
personnes_appairées : paires (gt, dt) de personnes appairées de la vérité terrain et de la détection
dts_non_appairées : personnes détectées non appairées
gts_non_appairées : personnes de la vérité terrain non appairées
seuil_score : seuil d'acceptation pour le score ks
Sorties : AP, AR

$vrais\_positifs = 0$, $faux\_positifs = 0$, $faux\_négatifs = 0$
pour chaque (gt, dt) dans *personnes_appairées* **faire**
  pour chaque point kp dans la représentation du squelette **faire**
    si $dt[kp]$ n'est pas vide et $gt[kp]$ est vide **alors**
      $faux\_positifs \mathrel{+}= 1$
    sinon si $dt[kp]$ est vide et $gt[kp]$ n'est pas vide **alors**
      $faux\_négatifs \mathrel{+}= 1$
    sinon
      si $\text{ks}(gt[kp], dt[gp]) > seuil\_score$ **alors**
        $vrais\_positifs \mathrel{+}= 1$
      sinon
        $faux\_positifs \mathrel{+}= 1$
        $faux\_négatifs \mathrel{+}= 1$

pour chaque *point* dans les *gts_non_appairées* **faire**
  $faux\_négatifs \mathrel{+}= 1$
pour chaque *point* dans les *dts_non_appairées* **faire**
  $faux\_positifs \mathrel{+}= 1$

$\text{AP} = \text{calculer\_AP}(vrais\_positifs, faux\_positifs)$
$\text{AR} = \text{calculer\_AR}(vrais\_positifs, faux\_négatifs)$

Le calcul de la métrique est alors effectué comme suit. Tout d'abord, nous calculons un score ks pour chaque point annoté et détecté. Un point est considéré comme correctement détecté, c'est-à-dire comme un vrai positif (TP), si son score ks dépasse un seuil fixé. Dans le cas contraire, nous considérons que le point prédit et sa vérité terrain ne sont pas appairés. Nous considérons ensuite tous points non appairés parmi les prédictions comme des faux positifs, et ceux de la vérité terrain comme des faux négatifs. Enfin, nous calculons la précision et le rappel pour chaque type de point.

Pour le choix du seuil, une possibilité serait de fixer une valeur unique. À la place, nous décidons de suivre l'approche du *benchmark* COCO. Ainsi, cette mesure est répétée avec différentes valeurs de seuils d'acceptation (de 0,5 à 0,95, avec un pas de 0,05) et une moyenne est calculée pour obtenir la performance finale de la méthode évaluée. Cette approche permet de s'affranchir de la dépendance de notre métrique au choix du seuil en lissant les performances entre un seuil assez permissif et un seuil plus strict. Nous

nommons cette métrique mAPK, pour *mean Average Precision Keypoint*. Sa méthode de calcul est résumée dans l'Algorithme 2.

3.3.3 Résultats

Nous pouvons maintenant évaluer la performance des mêmes réseaux de l'état de l'art, mais au niveau de la prédiction des points clés et non des personnes. Pour ce faire, nous recalculons les performances des modèles évalués précédemment (Tableaux 3.2 et 3.3) en utilisant la métrique mAPK (Tableaux 3.5 et 3.6).

		Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	SBl	45	69	61	54	64	62	60	**59**
	MSPN	49	76	60	53	62	47	40	55
	RSN	49	76	59	52	61	46	39	55
AR	SBl	82	86	83	80	80	82	80	83
	MSPN	87	88	87	84	82	85	85	**86**
	RSN	86	88	86	83	82	84	84	85

TABLEAU 3.5 – Performances de l'estimation de poses sur COCO, évaluées avec mAPK (en %).

Tout d'abord, et bien que les valeurs des métriques cocoAP et mAPK ne soient pas directement comparables, nous pouvons observer que les scores de rappel mesurés par les deux métriques sont dans un ordre de grandeur proche (environ 75 %) (Tableaux 3.2, 3.3, 3.5, et 3.6). À l'inverse, les scores de précision moyenne mesurés par mAPK sont plus faibles. Cette baisse de la précision s'explique par le nombre élevé de faux positifs pris en compte par mAPK mais ignorés par OKS, comme le montre le Tableau 3.7. Après analyse, nous avons déterminé que la plupart des faux positifs proviennent des points non annotés, et non d'une distance trop grande entre l'estimation et la vérité terrain. Ces résultats montrent que les réseaux sont trop confiants dans leur prédiction et ne sont pas en mesure de détecter correctement l'absence d'un point clé sur l'image. Cette information ne pouvait pas être observée sur les résultats d'évaluation mesurés avec cocoAP puisque la précision est calculée à l'échelle d'une personne.

		Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	SBl	28	85	79	92	91	76	15	**67**
	MSPN	25	80	77	90	91	77	13	65
	RSN	25	78	76	89	88	68	11	62
AR	SBl	89	91	93	96	88	63	9	75
	MSPN	96	87	96	97	92	77	45	**85**
	RSN	94	85	95	96	89	68	33	81

TABLEAU 3.6 – Performances de l'estimation de poses sur DriPE, évaluées avec mAPK (en %).

Cette surconfiance peut en partie être expliquée par la méthode de calcul de la fonction de coût utilisée pour ces réseaux. En effet, puisque la métrique cocoAP prend en considération uniquement les points annotés, l'erreur d'entraînement est également calculée uniquement pour ces derniers (Xiao et al., 2018). Ainsi, si un point est prédit mais non annoté, aucune erreur ne sera rétro-propagée dans le réseau. Nous montrons dans la

section suivante (Tableau 3.11) les conséquences de l'utilisation d'une fonction de coût calculée sur la totalité des points lors de l'entraînement.

On peut également noter que même si les points de la tête sont en général parmi les points les plus simples à détecter (souvent visibles, placements fixes les uns par rapport aux autres), les modèles entraînés ont atteint une précision moyenne très faible pour leur détection. En effet, le nombre total de faux positifs est environ trois fois plus élevé que le nombre de vrais positifs (Tableau 3.7). Ceci peut en partie s'expliquer par le fait que la politique d'annotation du jeu de données COCO n'annote pas les points de la tête occultés. Ces résultats soulignent que les modèles actuels ont des difficultés à ne pas détecter les points, c'est-à-dire à identifier lorsqu'un point n'est pas visible. Par ailleurs, les modèles montrent des performances sur DriPE très faibles pour la détection des chevilles, à la fois en termes de précision et de rappel. Les chevilles sont généralement difficiles à prédire, en particulier à l'intérieur d'une voiture où les membres inférieurs sont presque totalement occultés par le tableau de bord. Cette difficulté d'occultation, associée au faible contraste et à la faible luminosité, rend la détection des chevilles très difficile.

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
GT	17k	25k	21k	26k	26k	26k	11k	152k
VP	**16k**	21k	20k	23k	23k	18k	**2,8k**	124k
FP	**50k**	5,7k	6,4k	3,1k	3,1k	8,4k	**24k**	100k
FN	0,7k	3,8k	1,1k	2,9k	3,0k	8,3k	8,2k	28k

TABLEAU 3.7 – Nombre de points prédits par le modèle RSN sur le jeu DriPE avec la métrique mAPK (en %) avec : GT le nombre de points annotés, VP les vrais positifs, FP les faux positifs et FN les faux négatifs.

Enfin, nous comparons l'évaluation des réseaux après *fine-tuning* sur DriPE en utilisant mAPK (Tableau 3.8). Tout d'abord, nous pouvons observer que cette métrique confirme l'augmentation des performances de prédiction apportée par l'entraînement supplémentaire montrée préalablement par cocoAP (Tableau 3.4). Ensuite, on peut remarquer que le *fine-tuning* ne permet pas d'endiguer la surconfiance des réseaux puisque la précision mesurée par mAPK reste inférieure au rappel. Ces résultats soulignent l'importance de DriPE pour améliorer les performances des modèles actuels d'estimation de poses des personnes dans le contexte des habitacles de voiture. Mais ils attirent également l'attention sur les défis ouverts en matière de prédiction des points clés qui ne peuvent pas être résolus par un simple *fine-tuning* des modèles actuels sur un jeu de données spécifique. Ce constat est appuyé par la précision mesurée sur les points de la tête et des chevilles qui reste très faible (malgré une amélioration notable sur les chevilles). Il est intéressant de

		Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	SBl	24	91	79	93	98	98	40	**75**
	MSPN	25	89	79	91	97	94	38	73
	RSN	25	88	78	91	95	86	30	70
AR	SBl	94	97	98	98	98	98	95	**97**
	MSPN	97	97	98	99	98	94	87	96
	RSN	91	95	98	98	95	86	73	91

TABLEAU 3.8 – Performances sur le jeu DriPE après *fine-tuning*, évaluées avec mAPK (en %).

noter que Simple Baseline surpasse les méthodes plus récentes et complexes selon mAPK. Ce constat peut être observé sur les deux jeux de données, en particulier sur les valeurs de précision ; ce qui indique un nombre de faux positifs plus faible.

Nous avons donc proposé la métrique mAPK, une extension des métriques d'évaluation APK et OKS. Les résultats indiquent qu'elle permet de caractériser plus précisément les performances des méthodes d'estimation de poses en termes de détection des points, à la fois sur les jeux de données génériques et sur des images de conducteurs. Elle a permis de mettre en lumière que les réseaux de neurones de l'état de l'art étaient entraînés pour maximiser la confiance au détriment de la précision de la détection individuelle des points clés dans le but d'obtenir les meilleures performances en se basant sur la métrique cocoAP. Cette métrique ne permet pas de mettre en lumière de tels défauts.

3.4 Prédiction de la visibilité des points caractéristiques

Comme nous l'avons observé dans la section précédente, les réseaux sont surconfiants lors de la prédiction des points clés. La Figure 3.8 illustre cette problématique. En effet, on peut voir que le nez et l'épaule gauche (en rouge) sont prédits par le réseau avec des scores de confiance élevés, alors même qu'ils ne sont pas annotés car leur localisation exacte est incertaine du fait de l'occultation. Ce score important est d'autant plus problématique que des parties du corps comme le poignet gauche, entièrement visible et annoté, sont prédites avec un score inférieur aux deux points précédents. On peut en déduire que le score de confiance de ces modèles n'est pas utilisable pour filtrer les points indésirables à l'aide d'un simple seuil fixe.

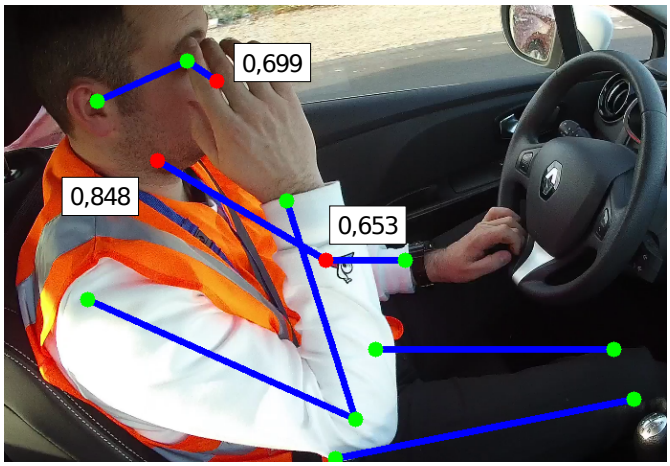


FIGURE 3.8 – Exemple de prédiction d'estimation de poses. Les points rouges représentent les faux positifs. Les scores de confiance sont indiqués dans les cellules (score maximum = 1,0).

Pour répondre à cette problématique, nous souhaitons trouver une alternative pour faciliter l'indication de la présence ou non d'un point dans les prédictions. Comme indiqué dans la Section 2.5.2.1, les jeux de données les plus récents fournissent des annotations de visibilité pour chaque point clé. Un point étiqueté peut être visible, ou non visible lorsque le point clé est légèrement occulté mais qu'il contient suffisamment d'informations pour être

localisé. Si le point est fortement occulté ou hors du champ de vision, il n'est pas annoté. Cependant, les méthodes de l'état de l'art ne tiennent pas compte de ces informations de visibilité. Les quelques architectures de la littérature cherchant à prédire la visibilité ne prennent en compte que les points annotés ou non annotés, et n'utilisent donc pas les trois classes de visibilité (Kumar et al., 2020; Stoffl et al., 2021). Nous proposons donc un métamodèle pour la prédiction de la visibilité des points. Cette structure se veut adaptable au plus grand nombre de réseaux d'estimation de poses de l'état de l'art, pour permettre à ces derniers de prédire à la fois la pose et la visibilité associée, sans détériorer les performances initiales.

3.4.1 Architecture du métamodèle

L'architecture proposée est divisée en trois parties : le module d'extraction de caractéristiques, celui d'estimation de coordonnées et enfin, celui de prédiction de la visibilité des points clés (Figure 3.9).

Tout d'abord, l'extracteur de caractéristiques traite l'image d'entrée pour générer un vecteur de caractéristiques. L'architecture précise de ce modèle n'est pas détaillée puisque n'importe quel extracteur de l'état de l'art peut être utilisé. On peut citer comme exemples les extracteurs utilisés pour l'estimation de poses (Tang and Wu, 2019; Li et al., 2019b; Artacho and Savakis, 2020a), ou les architectures plus génériques couramment utilisées dans la reconnaissance d'images telles que ResNet (He et al., 2016) ou EfficientNet (Tan and Le, 2019). Le vecteur généré sert ensuite d'entrée aux deux autres modules. L'estimation des coordonnées peut être effectuée par des modules tels qu'un décodeur complexe ou un module composé de plusieurs couches de convolution transposée, généralement suivis d'une couche de convolution qui génère les cartes de chaleur finales. (Tang and Wu, 2019; Li et al., 2019b; Artacho and Savakis, 2020a). La majorité des réseaux pour l'estimation de poses humaines peuvent être divisés en un module d'extraction de caractéristiques et un module de génération de cartes de chaleur, ce qui permet à la plupart des architectures d'être compatibles avec notre métamodèle.

En plus de ces deux modules habituels pour l'estimation de poses, nous ajoutons une branche de visibilité (Figure 3.10). Ce module prend en entrée le même vecteur

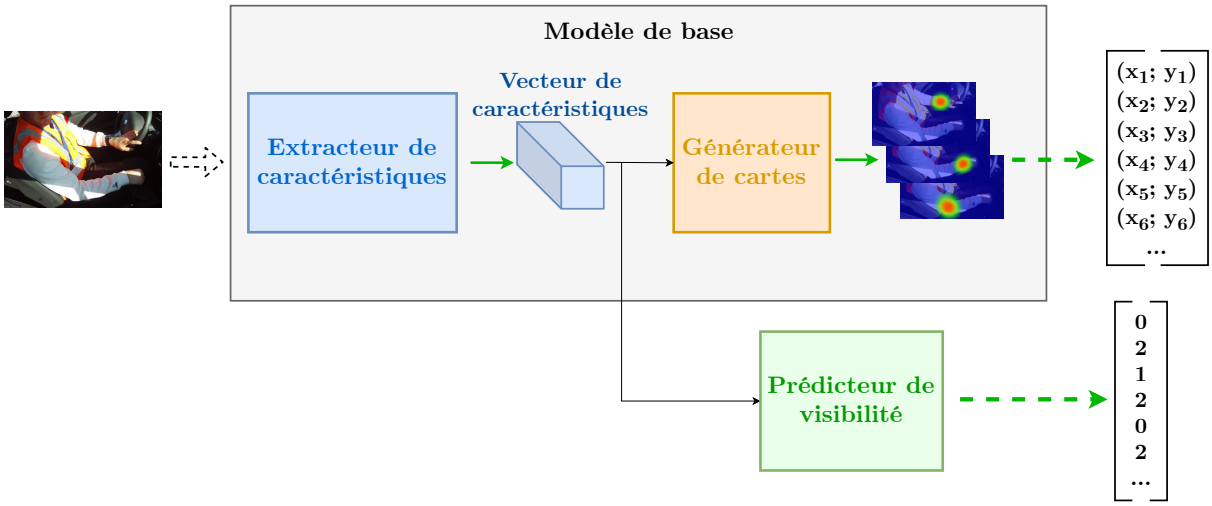


FIGURE 3.9 – Architecture du métamodèle multitâche proposé pour l'estimation de la pose et de la visibilité.

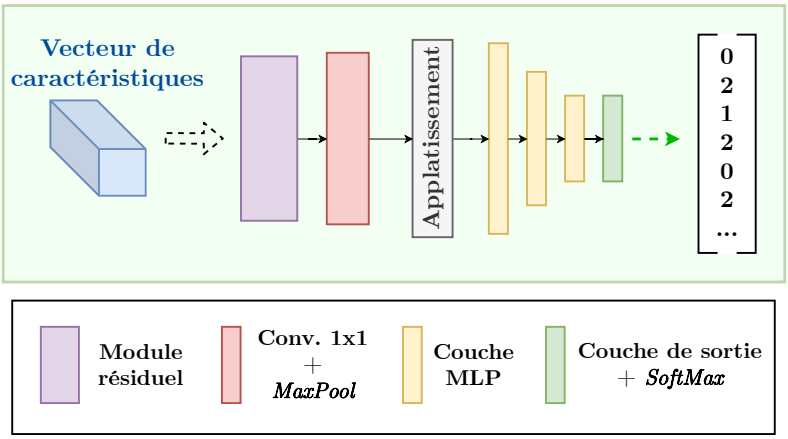


FIGURE 3.10 – Architecture de notre module de prédiction de visibilité.

de caractéristiques que le module d'estimation des coordonnées et produit la prédiction de visibilité pour chaque point clé. L'architecture détaillée est présentée dans la section suivante.

3.4.1.1 Module de visibilité

Nous modélisons le problème de prédiction de la visibilité comme une tâche de classification. Ainsi, à chaque point clé est associée une étiquette de visibilité modélisée par un entier (0 pour les points non annotés, 1 pour les points non visibles et 2 pour les points visibles). Le module de visibilité prend en entrée le vecteur de caractéristiques calculé par le module d'extraction de caractéristiques, et est composé d'un module convolutif suivi d'un réseau entièrement connecté qui génère les prédictions de visibilité finales.

Plus précisément, le module convolutif possède une architecture construite autour d'un bloc résiduel et d'une couche de convolution ayant montré de bons résultats dans le calcul des caractéristiques pour l'estimation de poses dans d'autres travaux (Newell et al., 2016; Tang and Wu, 2019). Un tel module convolutif a également l'avantage d'être léger et facilement modifiable. Ainsi, un bloc résiduel (architecture décrite dans le chapitre précédent, Figure 2.13) traite d'abord les caractéristiques d'entrée. Ensuite, une couche de convolution avec un noyau de taille 1×1, une couche de *BatchNorm* et une couche de *Max Pooling* 2×2 réduisent la taille et le nombre de canaux des caractéristiques. Enfin, le vecteur de caractéristique est aplati et un réseau entièrement connecté composé de trois couches cachées (4096, 2048 et 1024 neurones) suivi d'une couche de *SoftMax* produit les prédictions finales. Étant donné que le jeu de données COCO fournit 17 points clés annotés avec trois classes de visibilité possibles, la couche de sortie est composée de 51 neurones. La fonction *SoftMax* est appliquée par groupe de trois neurones de visibilité (un groupe représentant un point clé) pour obtenir les probabilités associées à chaque classe, et la visibilité avec la plus haute probabilité est sélectionnée comme prédiction finale.

3.4.1.2 Fonction de coût

La fonction de coût globale utilisée pour entraîner le réseau est définie comme suit :

$$L = (1 - \alpha).L_{HPE} + \alpha.L_V \tag{3.8}$$

où L_{HPE} est la fonction de coût du réseau d'estimation de poses, généralement la distance quadratique entre les cartes de chaleur prédites et celles de la vérité terrain. La fonction

L_V est la fonction d'entropie croisée (Goodfellow et al., 2016) appliquée aux prédictions des classes de visibilité, définie comme :

$$L_V(\hat{y}, y) = -\sum_{c=0}^{2} w_c \log \frac{\exp(\hat{y}_c)}{\sum_{i=0}^{2} \exp(\hat{y}_i)} y_c \tag{3.9}$$

avec $\hat{y}_c$ la probabilité prédite pour la classe c pour un point, y la vérité terrain associée et w_c le poids associé à chaque classe de visibilité. L'entropie croisée est pondérée afin de compenser la distribution déséquilibrée des points dans les trois classes de visibilité. Pour ce faire, les ratios du nombre de points clés dans chaque classe sont calculés sur le jeu d'entraînement. Enfin, α est le paramètre utilisé pour équilibrer le rapport entre les fonctions de perte associées aux deux tâches. Ce rapport représente l'impact de l'entraînement des deux tâches sur la mise à jour des poids de l'extracteur de caractéristiques.

3.4.2 Expérimentations

3.4.2.1 Modèles de base

Nous choisissons Simple Baseline (SBl) comme modèle de base pour les premières expérimentations, puisque son architecture simple combinant ResNet50 comme extracteur de caractéristiques et un décodeur basé sur plusieurs couches de convolution transposée se prête particulièrement bien au métamodèle.

3.4.2.2 Entraînement

Nous testons trois stratégies pour l'entraînement multitâche. Les poids de l'extracteur de caractéristiques sont les poids fournis par les auteurs sur le dépôt officiel initialisés sur ImageNet (Deng et al., 2009), et les poids du module de prédiction de visibilité sont initialisés aléatoirement avec des valeurs issues d'une distribution uniforme. Les trois stratégies sont définies comme telles :

- S1 : nous entraînons les tâches d'estimation de poses et de prédiction de la visibilité conjointement avec un α fixé à 0,25. Le réseau est ainsi entraîné durant 140 *epochs* avec un taux d'apprentissage de $1E^{-3}$, diminué d'un facteur de 10 après les *epochs* 90 et 120.
- S2 : nous pré-entraînons les modules d'extraction de caractéristiques et de prédiction de coordonnées sur le jeu de données COCO en suivant les recommandations des auteurs. Ensuite, nous ajoutons le module de visibilité et reprenons l'entraînement total du réseau pendant 80 *epochs*, tout en incrémentant α de 0,1 toutes les 20 *epochs* à partir de α=0.
- S3 : nous suivons le même procédé que pour S2 avec l'augmentation graduelle de α, à la différence que seul le module de visibilité est entraîné pendant cette étape tandis que les autres poids (extracteur de caractéristiques et estimateur de coordonnées) sont gelés.

À noter que nous entraînons la tâche d'estimation de poses avec comme fonction de coût une erreur quadratique calculée sur la totalité des cartes de chaleurs, et non uniquement celles correspondant aux points annotés comme c'était le cas dans les sections précédentes par les travaux de la littérature. Cela permet de réduire la surconfiance des réseaux et donc d'améliorer la précision. Nous utilisons des opérations d'augmentation

des données (retournement horizontal, rotation, translation, changement d'échelle) pour les deux jeux de données. Conformément aux travaux originaux, les images d'entrée sont rognées autour des sujets à l'aide de la vérité terrain, aussi bien pour l'entraînement que pour l'évaluation. Les poids sont mis à jour en suivant l'algorithme d'optimisation Adam (Kingma and Ba, 2014) avec des taux pour les moyennes mobiles exponentielles de $\beta_1 = 0.9$ et $\beta_2 = 0.999$. L'entraînement est effectué sur un ordinateur équipé d'une carte graphique Nvidia GTX 1080 avec 12 Go de VRAM, d'un processeur Intel Core i990k et de 32 Go de RAM. Les réseaux sont évalués à l'*epoch* où ils présentent la plus faible erreur sur le jeu de validation.

3.4.3 Résultats

Dans cette section, nous présentons et discutons les performances du métamodèle proposé. Plus précisément, nous étudions d'abord la qualité des prédictions de visibilité en utilisant différentes stratégies d'entraînement. Ensuite, nous étudions l'impact de la prédiction de la visibilité sur la tâche d'estimation de poses. Enfin, nous discutons des performances du métamodèle lorsque nous faisons varier différents hyperparamètres expérimentaux.

3.4.3.1 Prédiction de visibilité

Comme décrit dans la Section 3.4.2.2, nous essayons plusieurs stratégies pour entraîner le réseau. Les performances obtenues des trois réseaux sont présentées dans le Tableau 3.9.

Stratégie	Non annoté	Non visible	Visible	Total
S1	72	21	76	71
S2	75	34	79	74
S3	**77**	**37**	**80**	**76**

TABLEAU 3.9 – Score F1 du métamodèle pour la prédiction de la visibilité en fonction des classes de points sur COCO avec différentes stratégies d'apprentissage (en %).

Tout d'abord, nous pouvons observer que le pré-entraînement du réseau sur la tâche d'estimation de poses (S2 et S3) permet d'obtenir de meilleures performances que l'entraînement conjoint des trois modules (S1). En effet, nous pouvons noter une augmentation de 5 % du score F1 total entre S1 et S3. Cette amélioration est surtout perceptible pour les points de la classe non visible (gain de 16 %). Cependant, nous considérons que l'entraînement sur la tâche de visibilité en gelant le reste du réseau (S3) n'a pas d'impact sur la performance globale. En effet, nous avons entraîné plusieurs modèles et présentons dans le Tableau 3.9 les modèles avec la meilleure performance pour chaque stratégie. Malgré cela, nous observons peu de différence de performance entre les réseaux entraînés avec et sans avoir gelé l'extracteur de caractéristiques et le module d'estimation de poses. Cette expérience montre que les réseaux d'estimation de poses déjà entraînés peuvent atteindre des performances optimales pour la prédiction de visibilité sans nécessiter d'effectuer un nouvel entraînement complet. Cela permet d'économiser des ressources de temps et de calcul, en particulier avec un grand jeu d'entraînement tel que COCO.

En ce qui concerne la performance de la prédiction de la visibilité, les résultats du Tableau 3.9 montrent que les réseaux prédisent la visibilité des points clés avec un score F1 total allant jusqu'à 76 %. Cependant, les modèles ont des difficultés à prédire la classe

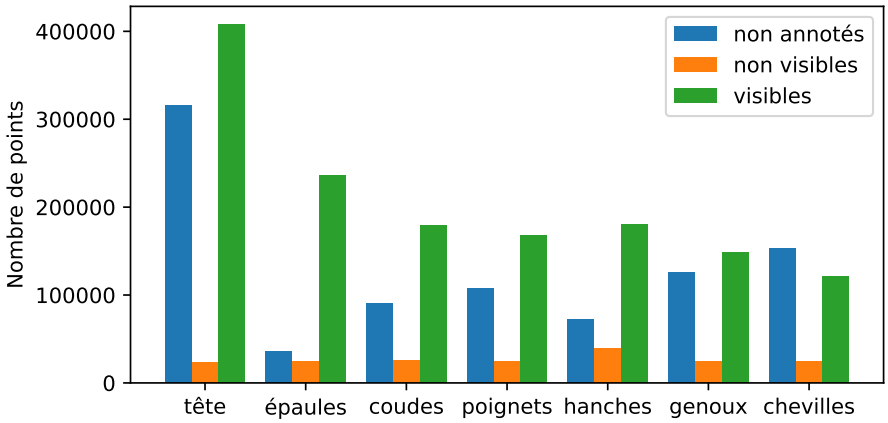


FIGURE 3.11 – Distribution des classes de visibilité des points dans le jeu de données COCO.

"non visible", avec un score F1 maximal de 37 %. Deux raisons peuvent expliquer cet écart. Premièrement, la notion de point non visible est subjective, puisqu'elle correspond aux points occultés mais pour lesquels nous disposons de suffisamment d'informations dans l'image pour déduire son emplacement. L'évaluation du niveau "d'informations suffisantes" est laissée à l'annotateur ; ce qui entraîne des incohérences dans les annotations. Deuxièmement, les points annotés comme non visibles ne représentent que 7 % des points présents dans le jeu de données COCO (Figure 3.11). Même si cet écart de distribution est pris en compte dans le calcul de la fonction de coût d'entropie croisée pondérée L_v, il nuit à la qualité du processus d'apprentissage.

Pour étudier l'impact de la distribution des trois classes de visibilité, nous effectuons un *fine-tuning* de notre réseau sur le jeu de données DriPE. Ce jeu de données présente une distribution plus homogène des classes de points, comme le montre la Figure 3.12.

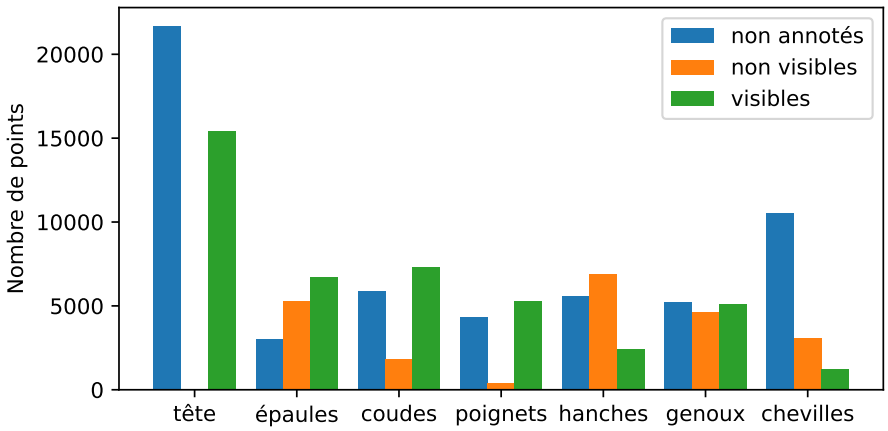


FIGURE 3.12 – Distribution des classes de visibilité des points dans le jeu de données DriPE.

Comme nous pouvons voir dans le Tableau 3.10, après être entraîné sur DriPE, le modèle atteint un score F1 de 70 % pour les points non visibles. Ce résultat démontre qu'avec une meilleure distribution des classes de visibilité, notre métamodèle est capable de mieux estimer la visibilité des points, en particulier pour les classes non visibles.

	Non annoté	Non visible	Visible	Total
Pré-entraînement COCO	77	37	80	76
Fine-tuning sur DriPE	**81**	**70**	**76**	**76**

TABLEAU 3.10 – Score F1 du réseau pour la prédiction de la visibilité sur le jeu de données DriPE après le pré-entraînement et après le *fine-tuning* (en %).

3.4.3.2 Impact sur l'estimation de poses

Nous étudions maintenant l'impact de l'ajout du module de visibilité sur la performance de l'estimation de poses. Nous utilisons pour cette étude la métrique mAPK afin d'avoir une mesure de la performance plus centrée sur la détection des points. Les performances sont mesurées sur les jeux de données COCO (Tableau 3.11) et DriPE après un *fine-tuning* (Tableau 3.12). Le réseau "SBl + visibilité" fait référence au réseau Simple Baseline utilisé avec le module de prédiction de visibilité entraîné avec la stratégie S2. Le terme "non 0" définit l'expérience dans laquelle toutes les coordonnées des points prédites par le module de visibilité comme "non annotés" sont rejetées.

	Configuration	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	SBl	66	76	73	70	74	74	74	72
	SBl + visibilité	66	76	72	70	73	73	73	72
	SBl + visibilité + non 0	71	78	77	73	73	76	74	**75**
AR	SBl	73	77	73	70	70	72	72	**72**
	SBl + visibilité	73	76	73	69	70	72	72	**72**
	SBl + visibilité + non 0	43	72	57	68	68	66	35	59

TABLEAU 3.11 – Comparaison des performances pour l'estimation de poses de différentes configurations sur COCO avec mAPK (en %).

Tout d'abord, nous observons que le métamodèle (SBl + visibilité) atteint des performances similaires à celles de SBl seul pour l'estimation de poses. Cela indique que l'ajout de la tâche de visibilité n'a pas d'impact négatif sur la tâche principale, quel que soit le jeu de données utilisé. Deuxièmement, la stratégie "non 0" améliore légèrement la précision moyenne de l'estimation de poses ; ce qui indique une diminution du nombre de faux positifs parmi les points prédits. Cependant, cette augmentation de la précision est contrebalancée par une diminution du rappel moyen causée par une augmentation des faux négatifs. La diminution du rappel est particulièrement significative pour les points de la tête, des coudes et des chevilles. Comme expliquée dans la Section 3.3.3, la prédiction de visibilité des points du visage est une tâche délicate puisque presque aucun de

	Configuration	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	SBl	85	90	94	96	98	95	68	89
	SBl + visibilité	84	90	94	96	98	95	68	89
	SBl + visibilité + non 0	86	90	94	97	98	96	72	**90**
AR	SBl	87	96	96	97	98	95	80	**93**
	SBl + visibilité	87	96	96	97	98	95	80	**93**
	SBl + visibilité + non 0	44	96	85	97	98	93	77	84

TABLEAU 3.12 – Comparaison des performances pour l'estimation de poses de différentes configurations sur DriPE après un *fine-tuning* avec mAPK (en %).

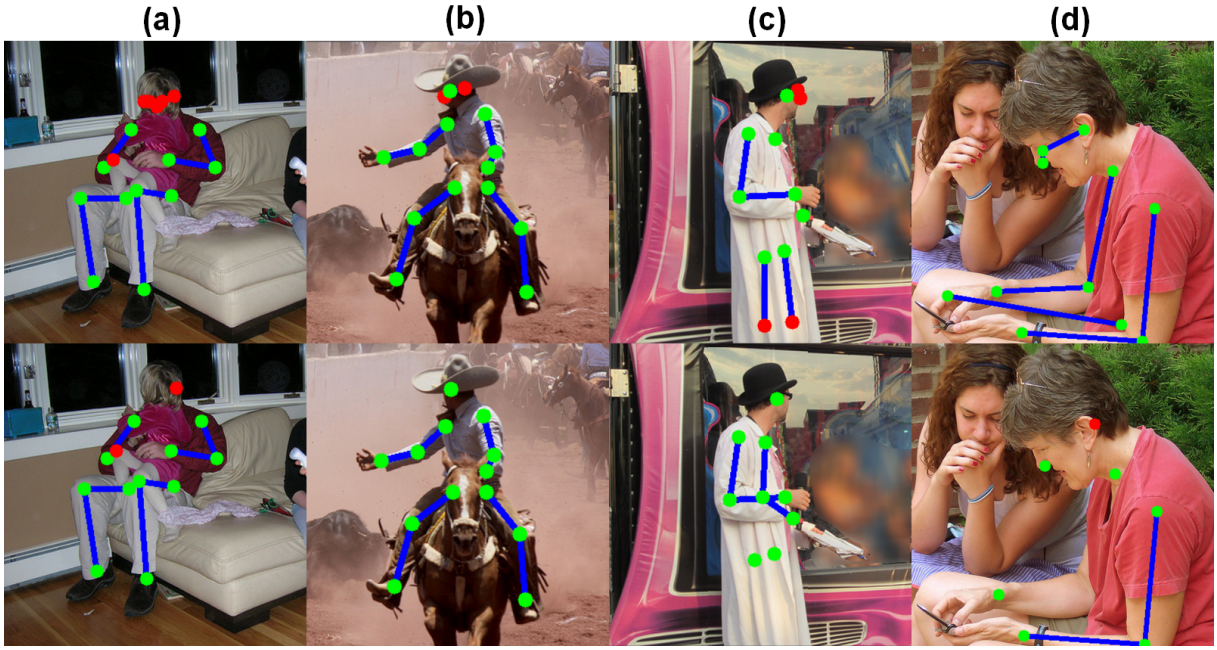


FIGURE 3.13 – Comparaison qualitative de la prédiction des points filtrés avec un seuil de confiance (rangée du haut) et avec la visibilité prédite par notre métamodèle (rangée du bas). Les points rouges représentent les faux positifs.

ces points n'est étiqueté comme non visible en raison du style d'annotation COCO. Les chevilles sont également des points difficiles à prédire dans un contexte général puisqu'ils sont légèrement moins souvent annotés.

Nous présentons des résultats qualitatifs dans la Figure 3.13. Comme observé dans les Tableaux 3.11 et 3.12, le gain de précision provient principalement des points du visage. Ceci est illustré dans la Figure 3.13-a et b où des points du visage initialement prédits malgré la forte occultation et le manque d'informations ont été retirés. Cependant, la précision de la prédiction d'autres parties du corps a également été améliorée, comme les genoux sur la Figure 3.13-c. Il est intéressant de noter que le coude gauche, qui n'est pas prédit sur l'image du haut en raison du manque de confiance dans les coordonnées prédites, est conservé à juste titre dans les images du bas. Enfin, le compromis négatif concernant le rappel est causé par les points clés correctement prédits par le module d'estimation de coordonnées mais prédits comme non annotés par le module de visibilité (Figure 3.13-d).

3.4.3.3 Etudes complémentaires

Impact du hyperparamètre α

Enfin, nous étudions l'impact du paramètre α sur les performances des réseaux. Pour rappel, ce paramètre est utilisé dans le calcul de la fonction de coût de notre métamodèle (Équation 3.8). Plus ce paramètre est élevé, plus l'erreur faite sur la prédiction de la visibilité des points aura un impact sur la fonction de coût total. Nous entraînons donc plusieurs instances de Simple Baseline en suivant la stratégie S1 avec des valeurs de α différentes. Les résultats de l'évaluation sont disponibles dans les Tableaux 3.13 et 3.14.

Nous pouvons observer plusieurs choses dans ces résultats. Premièrement, augmenter α nuit bien aux performances de l'estimation de poses puisque le réseau perd environ 0,05 de précision et rappel entre un α à 0,75 et 0,25. La réciproque pour la visibilité ne

α	Non annoté	Non visible	Visible	Total
0,25	72	21	76	71
0,50	73	32	78	**73**
0,75	68	18	70	66

TABLEAU 3.13 – Score F1 du réseau pour la prédiction de la visibilité sur COCO avec différentes valeurs de α (en %).

	α	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	0,25	66	74	72	69	71	73	71	**70**
	0,50	54	67	64	64	65	63	67	64
	0,75	40	48	43	39	44	46	43	43
AR	0,25	73	77	73	70	70	72	72	**72**
	0,50	61	70	65	64	64	62	66	65
	0,75	49	54	48	44	46	50	47	48

TABLEAU 3.14 – Performances pour l'estimation de poses pour plusieurs valeurs de α avec mAPK (en %).

semble cependant pas être vraie puisque, bien que l'on observe une légère augmentation des performances entre 0,25 et 0,50 sur la prédiction de visibilité des points, un α à 0,75 détériore grandement les performances avec une perte de prédiction de visibilité de près de 10 %. On peut donc en déduire que la prédiction de visibilité bénéficie de l'entraînement sur la tâche d'estimation de poses et donc que la qualité de la prédiction de la visibilité est liée à la qualité de l'estimation de poses. Cela pourrait être expliqué par le fait que l'erreur calculée sur les cartes de chaleurs est plus riche en information que l'erreur calculée sur les classes de visibilité ; ce qui permet un meilleur entraînement de l'extracteur de caractéristiques, en particulier vis-à-vis de la compréhension spatiale de l'image.

Autres modèles de base

Nous évaluons ensuite le métamodèle avec différentes architectures d'estimation de poses : Simple Baseline avec l'extracteur de caractéristiques original (ResNet50), Simple Baseline en remplaçant l'extracteur de caractéristiques par EfficientNet B0 et B6 (EfficientNet est disponible en plusieurs versions de plus en plus profondes, B0 et B6 possédant respectivement 5,3 M et 43 M de paramètres)(Tan and Le, 2019), et MSPN (Li et al., 2019b). L'entraînement des réseaux est fait en suivant la stratégie S3, autrement dit en gelant l'extracteur de caractéristiques lors de l'entraînement du module de visibilité. Les performances de ces implémentations sont présentées dans les Tableaux 3.16 et 3.17, et le Tableau 3.15 présente les performances originales des modèles de base.

Modèle de base	paramètres	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR	AR^{50}	AR^{75}	AR^{M}	AR^{L}
SBl	71,2M	73	92	70	69	71	75	93	82	73	80
SBl (EfficientNet B0)	55,6M	67	90	74	65	72	71	92	78	77	76
SBl (EfficientNet B6)	95,5M	76	92	81	70	78	76	93	84	73	82
MSPN 2-stg	104,6M	73	93	83	71	76	75	94	83	75	81

TABLEAU 3.15 – Performance de différents modèles de base pour l'estimation de poses 2D sur COCO (cocoAP en %).

Modèle de base	paramètres	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR	AR^{50}	AR^{75}	AR^{M}	AR^{L}
SBl (ResNet 50)	71,2M	72	92	69	69	76	75	93	82	72	80
SBl (EfficientNet B0)	55,6M	67	90	75	64	72	70	91	77	67	76
SBl (EfficientNet B6)	95,5M	72	92	80	70	77	76	93	83	73	81
MSPN 2-stg	104,6M	72	93	81	69	76	75	94	84	72	80

TABLEAU 3.16 – Performance du métamodèle avec différents modèles de base pour l'estimation de poses sur COCO (cocoAP en %).

Modèle de base	paramètres	non annoté	non visible	visible	total
SBl (ResNet50)	71,2M	77	37	80	76
SBl (EfficientNet B0)	55,6M	74	32	77	73
SBl (EfficientNet B6)	95,5M	75	34	80	76
MSPN 2-stg	104,6M	69	34	69	67

TABLEAU 3.17 – Score F1 du réseau pour la prévision de la visibilité sur COCO avec différents modèles de base (en %).

Comme nous pouvons l'observer dans le Tableau 3.16, les modèles préservent de bonnes performances sur l'estimation de poses avec des scores de précision et rappel comparables à ceux mesurés sur les réseaux entraînés sans le module de visibilité (Tableau 3.15). En outre, les métamodèles semblent capables de prédire la visibilité quelle que soit l'architecture de base utilisée. Ces résultats montrent que notre métamodèle peut être utilisé avec des réseaux de tailles et d'architectures variées tout en préservant les performances sur les deux tâches. Nous pouvons remarquer que les performances du réseau MSPN sont inférieures à ce que nous pourrions attendre pour un si grand nombre de paramètres. Cela pourrait s'expliquer par l'architecture multi-échelle et multi-étape du réseau MSPN contrairement aux autres architectures qui possèdent un unique extracteur de caractéristiques. La concaténation de vecteurs de caractéristiques issus de plusieurs niveaux d'échelle ou de la sortie de plusieurs étapes du réseau pourrait améliorer les résultats.

Architecture du module de visibilité

Nous comparons ensuite trois architectures pour le module de visibilité. Premièrement, nous utilisons uniquement un réseau entièrement connecté avec une couche cachée unique de 2048 neurones (Figure 3.14-a). Nous ajoutons ensuite le bloc résiduel et la couche de convolution avec *MaxPool* (Figure 3.14-b). Nous mettons ensuite les trois couches cachées entièrement connectées de l'architecture originale (Figure 3.14-c). Enfin, nous ajoutons une couche intermédiaire supplémentaire dans le réseau entièrement connecté (Figure 3.14-d). Nous entraînons les différentes architectures sur COCO avec la stratégie

Archi.	Conv.	F.C.	Non annoté	Non visible	Visible	Total
(a)	✗	2048	54	0	0	20
(b)	✓	2048	61	14	63	57
(c)	✓	4096 - 2048 - 1024	72	21	**76**	71
(d)	✓	4096 - 2048 - 2048 - 1024	**75**	**22**	76	**73**

TABLEAU 3.18 – Score F1 des différentes architectures pour la prédiction de la visibilité sur COCO (en %) (Conv : ajout des blocs convolutifs ; F.C. : taille des couches cachées).

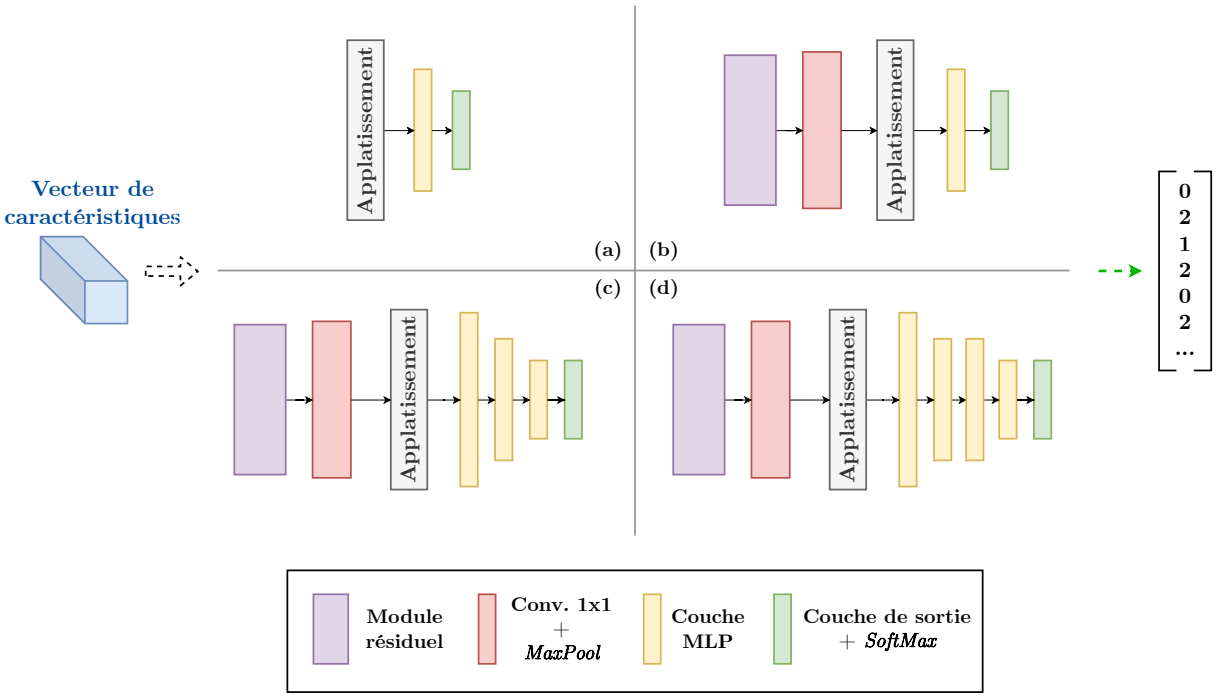


FIGURE 3.14 – Architectures de la branche de visibilité utilisées pour l'étude ablative.

d'entraînement S1. Les résultats comparés à l'architecture totale sont montrés dans le Tableau 3.18.

Ces résultats montrent tout d'abord que l'utilisation d'un réseau MLP seul ne permet pas de prédire la visibilité des points. En effet, le réseau entraîné converge très rapidement vers la prédiction de l'ensemble des points comme "non annotés". Une architecture similaire est pourtant utilisée pour la prédiction de la visibilité sur les points du visage dans la littérature (Kumar et al., 2020). Plusieurs raisons pourraient expliquer ce comportement. Premièrement, le vecteur de caractéristiques en entrée du module de prédiction de visibilité de Kumar *et. al.* est de taille 2048. A l'inverse, le vecteur de caractéristiques produit par Simple Baseline est de taille $8 \times 6 \times 2048$, soit près de 100 000 paramètres. Le travail de la couche d'entrée du MLP pour réduire cette dimension est donc beaucoup plus important. Deuxièmement, le visage possède une géométrie globale plus fixe que le corps entier ; ce qui peut faciliter la prédiction de la visibilité des points clés. Troisièmement, nous cherchons à prédire les trois classes de visibilité, contrairement à Kumar *et. al.* qui cherchent uniquement à prédire si les points clés sont annotés.

Nous pouvons voir, dans un second temps, que l'ajout des couches convolutives permet une première estimation de la visibilité. Ces couches ont plusieurs rôles. Le bloc résiduel et la couche de *MaxPool* permettent de concentrer l'information tout en tenant compte des informations spatiales. La couche de *MaxPool* ainsi que la convolution 1×1 permettent quant à elles de réduire efficacement la dimension du vecteur de caractéristiques. Enfin, l'ajout de couches cachées permet d'améliorer la qualité globale des prédictions. Cependant, on peut voir qu'ajouter plus de couches augmente très peu les performances tout en ajoutant 4,2 M de paramètres au réseau. L'architecture avec 3 couches permet donc d'obtenir de bonnes performances tout en limitant la taille du réseau.

3.5 Conclusion

Dans ce chapitre, nous avons cherché à étudier les performances des méthodes de l'état de l'art pour l'estimation de poses humaines 2D dans le contexte de l'intérieur d'une voiture. Les résultats préliminaires obtenus avec le réseau OpenPose sur une dizaine d'images semblaient indiquer que le réseau entraîné sur MPII et COCO, deux jeux de données génériques de l'état de l'art, rencontrait quelques difficultés pour la détection de certains points, en particulier le bas du corps. En outre, les variations de contraste créées par l'habitacle du véhicule et les éclairages irréguliers, ainsi que les occultations causées par l'angle de vue de la caméra pouvaient être d'autant plus d'obstacles à la bonne estimation de la pose. Cependant, un nombre plus important d'images avec une plus grande variabilité était nécessaire pour valider ces observations. Nous avons alors cherché à obtenir plus d'images annotées dans notre contexte pour l'estimation de poses humaines. Afin de pouvoir correctement tester les performances des réseaux, il était nécessaire d'obtenir des images avec différents conducteurs, différentes conditions d'enregistrement, mais aussi enregistrées dans un environnement extérieur puisque nous avons observé que les ombres et fortes lumières inhabituelles pouvaient perturber la détection de la personne. Cependant, les jeux de données disponibles publiquement pour l'estimation de poses dans notre contexte sont rares, et aucun ne rassemblait nos critères.

Nous avons donc décidé de créer notre propre jeu de données nommé DriPE. Ce jeu contient dix mille images extraites de clips vidéo enregistrés lors d'une précédente expérimentation sur Transpolis, un centre d'essais automobiles en conditions réelles. Les images ont été sélectionnées à partir de deux métriques : la similarité structurelle et la luminosité. Ces deux métriques ont été calculées entre deux *frames* successives dans le but de mettre en valeur les fortes variations de luminosité et de structure de l'image afin de sélectionner des images variées qui pourraient mettre en difficulté les réseaux. Afin de mesurer l'impact réel de cette approche, il serait nécessaire d'extraire un deuxième jeu de données en sélectionnant les images aléatoirement ou avec un pas fixe, et de mesurer la différence des performances des solutions de l'état de l'art sur les deux jeux. L'annotation manuelle des images étant cependant longue à réaliser, nous n'avons pas réalisé ce travail comparatif. La pose sur ces images a été annotée manuellement en suivant la nomenclature du jeu de données COCO qui fournit en plus des points du corps l'annotation des points principaux du visage et une boite englobante. Nous avons alors évalué plusieurs réseaux de l'état de l'art pour l'estimation de poses sur DriPE avec la métrique cocoAP, métrique officielle du *benchmark* COCO. Les performances mesurées sur DriPE des réseaux entraînés sur COCO étaient proches de celles mesurée sur le jeu de données COCO, alors qu'un entraînement supplémentaire de quelques *epochs* sur DriPE a permis d'obtenir des scores de performance presque parfaits. Ces premières mesures semblaient indiquer que la tâche d'estimation de poses humaines dans un véhicule était une tâche résolue par l'état de l'art.

Les observations qualitatives ne reflétaient cependant pas ces performances presque parfaites, en particulier sur le visage et le bas du corps. Nous nous sommes donc intéressés à la métrique d'évaluation et avons proposé mAPK, une nouvelle métrique pour l'estimation de poses mono-personne s'inspirant de la métrique APK et du score OKS. Cette métrique permet de mesurer la performance de l'estimation de poses au niveau des points clés et non des personnes comme le faisait cocoAP, tout en étant compatible avec les jeux de données suivant la nomenclature d'annotation COCO. L'évaluation des réseaux de l'état de l'art a montré que ces derniers étaient sujets à une surconfiance des prédictions. Cette

surconfiance n'avait aucun impact négatif sur la mesure de cocoAP puisque les points faux-positifs ne sont pas pris en compte dans le calcul. La stratégie d'entraînement utilisée par l'état de l'art peut en partie expliquer ce phénomène, puisqu'elle ne pénalise pas l'erreur faite sur les points clés non annotés afin d'optimiser les performances mesurées par cocoAP. Nous avons aussi mesuré lors de l'évaluation sur DriPE une moins bonne performance sur le visage et les chevilles, comme observé sur les résultats qualitatifs.

Enfin, nous avons proposé un métamodèle permettant de prédire à la fois la pose humaine et la visibilité des points. Cette méthode permet d'atteindre de bonnes performances de prédiction de visibilité tout en préservant la performance de l'estimation de poses des modèles de base. Ces observations ont été faites sur COCO, un jeu de données générique, ainsi que sur notre jeu de données DriPE ; ce qui a permis de montrer qu'une répartition plus équilibrée des classes de visibilité permettait d'améliorer les performances. Nous avons également montré que ce métamodèle était applicable avec des modèles d'estimation de poses de taille et d'architecture variées. Cependant, mieux adapter la récupération du vecteur de caractéristiques aux architectures plus complexes pourraient permettre d'améliorer les performances. Nous avons aussi testé une application simple de cette prédiction de visibilité dans le but d'améliorer l'estimation de poses. Cette approche a permis d'améliorer légèrement la précision des prédictions, mais cette amélioration s'est faite au détriment du rappel.

Les travaux présentés dans ce chapitre ont fait l'objet de deux publications internationales. Notre jeu de données DriPE ainsi que la métrique mAPK ont été présentés lors d'un *workshop* autour de la thématique des véhicules autonomes à la conférence ICCV 2021 (Guesdon et al., 2021). Le métamodèle pour la prédiction de visibilité a été présenté lors de la conférence VISAPP 2022 (Guesdon et al., 2022). Enfin, le jeu de données, les codes d'évaluation avec mAPK [11] ainsi que les codes pour notre métamodèle [12] sont disponibles publiquement sur des dépôts en ligne .

11. `https://gitlab.liris.cnrs.fr/aura_autobehave/dripe`
12. `https://gitlab.liris.cnrs.fr/aura_autobehave/vis-pred`

Chapitre 4

Génération de données pour l'estimation de poses humaines

Dans ce chapitre, nous présentons nos travaux sur l'augmentation de données pour l'estimation de poses. Ces travaux ont pour but de répondre au manque de données pour la généralisation de domaine. Nous nous intéressons d'abord à la création de données par réseau génératif. Nous implémentons un réseau de transfert de poses dans l'objectif de générer de nouvelles images de personnes dans des véhicules, mais prenant des poses différentes. L'entraînement de réseaux pour le transfert de poses demandent cependant un grand nombre de données diversifiées pour être entraînées, et nous ne disposons pas de suffisamment de données dans notre contexte pour converger les modèles vers des résultats satisfaisants. Nous nous intéressons donc à la génération d'images par modélisation 3D. Ainsi, nous proposons une méthode permettant de générer un grand nombre d'images synthétiques variées de personnes dans des habitacles de voitures. Nous montrons que les images générées suffisent à entraîner un réseau pour le transfert de poses dans le domaine synthétique. Finalement, nous tentons d'utiliser ces images pour entraîner un modèle de transfert de poses pour des images de passagers dans le domaine réel.

Sommaire

4.1 Introduction

Nous avons vu dans le chapitre précédent que les méthodes de l'état de l'art pour l'estimation de poses humaines permettaient d'obtenir de bonnes performances sur des images dans une voiture. En effet, nous mesurons par exemple autour de 70 % de précision et rappel sur DriPE pour des réseaux de neurones entraînés sur COCO, et ces performances montent à environ 90 % après un *fine-tuning* sur DriPE. Cependant, le modèle d'estimation de poses que nous cherchons à utiliser lors des expérimentations doit être capable de généraliser malgré des variations de conditions expérimentales. Les performances doivent être préservées sans nécessiter d'entraîner à nouveau le réseau ou d'annoter des données illustrant ces nouvelles conditions. Afin de pouvoir mesurer les capacités de généralisation des différents modèles, il est nécessaire d'évaluer ce dernier sur un jeu de données différent de ceux utilisés pendant l'entraînement. En effet, bien que les conditions dans DriPE soient relativement variées (différents acteurs, véhicules, cadrages, arrière-plans) et que les ensembles d'entraînement et d'évaluation ne partagent pas d'acteurs, la majorité des images ont été enregistrées lors de la même expérimentation, avec le même type de caméra, et illustrent donc le même environnement.

Nous commençons donc par évaluer les performances sur DriPE du réseau *Simple Baseline* entraîné sur COCO, le jeu de données utilisé pour entraîner la majorité des réseaux actuels d'estimation de poses dans l'état de l'art, afin de mesurer sa capacité de généralisation. Le modèle utilisé est celui dont les performances sont présentées dans le Tableau 3.11 et dont les conditions d'entraînement sont décrites dans la Section 3.4.2.2. À noter qu'à la différence des réseaux évalués dans le Tableau 3.3 et de ceux de la littérature, la fonction de coût est calculée sur la totalité des cartes de chaleur, et non uniquement sur celles correspondant aux points annotés, afin de diminuer le nombre de faux positifs. Puisqu'aucune image de DriPE n'est utilisée durant l'entraînement, nous pouvons évaluer les réseaux sur la totalité des images (ensembles d'entraînement, de validation et de test) et non uniquement sur le jeu d'évaluation. Cela permet de mesurer la performance sur une plus grande diversité d'images. Nous référerons dans la suite de nos travaux à ce jeu d'évaluation comme "DriPE complet". Les résultats de l'évaluation sont montrés dans le Tableau 4.1.

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	82	77	86	89	84	75	21	82
AR	85	76	78	82	62	40	03	67

TABLEAU 4.1 – Performances sur le jeu DriPE complet de *Simple Baseline* entraîné sur COCO, évaluées avec mAPK (en %).

Comme nous pouvons le constater, de bonnes performances sont globalement mesurées, avec une précision sur l'estimation des points clés de 82 % et un rappel de 67 %. On peut cependant remarquer que les performances sont beaucoup plus faibles sur le bas du corps, en particulier sur les chevilles avec une précision de 21 %. Ce constat est d'autant plus vrai pour le rappel, ce qui indique que beaucoup de points ne sont pas détectés par le réseau. On peut donc en conclure qu'un entraînement sur COCO permet d'atteindre de bonnes performances sur le haut du corps, mais le réseau éprouve des difficultés à détecter les points du bas du corps, et en particulier des chevilles. Bien que les chevilles soient des point difficiles à détecter sur DriPE, comme nous avons pu le montrer dans le Chapitre 3, les performances sur ces points clés restent tout de même très faibles.

Il semble donc nécessaire d'entraîner le réseau avec des images contenant des poses du bas du corps proches de celles observables à l'intérieur d'une voiture. Puisque nous souhaitons utiliser DriPE pour mesurer la capacité de généralisation des méthodes d'estimation de poses, nous décidons de l'exclure des jeux de données utilisés pour l'entraînement. Le seul autre jeu de données réelles dans correspondant à notre contexte proposant des pseudo-annotations pour l'estimation de poses est le jeu Drive&Act (Martin et al., 2019). Les annotations de ce jeu de données pour la pose ont cependant été obtenues en utilisant le réseau OpenPose (Cao et al., 2019) entraîné sur les jeux génériques COCO et MPII. Ainsi, la pose du bas des jambes est peu annotée (15 % des genoux et 2 % des chevilles). De plus, ces images ont été enregistrées dans un simulateur avec une caméra fixe ; ce qui limite la diversité des poses.

Il est donc nécessaire d'obtenir de nouvelles images suffisamment diversifiées de passagers avec les annotations correspondantes. Une possibilité serait d'annoter la pose d'images provenant d'un autre jeu de données publique tel que DmD (Ortega et al., 2020) ou de l'expérimentation réalisée entre-temps à Nantes pour le projet AutoBehave (Annexe A), tel que nous l'avons fait pour DriPE. Cependant, sélectionner les images et les annoter manuellement demanderait un temps très important. On pourrait également considérer d'utiliser des pseudo-annotations pour DmD générées par un réseau de neurones, similaires aux pseudo-annotations fournies avec le jeu Drive&Act. Ces approches ne répondent cependant pas à nos besoins, puisque, comme nous avons pu le voir, peu de genoux et de chevilles seraient annotées à cause de la difficulté à estimer ces points clés. Nous recherchons donc une méthode d'augmentation de données permettant d'obtenir des images et les vérités terrain associées.

4.2 Transfert de poses humaines

Nous décidons d'abord de nous intéresser à l'augmentation de données par réseau génératif. Cette approche consiste à générer de nouvelles images en utilisant des réseaux de neurones génératifs, généralement des réseaux antagonistes (GAN). Le principe de cette approche est d'utiliser un GAN pour produire des images appartenant à la même distribution que les données d'entraînement initiales. Ces images sont généralement plus variées visuellement et sémantiquement que ce que l'on peut obtenir en appliquant des méthodes d'augmentation de données classiques, tout en étant relativement réalistes. Cette approche peut, par exemple, être utilisée pour diverses tâches de classification (Antoniou et al., 2017), ou bien pour de la segmentation (Sandfort et al., 2019).

Dans notre contexte d'estimation de poses, la difficulté est double puisque nous cherchons à générer des images dans un environnement spécifique, mais nous avons aussi besoin de l'annotation de la pose associée. Nous nous intéressons donc aux méthodes de transfert de poses humaines (*Human Pose Transfer*, ou HPT). L'utilisation de cette approche dans notre contexte permettrait de générer à partir d'une image d'un passager un grand nombre d'images dans des poses variées. Cela augmenterait le nombre et la diversité des images utilisées pour l'entraînement sans nécessiter la mise en place d'expérimentations pour la capture de nouvelles données et l'annotation manuelle.

4.2.1 État de l'art

La tâche de transfert de poses humaines consiste à générer une image $\hat{I}_t$ d'une personne dans une pose P_t donnée à partir d'une image source I_s de cette même personne dans une pose P_s. Cette tâche est principalement utilisée dans deux contextes : les images de mode et de vidéosurveillance. Ces deux domaines correspondent aux deux jeux de données principaux de l'état de l'art pour cette tâche : Market-1501 (Zheng et al., 2015) et DeepFashion (Liu et al., 2016).

Les approches par réseaux de neurones utilisent un générateur qui encode l'image source, la pose source et la pose cible afin de générer l'image cible. Ainsi, PG2 (Ma et al., 2017) utilise un premier encodeur-décodeur U-Net (Ronneberger et al., 2015) pour générer une image grossière, puis un deuxième réseau entraîné de manière antagoniste pour raffiner l'image. DeformGan (Siarohin et al., 2018) encode I_s et P_s d'une part et P_t d'autre part, puis calcule la transformation entre chaque liaison du squelette source et cible afin d'appliquer ces transformations sur les vecteurs de caractéristiques calculés. L'architecture PATN (Zhu et al., 2019) se compose de plusieurs blocs successifs identiques qui utilisent un mécanisme d'attention calculé à partir des deux poses pour calculer un vecteur de caractéristiques. Ce mécanisme d'attention à partir de la pose a été beaucoup réutilisé par la suite. XingGAN (Tang et al., 2020) encode le style et la pose dans deux branches distinctes, et utilisent un mécanisme d'attention croisée entre les deux branches afin de partager les informations. Le réseau APS (Huang et al., 2020) encode d'abord le style et la pose source, puis injecte la représentation calculée dans le générateur pour produire l'image de sortie à partir de la pose cible. L'architecture PISE (Zhang et al., 2021) est entraînée à prédire une carte de segmentation des différentes parties du corps, utilisée par la suite pour guider l'encodage et la génération de l'image. Plus récemment, les auteurs de NTED (Ren et al., 2022) proposent un apprentissage de coefficients d'attention afin d'extraire individuellement les différentes textures. Enfin, le réseau SAPFN (Ma et al., 2023) calcule le flux optique entre les deux cartes de segmentation pour alimenter le mécanisme d'attention.

Ces méthodes sont toutes supervisées puisque la différence entre l'image générée et l'image attendue est prise en compte dans la fonction de coût total. Quelques approches non supervisées existent dans la littérature, principalement basées sur une contrainte cyclique (Pumarola et al., 2018; Song et al., 2019). Cette approche s'inspire de *Cycle* GAN (Zhu et al., 2017), et consiste à transférer l'image une première fois de la pose A à la pose B, puis de re-transférer l'image générée précédemment de la pose B à la pose A. L'erreur est alors calculée entre l'image d'entrée (avec la pose A) et l'image finale reconstruite, ce qui permet d'entraîner le réseau sans données appairées. Cela limite cependant les méthodes de supervision utilisables et produit donc des résultats de moins bonne qualité, ce qui explique que ces architectures soient moins représentées dans la littérature.

Nous décidons d'utiliser pour nos essais APS (Huang et al., 2020). Ce réseau produit des résultats réalistes sur les jeux de données Market-1501 et DeepFashion. De plus, il possède une architecture relativement simple par bloc qui permet de réaliser d'éventuelles modifications facilement, et ne nécessite pas de vérité terrain supplémentaire telles que des cartes de segmentation. Enfin, les auteurs fournissent un code pour l'entraînement et l'évaluation ainsi que des poids pré-entraînés pour les deux jeux de données principaux de l'état de l'art.

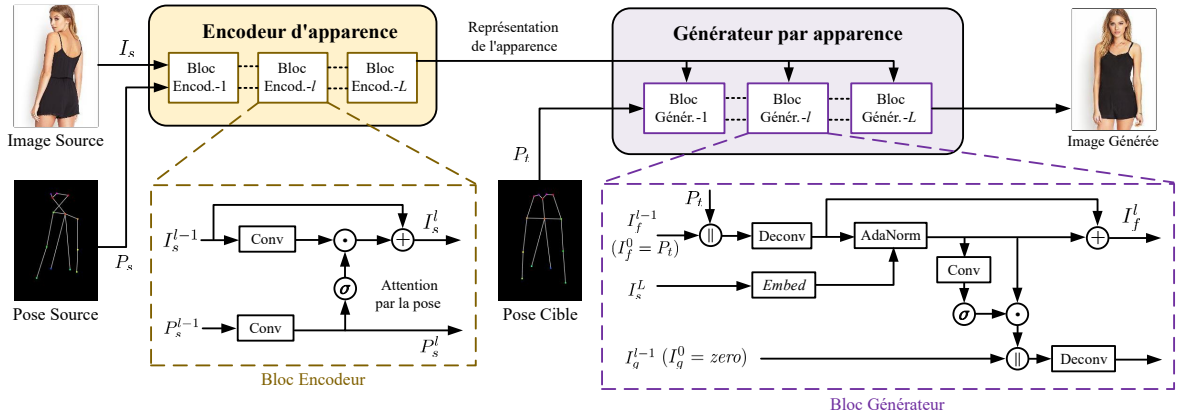


FIGURE 4.1 – Schéma du réseau APS pour le transfert de poses humaines (Huang et al., 2020). Notations : fonction d'activation sigmoïde σ, multiplication par élément $\cdot$, multiplication par élément $+$, concaténation par canal $||$.

Le réseau APS est divisé en deux parties distinctes : "l'encodeur d'apparence" et le "générateur par apparence" (Figure 4.1). L'encodeur est constitué de L blocs encodeurs identiques traversés par deux branches d'informations, prenant en entrée la sortie des deux branches du bloc précédent $l-1$ et produisant respectivement I_s^l et P_s^l. La première branche est chargée d'encoder l'image source tandis que la seconde branche encode la pose source. Un mécanisme d'attention est également utilisé en multipliant le vecteur de caractéristiques de l'image par celui de la pose. Ce mécanisme est mis en place dans l'objectif de pousser le réseau à se concentrer sur l'encodage de l'apparence de la personne, en particulier autour des points de la pose. Le vecteur de caractéristiques de l'apparence I_s^L ainsi calculé par l'encodeur est alors donné en entrée au générateur, composé également d'une succession de L blocs générateurs identiques et traversés par deux branches. Chaque bloc prend en entrée le vecteur d'apparence I_s^L, la pose cible P_t, ainsi que les sorties du bloc précédent $l-1$, et produit la sortie respective des deux branches I_f^l et I_g^l. Dans la première branche, la pose cible est concaténée avec la sortie du bloc précédent avant de passer à travers une couche de convolution fractionnée (notée "Deconv"). Ce nouveau vecteur est alors normalisé, en utilisant une méthode appelée "*Adaptive Patch Normalization*" (noté "*AdaNorm*") (Huang et al., 2020), et multiplié par le vecteur d'apparence calculé par le module encodeur. La méthode "*AdaNorm*" cherche à normaliser le vecteur par *patch* et non par canal ou par *batch* tout entier comme cela est fait habituellement (Ioffe and Szegedy, 2015; Ulyanov et al., 2016). L'intuition derrière cette approche est que les points du corps sont généralement situés approximativement dans les mêmes régions de l'image, et qu'il est donc intéressant d'être capable de normaliser indépendamment les régions les unes des autres. Une fois cette fusion effectuée entre le vecteur d'apparence et la pose cible, un mécanisme d'auto-attention est utilisé. Ce mécanisme consiste à calculer la carte d'attention partir du vecteur lui-même et non d'un vecteur extérieur. La sortie de cette étape d'attention est alors concaténée dans la seconde branche, avant de passer à travers une dernière étape de convolution afin de générer l'image totale.

Trois fonctions de coût sont utilisées pour entraîner ce réseau :

$$L_{L1} = \|I_t - \hat{I}_t\|_1 \tag{4.1}$$

$$L_{\text{per}} = \|\phi(I_t) - \phi(\hat{I}_t)\|_1 \tag{4.2}$$

$$L_{\text{GAN}} = \mathbb{E}[\log D_a(I_s, I_t) + \log(1 - D_a(I_s, \hat{I}_t)) + \log D_p(P_t, I_t) + \log(1 - D_p(P_t, \hat{I}_t))] \tag{4.3}$$

où ϕ représente les deux premières couches de convolution du réseau VGG16 (Simonyan and Zisserman, 2015) entraîné sur ImageNet, et D_a et D_p sont deux discriminateurs prenant en entrée la concaténation de l'image source et cible d'une part, et l'image et pose cible d'autre part. On a donc L_{L1} qui calcule la différence entre l'image générée et l'image source, et L_{per} qui fait de même entre les vecteurs de caractéristiques encodés par les premières couches de VGG. L'objectif de L_{per} est d'effectuer une comparaison des caractéristiques perçues, et non pixel par pixel. Enfin, L_{GAN} est la fonction d'entropie croisée utilisée pour la contrainte adversaire des réseaux génératifs. Ces trois fonctions sont additionnées pour calculer la fonction de coût total, avec trois hyperparamètres λ_1, λ_2 et α pour équilibrer leur impact sur l'entraînement, comme suit :

$$L = \lambda_1 L_{L1} + \lambda_2 L_{\text{per}} + \alpha L_{\text{GAN}} \tag{4.4}$$

4.2.2 Application de APS

Inférence directe

Nous commençons par tester la capacité de généralisation des modèles entraînés sur les jeux de données de l'état de l'art, puisqu'ils contiennent un grand nombre d'images variées. Nous réalisons donc une inférence sur des images de conducteur du réseau APS pré-entraîné. Nous choisissons d'utiliser les poids fournis par les auteurs du modèle entraîné sur le jeu de données Market-1501 car ce réseau présente des images avec plus de variations dans l'apparence, en particulier dans l'arrière-plan (les images de DeepFashion provenant d'un studio à fond blanc). Nous donnons en entrée des images du jeu de données DriPE, où chaque paire d'images (source ; cible) est tirée aléatoirement parmi les images de chaque conducteur.

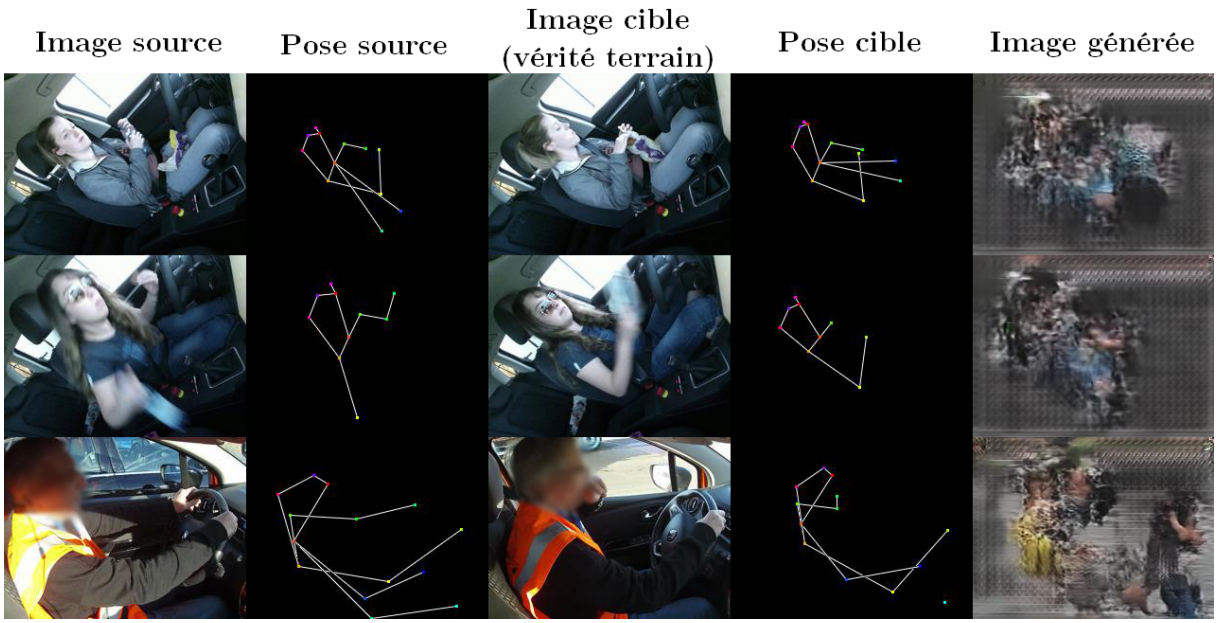


FIGURE 4.2 – Exemple d'inférence sur des images de conducteurs du réseau APS pré-entraîné sur Market-1501.

Comme montré dans la Figure 4.2, le réseau n'est pas capable de générer une image réaliste, et ce malgré une pose relativement proche entre la source et la cible. En effet, on peut voir que les images générées sont composées de *patchs* bruités de différentes textures, placés autour des points clés de la pose cible. On peut également observer qu'une importante partie de l'image est grise alors que cette couleur n'apparaît pas dans l'image source. Ce gris est caractéristique des images du jeu de données Market-1501 sur lequel a été entraîné le modèle, puisqu'il représente le gris du sol de la place où ont été capturées les images. À partir de ces observations, on peut déduire que ce réseau de transfert de poses possède une très mauvaise capacité à généraliser sur des données trop éloignées de celles d'entraînement, et qu'il est nécessaire de ré-entraîner un modèle dans notre contexte.

Entraînement de APS

Nous souhaitons donc entraîner APS avec des images de passagers de véhicules. Les jeux de données de l'état de l'art utilisés pour entraîner les solutions de transfert de poses sont des jeux relativement larges avec plusieurs dizaines de milliers de paires d'images et plusieurs milliers de sujets différents (Tableau 4.2). Nous constituons donc un jeu de données en rassemblant des images provenant de DriPE et de Drive&Act (Martin et al., 2019). Nous extrayons ainsi 15 k images des clips vidéo de Drive&Act, ce qui permet de constituer un jeu total 25 k images, dont 20 k pour l'entraînement et 5 k restantes équitablement réparties entre un jeu de validation et un jeu de test. Les trois ensembles d'images ainsi constitués ne partagent pas d'acteurs. Nous créons ensuite des paires tirées aléatoirement, composées de deux images d'un même acteur. Nous composons ainsi aléatoirement 120 k paires pour l'entraînement et 2,5 k paires pour la validation et l'évaluation, respectivement.

Nous entraînons le réseau avec ce jeu de données en suivant les hyperparamètres utilisés par les auteurs. Ainsi, le réseau est entraîné durant 800 *epochs*, avec un taux d'apprentissage initial de $2E^{-4}$ que nous réduisons linéairement jusqu'à 0 durant la deuxième moitié de l'entraînement. Deux ensembles de poids pour les fonctions de coût $(\alpha, \lambda_1, \lambda_2)$ (Équation 4.3) sont utilisés par les auteurs en fonction du jeu d'entraînement : $(5, 10, 10)$ pour Market-1501 et $(5, 1, 1)$ pour DeepFashion. Puisque les auteurs ne justifient pas cette différence d'hyperparamètres entre les deux jeux de données, nous réalisons un entraînement avec chaque ensemble. Les images sont recadrées autour de la personne et redimensionnées à 256×192 pixels, et des opérations d'augmentation de données sont appliquées (rotation, translation et changement d'échelle). Les poids des modèles sont mis à jour en utilisant l'algorithme d'optimisation ADAM (Kingma and Ba, 2014) avec des taux pour les moyennes mobiles exponentielles de $\beta_1 = 0.5$ et $\beta_2 = 0.999$. Les entraînements sont réalisés sur le centre de calcul de l'IN2P3 [13] avec deux cartes graphiques Nvidia Tesla V100 équipées de 32Go de VRAM chacune.

Nous observons d'abord que le réseau n'est pas capable de converger correctement lorsque nous utilisons les poids $(5, 1, 1)$ pour la fonction de coûts. En effet, le réseau uniquement des images noires produit au bout de quelques dizaines d'*epochs*. La forte contrainte imposée par le générateur par le biais de L_{GAN} en comparaison avec les deux autres fonctions de coût pourrait expliquer ce comportement. Le discriminateur peut converger trop rapidement par rapport au générateur, puisque les données d'entraînement utilisées contiennent un faible nombre de sujets différents par rapport aux jeux de données de l'état de l'art pour le transfert de poses. Nous pouvons alors nous retrouver dans une

13. Centre de calcul de l'IN2P3 : `https://cc.in2p3.fr/`

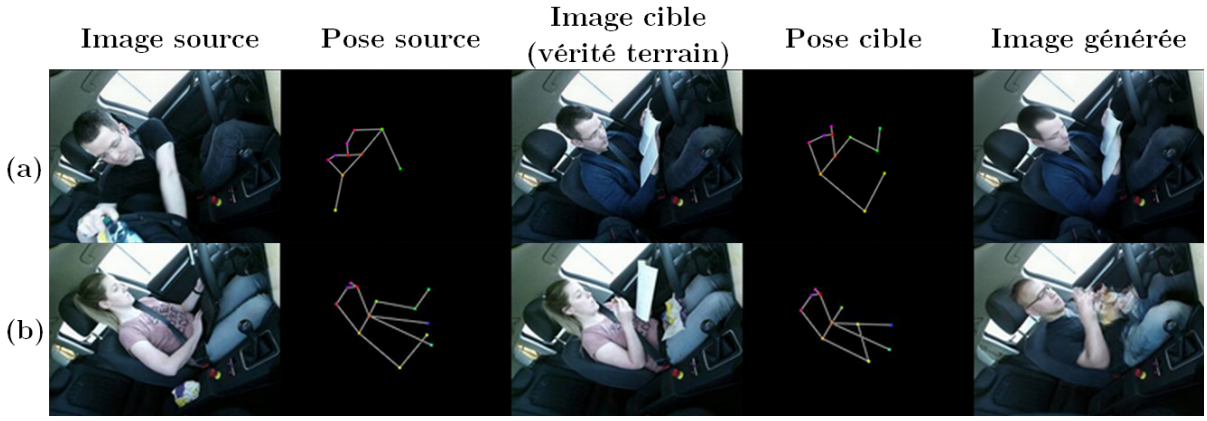


FIGURE 4.3 – Illustrations des résultats produits par le réseau APS entraîné sur les jeux de données DriPE et Drive&Act sur : (a) - une image d'entraînement, et (b) - une image de test.

situation de *mode collapse*, empêchant la convergence du générateur vers le comportement souhaité. (Arjovsky and Bottou, 2017).

À l'inverse, l'utilisation des hyperparamètres $(5, 10, 10)$ permettent au réseau de converger rapidement, puisque nous observons des résultats très réalistes sur les images d'entraînement au bout de 100 *epochs*, comme montré dans la Figure 4.3-a. Les images générées semblent cependant "trop parfaites", puisqu'on peut voir que le réseau a recréé le gilet et le livre du sujet alors qu'ils n'étaient pas présents dans l'image source. On peut donc en conclure que le réseau a totalement sur-appris sur le jeu d'entraînement. Cette hypothèse est validée lors de l'inférence sur des images de test (Figure 4.3-b), où nous pouvons voir que le réseau a remplacé l'identité de la personne source (visage, habits, accessoires). On remarque cependant que le réseau a reconstruit une image respectant la pose cible. Nous pouvons donc en déduire que le réseau n'est pas capable de généraliser sur les données d'entraînement concernant l'apparence des personnes et des images, ce qui laisse supposer un manque de diversité dans les images pour cette tâche.

Nous tentons d'appliquer des méthodes d'augmentation de données supplémentaires sur les images d'entrée pour résoudre ce problème, comme une inversion horizontale, des rotations, des découpages, ou des modifications du spectre de couleur. Ces transformations

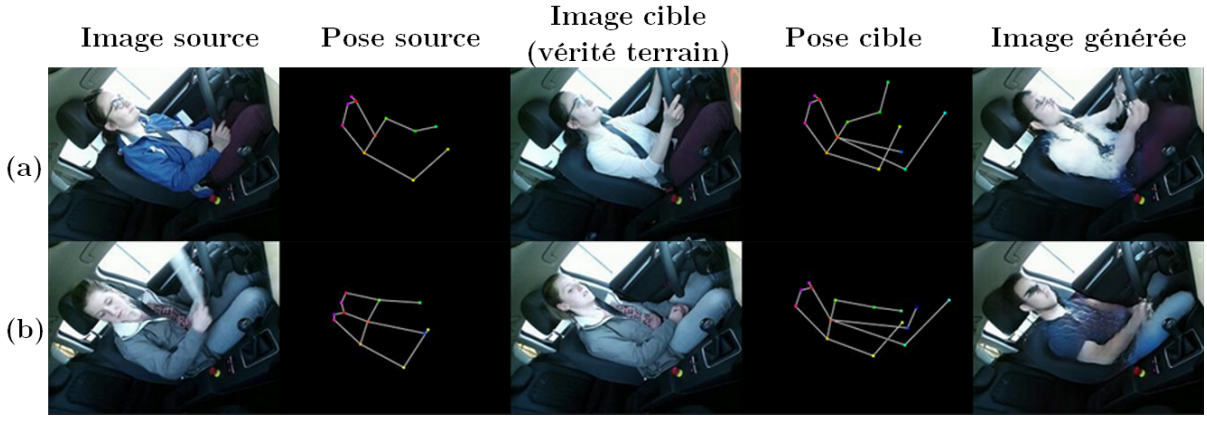


FIGURE 4.4 – Illustrations des résultats produits par le réseau APS entraîné sur les jeux de données DriPE et Drive&Act avec augmentation de données sur : (a) - une image d'entraînement, et (b) - une image de test.

ralentissent la convergence du réseau et réduisent la qualité des images générées, mais ne permettent pas de résoudre la problématique de sur-apprentissage, comme montré dans la Figure 4.4. Nous en déduisons donc que, dans l'état actuel, les données à notre disposition ne permettent pas d'entraîner correctement un réseau de neurones génératif pour le transfert de poses.

4.3 Génération des données synthétiques

Nous avons pour le moment tenté de résoudre le manque de données pour la généralisation des méthodes d'estimation de poses dans notre contexte à l'aide de réseaux génératifs, sans succès par manque de diversité dans les données initiales. Nous recherchons donc une méthode qui ne s'appuie pas sur des données pré-existantes pour générer de nouvelles images et annotations. Nous nous intéressons dans cette section à la génération d'images synthétiques par des méthodes de modélisation 3D.

Comme décrit dans la Section 2.5.2.2, il existe peu de jeux de données synthétiques utilisables pour l'estimation de poses humaines à l'intérieur d'une voiture (Cruz et al., 2020; Katrolia et al., 2021). De plus, ces jeux de données ont été développés principalement pour d'autres tâches telles que la reconnaissance d'actions ou la détection de personnes, et présentent donc peu de variété de poses et de sujets. L'angle de vue présenté dans ces jeux de données est également très différent de celui de notre étude et les auteurs ne fournissent aucune application permettant de générer de nouvelles images sous différents angles. Les auteurs de (Zhao et al., 2020) proposent une méthode pour générer des modèles 3D prenant des postures de conduite extraites de capture de mouvement. Ces travaux ne s'intéressent cependant pas à la modélisation de l'environnement, et aucune donnée n'a été publiée à notre connaissance. Nous développons donc une méthode permettant de générer des images répondant à nos besoins. L'approche adoptée est basée sur la modélisation de scènes 3D, suivie d'une génération de rendus images comparable à l'approche théorique présentée dans (Canas et al., 2022). Cette méthode est illustrée dans la Figure 4.5. Nous décrivons plus en détail dans cette section les étapes effectuées puis réalisons une première évaluation du jeu de données ainsi généré.

4.3.1 Méthode de génération

4.3.1.1 Modèles 3D

Pour construire une scène 3D représentant notre contexte, il est nécessaire de modéliser deux catégories d'objets : des voitures et des humains. Nous utilisons pour les modèles humains le logiciel libre de droit *MakeHuman Community* (MakeHuman, 2022). Ce logiciel permet de générer des modèles 3D en définissant différents paramètres comme l'âge, la taille, la masse musculaire, l'origine ethnique, etc. Ces différents paramètres sont alors utilisés pour définir l'ensemble des paramètres physiques du modèle 3D. Les modèles sont générés avec un squelette, ce qui permet de les animer facilement. Pour habiller ces modèles, nous utilisons les habits fournis par défaut dans le logiciel ainsi que plusieurs modèles fournis par la communauté. Puisque nous cherchons à générer un grand nombre de modèles humains variés, nous utilisons le module complémentaire *Mass Produce*. Cet outil permet de définir des intervalles pour chaque paramètre puis de générer aléatoirement un nombre prédéfini de modèles.

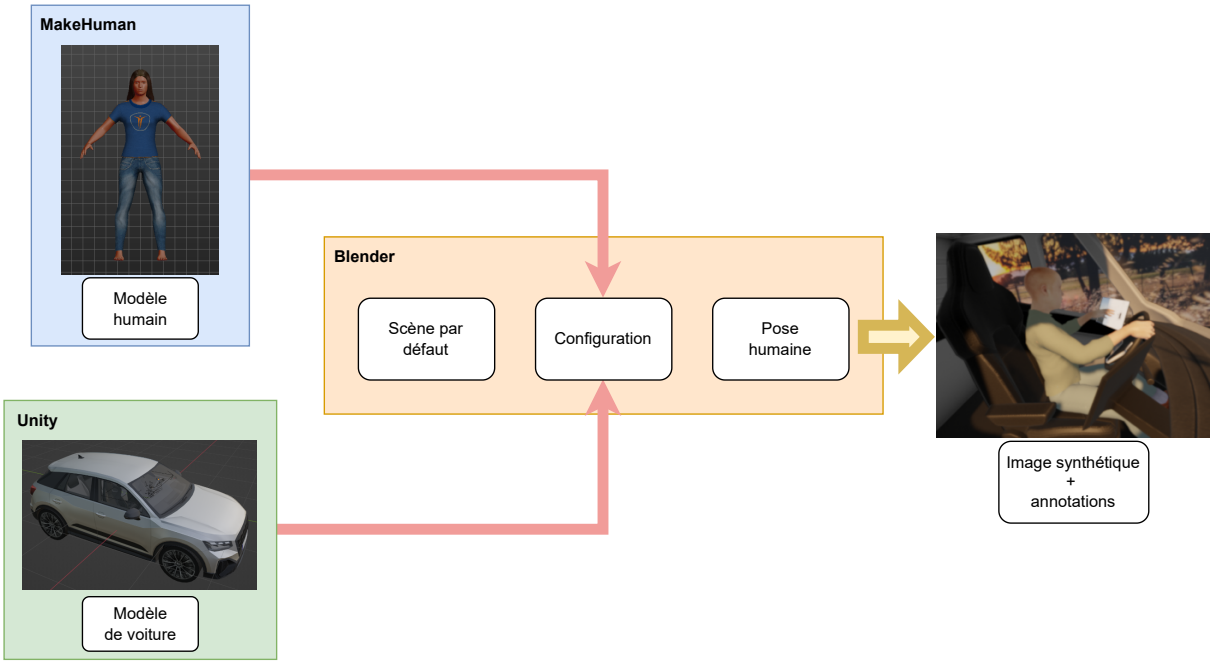


FIGURE 4.5 – Schéma de la méthode globale utilisée pour la génération des images synthétiques.

Les modèles 3D des voitures sont quant à eux issus du *Unity Asset Store* (Unity, 2022). Nous sélectionnons différents types de véhicules afin de représenter une variété d'habitacles (voitures familiales, voitures de sport, *pick-ups*, etc.) avec des équipements dans l'habitacle allant d'un simple tableau de bord à des écrans tactiles.

4.3.1.2 Génération des poses

Les modèles humains sont animés à l'aide du squelette intégré (Figure 4.6-a). Par défaut, chaque os du modèle peut effectuer une rotation libre autour de l'articulation de sa tête, ce qui lui confère trois degrés de liberté. Cependant, plusieurs contraintes supplémentaires sont applicables dans notre contexte. Premièrement, aucun os du corps humain réel ne peut effectuer une rotation complète dans une direction, puisqu'ils sont contraints par l'anatomie générale, le type de liaison, les muscles et ligaments, etc. Si l'on prend l'avant-bras comme exemple et que l'on considère sa position initiale comme étant entièrement ouverte, l'os peut approximativement effectuer une rotation de 0 à 150° le long des axes latéral et longitudinal et ne peut pas tourner autour de l'axe vertical (la rotation du bras dans cette direction est anatomiquement possible grâce à une rotation longitudinale de l'arrière-bras) (Maik et al., 2010). Ensuite, l'habitacle étant un espace restreint, il est nécessaire d'appliquer des contraintes supplémentaires sur la pose afin d'empêcher le modèle humain et celui de la voiture d'entrer en collision.

Afin de prendre en compte ces problématiques, nous procédons de la manière suivante :

1. Nous définissons une pose par défaut correspondant à la personne assise sur le siège conducteur, le dos et la tête droits et les bras proches du tronc.
2. Nous effectuons une petite rotation aléatoire de la tête, du dos et des jambes en considérant les contraintes du corps humain et du modèle 3D de l'habitacle.
3. Nous définissons aléatoirement deux cibles pour les poignets. Ces cibles sont des points de l'espace choisis dans les deux demi-sphères centrées sur les épaules, devant

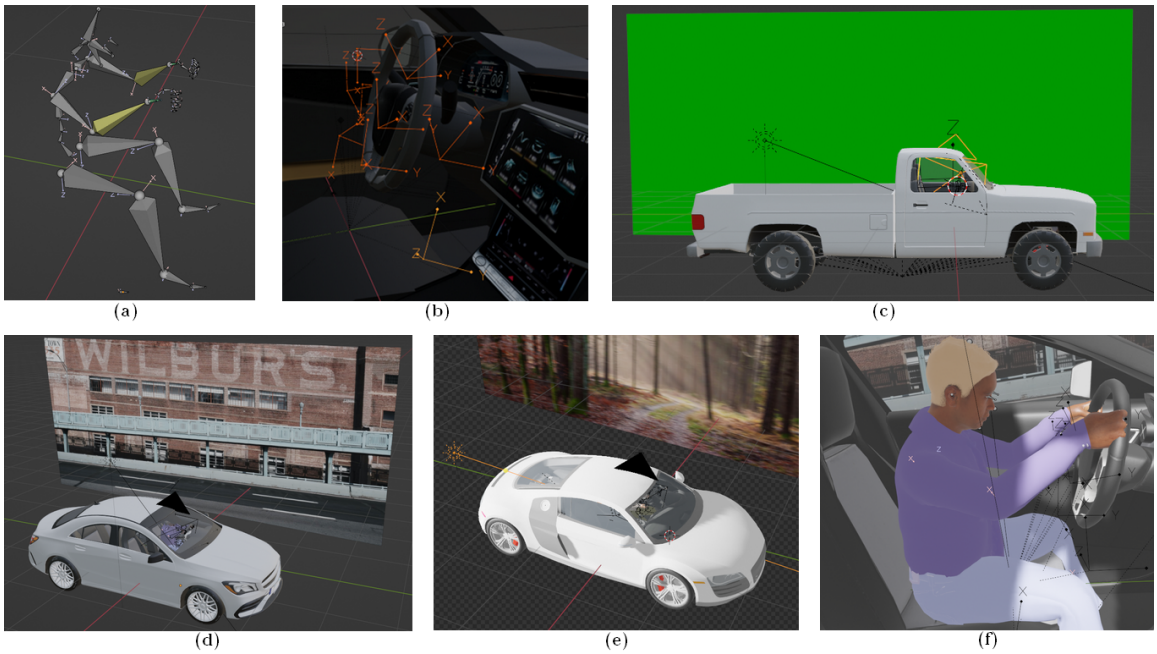


FIGURE 4.6 – Illustrations de la méthode de génération dans Blender avec (a) le squelette, (b) les cibles fixes pour les poignets (utilisées uniquement pour les images de conduite supplémentaires), (c) la scène par défaut, un exemple des scènes finales (d) sans, (e) avec le rendu de la lumière, et (f) une vue de la caméra.

la personne, et de rayon égal à la longueur des bras. Afin de simplifier la gestion des collisions qui peuvent être difficiles à généraliser étant donnée la diversité des modèles 3D des véhicules et leur provenance, nous ajoutons au préalable un volume 3D représentant l'espace disponible devant le passager dans chaque voiture. Les cibles ne peuvent donc pas être choisies en dehors de ce volume.

4. Nous utilisons de la cinématique inversée pour placer les poignets sur les cibles. Cette approche permet de déterminer les rotations des articulations en fonction de la position finale souhaitée des poignets. Nous déplaçons uniquement les bras (arrière et avant-bras) lors de cette étape puisque nous avons observé que cela permettait d'obtenir les poses les plus variées et réalistes. Le système de cinématique inverse est résolu en utilisant une méthode basée sur la représentation Jacobienne du problème (iTaSC, 2022), tout en prenant compte des contraintes de rotations anatomiques que nous imposons.

Cette méthode permet de générer facilement un grand nombre de poses aléatoires anatomiquement plausibles prenant en compte les contraintes imposées par l'habitacle de la voiture. Cependant, ce positionnement aléatoire des poignets a une probabilité très faible de donner des postures de conduite réalistes comme les mains sur le volant ou sur le levier de vitesse. Cela n'est pas problématique si nous considérons le véhicule comme autonome avec un niveau d'autonomie de niveau 2 ou plus, mais est moins réaliste pour la conduite manuelle d'un véhicule. Nous décidons donc de placer manuellement des points supplémentaires sur le volant, le levier de vitesse et le tableau de bord de chaque modèle de voiture (Figure 4.6-b). Ces cibles sont utilisées à la place des cibles aléatoires pour générer des images supplémentaires illustrant des postures de conduite, avec l'objectif d'obtenir un jeu de données varié.

4.3.1.3 Génération globale

Pour construire la scène globale et générer les images finales, nous utilisons le logiciel de modélisation Blender 3.2 (Blender, 2022). Notre choix se porte sur ce logiciel puisqu'il est gratuit et libre de droits, relativement accessible pour des utilisateurs extérieurs au domaine de la modélisation 3D, et il peut être entièrement automatisé par le biais du langage Python. La méthode globale est résumée dans la Figure 4.5.

Nous construisons d'abord une scène par défaut en installant une caméra fixe, une source lumineuse et un panneau pour accueillir l'arrière-plan (Figure 4.6-c). Nous utilisons des images de haute définition en tant qu'arrière-plan ; ce qui nous permet de pouvoir profiter facilement d'un grand nombre d'arrière-plans variés issus de jeux de données d'images libres de droits.

Ensuite, nous définissons aléatoirement plusieurs configurations, où une configuration est constituée d'un modèle humain, d'un modèle de voiture, d'un arrière-plan, de paramètres d'éclairage ainsi que de légères variations des paramètres de la caméra (Figures 4.6-d, e. À noter que le triangle noir dans les images représente uniquement le sens de la caméra). Dans le but d'augmenter la variété des modèles humains, nous changeons aléatoirement la couleur des habits et des cheveux pour chaque configuration. Pour la paramétrisation de l'éclairage, nous utilisons un module complémentaire de Blender permettant de déplacer le soleil de manière réaliste à partir de coordonnées GPS et d'une heure de la journée que nous définissons aléatoirement. Nous modélisons également des conditions de nuit en sélectionnant des arrière-plans adéquats et en réduisant la luminosité. Ces paramètres nocturnes sont utilisés dans 20 % des images totales.

Enfin, pour chaque configuration, nous générons plusieurs poses en utilisant la méthode décrite dans la Section 4.3.1.2 (Figure 4.6-f) avant d'effectuer un rendu des images finales. Nous enregistrons également les coordonnées 2D et 3D de la pose, la boite englobante ainsi que les paramètres intrinsèques et extrinsèques de la caméra. Un échantillon des images ainsi générées est présenté dans la Figure 4.7.

FIGURE 4.7 – Illustration des images synthétiques générées.

4.3.2 Évaluation des données synthétiques

Pour la suite de nos travaux, nous définissions un total de 1 000 configurations en choisissant aléatoirement parmi 7 modèles de voitures et 100 modèles humains. Pour chacune de ces configurations, 200 poses sont générées, ce qui nous permet d'obtenir un jeu de données de 200 k images, que nous nommons "PoseSynth". Nous comparons dans le Tableau 4.2 ce jeu de données synthétiques avec plusieurs autres jeux de la littérature. Comme nous pouvons le constater, notre jeu de données possède un plus grand nombre d'images que les autres jeux, qu'ils soient synthétiques ou réels. L'unique exception est Drive&Act qui est constitué de clips vidéo ce qui multiplie le nombre total d'images sans apporter de variété supplémentaire. Enfin, notre jeu de données présente beaucoup plus de modèles de conducteurs que ceux de l'état de l'art.

Dataset	SVIRO	TICaM	Drive&Act	DriPE	Market-1501	Deep-Fashion	PoseSynth
Année	2020	2021	2019	2021	2015	2016	2022
#Images	25 k	126 k	9.6 M	10 k	33 k	54 k	200 k
#Sujets	22 adultes	13	15	19	$\sim$ 3 k	$\sim$ 10 k	100
Synthétique / Réel	Synthétique	Synthétique et réel	Réel	Réel	Réel	Réel	Synthétique
Modalités	Profondeur, RGB, IR	Profondeur, RGB, IR	Profondeur, RGB, IR	RGB	RGB	RGB	RGB
Annotations	Classification, boîtes 2D, poses 2D	Boîtes 2D+3D, masques 3D, activités	Activités, poses 2D+3D	Boîtes et poses 2D	Poses 2D	Poses 2D	Poses et boîtes 2D+3D

TABLEAU 4.2 – Tableau de comparaison des différents jeux de données liés à notre étude avec notre jeu d'images synthétiques avec dans l'ordre : quatre jeux de données synthétiques et réels à l'intérieur d'une voiture, les deux jeux principaux pour le transfert de poses, et notre jeu de données.

Nous comparons ensuite la qualité de nos images synthétiques avec celles des autres jeux de l'état de l'art. Pour cela, nous utilisons deux métriques : l'*Inception Score* (IS) (Salimans et al., 2016) et le *Cumulative Probability Blur Detection* (CPBD) (Narvekar and Karam, 2011).

L'*Inception Score* vise à évaluer la qualité d'un ensemble d'images en s'appuyant sur les prédictions faites par le réseau de classification *Inception-v3* (Szegedy et al., 2016) entraîné sur le jeu de données Imagenet (Deng et al., 2009), qui produit une probabilité $p(y|x)$, ou score, pour chaque image x d'appartenir à la classe y. Cette métrique est basée sur deux préceptes. Premièrement, chaque image devrait contenir un nombre limité d'objets correctement identifiables par le réseau *Inception-v3*. La distribution $p(y|x)$ devrait donc avoir une entropie faible. Deuxièmement, l'ensemble des images devrait être le plus varié possible, donc la somme $\int p(y|x)$ devrait être la plus élevée possible (Barratt and Sharma, 2018). Le calcul de cette métrique s'exprime alors comme suit :

$$\text{IS} = \exp \mathbb{E}_x[\text{KL}(p(y|x)||p(y))] \tag{4.5}$$

où KL est la divergence de Kullback-Leibler :

$$\text{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{4.6}$$

avec P et Q deux distributions de probabilités discrètes.

La *Cumulative Probability Blur Detection* est une métrique s'intéressant à la netteté des contours sur l'image. Plus précisément, une détection de contours est réalisée sur l'image en appliquant un filtre de Sobel (Sobel, 2014), et pour chaque pixel de contour, une "probabilité de détecter le flou" est calculée comme :

$$P_{\text{BLUR}}(e_i) = 1 - \exp\left(-\left|\frac{w(e_i)}{w_{JNB}}\right|^{\beta}\right) \tag{4.7}$$

où e_i est le pixel évalué, $w(e_i)$ sa largeur, w_{JNB} une largeur seuil représentant le flou visible (*Just Noticeable Blur*) dépendant du contraste dans le voisinage du pixel, et β un paramètre de régularisation. Cette équation est utilisée pour calculer un histogramme de probabilité, et la métrique finale est définie comme :

$$\text{CPBD} = P(P_{\text{BLUR}} < P_{\text{JNB}}) = \sum_{P_{\text{BLUR}}=0}^{P_{\text{JNB}}} P(P_{\text{BLUR}}) \tag{4.8}$$

avec $P_{\text{JNB}} = 63$ %.

Il est important de noter que l'*Inception Score* dépend de l'implémentation et des poids utilisés pour le réseau *Inception-v3*. Ainsi, nous avons observé des différences entre les scores annoncés par les auteurs de APS, les scores mesurés avec les codes d'évaluation fournis par ces mêmes auteurs et les scores mesurés par une autre implémentation de l'*Inception Score* (Pytorch metrics, 2022)(Tableau 4.3). Nous utilisons donc dans la suite de ces travaux l'implémentation tierce qui mesure des performances proches de celles annoncées par les auteurs et qui utilise la même librairie que le reste de notre code, à savoir PyTorch (Paszke et al., 2019). Puisque l'*Inception Score* est sensible à la taille des images, tous les jeux de données sont redimensionnés dans le but de se rapprocher de la même taille, tout en conservant le ratio largeur / hauteur original de chaque jeu de données. Nous choisissons comme taille de référence une taille de 49 152 pixels, ce qui correspond à une taille de 192×256 pixels. Les résultats de cette évaluation sont montrés dans le Tableau 4.4.

	Score article APS	Implémentation auteurs APS	Implémentation tierse
Inception Score	3,295	3,038	3,183

TABLEAU 4.3 – Évaluation de l'*Inception Score* des images générées par le réseau APS entraîné par les auteurs sur le jeu de test DeepFashion avec différentes implémentations.

Premièrement, nous observons que les deux jeux de données utilisés pour le transfert de poses, à savoir DeepFashion et Market-1501, montrent un *Inception Score* beaucoup plus élevé que ceux mesurés sur les jeux de données dans un véhicule. Cela peut être expliqué par le fait que l'*Inception Score* mesure à la fois la qualité individuelle de chaque image et la diversité du jeu de données. Puisque les images de passagers présentent un faible nombre de personnes différentes, avec un habitacle en arrière-plan variant peu et occupant une large partie de l'image, un score plus bas pouvait être attendu. Cependant, notre jeu de données d'images synthétiques obtient un meilleur score que les autres jeux

Jeu de données Optimum	*Inception Score* (IS) ↑ $+\infty$	CPBD ↑ 1
DeepFashion	4,247	0,789
Market-1501	4,223	0,669
DriPE	1,481	0,644
Drive&Act	1,343	0,656
SVIRO	1,902	0,628
TICaM - synthétique	1,276	0,642
TICaM - réel	1,662	0,664
PoseSynth	2,391	0,611

TABLEAU 4.4 – Évaluation de la qualité des images de différents jeux de données.

de données du même contexte. Cela suggère que les images générées possèdent une qualité au moins équivalente à celles des autres jeux de données, tout en étant plus variées.

Concernant *Cumulative Probability Blur Detection*, on peut constater que le score mesuré sur le jeu de données DeepFashion est plus élevé que sur les autres jeux. Cela peut être expliqué par le fait que les images ont été capturées dans un studio avec un fond blanc ; ce qui permet d'avoir un sujet parfaitement exposé et aucune texture en arrière-plan, donc des contours nets. On mesure sur les autres jeux de données réels un score approximativement équivalent, et légèrement supérieur aux données synthétiques. Enfin, on mesure un score sur nos images du même ordre de grandeur que ceux mesurés sur les autres jeux de données synthétiques.

4.4 Application des images synthétiques au transfert de poses

Nous avons généré un grand nombre d'images synthétiques variées. Cependant, ces images sont peu réalistes visuellement en comparaison aux images des jeux de données réels. Générer des images à l'aspect réel permettrait de réduire la distance entre le domaine d'entraînement et notre domaine d'application. Nous souhaitons donc utiliser les données synthétiques générées afin d'aider à entraîner un modèle de transfert de poses à l'intérieur d'une voiture dans le domaine réel.

4.4.1 Transfert de poses dans le domaine synthétique

Nous commençons par entraîner et évaluer le réseau APS sur nos données synthétiques. Les images sont redimensionnées à la taille de 256×192 et sont divisées en un ensemble d'entraînement de 180 k images et deux ensembles de validation et de test de 10 k images chacun, ces ensembles ne partageant aucun modèle de personne. Pour chaque image, nous définissons une paire avec une autre image choisie aléatoirement de la même configuration de scène.

Les premières expérimentations montrent que le réseau converge plus rapidement que sur les jeux de données de l'état de l'art. Nous entraînons donc le modèle durant 400 *epochs* avec un taux d'entraînement initial de $2E^{-4}$. Suivant la recommandation des auteurs, nous faisons diminuer linéairement le taux d'entraînement jusque 0 à partir de l'*epoch* 200. Cependant, en s'appuyant sur les erreurs mesurées durant l'entraînement sur le jeu

de validation, nous présentons les meilleurs résultats, réalisés à l'*epoch* 310. Enfin, nos essais préliminaires montrent que les hyperparamètres pour la fonction de coût $(5, 1, 1)$ permettent d'obtenir les meilleurs résultats.

Des résultats qualitatifs sont montrés dans la Figure 4.8. Tout d'abord, nous pouvons observer que le réseau a correctement appris à effectuer le transfert de poses. En effet, les personnes présentées sur les images générées respectent la pose cible tout en préservant l'identité de la personne et l'environnement de l'image source. Ces résultats montrent que le jeu de données synthétiques est composé de suffisamment d'images diversifiées pour éviter un sur-apprentissage sur les identités des différents modèles. On peut cependant noter que le modèle présente des difficultés à reconstruire les hautes fréquences des textures, particulièrement celles des textures du modèle humain (cheveux, visages, motifs de vêtement). De la même manière, le réseau tend à générer des personnes portant un T-shirt aux manches mi-longues bien que le modèle source porte des manches courtes ou un débardeur. Cependant, le troisième exemple montre que le réseau conserve les hauts suffisamment éloignés d'un T-shirt, comme un haut "*crop top*" à manches longues.

Nous évaluons ensuite les performances du réseau à l'aide de plusieurs métriques et les comparons avec les performances des instances de APS entraînées et évaluées les jeux de données de l'état de l'art. Pour ce faire, nous utilisons les métriques *Inception Score* et *Cumulative Probability Blur Detection* utilisées dans la Section 4.3.2. De plus, nous intégrons plusieurs métriques visant à mesurer la distance de qualité perçue entre les images sources et celles générées :

- la *Fréchet Inception Distance* (FID) (Heusel et al., 2017) : cette métrique s'inspire de l'*Inception Score*, mais calcule la distance entre les distributions des images générées

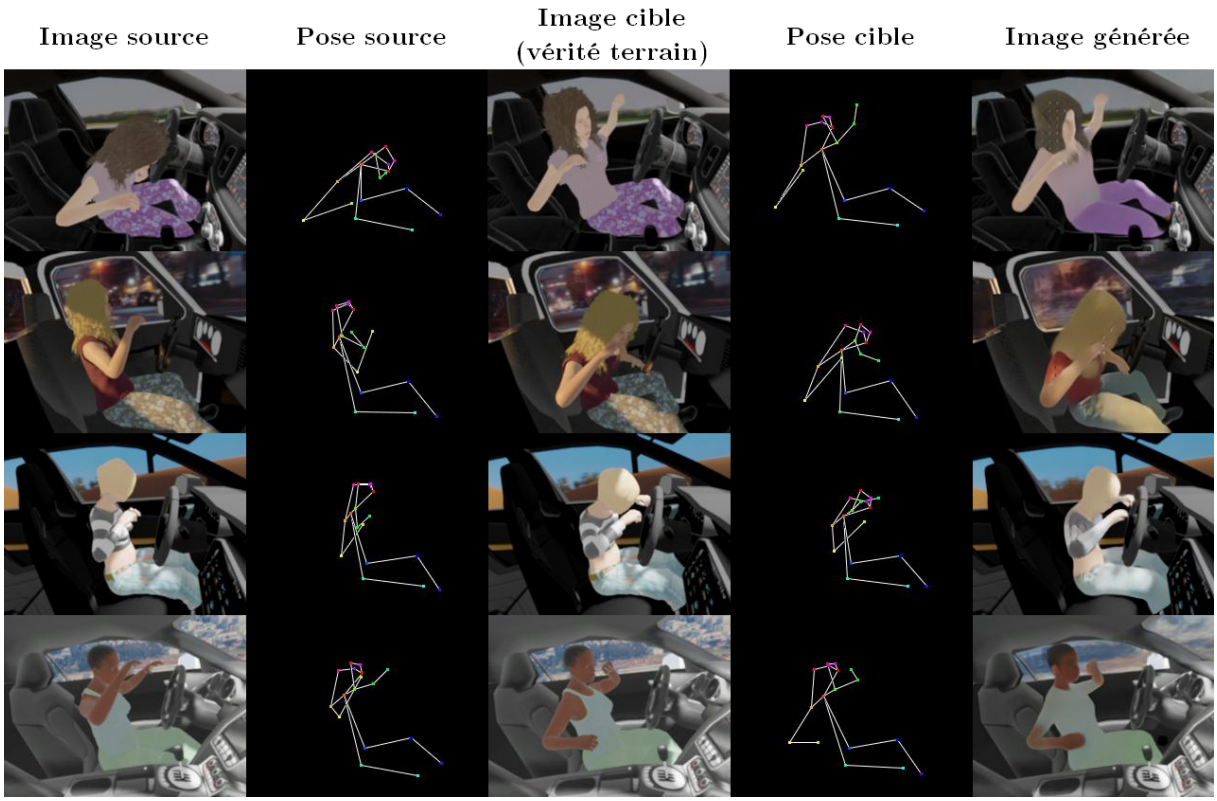


FIGURE 4.8 – Illustrations des résultats du transfert de poses sur le jeu de test des images synthétiques.

et des vérités terrain issues de *Inception-v3* pour comparer la qualité perçue. La métrique est calculée comme suit :

$$\text{FID}(X, Y) = ||\mu_X - \mu_Y||^2 + Tr(C_X + C_Y - 2\sqrt{C_X * C_Y}) \tag{4.9}$$

où X et Y sont les deux distributions comparées, μ_i la moyenne, C_i la matrice de covariance de distribution respectives, et Tr la trace de la matrice.

- la Similarité Structurelle (SSIM) définie dans la Section 3.2.1.

Les mesures des métriques sont présentées dans le Tableau 4.5. Tout d'abord, nous pouvons observer que l'*Inception Score* ainsi que la *Cumulative Probability Blur Detection* des images générées sont proches de ceux mesurés sur le jeu de données synthétiques dans le Tableau 4.4. Ensuite, la *Fréchet Inception Distance* entre les images de conducteurs générées par le GAN et les images de la vérité terrain est proche de celle mesurée sur le jeu de données Market-1501.

Jeu d'évaluation	IS ↑	CPBD ↑	FID ↓	SSIM ↑
Optimum	$+\infty$	1	0	1
DeepFashion	3,565	0,795	16,84	0,669
Market-1501	3,144	0,680	41,49	0,312
PoseSynth	2,456	0,612	38,06	0,810

TABLEAU 4.5 – Évaluation de la qualité des images générées par APS entraîné et évalué sur différents jeux de données.

Nous pouvons remarquer également que la FID mesurée sur le jeu de données DeepFashion est meilleure que celles mesurées sur le jeu de données Market-1501 et sur les images synthétiques. Cela peut s'expliquer par la simplicité du contexte des images de mode, en particulier l'absence d'arrière-plan complexe, de parties du corps entièrement visibles, etc. Enfin, on peut observer que la Similarité Structurelle (SSIM) mesurée est plus élevée sur notre jeu de données synthétiques que sur les jeux de données DeepFashion et Market-1501. La raison de cet écart est que plus de la moitié de la surface des images de conducteurs est composée d'un arrière-plan fixe (habitacle du véhicule et paysage) que le réseau GAN peut facilement préserver puisqu'il ne change pratiquement pas pendant le transfert de poses. À l'inverse, les images composant le jeu de données DeepFashion sont plus cadrées autour de la personne. Ces mesures valident le fait que les images synthétiques générées permettent d'entraîner un réseau pour le transfert de poses humaines dans un habitacle de voiture, dans le domaine synthétique.

4.4.2 Application dans le domaine réel

Nous souhaitons ensuite utiliser les images synthétiques pour entraîner un modèle capable d'effectuer du transfert de poses humaines sur des images réelles de passagers d'une voiture. La première approche consiste à réutiliser directement le modèle d'APS entraîné pour le transfert de poses dans le domaine synthétique dont les performances ont été présentées dans le Tableau 4.5. Nous effectuons d'abord une inférence directe avec les images issues de Drive&Act et DriPE utilisées dans la Section 4.2. Comme montré dans la Figure 4.9-a, le réseau produit des images avec un phénomène similaire à celui observé dans la Figure 4.2 avec le réseau pré-entraîné sur Market-1501. En effet, nous pouvons

voir une forme approximative de l'habitacle de voiture avec une personne au milieu. Bien que les formes soient plus proches du résultat souhaité qu'avec le réseau pré-entraîné sur Market-1501, les textures ne correspondent pas à celles de l'image source et l'image est dénuée de hautes fréquences.

Nous effectuons donc un *fine-tuning* de ce même modèle sur le jeu de données utilisé précédemment constitué des images de Drive&Act et DriPE. Plus précisément, nous effectuons un apprentissage durant 10 *epochs* avec un taux d'apprentissage de $2E^{-5}$. Des exemples d'inférences d'images du jeu de test sont montrés dans la Figure 4.9-b. Comme nous pouvons le voir, l'identité de la personne cible n'est pas conservée dans les deux premières images générées puisqu'on peut reconnaître une silhouette masculine habillée avec un polo. Le troisième exemple provenant du jeu DriPE ne produit pas une image exploitable, bien que l'on semble reconnaître la forme de l'habitacle des images de Drive&Act. Prolonger l'entraînement permet d'améliorer la qualité de l'image générée, mais renforce également le sur-apprentissage sur le jeu d'entraînement. Le pré-entraînement sur les images synthétiques ne semble donc pas être suffisant pour empêcher le phénomène de

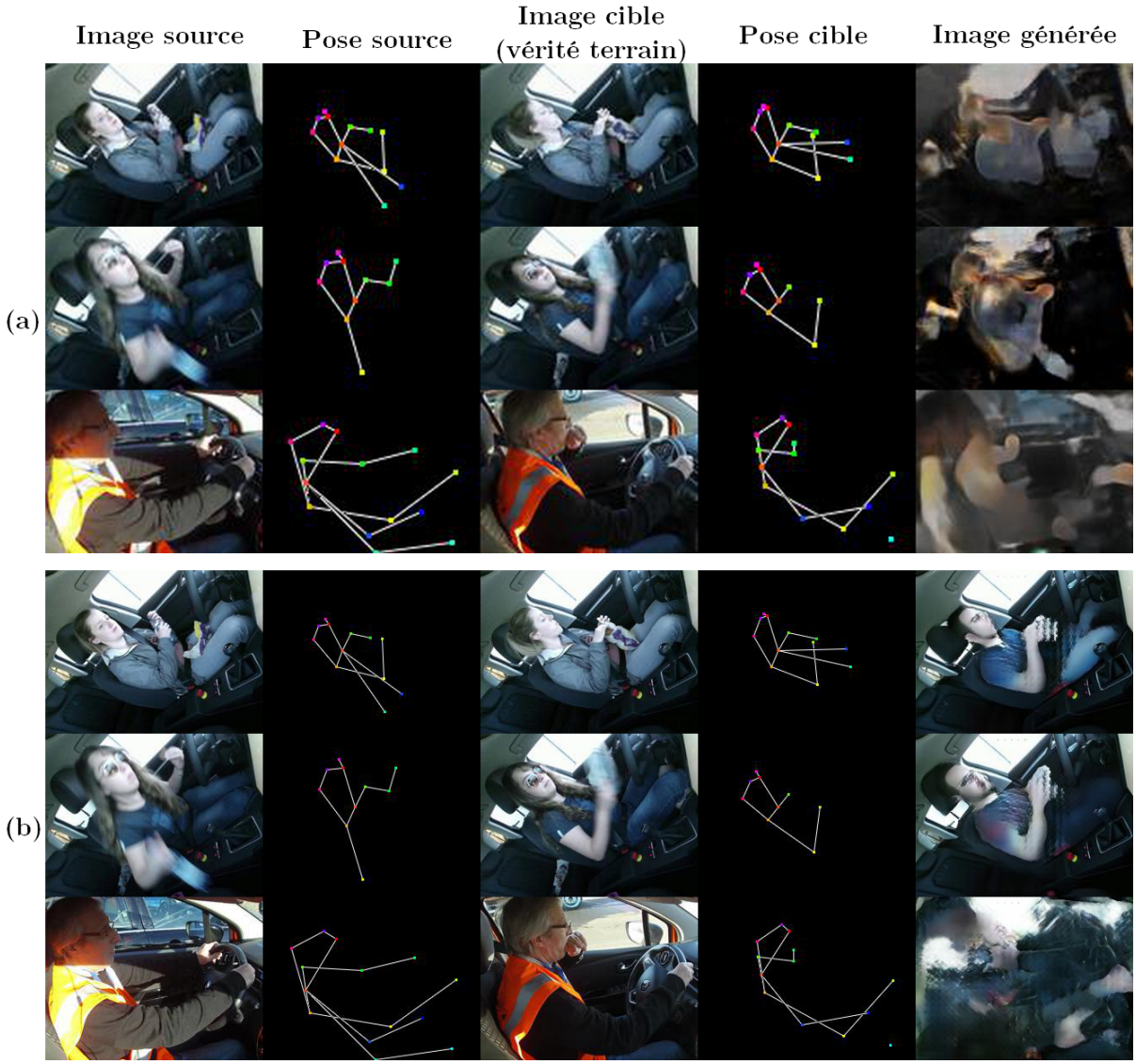


FIGURE 4.9 – Illustrations des résultats produits par le réseau APS entraîné sur les images synthétiques : (a) - en inférence directe (b) - après *fine-tuning* sur Drive&Act et DriPE.

sur-apprentissage.

Nous décidons alors d'entraîner un nouveau modèle sur un jeu d'entraînement mélangeant les images synthétiques et les images réelles. L'objectif est de limiter le sur-apprentissage sur les images réelles en continuant à fournir au fur et à mesure de l'entraînement des images synthétiques qui sont plus diversifiées. Nous utilisons donc dans chaque *batch* 15 % d'images réelles et 85 % d'images synthétiques. De plus, nous implémentons plusieurs augmentations de données afin de tenter de réduire la différence perçue par le réseau entre les deux sources d'images. Nous utilisons ainsi :

- une rotation fixe de 40° sur les images synthétiques afin d'aligner les poses sur celles de Drive&Act (en plus des rotations aléatoires appliquées par défaut lors de tous les entraînements),
- une normalisation de la couleur des images,
- une inversion aléatoire des canaux de couleurs RGB,
- une translation aléatoire correspondant à 10 % de la taille de l'image,

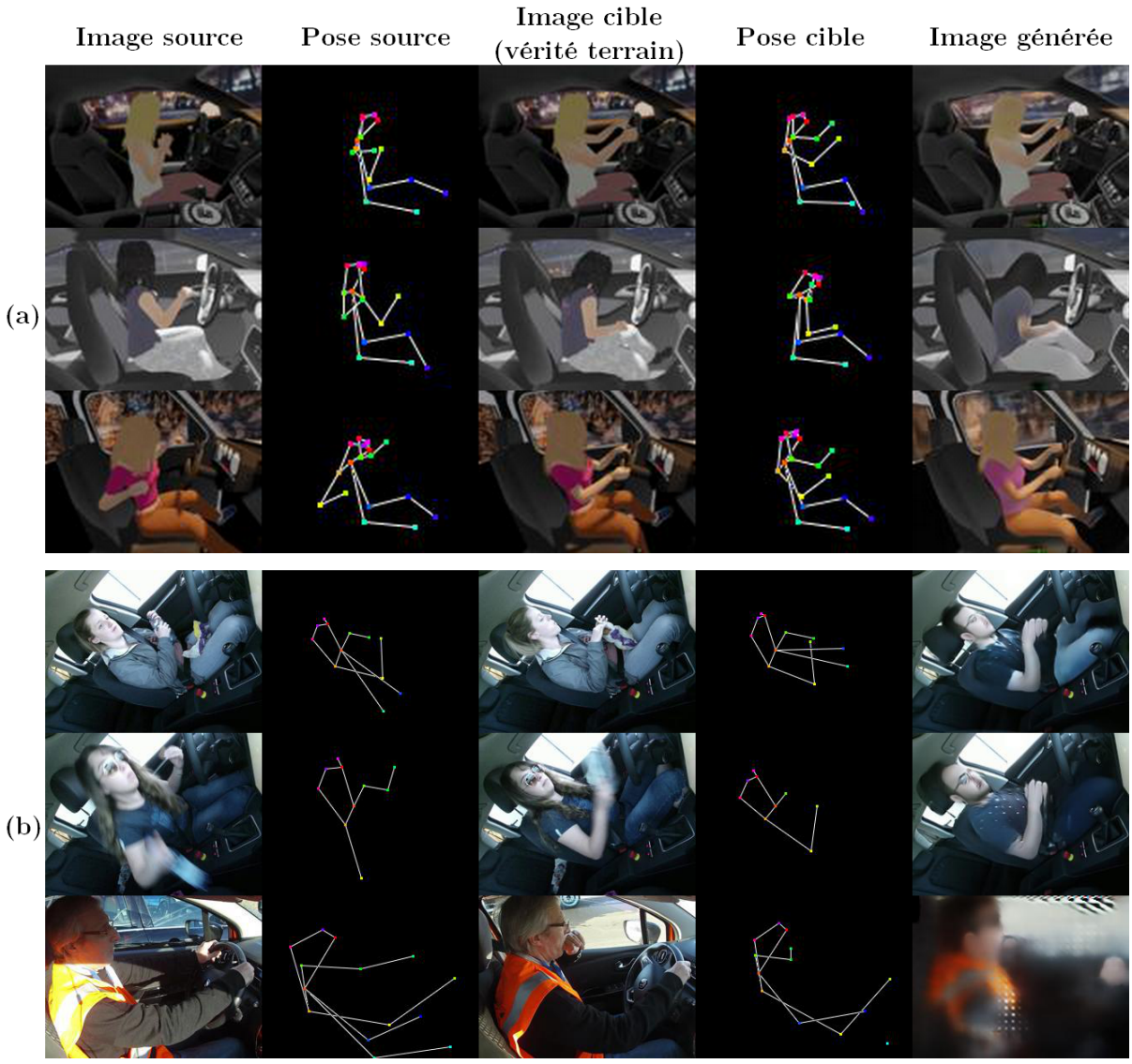


FIGURE 4.10 – Illustrations des résultats de l'évaluation du réseau APS entraîné à la fois sur les données réelles et synthétiques.

- une modification aléatoire de l'échelle de l'image de ±10 %.

Ces augmentations de données ne sont pas appliquées lors de l'évaluation. En plus de ces augmentations de données, nous décidons d'utiliser 4 blocs encodeurs et générateurs dans l'architecture APS au lieu de 5 afin de réduire la taille du réseau et donc son potentiel de sur-apprentissage.

Des images de l'évaluation sont montrées dans la Figure 4.10. Comme nous pouvons le voir, le réseau est capable de discerner les images synthétiques et les images réelles. En effet, le réseau est capable de généraliser dans le domaine synthétique puisque l'identité de la personne source est conservée sur les images synthétiques (Figure 4.10-a). À l'inverse, le sur-apprentissage sur les images réelles est toujours présent puisque le réseau reproduit l'apparence des personnes du jeu d'entraînement dans l'image générée (Figure 4.10-b). Dans le but de diminuer l'impact des images synthétiques sur l'apprentissage, nous avons reproduit cette expérience en augmentant le ratio d'images réelles. Nous avons pour un premier entraînement utilisé 1/3 d'images réelles et 2/3 d'images synthétiques dans les *batchs*. Lors d'un second entraînement, nous commençons avec 15 % d'images réelles et augmentons progressivement ce ratio de 15 % toutes les 100 *epochs*, terminant ainsi avec 60 % d'images réelles. Aucun de ces deux entraînements n'a montré de résultats différents de ceux présentés dans la Figure 4.10. Nous pouvons en déduire que les méthodes d'augmentation de données et la réduction de la taille du réseau n'ont pas été suffisantes pour empêcher le réseau de différencier les deux modalités d'images et de sur-apprendre sur les images réelles.

4.5 Conclusion

Dans ce chapitre, nous avons cherché à augmenter la quantité et la diversité des données à disposition pour l'estimation de poses humaines. En effet, nous nous intéressons à la généralisation de domaine, et avons donc décidé d'utiliser DriPE pour évaluer la généralisation des réseaux entraînés. Ainsi, nous avons montré que des données spécifiques à notre contexte semblaient nécessaires pour améliorer la performance des réseaux entraînés sur COCO, en particulier pour le bas du corps.

Nous nous sommes donc intéressés aux méthodes de génération de nouvelles données. La première approche que nous avons étudiée est celle du transfert de poses humaines. Cette méthode étant principalement utilisée dans les contextes des images de mode et de la vidéosurveillance, nous avons expérimenté l'application du transfert de poses dans notre contexte. Une inférence directe d'un réseau pré-entraîné nous a montré que le réseau générateur utilisé n'était pas capable de généraliser sur d'autres jeux de données. Nous avons donc effectué d'une part un *fine-tuning* de ce réseau et d'autre part un entraînement complet sur des images de passagers de voitures. Les modèles ainsi entraînés ont tous deux montré un fort sur-apprentissage puisque les images générées lors de l'évaluation illustrent des personnes du jeu d'entraînement et non celles de l'image source.

Puisque les données à disposition n'étaient pas suffisantes pour entraîner un réseau génératif, nous nous sommes intéressés aux données synthétiques. Les jeux de données synthétiques de l'état de l'art dans notre contexte présentent peu de diversité dans les modèles humains utilisés et aucune application permettant de générer directement nos propres données n'a été publiée. Nous avons donc proposé une méthode basée sur Blender, un logiciel de modélisation 3D, pour générer un grand nombre d'images synthétiques de personnes dans un véhicule. L'utilisation du logiciel MakeHuman nous a permis d'utiliser

un grand nombre de modèles humains avec une importante diversité. Nous avons ainsi proposé PoseSynth, un jeu de données de 200 k images synthétiques avec 100 modèles humains différents dans des poses aléatoires et des poses de conduite, avec différents modèles de voitures, arrière-plans, conditions d'éclairages, etc. Nous avons alors montré que la qualité mesurée sur les images générées était comparable à celles mesurées sur les différents jeux de l'état de l'art dans notre contexte.

Enfin, nous avons appliqué PoseSynth aux réseaux de transfert de poses humaines afin de générer des images à l'aspect réel. Nous avons d'abord montré qu'un réseau entraîné sur les nouvelles images était capable de réaliser efficacement le transfert de poses dans le domaine synthétique. Cela confirme que les images générées possèdent une qualité et une diversité suffisantes pour entraîner un réseau sur une tâche demandant beaucoup de données. Nous avons alors associé ces images synthétiques aux images réelles en cherchant à résoudre la problématique de sur-apprentissage. Les résultats ont cependant montré que le réseau était capable de différencier les images réelles des images synthétiques et ainsi de sur-apprendre sur les images réelles, et ce quelle que soit la méthode d'entraînement utilisée. Le transfert de poses ne semble donc pas être une solution applicable dans notre situation puisque les modèles demandent un nombre de données réelles trop important pour être entraînés, et sont fortement dépendants du domaine des images d'entraînement. Des méthodes de *transfert learning* plus poussées pourraient être testées afin de réduire l'écart de domaine, bien qu'aucun article n'ait spécifiquement été publié à ce sujet dans le cadre du transfert de poses.

Une approche alternative envisageable serait l'utilisation de réseaux de transfert de poses entraînés de manière non supervisée (Pumarola et al., 2018; Song et al., 2019). L'avantage d'une telle approche est que le générateur peut plus difficilement sur-apprendre sur le jeu d'entraînement puisqu'aucune image n'est donnée en tant que vérité terrain. Les approches non supervisées requièrent cependant généralement plus de données pour converger et n'empêchent pas totalement un sur-apprentissage. Une autre approche pour générer davantage de données réalistes pourrait être de rendre plus réalistes les données synthétiques. En effet, nous avons utilisé des méthodes de modélisation simples (textures, luminosité, etc.). Avec plus de connaissances en modélisation et en animation, il serait possible de générer des images plus proches des images réelles. Une autre approche pourrait être d'utiliser des réseaux de neurones pour le transfert de style afin d'appliquer sur nos images synthétiques l'apparence d'images réelles (Isola et al., 2017; Karras et al., 2019; Park et al., 2020). Cependant, ces architectures étant des réseaux génératifs, leur entraînement pourrait rencontrer les mêmes difficultés que l'entraînement des réseaux de transfert de poses sur les images réelles.

Les travaux présentés dans ce chapitre ont fait l'objet d'une publication internationale. Notre méthode de génération des images synthétiques a été présentée lors de la conférence VISAPP 2023 (Guesdon et al., 2023). Les codes utilisés ainsi que le jeu de données généré sont disponibles publiquement sur un dépôt en ligne [14].

14. `https://gitlab.liris.cnrs.fr/aura_autobehave/synthetic_drivers`

Chapitre 5

Généralisation de domaine pour l'estimation de poses

Dans ce chapitre, nous étudions l'impact des données sur la généralisation de domaine des méthodes pour l'estimation de poses humaines. Nous utilisons d'abord le jeu de données PoseSynth créé dans le chapitre précédent pour entraîner une instance de *Simple Baseline* capable de généraliser sur DriPE. En s'appuyant sur les observations faites lors de ces expérimentations, nous mettons à jour notre méthode de génération des images synthétiques afin de diminuer l'écart de domaine avec les images réelles. Nous étudions alors l'efficacité de plusieurs méthodes de traitement de données sur la généralisation de domaine. Enfin, nous proposons deux nouvelles architectures extrayant des caractéristiques propres à chaque domaine afin de mieux tirer parti des spécificités de chaque jeu de données.

Sommaire

5.1 Motivations

Comme introduit dans le chapitre précédent, nous cherchons à entraîner un réseau pour l'estimation de poses capable de généraliser, c'est-à-dire montrant de bonnes performances sur des images enregistrées dans des conditions différentes de celles vues lors de l'entraînement. Nous avons généré dans le Chapitre 4 le jeu de données synthétiques PoseSynth, afin d'ajouter aux images d'entraînement des images illustrant des personnes en position de conduite. L'objectif est d'utiliser ces nouvelles données afin d'améliorer la performance des modèles entraînés sur COCO, un jeu de données générique, et en particulier les performances sur le bas de corps. Nous commençons donc par appliquer des méthodes de transfert d'apprentissage sur le réseau *Simple Baseline* étudié précédemment, dont les performances initiales sur DriPE sont montrées dans le Tableau 4.1. Plus précisément, nous réalisons d'abord un *fine-tuning* du réseau avec les images de PoseSynth. Le réseau est entraîné sur les images synthétiques durant 10 *epochs* avec un taux d'apprentissage de $10E^{-5}$, et nous conservons les poids donnant la plus faible erreur sur l'ensemble de validation (*epoch* 6 dans notre cas). Les autres hyperparamètres et augmentations de données utilisés sont ceux présentés dans la Section 3.2.3.2 (retournement horizontal, rotation et changement d'échelle aléatoires, algorithme d'optimisation ADAM (Kingma and Ba, 2014)). Ce procédé d'entraînement avec ces hyperparamètres sera réutilisé tout au long de ce chapitre et référé sous le nom de "procédé de *fine-tuning* par défaut". Les résultats de l'évaluation sur PoseSynth et sur DriPE complet sont montrés dans les Tableaux 5.1 et 5.2, respectivement.

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	98	100	97	97	100	100	100	99
AR	98	100	97	98	100	100	99	99

TABLEAU 5.1 – Performances de *Simple Baseline* sur le jeu d'évaluation de PoseSynth après un *fine-tuning*, évaluées avec mAPK (en %).

Nous pouvons tout d'abord observer dans le premier tableau que le modèle a parfaitement appris à réaliser l'estimation de poses sur les images synthétiques. En effet, on mesure des performances entre 97 et 100 % sur toutes les parties du corps, aussi bien pour la précision que pour le rappel. Cependant, ces performances ne semblent pas se transposer sur DriPE, comme nous pouvons l'observer dans le Tableau 5.2. Premièrement, les performances globales mesurées sur DriPE sont plus faibles que celles du réseau entraîné uniquement sur COCO (Tableau 4.1), puisque la précision et le rappel totaux étaient respectivement de 82 et 67 %, soit une perte d'environ 30 % de précision et 15% de rappel. Cette importante diminution de la précision démontre une surconfiance du réseau, qui peut être expliquée par la similarité entre les poses prises et l'apparence visuelle dans les images synthétiques. Les caractéristiques apprises sur PoseSynth ne permettent donc pas au modèle de prédire correctement la pose sur les images de DriPE. Cependant, on peut

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	30	52	54	63	44	38	27	43
AR	66	57	58	65	43	32	17	52

TABLEAU 5.2 – Performances sur le jeu DriPE complet après un *fine-tuning* de *Simple Baseline* sur PoseSynth, évaluées avec mAPK (en %).

constater que la perte de rappel sur le haut du corps est en partie compensée par une augmentation de 14 % de rappel sur les chevilles. Cette augmentation de rappel semble indiquer que l'entraînement sur les images synthétiques permet bien d'habituer le réseau à l'angle de vue particulier pour le bas du corps, partiellement occulté dans notre contexte. Cependant, cet entraînement semble diminuer les capacités du réseau à généraliser sur le haut du corps puisqu'il détecte correctement moins de points clés que le modèle de base entraîné sur COCO. Enfin, on peut constater une baisse plus importante de précision sur le visage que sur le reste du corps. Cette différence de résultats entre les deux jeux de données semble confirmer qu'il existe un écart de domaine trop important entre les images de PoseSynth et les images de DriPE qui empêche le réseau de généraliser. Cet écart avait déjà été observé dans la Section 4.4.2 lorsque les données de PoseSynth n'avaient pas permis d'entraîner un réseau de transfert de poses dans le domaine réel.

Il semble donc nécessaire de réduire l'écart de domaine entre le jeu de données synthétiques et le domaine cible afin d'améliorer les performances de généralisation. Pour ce faire, nous appliquons dans un premier temps des augmentations de données génériques supplémentaires. En effet, les images de PoseSynth sont toutes approximativement cadrées de manière identique, puisque peu de mouvements de caméra ont été intégrés. Ce choix avait été réalisé afin de maximiser la probabilité que la totalité de la personne soit dans le cadre de l'image. Cela limite cependant la diversité de placement du corps vis-à-vis du centre de l'image. De plus, les augmentations de données utilisées initialement pour l'entraînement de *Simple Baseline*, c'est-à-dire la rotation aléatoire, le retournement horizontal et le changement d'échelle, ne permettent pas de déplacer le centre de l'image. Nous choisissons donc d'ajouter une translation aléatoire horizontale et verticale d'un maximum de 20 % de la largeur et de la hauteur de l'image, respectivement, afin de déplacer le centre de l'image lors de l'entraînement.

Ensuite, nous nous intéressons à la différence d'aspect visuel global entre les images synthétiques et les images réelles. En effet, bien que les images de PoseSynth illustrent des modèles humains dans des véhicules réalistes, ces images possèdent une apparence globale moins détaillée et précise que les images réelles, en particulier pour les textures et les illuminations. Cet écart de domaines peut causer une différence de performances entre l'entraînement sur des images synthétique et l'évaluation sur des images réelles (Tremblay et al., 2018). Pour répondre à cette problématique, nous choisissons d'appliquer une méthode de changement aléatoire de style (*style randomization*) (Jackson et al., 2019; Wang et al., 2022). Cette approche cherche à rendre le modèle entraîné invariant au style de l'image, en modifiant aléatoirement le style lors de l'entraînement à l'aide d'un réseau de neurones génératif. Nous utilisons pour notre application le réseau StyleAugmentor (Jackson et al., 2019) puisqu'il permet de facilement altérer le style des images, et les auteurs fournissent une implémentation complète avec les poids entraînés. Ce réseau consiste en un auto-encodeur composée de plusieurs blocs résiduels, utilisé pour ajouter un vecteur de style généré aléatoirement au vecteur de caractéristiques de l'image. Un exemple d'images que l'on peut obtenir est illustré dans la Figure 5.1.

Nous effectuons ainsi deux nouveaux entraînements de *Simple Baseline* avec PoseSynth pré-entraîné sur COCO, en suivant le "procédé de *fine-tuning* par défaut" et en ajoutant d'une part la translation et d'autre part le changement de style. Les résultats sont présentés dans le Tableau 5.3.

Tout d'abord, nous n'observons aucune différence significative après l'ajout de translations, avec au total un gain de précision de 1 % pour une perte de rappel de 1 % ; ce qui indique que le décentrage des images n'impacte pas les performances du réseau sur

FIGURE 5.1 – Illustrations d'images de PoseSynth avec le style changé aléatoirement.

Augment. supp.		Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
-	AP	30	52	54	63	44	38	27	**43**
translation		28	49	53	62	45	38	28	42
style		29	48	52	59	44	41	29	42
-	AR	66	57	58	65	43	32	17	52
translation		66	57	60	66	45	37	21	**53**
style		65	52	53	60	44	37	22	50

TABLEAU 5.3 – Performances sur le jeu DriPE complet après un *fine-tuning* de *Simple Baseline* sur PoseSynth avec différentes augmentations de données supplémentaires, évaluées avec mAPK (en %).

DriPE. Par ailleurs, la modification de style semble légèrement réduire les performances du modèle, et en particulier le rappel avec une diminution de 2 % de rappel global. Cela démontre une augmentation du nombre de faux négatifs ; ce qui indique que le réseau a légèrement perdu en confiance. Ce résultat pourrait-être expliqué par le bruit ajouté par les différents styles. On peut également relever un léger gain de rappel sur les genoux et les chevilles respectivement de 4 et 5 %. Cette hausse est cependant compensée pour les deux augmentations par une diminution des performances sur les autres parties du corps, puisque le rappel total est supérieur de seulement 1 % pour la translation, et inférieur pour le changement de style. On peut donc en conclure que ces augmentations de données ne permettent pas d'améliorer les capacités de généralisation du modèle.

Puisque ces augmentations de données génériques ne semblent pas être efficaces, nous nous intéressons à une transformation plus spécifique à notre contexte. En effet, une des particularités des images enregistrées à l'intérieur d'une voiture est le cadrage des caméras autour du sujet. Selon le véhicule et la caméra utilisée, il est parfois difficile de capturer la totalité du corps du passager. De plus, le véhicule en mouvement peut induire d'importants mouvements de caméra dus aux vibrations et aux secousses. Ainsi, la caméra peut se décentrer involontairement au cours de l'expérimentation, comme on peut l'observer dans le jeu de données DriPE ; ce qui a pour conséquence de faire sortir une partie du sujet du champ de vision, par exemple la tête (Figure 5.2-gauche). Cette situation n'est pas prévue dans les images de PoseSynth, ce qui peut pousser les réseaux entraînés sur ces dernières à toujours prédire les points du visage, causant une diminution de la précision.

Nous décidons donc d'implémenter un rognage plus important des images d'entraînement lors de la phase d'augmentation de données afin de mieux représenter notre contexte (Figure 5.2-droite). Ainsi, l'image est rognée en haut dans un intervalle de ±10 % de la hauteur totale de l'image autour du nez, et en bas entre 0 et 20 % depuis la bordure

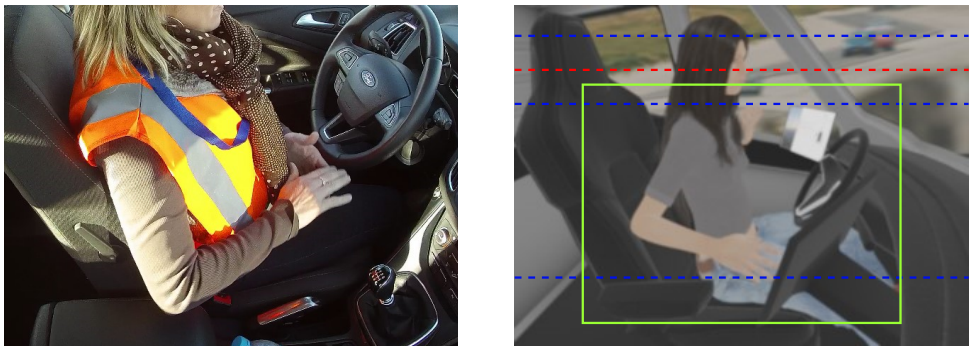


FIGURE 5.2 – Illustration d'une image de DriPE avec la caméra décentrée (à gauche) et de la méthode de rognage utilisée pour l'augmentation de données (à droite). Les lignes bleues représentent les intervalles de rognage, la ligne rouge le niveau du nez et le cadre vert un exemple de rognage possible.

inférieure. L'image est ensuite rognée à gauche et à droite de manière symétrique afin de retrouver les proportions de l'image de départ, puis est redimensionnée à la taille originale. Ce rognage est appliqué sur les images d'entraînement avec une probabilité de 50 %.

Nous effectuons à nouveau un *fine-tuning* de notre réseau sur PoseSynth avec ce rognage en suivant le "procédé de *fine-tuning* par défaut". Les résultats d'évaluation de ce modèle sont donnés dans le Tableau 5.4. Ces résultats montrent une légère augmentation du rappel, en particulier sur la tête et les genoux avec une augmentation de 4 et 8 %, respectivement. L'ajout du rognage semble donc aider à habituer le réseau à des cadrages imparfaits où certaines parties du corps comme la tête ne sont alors plus visibles. Cependant, cette augmentation n'a pas permis d'augmenter significativement la capacité du modèle à généraliser. En effet, les performances globales restent faibles, en particulier la précision, ce qui démontre que le modèle continue à sur-prédire les points clés sur les images.

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	30	50	52	55	46	42	26	42
AR	70	59	60	62	46	40	17	54

TABLEAU 5.4 – Performances sur le jeu DriPE complet après un *fine-tuning* de *Simple Baseline* sur PoseSynth avec rognage, évaluées avec mAPK (en %).

Outre la différence visuelle entre les images synthétiques et les images réelles, une raison pouvant expliquer la surconfiance observée est le fait que tous les points clés dans le jeu de données PoseSynth soient annotés. En effet, puisque les images sont générées à partir d'une scène 3D, la position dans l'espace de tous les points est connue. En utilisant les paramètres de la caméra virtuelle, tous les points clés peuvent être projetés sur le plan de l'image. L'annotation de PoseSynth ne prend donc pas en compte la visibilité des points, puisque les points occultés du point de vue de la caméra sont annotés, ce qui n'est pas le cas pour une image réelle comme dans DriPE. Seuls les points en dehors de l'image sont retirés de l'annotation finale de PoseSynth. Ainsi, entraîner un réseau sur ce jeu de données entraîne le modèle à toujours prédire la totalité des points clés dans le cadre de l'image. Ce comportement réduit alors la précision lors de l'évaluation sur DriPE puisque les points prédits mais non annotés sont comptés comme des faux positifs.

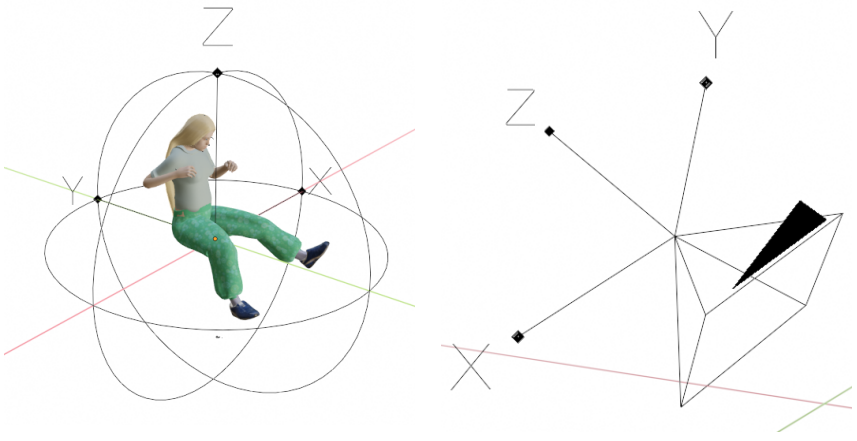
Pour enlever ce biais, il serait nécessaire d'annoter la visibilité des points clés dans les

images synthétiques. Cela requiert de définir les trois classes de visibilité d'une manière rigoureuse, pour pouvoir ensuite appliquer un algorithme d'annotation lors de la génération des images. La classe des points visibles est facilement définissable : le point est visible par la caméra, c'est-à-dire qu'aucun autre objet de la scène ne l'occulte. Pour déterminer cela, il est possible de tracer un rayon entre le centre de la caméra et le point, et de vérifier s'il traverse un autre objet. Cependant, un grand nombre de points clés à l'intérieur d'une voiture sont annotés comme "non visibles", c'est-à-dire qu'ils sont occultés mais qu'il reste suffisamment d'informations pour définir la position du point. Comme discuté dans le Chapitre 3, la notion "d'information suffisante" reste subjective, et il est difficile de définir rigoureusement ce concept. En effet, l'annotateur s'appuie sur différentes indications pour prendre sa décision : la structure du squelette, la visibilité des parties du corps entourant le point clé, les dimensions de ces parties déductibles par symétrie, etc. Toutes ces notions sont complexes à modéliser, et il serait nécessaire de les coupler à un algorithme décisionnel afin de déterminer la visibilité finale. Ainsi, il est difficile de discerner les points non visibles des points non annotables lors de la génération des images synthétiques. Nous pouvons cependant noter que cette problématique autour des points non visibles ne s'applique pas aux points de la tête, puisque seuls les points visibles sont annotés sur le visage, conformément aux consignes d'annotation du jeu de données COCO. Nous présenterons donc dans la section suivante les modifications apportées à la méthode de génération de nos images synthétique afin d'automatiser l'annotation de la visibilité des points de la tête.

5.2 Mise à jour du jeu de données PoseSynth

5.2.1 Modifications de la méthode de génération

Suite aux observations faites sur les résultats précédents, nous décidons de mettre à jour notre méthode de génération d'images synthétiques afin de créer de nouvelles données. Tout d'abord, nous augmentons la diversité des angles de vue. En effet, bien que des légers mouvements aléatoires de caméra aient été intégrés à la génération, les images générées restent entièrement cadrées autour de la personne. Cela permet d'obtenir des images où



(a) (b)

FIGURE 5.3 – Illustrations des axes de coordonnées de l'environnement de modélisation (a) et de la caméra virtuelle (b).

Algorithme 3 : Algorithme de placement de la caméra pour PoseSynth V2

Entrées : caméra C, modèle_humain M
Sorties : rotation $\text{rot}(C)$, localisation $\text{loc}(C)$, angle_de_vue(C)

$\alpha \leftarrow \text{aléa}(-200°, -140°)$
$\text{loc}_X(C) \leftarrow 1 \text{ m} \cdot \cos\alpha$
$\text{loc}_Y(C) \leftarrow 1 \text{ m} \cdot \sin\alpha$
$\text{loc}_Z(C) \leftarrow \text{aléa}(50 \text{ cm}, 80 \text{ cm})$
$\text{rot}_X(C) \leftarrow \arctan \| \frac{\text{loc}_X(C)}{\text{loc}_Z(C)} \|$
$\text{rot}_Y(C) \leftarrow \arctan \| \frac{\text{loc}_Y(C)}{\text{loc}_X(C)} \|$
$\text{rot}_X(C) \leftarrow \text{rot}_X(C) + \text{aléa}(-10°, 0°)$
$\text{rot}_Y(C) \leftarrow \text{rot}_Y(C) + \text{aléa}(-5°, 25°)$
$\text{rot}_Z(C) \leftarrow \text{rot}_Z(C) + \text{aléa}(-5°, 5°)$
angle_de_vue$(C) \leftarrow \text{aléa}(45°, 64°)$

la quasi-totalité du corps est visible (les chevilles peuvent rester cachées par l'habitacle), ce qui fournit le maximum d'informations sur la pose du sujet. Cependant, cela n'est pas représentatif des angles de vue que l'on peut rencontrer dans des images en conditions réelles, comme on peut le voir dans la Figure 5.2.

Nous décidons donc de générer des poses de caméra aléatoires plus diversifiées, tout en conservant un angle de vue de côté. Nous procédons selon l'Algorithme 3, en utilisant les repères de coordonnées pour le modèle humain et la caméra illustrés respectivement dans la Figure 5.3-a et b. Les différentes valeurs de rotation ("rot") et localisation ("loc") sont choisies afin de modéliser un grand nombre de placements et de paramètres de caméra, ainsi que différents mouvements indésirés pouvant apparaître en conditions réelles. En plus de la variation du placement de la caméra, nous annotons la visibilité des points du visage. Pour ce faire, un rayon est projeté depuis le centre de la caméra vers chaque point clé. Si ce rayon traverse un objet quelconque de la scène avant d'atteindre le point clé, ce dernier est considéré comme occulté et sa visibilité est mise à 0. Nous ne traiterons pas dans ces travaux du calcul de la visibilité des autres points clés du fait de la complexité du problème, comme expliqué dans la section précédente.

Nous générons avec cette méthode une mise à jour du jeu PoseSynth V2 contenant

FIGURE 5.4 – Illustration des images synthétiques de PoseSynth V2 générées avec l'Algorithme 3.

120 k nouvelles images. Mis à part les modifications décrites dans cette section, nous suivons la méthode décrite dans le Chapitre 4. Pour créer ces images, nous ré-utilisons l'ensemble des 100 modèles humains et 7 modèles de voitures, et produisons ainsi un total de 5 000 configurations (voir Section 4.3.1 pour plus de détails). Un échantillon des nouvelles images est illustré dans la Figure 5.4.

5.2.2 Entraînement avec PoseSynth V2

Afin de mesurer la capacité de génération des modèles apportée par un entraînement sur ces nouvelles données, nous effectuons un *fine-tuning* du modèle de *Simple Baseline* pré-entraîné sur COCO. Le modèle est entraîné sur 100 k images de PoseSynth V2, et les images restantes sont divisées en deux ensembles de validation et de test de 10 k images chacun. Nous suivons le "procédé de *fine-tuning* par défaut", et les résultats de l'évaluation sont présentés dans le Tableau 5.5.

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	58	67	77	84	49	64	52	66
AR	71	74	77	82	37	46	15	62

TABLEAU 5.5 – Performances sur le jeu DriPE complet de *Simple Baseline* après un *fine-tuning* sur les images de PoseSynth V2, évaluées avec mAPK (en %).

Nous pouvons tout d'abord constater que les performances ont grandement augmenté par rapport au *fine-tuning* sur le premier jeu de données synthétiques (Tableau 5.2). En effet, on peut voir une hausse de la précision et du rappel sur la majorité des points clés, avec un gain de plus de 20 % de précision sur toutes les parties du corps, exceptées les hanches et les genoux. Cette augmentation globale de la précision est confirmée par l'augmentation de la précision totale de 23 % en comparaison avec le *fine-tuning* sur la première version de PoseSynth. On peut également constater une augmentation du rappel sur le haut du corps d'environ 10 %, qui est cependant en partie compensée par une perte de rappel sur les hanches et les chevilles. L'ensemble de ces augmentations semble valider que la hausse de la diversité des angles de vue dans les images synthétiques permet d'aider le réseau à généraliser sur un plus grand nombre de poses, en rapprochant le domaine d'entraînement du domaine cible. On peut également observer que la précision sur les points clés de la tête a plus que doublé ; ce qui indique une forte diminution du nombre de faux positifs. L'annotation de la visibilité des points clés sur cette partie du corps a donc permis d'atteindre une bonne performance pour l'estimation de ces points.

Enfin, on peut noter que les performances mesurées sur les hanches ont baissé par rapport à celles mesurées après l'entraînement sur la première version de PoseSynth. Ce phénomène s'explique par le fait que les hanches dans cette nouvelle version ne sont plus constamment placées au même emplacement dans l'image, grâce aux changements de cadre intégrés dans la nouvelle version du jeu de données. Ces déplacements apportent du bruit sur la localisation de ces points clés lors de l'entraînement, réduisant les performances. Ces déplacements permettent cependant d'éviter que le réseau prédise la position des hanches toujours au même endroit sans s'appuyer sur de réelles observations, même si ces prédictions "à l'aveugle" pouvaient parfois être correctes.

Ainsi, cette mise à jour de la méthode de génération des images synthétiques a permis de diminuer en grande partie les pertes de performances observées après le *fine-tuning* sur

ces dernières. Cette nouvelle version conserve également le gain de performance sur l'estimation de la pose des chevilles, et de rappel sur l'estimation des genoux dans une moindre mesure, observé sur PoseSynth. La perte de performances générales en comparaison avec celle de *Simple Baseline* entraîné uniquement sur COCO reste cependant importante, puisque nous relevons une différence de précision totale de 16 %.

5.2.3 Combinaisons des données

Les résultats précédents nous ont montré que la génération d'images synthétiques plus variées et avec des annotations plus réalistes permettait d'augmenter les performances des modèles entraînés. Cependant, bien que les performances sur le bas du corps (genoux et chevilles) aient partiellement augmenté en comparaison avec les résultats du réseau entraîné uniquement sur COCO (Tableau 4.1), les performances sur le reste du corps ont été nettement détériorées. Cela occasionne une diminution indésirée de la performance totale. Ainsi, bien que les images de COCO soient peu représentatives de notre contexte, il semblerait qu'elles puissent grandement améliorer l'entraînement et donc les performances finales du réseau. Cela pourrait s'expliquer par deux points. Premièrement, les images de COCO sont des images réelles, et donc plus riches en détails dans les textures, les illuminations, etc., mais aussi plus bruitées du fait des différents mouvements de la caméra. On peut ainsi supposer qu'intégrer des images réelles aux données d'entraînement permettrait de diminuer l'écart avec le domaine cible, en habituant le modèle à ces types de bruits et en le poussant à enrichir son analyse de l'image. Deuxièmement, bien que la génération d'images synthétiques permette en théorie d'obtenir une diversité infinie, les images générées illustrent toujours un habitacle de véhicule. À l'inverse, les images de COCO sont beaucoup plus variées puisqu'elles illustrent chacune des personnes différentes dans des contextes et environnement diverses. La diversité des jeux de données génériques aide à la généralisation du réseau, comme le montre la perpétuelle croissance de la taille des jeux de données utilisés pour l'entraînement (Johnson and Everingham, 2010; Andriluka et al., 2014; Lin et al., 2014). Cette diversité n'est cependant pas toujours suffisante pour représenter des contextes spécifiques, comme nous avons pu le montrer au cours des évaluations des méthodes de l'état de l'art dans le Chapitre 3.

Nous décidons donc d'étudier l'impact de l'ajout d'images de COCO aux images synthétiques lors de l'entraînement sur les performances du réseau. Nous créons ainsi trois jeux d'entraînement constitués de combinaisons d'images de COCO et de PoseSynth V2, en faisant varier les ratios utilisés. Les trois jeux sont composés de 145 k images au total et partagent le même nombre de configurations (modèles de personnes, véhicules, environnement) représentées dans les images synthétiques sélectionnées. Nous veillons également à ce que chaque *batch* suive la même proportion des deux sources de données. De la même manière, nous créons trois jeux de validation de 6,5 k images chacun. Nous effectuons un *fine-tuning* d'un modèle de *Simple Baseline* pré-entraîné sur COCO avec chacun de ces trois jeux d'entraînement, en suivant le "procédé de *fine-tuning* par défaut". Les résultats d'évaluation sur DriPE sont présentés dans le Tableau 5.6.

Nous observons tout d'abord que l'augmentation du ratio d'images synthétiques permet d'augmenter les performances d'estimation sur le bas du corps, avec un gain de précision de près de 20 % sur les chevilles entre les modèles entraînés sur 25 % et su 75 % d'images de COCO. Ces résultats confirment les observations faites suite au *fine-tuning* sur PoseSynth V2. Ce gain de performances est compensé par une diminution des performances sur le haut du corps, en particulier sur le visage, entraînant une diminution des

% de COCO		Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
75		71	73	76	82	73	70	24	**73**
50	AP	66	68	79	87	62	68	31	71
25		65	65	80	87	45	64	43	68
75		82	77	75	79	59	44	09	**66**
50	AR	76	72	76	81	44	42	10	62
25		77	76	79	83	34	41	15	62

TABLEAU 5.6 – Performances sur le jeu DriPE complet après un *fine-tuning* de *Simple Baseline* sur PoseSynth V2 + COCO, évaluées avec mAPK (en %).

performances globales de jusqu'à 4 % de précision et 7% de rappel. Ainsi, la combinaison d'images générales et spécifiques, apportées respectivement par COCO et PoseSynth V2, permet de partiellement compenser les défauts de chaque jeu de données. Cependant, on peut constater qu'il n'existe pas de ratio idéal, puisque l'ajout d'images synthétiques diminue les performances globales mais améliore l'estimation de poses sur le bas du corps.

Ajout d'images de Drive&Act

Afin d'aider à réduire l'écart entre le domaine d'entraînement et le domaine cible, il serait pertinent d'intégrer dans les images d'entraînement des données réelles illustrant notre contexte. Comme mentionné dans le chapitre précédent, nous ne disposons pas de données annotées répondant à nos besoins. Nous décidons donc d'ajouter des images du jeu de données Drive&Act (Martin et al., 2019) à l'ensemble d'entraînement, malgré le fait que les pseudo-annotations aient été obtenues grâce à un réseau d'estimation de poses et non manuellement. Ainsi, nous constituons un nouveau jeu de données incluant des images de Drive&Act composé du même nombre d'images que les jeux de la section précédente, soit 145 k images d'entraînement et 6,5 k images de validation. Nous conservons les proportions ayant montré les meilleures performances totales dans l'expérimentation précédente, soit 75 % d'images génériques et 25 % d'images spécifiques à notre contexte. Afin de ne pas donner trop de poids aux pseudo-annotations, les images de Drive&Act ne constituent que 5 % du jeu de données total. Le reste du jeu de données est donc constitué de 75 % d'images du jeu COCO et 20 % d'images de PoseSynth V2. Nous nommerons ce jeu de données pour la suite de cette thèse "jeu triple-données". Afin d'aligner les poses de Drive&Act sur celles des images synthétiques, et plus généralement sur les poses enregistrées en dehors d'un simulateur, nous ajoutons une rotation fixe de 40° sur les images de Drive&Act lors de l'entraînement (en plus des rotations aléatoires appliquées par défaut avec *Simple Baseline*). Nous réalisons donc un entraînement avec ce nouveau jeu de données en suivant le "procédé de *fine-tuning* par défaut", et les résultats d'évaluation sur DriPE sont donnés dans le Tableau 5.7.

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	66	70	81	86	77	63	25	71
AR	86	75	83	86	69	53	13	72

TABLEAU 5.7 – Performances sur le jeu DriPE complet de *Simple Baseline* après un *fine-tuning* sur un jeu de données incluant des images de Drive&Act, évaluées avec mAPK (en %).

Nous pouvons ainsi constater que l'ajout des images de Drive&Act a pour effet d'augmenter le rappel sur le bas du corps, avec une augmentation d'environ 10 % sur les hanches et genoux, et de 4 % sur les chevilles. En outre, on observe une augmentation de la précision sur les hanches, bien qu'elle soit compensée par une diminution des performances sur les genoux. Enfin, on peut relever une augmentation du rappel sur le visage, compensé par une diminution de la précision. Au global, l'ajout d'images de Drive&Act semble avoir permis d'équilibrer le rappel vis-à-vis de la précision, puisque l'on passe d'un score F1 de 69% à 71 %. Ce constat est particulièrement valable sur le bas du corps.

Sur-échantillonnage

Comme nous l'avons observé avec les précédentes évaluations, les chevilles semblent toujours poser des difficultés aux modèles. Bien que l'ajout des images synthétiques ait permis une augmentation des performances, elles restent cependant assez basses sur cette partie du corps (moins 30 % de précision et 20 % de rappel). Augmenter fortement le taux d'images synthétiques dans les images d'entraînement permet d'améliorer l'estimation de ces points clés, mais cela nuit fortement aux performances globales du réseau. De plus, l'ajout d'images provenant de Drive&Act permet d'améliorer l'estimation de poses à l'intérieur du véhicule, mais n'augmente que très peu les performances sur les chevilles, comme nous l'avons constaté dans la section précédente.

Dans le but d'améliorer l'estimation sur le bas du corps, nous décidons d'utiliser une méthode de sur-échantillonnage afin d'augmenter la proportion d'images avec les points clés du bas du corps annotés. Ainsi, nous dupliquons dans le jeu d'entraînement toutes les images avec le bas du corps annoté, jusqu'à atteindre un taux de représentation d'au moins 20 % pour chaque point clé. Ce taux est calculé individuellement pour chaque jeu de données (COCO, PoseSynth V2, et Drive&Act). Nous fixons le taux à 20 % afin de ne pas trop dupliquer les mêmes images, puisque certains points tels que les chevilles droites sont peu représentés. Comme on peut le constater dans la Figure 5.5, le sur-échantillonnage permet d'augmenter significativement la représentation du bas du corps dans Drive&Act et dans les images de PoseSynth V2. Les proportions de COCO restent inchangées puisqu'elles sont déjà supérieures à 20 %. Nous effectuons un nouveau *fine-tuning* avec ces

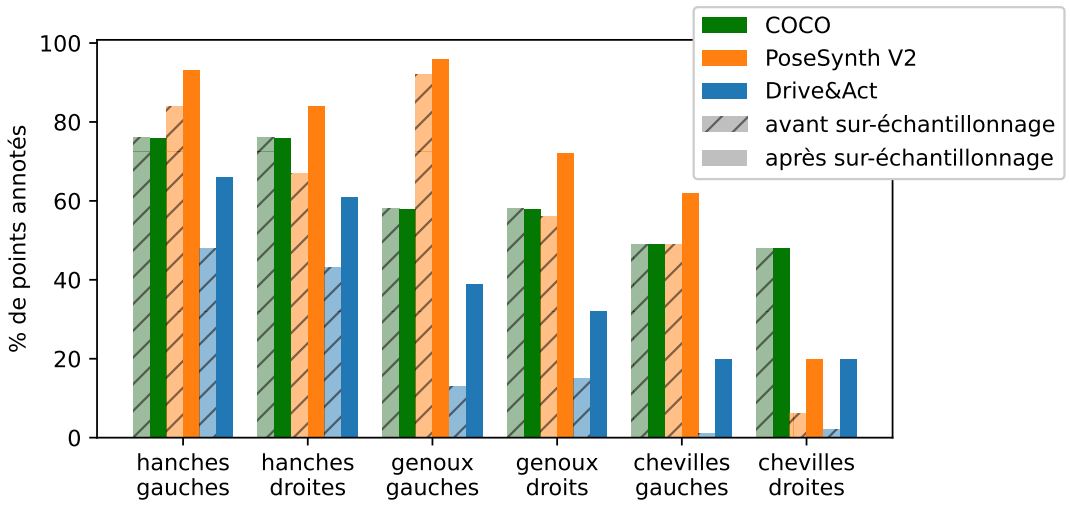


FIGURE 5.5 – Histogramme de la représentation des points clés du bas du corps dans chaque jeu de données.

jeux sur-échantillonnés, tout en conservant les proportions du "jeu triple-données" utilisées dans la section précédente (75 % COCO, 20 % PoseSynth V2, 5 % Drive&Act). Nous suivons le "procédé de *fine-tuning* par défaut", et les résultats d'évaluation sur DriPE sont présentés dans le Tableau 5.8.

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	63	70	79	85	84	72	32	73
AR	85	73	79	83	66	48	17	69

TABLEAU 5.8 – Performances sur le jeu DriPE complet de *Simple Baseline* après un *fine-tuning* avec sur-échantillonnage, évaluées avec mAPK (en %).

Comme nous pouvons le constater, les performances globales restent proches de celles observées dans le Tableau 5.7. Cependant, on peut constater un gain de performance sur les points clés du bas du corps, en particulier les genoux et les chevilles avec un gain de 7 à 9 % de précision après l'application du sur-échantillonnage. Ce gain est nuancé par une légère baisse de rappel sur le haut du corps.

Ainsi, entraîner un réseau pour l'estimation de poses en utilisant plusieurs jeux de données semble être une piste intéressante pour aider à la généralisation du réseau. En effet, les résultats ont montré que mélanger des images génériques et des images spécifiques permet de conserver des performances correctes pour l'estimation de poses sur l'ensemble du corps, tout en augmentant légèrement les performances sur le bas du corps. Le sur-échantillonnage des images présentant des genoux et des chevilles a également montré de bons résultats sur le bas du corps. Cependant, malgré le gain de performance sur ces points clés, la performance globale reste inférieure à un entraînement uniquement sur COCO, avec une perte de 9 % de précision pour un gain de seulement 2 % de rappel total ; ce qui indique que nous ne parvenons pas à tirer efficacement avantage de chaque jeu de données.

5.3 Architecture multi-branches

Nous avons montré dans la section précédente que réaliser le *fine-tuning* d'un réseau de neurones avec une combinaison d'images génériques et spécifiques, ainsi que synthétiques et réelles, permettait d'améliorer les performances d'estimation de poses sur le bas du corps dans notre contexte. Ce gain se fait cependant au détriment de l'estimation du haut du corps. Il semble donc difficile de trouver une méthode d'entraînement permettant de conserver à la fois les bonnes performances globales et en particulier sur le haut du corps apportées par l'entraînement sur COCO, tout en profitant de la spécificité des images synthétiques permettant d'améliorer les performances sur les genoux et les chevilles. Nous cherchons donc une nouvelle approche permettant de profiter des apports de chaque jeu de données. Nous nous intéressons dans cette section aux architectures pour la généralisation de l'estimation de poses, et proposons deux nouvelles architectures dérivées de *Simple Baseline*. Ces architectures s'appuient sur plusieurs instances de *Simple Baseline* entraînées individuellement sur les jeux de données COCO et PoseSynth V2, et fusionnent ensuite les vecteurs produits pour obtenir une prédiction finale. Nous étudions dans cette section deux approches de fusion : la fusion de caractéristiques (Section 5.3.1) et la fusion des cartes de chaleur (Section 5.3.2).

5.3.1 Fusion de caractéristiques

Tout d'abord, nous nous intéressons à la fusion des vecteurs de caractéristiques. L'architecture *Simple Baseline* est constituée d'un encodeur basé sur ResNet50 suivi d'un décodeur produisant les cartes de chaleur finales. Nous nous inspirons donc de travaux sur la généralisation de domaine pour différentes tâches (Mancini et al., 2018; Zhou et al., 2021; Qin et al., 2022) et dupliquons les encodeurs afin de produire un vecteur de caractéristiques propre à chaque jeu de données. À notre connaissance, il n'existe pas de travaux proposant une approche similaire pour l'estimation de poses humaines.

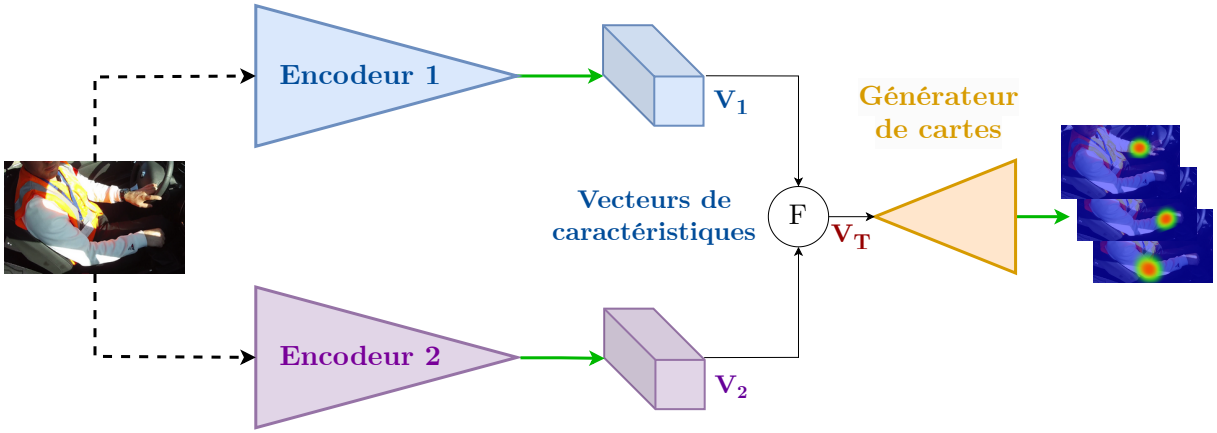


FIGURE 5.6 – Schéma de l'architecture double par fusion de caractéristiques.

Ainsi, notre architecture reprend la structure de base de *Simple Baseline*, mais nous dédoublons l'extracteur de caractéristiques (Figure 5.6). Chaque extracteur reçoit la même image en entrée et génère un vecteur de caractéristiques de dimension $8 \times 6 \times 512$. Il est alors nécessaire de fusionner les deux vecteurs produits V_1 et V_2 en un vecteur V_T de même dimension pour donner en entrée du décodeur. Pour ce faire, nous envisageons deux méthodes :

- **fusion par addition** : les deux vecteurs de caractéristiques sont additionnés, ce qui permet de faire ressortir les valeurs élevées. On a donc :

$$V_T = V_1 + V_2 \tag{5.1}$$

- **fusion par concaténation** : les deux vecteurs sont concaténés le long de la dimension des canaux. Cela permet de conserveur l'ensemble de l'information provenant des deux sources. Le vecteur obtenu est cependant de dimension $8 \times 6 \times 1024$, il est donc nécessaire de ramener le nombre de canaux à 512, soit le nombre de canaux d'entrée du décodeur de *Simple Baseline*. Pour ce faire, nous utilisons une couche de convolution C avec un noyau de 1×1. Ainsi, on a :

$$V_T = C(V_1 \| V_2) \tag{5.2}$$

Le décodeur final prend donc en entrée le vecteur de caractéristiques fusionné V_T et produit les prédictions sous la forme de cartes de chaleur, de manière similaire à *Simple Baseline*. Chaque extracteur est pré-entraîné individuellement pour l'estimation de poses humaines. Ainsi, les poids de l'encodeur 1 est issu d'un modèle *Simple Baseline* entraîné sur COCO, tandis que le ceux de l'encodeur 2 sont issus d'un modèle après un *fine-tuning* sur PoseSynth V2. Nous ré-utilisons les extracteurs de caractéristiques des modèles

entraînés précédemment, dont les performances sont montrées respectivement dans les Tableaux 4.1 et 5.5. Par la suite, les poids des encodeurs sont gelés, et seul le décodeur et la couche de convolution supplémentaire, dans le cas de la fusion par concaténation, sont entraînés. Cet entraînement est réalisé sur le "jeu triple-données" durant 10 *epochs* avec un taux d'apprentissage de $1E^{-5}$, et nous conservons les poids montrant la plus faible erreur sur l'ensemble de validation. Les autres hyperparamètres et les augmentations de données utilisées sont ceux de base pour l'entraînement de *Simple Baseline*, présentés dans la Section 3.2.3.2. Les résultats d'évaluation sont montrés dans le Tableau 5.9.

	Fusion	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	addition	65	71	79	81	73	72	38	69
	concat.	67	73	78	84	65	73	47	**71**
AR	addition	78	71	71	75	61	48	08	64
	concat.	79	80	80	85	56	57	14	**70**

TABLEAU 5.9 – Performances sur le jeu DriPE complet du réseau de fusion des caractéristiques par addition et par concaténation (concat.), évaluées avec mAPK (en %).

Tout d'abord, nous constatons que l'ordre de grandeur des performances totales mesurées est proche de celui mesuré sur *Simple Baseline* seul entraîné sur le même jeu de données (Tableau 5.7), et ce quelle que soit la méthode de fusion. Nous pouvons également observer un important gain de précision sur la détection des genoux et des chevilles, avec un gain allant jusqu'à 8 % sur les genoux et 22 % sur les chevilles. On peut en déduire que l'utilisation des deux extracteurs de caractéristiques permet de conserver d'une part les capacités d'estimation de poses sur le haut du corps apportées par l'entraînement sur COCO, et d'autre part le gain de performance sur le bas du corps apporté par l'entraînement sur PoseSynth V2. Enfin, nous pouvons constater que la fusion par concaténation permet d'obtenir au global de meilleurs résultats, et en particulier sur les chevilles avec un gain de 9 % de précision et 6 % de rappel.

Nous entraînons ensuite la même architecture en intégrant le sur-échantillonnage pour les parties basses du corps présenté dans la Section 5.2.3. Les résultats sont présentés dans le Tableau 5.10. On observe ainsi le même comportement que sur *Simple Baseline*, avec un gain de précision d'environ 5 % sur les genoux et 7 % sur les chevilles en comparaison avec les résultats du Tableau 5.9. Ces résultats confirment également la capacité de cette architecture à conserver les performances des extracteurs apprises sur les différents jeux de données. Finalement, la fusion par concaténation avec le sur-échantillonnage permet de gagner en moyenne 14 % de rappel sur les genoux et chevilles, et 24 % de précision sur les chevilles par rapport au modèle de *Simple Baseline* entraîné sur COCO seul. Ce gain de performances ne permet cependant pas de compenser la perte de précision sur le reste du corps, avec une perte de précision totale de 8 %.

	Fusion	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	addition	66	70	78	84	75	77	45	73
	concat.	69	72	78	83	72	78	55	**75**
AR	addition	76	73	79	76	69	55	12	65
	concat.	77	80	79	84	56	53	18	**69**

TABLEAU 5.10 – Performances sur le jeu DriPE complet du réseau de fusion de caractéristiques avec sur-échantillonnage, évaluées avec mAPK (en %).

5.3.2 Fusion des cartes de chaleur

Nous nous intéressons ensuite à la fusion des cartes de chaleurs générées par les réseaux d'estimation de poses humaines. L'objectif est similaire à celui de l'architecture par fusion de caractéristiques de la Section 5.3.1 : utiliser deux réseaux indépendants pré-entraînés sur les deux jeux de données COCO et PoseSynth V2 pour pouvoir tirer au mieux parti des forces de ces deux sources.

Cette nouvelle architecture est donc composée de deux instances indépendantes de *Simple Baseline*, suivies d'un module de fusion des cartes de chaleur (Figure 5.7). A la différence de l'architecture précédente pour la fusion de caractéristiques (Figure 5.6), nous conservons l'intégralité des deux modèles *Simple Baseline*, et deux ensembles de cartes de chaleur H_1 et H_2 sont donc générés de dimension $64 \times 48 \times 17$, soit la dimension standard des cartes de chaleur produites par *Simple Baseline*. Les deux prédictions sont alors concaténées avant d'être données en entrée d'un auto-encodeur pour effectuer la fusion. Pour ce faire, nous utilisons l'architecture convolutive U-Net (Ronneberger et al., 2015) présentée dans la Section 2.1.4. Dû à la petite taille des cartes de chaleur produites par *Simple Baseline* (64×48), nous utilisons une architecture U-Net à trois niveaux. Enfin, deux couches de convolution avec un noyau de taille 1 sont utilisées pour générer les cartes de chaleur finales.

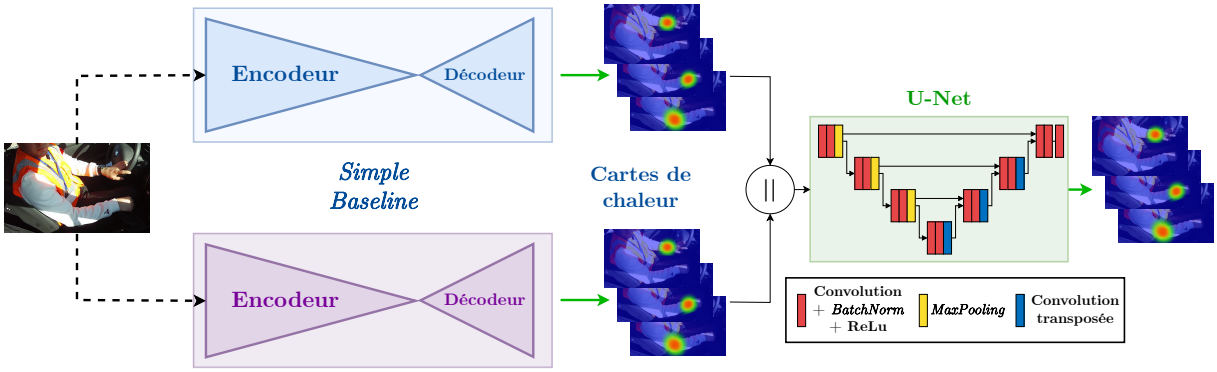


FIGURE 5.7 – Schéma de l'architecture double par fusion des cartes de chaleur. Notations : concaténation par canal ||.

Un des défauts de cette architecture est que l'auto-encodeur n'a pas accès directement aux informations de l'image d'entrée. En effet, les cartes de chaleur concaténées en entrée fournissent uniquement des informations sur la position des points clés. Ainsi, aucune information sur le contexte (environnement, textures, etc.) n'est utilisée pour générer les cartes de chaleur finales. Nous décidons donc de tester l'apport de l'ajout d'informations supplémentaires dans la fusion des cartes de chaleur. Pour ce faire, nous intégrons une nouvelle branche en parallèle des deux réseaux d'estimation de poses pour encoder l'image d'entrée et l'intégrer à la fusion (Figure 5.8).

Pour encoder l'image, nous intégrons un auto-encodeur pré-entraîné sur la tâche auto-supervisée de reconstruction d'images (Section 2.1.4). Nous utilisons à nouveau l'architecture U-Net à trois niveaux. Nous retirons cependant les connexions intermédiaires de l'architecture puisque nous utiliserons par la suite uniquement l'encodeur, et souhaitons donc pré-entraîner le réseau sans ces connexions. Ainsi, l'encodeur de caractéristiques est constitué de trois niveaux d'encodeurs U-Net. Il génère à partir de l'image d'entrée un vecteur de caractéristiques V_I de dimension $64 \times 48 \times 64$. Ce vecteur est concaténé avec les cartes de chaleur H_1 et H_2 générées par les deux instances de *Simple Baseline*, avant

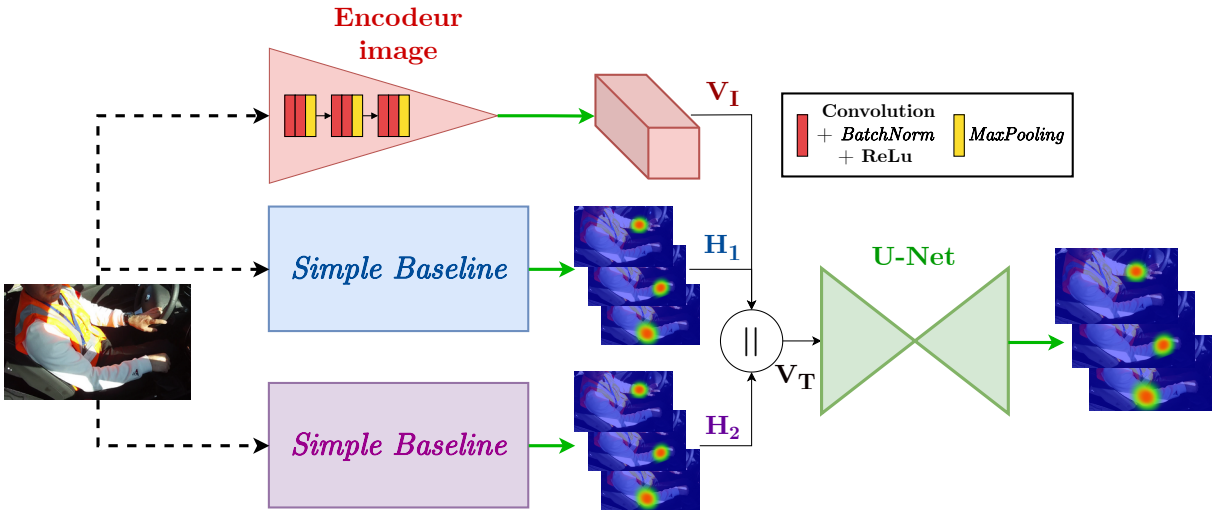


FIGURE 5.8 – Schéma de l'architecture double par fusion des cartes de chaleur avec la branche image. Notations : concaténation par canal ||.

d'être donné en entrée de l'auto-encodeur final. On a donc :

$$V_T = V_I \| H_1 \| H_2 \tag{5.3}$$

où $\|$ représente la concaténation par canal.

Les réseaux de fusion de cartes de chaleurs sont entraînés de manière similaire à celui de la fusion de caractéristiques. Les deux modèles *Simple Baseline* sont pré-entraînés individuellement sur COCO et PoseSynth V2 respectivement, puis leurs poids sont gelés. L'auto-encodeur des cartes et les couches de convolution finales sont initialisés aléatoirement, et entraînés sur le "jeu triple-données" durant 10 *epochs* avec un taux d'apprentissage de $1E^{-5}$. L'auto-encodeur de l'image est entraîné sur le "jeu triple-données" avec une fonction de coût L1 calculée entre l'image d'entrée et l'image reconstruite par le modèle. L'entraînement est réalisé avec un taux d'apprentissage de $1E^{-2}$ durant 80 *epochs*. Une fois l'auto-encodeur pré-entraîné, les poids de l'encodeur sont gelés et utilisés dans la branche image du réseau de fusion. Le réseau de fusion est alors entraîné en utilisant les mêmes hyperparamètres que pour la fusion de caractéristiques, c'est à dire sur le "jeu triple-données" durant 10 *epochs* avec un taux d'apprentissage de $1E^{-5}$.

Les résultats sont donnés dans les Tableaux 5.11 et 5.12. Comme on peut l'observer, le réseau par fusion des cartes de chaleur présente un gain de 6 à 8% de précision en comparaison avec la fusion des vecteurs de caractéristiques (Tableau 5.9). Ce gain est observé sur presque l'ensemble des points clés, en particulier sur le haut du corps et les hanches, bien qu'il soit légèrement compensé par une perte de précision sur le bas du

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	74	76	85	90	71	72	45	77
AR	80	79	80	84	53	42	10	67

TABLEAU 5.11 – Performances sur le jeu DriPE complet du réseau de fusion des cartes de chaleur, évaluées avec mAPK (en %).

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	73	73	82	89	69	69	42	75
AR	79	76	81	85	50	41	08	66

TABLEAU 5.12 – Performances sur le jeu DriPE complet du réseau de fusion des cartes de chaleur avec la branche image, évaluées avec mAPK (en %).

corps. Cependant, ces résultats montrent que la fusion des cartes de chaleur permet de préserver de bonnes performances sur le haut comme sur le bas corps.

Nous observons également que l'ajout de la branche image entraîne une diminution des performances du modèle sur la plupart des parties du corps, ce qui semble indiquer que l'ajout de l'image nuit globalement aux performances. On mesure une perte de performances sur la majorité des parties du corps entre 1 et 3%, résultant en une diminution de 2 % de précision et 1 % de rappel total. On peut supposer que ces résultats sont expliqués par le fait que le vecteur de caractéristiques provenant de l'image se combinent mal avec les cartes de chaleur, et apporte du bruit et non des informations supplémentaires lors de la fusion.

Nous intégrons ensuite à l'entraînement le sur-échantillonnage sur les données. Les résultats pour les réseaux sans et avec la branche image sont montrés dans les Tableaux 5.13 et 5.14, respectivement. Ces résultats confirment que le sur-échantillonnage permet d'augmenter efficacement les performances de l'estimation de poses sur les points clés du bas du corps, avec une précision et un rappel sur les chevilles de 56 et 24 %, respectivement, sans la branche image. Nous pouvons également observer que les performances du réseau avec la branche image restent inférieures même après l'ajout des données sur-échantillonnées. En effet, on mesure une diminution de 1 % de précision et 3 % de rappel, malgré un gain de 1 % de précision et de rappel sur les points de la tête.

Finalement, en comparaison avec le modèle *Simple Baseline* entraîné sur COCO (Tableau 4.1), on mesure pour le réseau sans branche image un gain de 25% de précision sur les chevilles, ainsi qu'une augmentation de 20% de rappel sur les genoux et de chevilles. Cela se retranscrit par une hausse du rappel total de 5 %. La précision totale est par ailleurs réduite en raison d'une diminution de 10% de précision sur le visage. Cependant, le score F1 total a bien augmenté puisque l'on mesure un score de 75 % pour le réseau de fusion des cartes de chaleur, contre 74 % pour *Simple Baseline* seul.

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	72	76	84	90	80	79	56	79
AR	80	80	82	85	61	60	24	72

TABLEAU 5.13 – Performances sur le jeu DriPE complet du réseau de fusion des cartes de chaleur avec sur-échantillonnage, évaluées avec mAPK (en %).

	Tête	Épa.	Coude	Poign.	Hanche	Gen.	Chev.	Total
AP	73	73	83	89	77	78	54	78
AR	81	77	81	84	51	56	17	69

TABLEAU 5.14 – Performances sur le jeu DriPE complet du réseau de fusion des cartes de chaleur avec la branche image et le sur-échantillonnage, évaluées avec mAPK (en %).

5.4 Conclusion

Dans ce chapitre, nous avons cherché à améliorer les performances des réseaux d'estimation de poses humaines, et en particulier leur capacité de généralisation sur un domaine inconnu. Nous sommes partis du réseau de référence *Simple Baseline* entraîné sur le jeu de données générique COCO avec pour objectif d'améliorer les performances sur l'estimation des jambes, qui est une partie du corps difficile dans notre contexte dû à l'angle de vu et aux fortes occultations, tout en conservant les bonnes performances du réseau sur le reste du corps. Nous avons tout d'abord montré qu'un simple *fine-tuning* sur des images synthétiques permettait bien d'améliorer les performances sur le bas du corps, mais entraînait une perte de performances trop importante sur les autres points clés du squelette. Il en va de même en utilisant des méthodes d'augmentation de données. Améliorer la diversité des images de PoseSynth et leurs annotations a cependant permis de légèrement réduire la perte de performances, en particulier pour les points clés du visage.

Nous avons alors entraîné le réseau en mélangeant les images synthétiques et les images réelles, afin de conserver la diversité apprise sur COCO tout en apprenant les spécificités de notre contexte sur les images de PoseSynth V2. Selon les proportions choisies, nous parvenons à réduire les pertes de performances par rapport à un simple entraînement sur COCO. La précision et le rappel totaux restent cependant inférieurs malgré le gain de performances sur les genoux et les chevilles. Ce gain a été également accentué en utilisant une méthode de sur-échantillonnage afin d'augmenter la proportion d'images illustrant des chevilles annotées dans l'ensemble d'entraînement.

Dans le but de mieux préserver les forces et les spécificités issues de l'entraînement sur chaque jeu de données, nous avons proposé deux architectures combinant une paire d'instances de *Simple Baseline* entraînées individuellement sur chaque jeu. Nous avons tout d'abord réalisé une fusion des vecteurs de caractéristiques, en utilisant deux encodeurs pré-entraînés pour l'estimation de poses sur leur jeu de données respectifs, suivi d'un décodeur pour générer les prédictions finales. La deuxième architecture utilise deux instances complètes de *Simple Baseline* pré-entraînées, et fusionne les cartes de chaleur à l'aide d'un auto-encodeur. Ces architectures ont permis d'améliorer les performances globales de détection de la pose dans l'intérieure d'une voiture, en particulier le réseau de fusion des cartes de chaleur. De plus, bien que ces réseaux possèdent environ deux fois plus de poids que le réseau d'estimation de poses équivalent, ils ont l'avantage d'être rapides à entraîner si les pré-entraînements sont effectuées en amont, puisque seuls les poids de la tête de fusion sont mis à jour.

Il pourrait être intéressant dans de futurs travaux d'étudier l'impact sur les performances et la généralisation de l'ajout de branches supplémentaires entraînées sur des jeux de données différents. En outre, on peut se demander si utiliser des architectures plus complexes pour les différentes branches d'estimation de poses et pour l'auto-encodeur des cartes de chaleur permettrait de gagner en performances.

Conclusion générale et perspectives

Dans cette thèse, nous avons abordé l'estimation de poses humaines à l'intérieur d'une voiture par apprentissage profond. Nous avons d'abord étudié l'état de l'art afin d'évaluer les performances des méthodes existantes ainsi que la pertinence des jeux de données et des métriques d'évaluation existantes. Nous avons proposé différentes contributions pour répondre aux spécificités inhérentes à notre contexte, telles qu'un jeu de données annotées et une métrique d'évaluation. Nous nous sommes ensuite intéressés à la généralisation de domaine appliquée à l'estimation de poses humaines, en cherchant à utiliser des données spécifiques afin de pallier les difficultés rencontrées dans un habitacle de voiture par les réseaux génériques. Nous nous sommes donc intéressés à différentes méthodes d'augmentation de données avant de proposer une méthode de génération d'images synthétiques de passagers. Ces nouvelles images ont pu être utilisées pour entraîner différents modèles dans le but d'augmenter leur capacité de généralisation. Parmi ces réseaux, nous avons proposé deux nouvelles architectures de fusion cherchant à tirer profit des spécificités de différents domaines.

Mise en place d'outils spécifiques à notre contexte

Nous avons utilisé dans ces travaux des données issues de différentes sources. Nous nous sommes d'abord appuyés sur les jeux de données génériques de l'état de l'art, et principalement le jeu de données COCO. Ce de jeu de données est le plus utilisé pour l'entraînement de réseaux de neurones profonds pour l'estimation de poses puisqu'il contient un très grand nombre d'images illustrant des situations et des contextes variés, extraites de différentes sources, et annotées. Cela permet d'entraîner des réseaux performants et capables de généraliser correctement dans un grand nombre de tâches. Afin d'évaluer les performances des méthodes de l'état de l'art dans notre contexte, nous avons créé DriPE, notre propre jeu de données pour l'estimation de poses humaines à l'intérieur d'une voiture. En effet, aucun jeu de l'état de l'art n'était publiquement disponible, à notre connaissance, avec des images enregistrées dans une voiture en conditions réelles et annotées manuellement pour l'estimation de poses humaines. Nous avons donc extrait des images d'enregistrements vidéo et les avons annotées. Pour choisir les images à extraire, nous avons décidé de nous appuyer sur deux métriques, la similarité structurelle et la différence de luminosité. Ce choix avait pour but de sélectionner des images plus variées et mettant les réseaux en difficulté, afin d'obtenir l'entraînement et l'évaluation les plus représentatifs possibles. Nous n'avons cependant pas mesuré la pertinence d'un tel choix en comparaison à un jeu de données constitué d'images tirées aléatoirement ou de manière séquentielle, puisque cela aurait demandé d'annoter de nouvelles images. Il serait également intéressant de mesurer l'impact du nombre d'images total sur l'entraînement des réseaux. En effet, on cherche généralement à maximiser le nombre d'images utilisées durant l'entraînement

afin d'augmenter la diversité des images vues par le réseau. Cependant, nous avions à disposition des clips vidéo présentant un nombre limité de sujets et enregistrés dans des conditions relativement proches. Nous avons alors décidé de fixer un nombre d'images arbitraire en s'appuyant sur l'état de l'art et sur nos capacités d'annotation. Il pourrait donc être intéressant d'évaluer à partir de quelle taille de jeu de données les modèles entraînés "saturent" et l'ajout de nouvelles images n'est plus pertinent. Enfin, les images ont été annotées manuellement afin d'obtenir une vérité terrain de la pose. Cette annotation a été réalisée en collaboration avec les membres du laboratoire. Réaliser une telle tâche a posé des questions sur l'objectivité des annotations et sur la différence de perception des scènes par les différents annotateurs. En particulier, l'estimation de la visibilité des points clés et de la position de ceux occultés est laissée à l'appréciation de l'annotateur. Cela a pour conséquence une inconsistance des annotations au sein d'un même jeu de données, en particulier sur les parties difficiles comme les chevilles. Une première solution est l'annotation d'une même image par plusieurs annotateurs, permettant également d'éliminer des erreurs d'inattention. Cette solution démultiplie grandement le temps déjà important requis pour annoter un jeu de données. Plus généralement, il serait intéressant de définir rigoureusement les classes de visibilité des points clés et de leur annotation afin de limiter la variance entre les jeux de données pour l'estimation de poses humaines.

Dans le but de proposer une évaluation plus détaillée de la pose mono-personne, nous avons également proposé mAPK, une nouvelle métrique permettant d'évaluer plus précisément la qualité des estimations pour chaque point du corps. Cette métrique nous a permis de mettre en lumière un biais d'entraînement mis en place pour maximiser la métrique cocoAP ayant pour conséquence une faible précision de l'estimation de la position individuelle des points clés. Nous avons cependant montré qu'en appliquant la fonction de coût L2 sur la totalité des points clés, et non uniquement sur ceux annotés, ce biais pouvait être allégé.

Génération d'images synthétiques

Nous avons également utilisé dans ces travaux des données synthétiques afin de combler le manque de données réelles dans notre contexte. Pour ce faire, nous avons utilisé Blender, un logiciel de modélisation 3D. Grâce à ce logiciel, nous avons mis en place une méthode permettant de générer de manière automatique un grand nombre d'images avec différents modèles de véhicules et de personnes. L'utilisation d'un tel procédé permet également de facilement modifier les paramètres expérimentaux tels que l'illumination, le placement de la caméra, etc. ; ce qui est difficile lors d'une expérimentation en conditions réelles. Nous avons ainsi pu générer une première version de jeu de données synthétiques appelé PoseSynth illustrant un grand nombre de passagers dans différents véhicules et environnements, et prenant différentes poses. Une deuxième version du jeu de données, PoseSynth V2, a ensuite été créée afin de diversifier les angles de vue de la caméra ainsi que d'annoter la visibilité des points clés du visage. L'approche de génération par modélisation permet de générer théoriquement une infinité d'images diversifiées puisque tout est entièrement paramétrable et ne nécessite pas de données préalables, contrairement aux méthodes utilisant des réseaux génératifs. Cependant, plusieurs difficultés sont apportées par l'utilisation d'une telle méthode. Tout d'abord, il peut être difficile d'assurer le réalisme des images. En effet, un important travail est nécessaire pour générer des images avec des textures et des illuminations photo-réalistes. Il est possible d'intégrer des textures et environnements plus détaillés au procédé de génération, mais cela demande de

modifier manuellement les modèles 3D et d'affiner les paramètres de modélisation et de rendu, ce qui requiert du temps. Nous avons également vu qu'il était difficile de générer aléatoirement des poses vraisemblables tout en respectant l'ensemble des collisions. Une possibilité serait de définir des poses réalistes fixes, mais cela réduit la diversité des poses représentées.

Nous nous sommes également intéressés dans ces travaux à la création de données à l'aide de réseaux de neurones génératifs. Nous avons étudié en particulier l'approche par transfert de poses. L'utilisation d'une telle approche avait pour intérêt de générer des images conservant un aspect et des détails issus d'images réelles. Nous ne sommes cependant pas parvenus à entraîner un tel réseau génératif et donc à générer des images à l'apparence réelle dans notre contexte. Nous expliquons ce résultat par un manque de diversité dans les images à notre disposition pour entraîner le réseau à l'intérieur d'une voiture. Cette hypothèse est confirmée par le fait que nous ayons pu entraîner les mêmes réseaux avec des données synthétiques. Ces réseaux semblent cependant fortement dépendants du domaine sur lequel ils sont entraînés. En effet, nous avons vu que les modèles entraînés pour le transfert de poses avaient une très faible capacité à généraliser. En outre, mélanger plusieurs sources de données menait à un réseau capable de différencier les deux domaines, ou un réseau générant des images floues, et ne permettait donc pas d'aider le réseau à généraliser. Il pourrait être intéressant d'entraîner des modèles de transfert de poses sur un jeu de données générique comparable à COCO. Cela permettrait de tester les capacités de généralisation des réseaux, voire de proposer des poids entraînés pour un potentiel *fine-tuning* sur un jeu de données plus spécifique. Il n'existe cependant à notre connaissance aucun jeu de données générique utilisable pour le transfert de poses humaines, c'est-à-dire un jeu de données constitué de paires d'images de personnes prises dans des contextes et environnement variés, et avec les poses annotées. Bien que les deux jeux principaux de l'état de l'art, DeepFashion et Market-1501, proposent un grand nombre d'images et de sujets, les images ont été capturées dans leur contexte respectif avec des conditions expérimentales variant peu (arrière-plan, cadre, illumination, etc.).

Une autre approche qui aurait pu être étudiée plus en profondeur dans ces travaux est le transfert de style par réseaux génératifs. En effet, nous avons utilisé une méthode de modification de style générique pour tenter de partiellement modifier l'apparence et donc d'aider à l'adaptation de domaine. Cependant, un réseau génératif aurait pu être utilisé afin de transférer les images synthétiques dans le domaine des images réelles, ou encore de modifier l'environnement des images réelles en notre possession. Des architectures basées sur les GAN cycliques pourraient être considérées, par exemple, pour changer des images de zèbres en chevaux, ou encore des photos de paysages en tableaux de peintres (Isola et al., 2017; Park et al., 2020). Ce type de réseaux requiert cependant un grand nombre de données variées afin d'être entraînés, et nos premiers essais réalisés n'ont pas apporté de résultats concluants.

Généralisation de domaine pour l'estimation de poses

Nous avons orienté nos travaux autour de la généralisation de domaine pour l'estimation de poses humaines. L'objectif de l'utilisation d'une telle approche était d'entraîner un modèle capable de généraliser, c'est-à-dire d'obtenir de bonnes performances dans des conditions d'application inconnues. L'entraînement de ces réseaux sur des jeux de données

génériques tels que COCO leur permet généralement de performer efficacement dans un grand nombre de contextes. Les évaluations initiales des méthodes de l'état de l'art sur notre jeu de données DriPE ont cependant montré que, bien que les performances globales étaient bonnes, l'estimation de poses sur le bas du corps était mauvaise, en particulier sur les chevilles. Nous en avons donc conclu qu'un entraînement sur le jeu COCO ne permettait pas de généraliser efficacement sur une situation complexe telle que l'estimation de poses à l'intérieur d'un véhicule.

Nous avons alors décidé d'utiliser des images synthétiques de passagers pour améliorer les performances sur les points clés du bas du corps montrant de moins bonnes performances. En effet, nous ne disposions pas d'autres images réelles pour l'estimation de poses humaines que celles de DriPE. Puisque nous souhaitons évaluer la capacité de généralisation des réseaux sur des images réelles à l'intérieur d'une voiture, nous sommes contraints d'utiliser DriPE uniquement pour l'évaluation et non pour l'entraînement. Nous avons ainsi montré qu'en effectuant un *fine-tuning* sur des images synthétiques spécifiques, nous pouvions augmenter significativement la qualité des estimations des points clés sur le bas du corps. Ce gain se faisait cependant au détriment d'une importante perte de performances sur le haut du corps et par conséquent sur les performances totales. L'utilisation de méthodes d'augmentation de données ne permettaient pas d'empêcher cette détérioration. Nous avons alors étudié l'impact sur les performances de l'utilisation d'un jeu d'entraînement composé de plusieurs jeux de données. Ainsi, nous avons montré que le mélange de données génériques réelles et de données synthétiques spécifiques à notre contexte permettait de partiellement conserver les bonnes performances générales ainsi que le gain de performance sur le bas du corps. Cependant, il ne semble pas exister de ratio "parfait" de répartition des différents jeux de données, et il est donc nécessaire d'adapter ce paramètre aux contraintes d'applications. Nous avons également montré que l'ajout de données réelles illustrant notre contexte telles que celles de Drive&Act pouvait être bénéfique malgré les pseudo-annotations. Enfin, nous avons utilisé du sur-échantillonnage afin d'augmenter le nombre d'images possédant des chevilles annotées dans le jeu d'entraînement. Cette approche permet d'augmenter significativement les performances d'estimation de la position de ces points clés. Il serait intéressant de faire varier le taux de sur-échantillonnage afin d'étudier la limite d'une telle approche lorsque trop d'images sont répétées dans le jeu d'entraînement. Nous n'avons malheureusement pas eu le temps de réaliser cette étude. Finalement, l'utilisation de plusieurs jeux de données couplée au sur-échantillonnage a permis de réduire significativement le contrecoup de l'utilisation des images synthétiques. Les performances globales restaient cependant inférieures au réseau uniquement entraîné sur COCO.

Afin de mieux tirer parti des jeux de données génériques et spécifiques, nous avons alors proposé deux architectures réalisant la fusion des vecteurs de caractéristiques et des cartes de chaleur, respectivement, produits par deux instances d'une même architecture d'estimation de poses. Cette fusion est réalisée après l'extracteur de caractéristiques ou à la sortie du réseau. Les deux branches sont pré-entraînées sur COCO et sur les images synthétiques indépendamment, puis les poids sont gelés afin de préserver leurs spécificités. Nous avons ainsi montré que cette approche par fusion permettait d'améliorer les performances globales tout en conservant de bonnes performances pour l'estimation de poses du bas du corps. De plus, la fusion des cartes de chaleur produites par le réseau semble plus efficaces que la fusion des vecteurs de caractéristiques, et ce, malgré l'absence d'informations sur le contexte des images. Nous avons tenté d'injecter ces informations

lors de la fusion des cartes de chaleur, mais avons observé que cet apport agissait comme un bruit et détériorait légèrement les performances.

Du fait du manque de jeux de données annotés pour notre contexte dans l'état de l'art, nous avons uniquement pu évaluer les réseaux obtenus sur DriPE. De plus, bien que DriPE n'ait pas été utilisé directement dans les procédés d'entraînement, toutes les observations et les conclusions réalisées s'appuient sur les évaluations sur ce jeu de données. Il serait donc pertinent d'évaluer les différentes méthodes proposées sur d'autres images de passagers afin de valider solutions proposées. Il pourrait aussi être intéressant de tester l'utilisation de tels réseaux de fusion avec des données synthétiques dans d'autres contextes difficiles pour l'estimation de poses, tels que l'analyse d'images sportives (Badiola-Bengoa and Mendez-Zorrilla, 2021). Nous pouvons cependant effectuer une première évaluation qualitative sur les images extraites de l'expérimentation à Nantes. Un résultat de l'inférence du réseau par fusion des cartes de chaleur est montré dans la Figure 6.1. Nous pouvons constater que le réseau a globalement été capable de détecter aussi bien les points clés du haut que du bas du corps. Des imprécisions peuvent être relevées sur le visage, bien que le masque puisse également causer des perturbations pour l'estimation des points. Enfin, on peut voir que, bien que le réseau soit en mesure de détecter les jambes, des occlusions inhabituelles telles qu'un journal ou une pochette posée sur les cuisses perturbent l'estimation des points. Cependant, ces résultats qualitatifs confirment partiellement les capacités de généralisation du réseau. On peut alors se demander si l'ajout de branches supplémentaires entraînées sur des jeux de données différents pourrait encore améliorer les performances. En outre, il pourrait être intéressant d'implémenter les réseaux multi-branches avec des architectures plus complexes pour les différentes branches d'estimation de poses et pour l'auto-encodeur des cartes de chaleur, afin d'étudier si cette approche est applicable à d'autres réseaux et si cela permettrait de gagner en performances.

Performances de l'estimation de poses dans notre contexte

Puisque ces travaux se sont concentrés principalement sur l'estimation de poses dans notre contexte spécifique, et en particulier sur la généralisation de domaine, nous nous sommes peu intéressés aux performances des méthodes d'estimation de poses humaines dans un cas d'étude classique, c'est-à-dire lorsque le réseau est entraîné et évalué sur des images issues du même domaine.

Nous nous sommes cependant intéressés à l'utilisation de la visibilité des points clés afin d'apporter une information complémentaire à la pose. Nous avons donc proposé un métamodèle adaptable à une majorité d'architectures d'estimation de poses humaines permettant de prédire la visibilité des points clés. Ce métamodèle couplé à différentes architectures de l'état de l'art a permis d'estimer efficacement la visibilité. Des difficultés ont cependant été rencontrées sur la classe de visibilité "non visible", dû à l'ambiguïté inhérente à sa définition et à sa faible représentation dans les jeux de données. Nous avons par ailleurs montré que l'application sur un jeu de données avec des classes de visibilité mieux réparties permettaient d'augmenter efficacement les performances des prédictions de la visibilité. Nous avons alors tenté d'utiliser cette visibilité prédite pour améliorer l'estimation de la pose, et en particulier augmenter la précision en diminuant le nombre de faux positifs. Bien qu'une amélioration ait été observée dans ce sens, elle est associée à une importante diminution du rappel causée par l'élimination excessive de points prédits. Pour pouvoir appliquer une telle méthode dans le but d'augmenter les

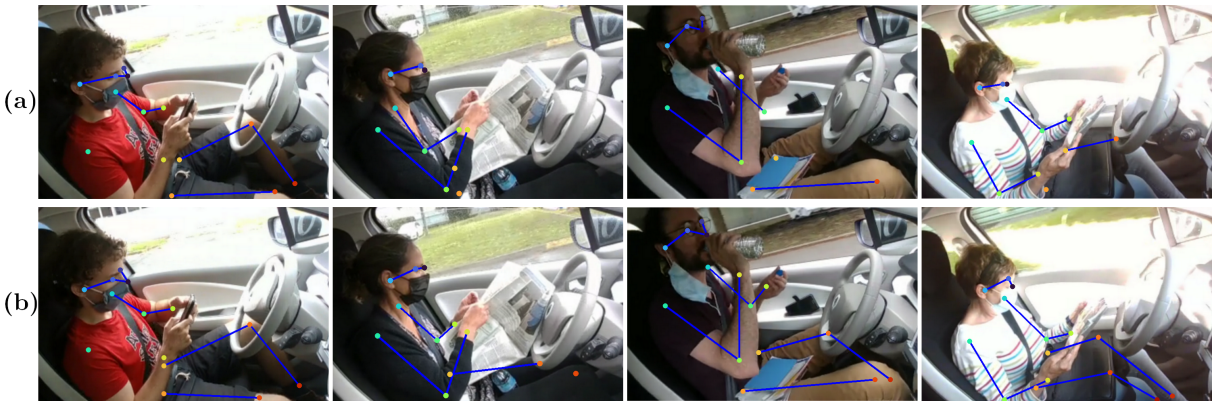


FIGURE 6.1 – Inférence d'images de l'expérimentation de Nantes avec *Simple Baseline* entraîné sur COCO (a) et notre réseau par fusion des cartes de chaleur (b).

performances globales de l'estimation de la pose, il serait donc nécessaire de réduire le nombre de faux positifs pour la classe de points "non annotés". Une possibilité serait d'adapter les poids utilisés pour la fonction d'entraînement du module de visibilité afin de plus fortement pénaliser ces mauvaises prédictions. Une architecture plus complexe pour le module de visibilité pourrait également améliorer ces performances, bien que nous ayons montré que l'ajout de couches totalement connectées supplémentaire amené à la saturation des performances tout en augmentant grandement le poids total du réseau. Une autre approche pour l'utilisation de la visibilité serait d'ajouter un module prenant en entrée la visibilité afin d'affiner les cartes de chaleur prédites, en utilisant par exemple un mécanisme d'attention tel que vu dans certains réseaux traités dans ces travaux.

Finalement, l'architecture multi-branches de fusion des cartes de chaleur à laquelle nous avons abouti au terme de ces travaux permet de réaliser l'estimation de poses à l'intérieur d'une voiture sur un domaine inconnu. Cette solution montre entre 70 et 80 % de performances globales. Nous avons cependant pu voir que la partie basse du corps restait difficile à estimer, avec en particulier un rappel plus bas que sur le reste du corps. Il pourrait être intéressant de tester des architectures plus récentes pour l'estimation de poses telles que les *transformers* qui ont montré des bonnes performances dans beaucoup de domaines, voir des modèles de fondations (Xu et al., 2022b). Une autre approche pourrait être de s'éloigner des contraintes fixées pour ces travaux telles que l'utilisation uniquement d'images RGB. Ainsi, l'ajout d'information de profondeur sur la scène pourrait permettre d'aider la détection des jambes parfois difficiles à détecter dû au faible contraste (Zimmermann et al., 2018; Pascual-Hernández et al., 2022). L'utilisation d'un modèle squelettique paramétrable pour l'estimation de la pose pourrait également aider le réseau à placer les jambes dans la scène. Cette approche requerrait soit de passer par une projection du squelette sur l'image, soit l'utilisation d'un modèle 3D (Chen and Ramanan, 2017; Chen et al., 2022b). Enfin, l'estimation de la pose sur un clip vidéo et non des images individuelles permettrait de corriger certaines mauvaises estimations grâce à la contrainte temporelle (Artacho and Savakis, 2020b; Zheng et al., 2021). On peut cependant considérer que cette approche aiderait peu pour l'estimation des chevilles dans nos travaux puisque ces points clés sont que ponctuellement détectés, comme montré par le faible rappel, ce qui laisse une importante incertitude. Toutes ces nouvelles modalités demanderaient cependant l'apport de nouvelles données spécifiques avec les annotations associées qui peuvent être difficile à obtenir, par exemple, les annotations 3D.

Contributions au projet AutoBehave

Enfin, ces travaux se sont inscrits dans le projet AutoBehave cherchant à étudier le comportement des utilisateurs de véhicules autonomes. Nous avons donc cherché à développer une solution pour l'estimation de poses applicable dans tous les contextes expérimentaux qui pourraient être rencontrés dans ce projet. Nous n'avons cependant pas pu évaluer quantitativement la performance de notre méthode sur les images collectées au cours de l'expérimentation à sur le campus de Nantes (Annexe A) par manque d'annotations.

Cependant, ce travail sur l'estimation de la pose à l'intérieur d'une voiture pourrait bénéficier à plusieurs domaines d'étude dans le cadre du projet AutoBehave. Par exemple, un certain nombre de méthodes de reconnaissance d'actions sont basées directement sur la pose détectée du sujet (Zhao et al., 2019; Si et al., 2019). Ainsi, la capacité à capturer avec précision la pose des occupants dans l'habitacle malgré les occultations peut directement améliorer les performances de reconnaissance des actions prises par les passagers (Yang et al., 2021). En outre, la pose des individus est une information pertinente pour analyser le comportement et les réactions des passagers. Les poses détectées à l'intérieur de l'habitacle pourraient donc permettre de mieux comprendre les interactions entre les humains, les véhicules et l'environnement lors de l'utilisation des nouvelles voitures autonomes en aidant à automatiser l'analyse des données vidéo. Enfin, l'obtention de la pose du bas du corps pourrait également fournir des informations intéressantes pour la conception de nouveaux sièges et d'espaces intérieurs adaptés aux futurs véhicules entièrement autonomes.

Travaux réalisés

Publications en conférence internationale

Guesdon, R., Crispim-Junior, C., and Tougne, L. (2021). Dripe : A dataset for human pose estimation in real-world driving settings. In *Workshop Autonomous Vehicle Vision (AVVision), Proceedings of the IEEE International Conference on Computer Vision (ICCV).*, pages 2865–2874

Guesdon, R., Crispim-Junior, C., and Tougne, L. (2022). Multitask metamodel for keypoint visibility prediction in human pose estimation. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5 : VISAPP*, pages 428–436. INSTICC, SciTePress

Guesdon, R., Crispim-Junior, C., and Tougne Rodet, L. (2023). Synthetic driver image generation for human pose-related tasks. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2023) - Volume 5 : VISAPP*, pages 762–769. INSTICC, SciTePress

Crispim-Junior, C., Guesdon, R., Jallais, C., Laroche, F., Corvec, S. S.-L., and Rodet, L. T. (2023). Autoexp : A multidisciplinary, multi-sensor framework to evaluate human activities in self-driving cars. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE

Dépôts publics

Jeu de données DriPE et code d'évaluation mAPK : `https://gitlab.liris.cnrs.fr/aura_autobehave/dripe`

Métamodèle pour la prédiction de la visibilité : `https://gitlab.liris.cnrs.fr/aura_autobehave/vis-pred`

Méthode de génération d'images synthétiques par modélisation 3D + jeu de donnée synthétique : `https://gitlab.liris.cnrs.fr/aura_autobehave/synthetic_drivers`

Autres logiciels

Logiciel d'annotation collaboratif d'images pour l'estimation de poses (publié au sein du laboratoire)

Annexes

A Expérimentation AutoBehave

Une expérimentation a été réalisée dans le cadre du projet AutoBehave afin de collecter différentes données (Crispim-Junior et al., 2023). Cette expérimentation avait pour objectif de mettre en conditions réelles des participants dans un véhicule autonome roulant. Ainsi, une trentaine de participants ont circulé sur le campus de l'École Centrale de Nantes dans une Renault Zoé robotisée (Figure A.1-a). Grâce à différentes caméras et capteurs LIDAR, le véhicule atteint un niveau d'autonomie 3-4 ; ce qui le rend capable de suivre un trajet préenregistré tout en adaptant sa trajectoire et son allure à l'environnement (Figure A.1-c). Le passager est assis sur le siège conducteur (Figure A.1-d), et un conducteur de sécurité est assis sur le siège passager afin de reprendre le contrôle du véhicule au besoin. Dans l'habitacle, une caméra couleur et profondeur Intel RealSense D435i enregistre le participant de côté, tandis qu'une caméra GoPro filme le visage et le haut du corps (Figure A.1-b).

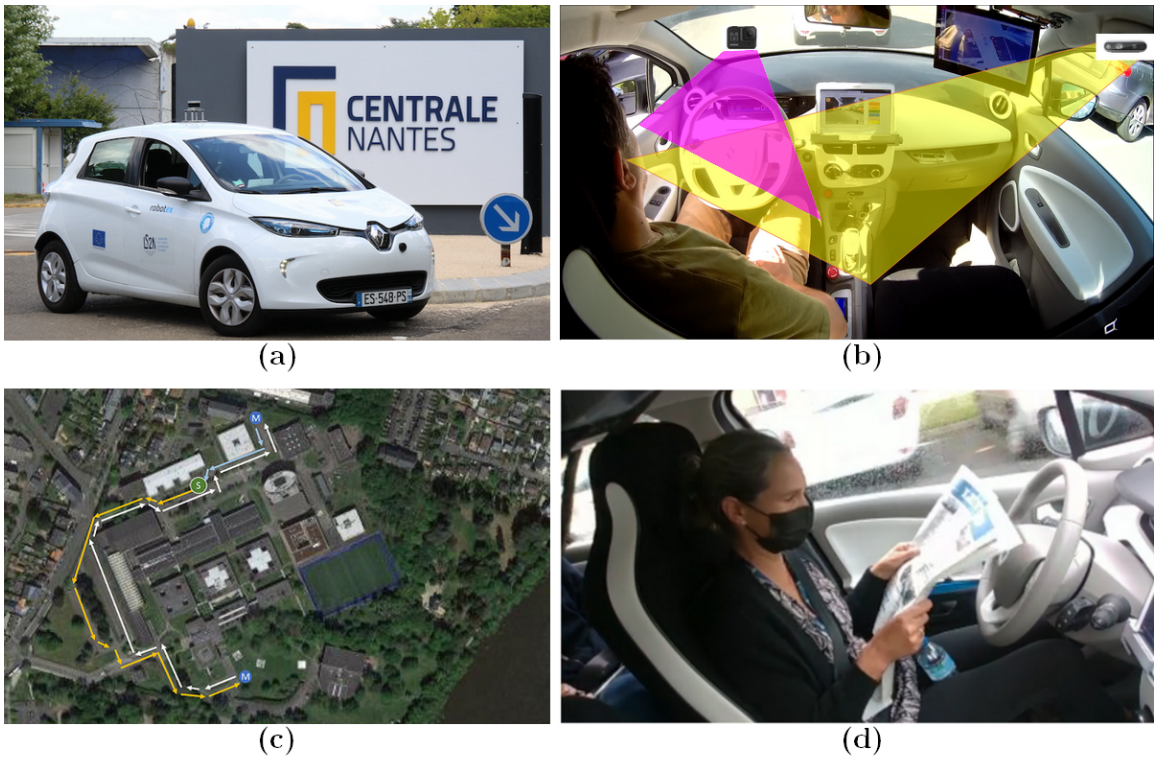


FIGURE A.1 – Illustrations de l'expérimentation à Nantes pour le projet AutoBehave avec : la voiture autonome (a), l'installation dans l'habitacle (b), le trajet parcouru (c) et un exemple d'image enregistrée (d) (Crispim-Junior et al., 2023).

La totalité de l'expérimentation s'est déroulée à l'intérieur du campus de l'École Centrale de Nantes. Un parcours fixe a été défini entre les différents parkings, consistant en une boucle d'environ 1,6 km (Figure A.1-c). Pendant la réalisation du trajet défini, le véhicule devait adapter sa trajectoire et son allure afin de franchir des obstacles fixes (véhicules en stationnement, dos d'âne) et gérer les interactions avec les autres usagers de la route (piétons, bicyclettes et autres véhicules).

L'expérimentation s'est divisée en deux phases de conduite pour un total de cinq tours. Tout d'abord, une phase libre de trois tours permettant au participant de découvrir la conduite autonome, le trajet, le véhicule, etc. Durant cette première phase, il a été demandé aux passagers de se comporter comme s'ils effectuaient un trajet quotidien entre leur domicile et leur lieu de travail. Ce scénario visait à collecter des données pour évaluer leur acceptation et leur réaction face au véhicule autonome, ainsi que les actions réalisées spontanément par les participants. Ensuite, une phase scriptée d'activités sur deux tours a eu lieu, durant laquelle l'expérimentateur demandait d'effectuer une série d'actions telles que la lecture d'un journal, l'utilisation d'une tablette, etc. L'ordre et la durée des actions ont été choisis aléatoirement. Cette approche a permis d'enregistrer à la fois les réactions spontanées des participants, et de mettre ces derniers en situation afin d'obtenir des images plus spécifiques. En plus des deux phases de conduite, des questionnaires ont également été intégrés à l'étude avant, durant, et après l'expérimentation. Ces questionnaires ont pour but de mesurer l'évolution du ressenti et des émotions des participants vis-à-vis du véhicule autonome.

Cette expérimentation a permis d'acquérir des clips vidéo représentant 29 participants réalisant 9 actions. Au total, près de 30h d'enregistrement ont été obtenues de vidéos RGB de face et de profil, ainsi que d'images de profondeur de profil Figure A.2. Parmi ces enregistrements, 500 minutes de vidéos illustrant des actions qui pourraient être réalisées

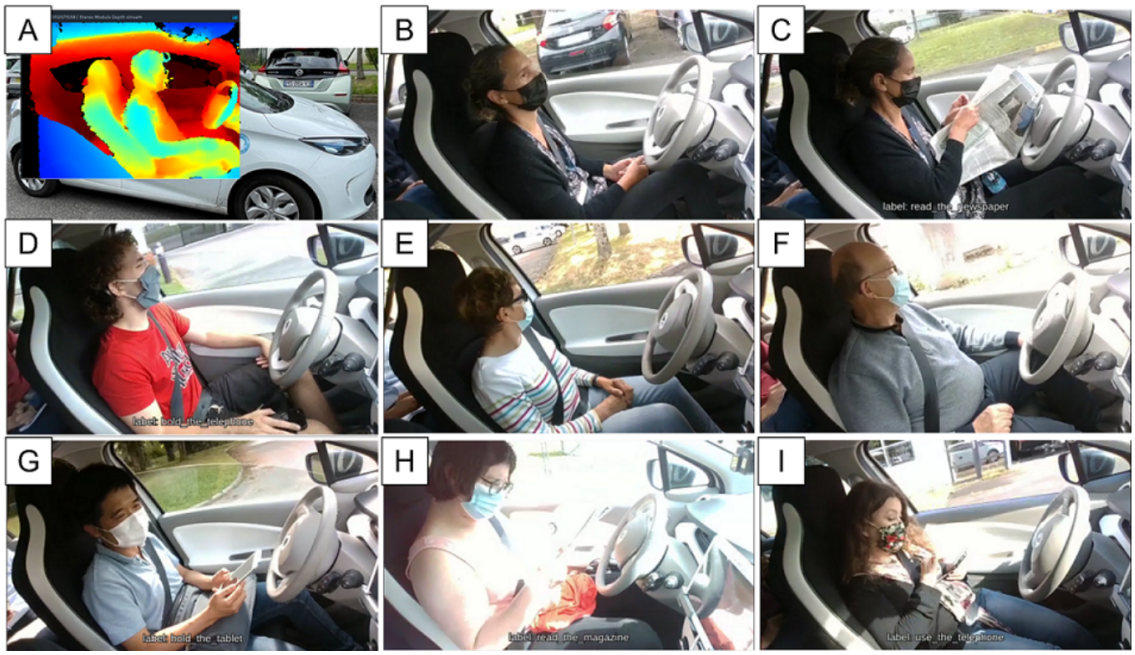


FIGURE A.2 – Exemples d'enregistrements lors de l'expérimentation à Nantes pour le projet AutoBehave avec : l'image de profondeur à l'intérieur de la voiture autonome (A), une participante durant la phase libre (B) et scénarisée (C), et différents exemples d'illuminations, de participants et d'activités (D-I) (Crispim-Junior et al., 2023).

dans une voiture avec une autonomie de niveau 4 ont été collectées. Une première analyse du comportement des passagers à l'intérieur d'un véhicule autonome a également pu être réalisée suite à cette expérimentation (Crispim-Junior et al., 2023). Enfin, des travaux réalisés en parallèle de cette thèse ont permis de former un jeu de données pour la reconnaissance d'action de passagers d'un véhicule autonome. Ce jeu de données est le seul jeu de la littérature de ce type enregistré en conditions réelles de roulage. Bien que la pose n'ait pas été annotée sur ce jeu de données, des résultats d'inférence comparant une solution de l'état de l'art (*Simple Baseline* entraîné sur COCO) et notre réseau de fusion sont montrés dans la Figure 6.1.

B Jeux de données

	Jeu de données	Année	Nb. d'images	Nb. de personnes	Réel / Synthétique	Contexte	Notes	Index
État de l'art	COCO	2015	120 k	270 k	Réel	Générique		33, 47, 52, 58, 63, 66–68, 70, 74, 97, 99, 102–103
	MPII	2014	25 k	38 k	Réel	Générique		33, 47
	Drive&Act	2019	9,6 M	15	Réel	Simulateur de conduite	Pseudo-annotations	34, 104
Réalisations de la thèse	DriPE	2021	10 k	19	Réel	Conduite réelle		47, 53, 58, 65–66, 74, 96–97, 99, 102, 106, 108, 110–111
	PoseSynth	2022	200 k	100	Synthétique	Simulation de roulage		96–97, 99
	PoseSynth V2	2023	120 k	100	Synthétique	Simulation de roulage	Visibilité du visage, plus grande diversité d'angles de caméra	100, 102–103
Combinaisons de jeux	"Jeu triple-données"	-	151,5 k	270 k + 15 + 100	-	-	75 % COCO + 20 % PoseSynth V2 + 5% Drive&Act	104, 106, 108, 110–111

Tableau B.1 – Tableau des jeux de données utilisés dans nos travaux pour l'estimation de poses.

Bibliographie

Akhter, I. and Black, M. J. (2015). Pose-conditioned joint angle limits for 3d human pose reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1446–1455.

Amazonmturk (2022). Amazon mechanical turk. `https://www.mturk.com/`. Dernier accès : 2022-11-01.

Andriluka, M., Pishchulin, L., Gehler, P., and Schiele, B. (2014). 2d human pose estimation : New benchmark and state of the art analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Andriluka, M., Roth, S., and Schiele, B. (2009). Pictorial structures revisited : People detection and articulated pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1014–1021. IEEE.

Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J. (2005). Scape : shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416.

Antoniou, A., Storkey, A., and Edwards, H. (2017). Data augmentation generative adversarial networks. *arXiv preprint arXiv :1711.04340*.

Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv :1701.04862*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Artacho, B. and Savakis, A. (2020a). Unipose : Unified human pose estimation in single images and videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7035–7044.

Artacho, B. and Savakis, A. (2020b). Unipose : Unified human pose estimation in single images and videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7035–7044.

AutoBehave (2023). Aura autobehave project. `https://perso.liris.cnrs.fr/carlos.crispim-junior/aura_autobehave2019.html`. Dernier accès : 2023-03-29.

Badiola-Bengoa, A. and Mendez-Zorrilla, A. (2021). A systematic review of the application of camera-based human pose estimation in the field of sport and physical exercise. *Sensors*, 21(18) :5996.

Barratt, S. and Sharma, R. (2018). A note on the inception score. *arXiv preprint arXiv :1801.01973*.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2) :157–166.

Bhattacharya, S., Das, N., Sahu, S., Mondal, A., and Borah, S. (2021). Deep classification of sound : A concise review. In *Proceeding of First Doctoral Symposium on Natural Computing Research : DSNCR 2020*, pages 33–43. Springer.

Blender (2022). Blender. `https://www.blender.org/`. Accessed : 2022-11-01.

Borghi, G., Pini, S., Vezzani, R., and Cucchiara, R. (2020). Mercury : a vision-based framework for driver monitoring. In *International Conference on Intelligent Human Systems Integration*, pages 104–110. Springer.

Borghi, G., Venturelli, M., Vezzani, R., and Cucchiara, R. (2017). Poseidon : Face-from-depth for driver pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4661–4670.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv :1809.11096*.

Bulat, A., Kossaifi, J., Tzimiropoulos, G., and Pantic, M. (2020). Toward fast and accurate human pose estimation via soft-gated skip connections. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pages 8–15. IEEE.

Cai, Y., Wang, Z., Luo, Z., Yin, B., Du, A., Wang, H., Zhou, X., Zhou, E., Zhang, X., and Sun, J. (2020). Learning delicate local representations for multi-person pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Canas, P. N., Ortega, J. D., Nieto, M., and Otaegui, O. (2022). Virtual passengers for real car solutions : synthetic datasets. *arXiv preprint arXiv :2205.06556*.

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., and Sheikh, Y. A. (2019). Openpose : Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Carreira, J., Agrawal, P., Fragkiadaki, K., and Malik, J. (2016). Human pose estimation with iterative error feedback. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cetinic, E., Lipic, T., and Grgic, S. (2018). Fine-tuning convolutional neural networks for fine art classification. *Expert Systems with Applications*, 114 :107–118.

Chen, C.-H. and Ramanan, D. (2017). 3d human pose estimation= 2d pose estimation+ matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7035–7043.

Chen, K., Zhuang, D., and Chang, J. M. (2022a). Discriminative adversarial domain generalization with meta-learning based cross-domain validation. *Neurocomputing*, 467 :418–426.

Chen, S., Xu, Y., Pu, Z., Ouyang, J., and Zou, B. (2022b). Skeletonpose : Exploiting human skeleton constraint for 3d human pose estimation. *Knowledge-Based Systems*, 255 :109691.

Chen, Y., Shen, C., Wei, X.-S., Liu, L., and Yang, J. (2017). Adversarial posenet : A structure-aware convolutional network for human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Chen, Y., Tian, Y., and He, M. (2020). Monocular human pose estimation : A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192 :102897.

Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., and Sun, J. (2018). Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR.

Chu, X., Yang, W., Ouyang, W., Ma, C., Yuille, A. L., and Wang, X. (2017). Multi-context attention for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J. (1995). Active shape models-their training and application. *Computer vision and image understanding*, 61(1) :38–59.

Crispim-Junior, C., Guesdon, R., Jallais, C., Laroche, F., Corvec, S. S.-L., and Rodet, L. T. (2023). Autoexp : A multidisciplinary, multi-sensor framework to evaluate human activities in self-driving cars. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE.

Cruz, S. D. D., Wasenmuller, O., Beise, H.-P., Stifter, T., and Stricker, D. (2020). Sviro : Synthetic vehicle interior rear seat occupancy dataset and benchmark. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 973–982.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet : A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE.

Ding, Y., Deng, W., Zheng, Y., Liu, P., Wang, M., Cheng, X., Bao, J., Chen, D., and Zeng, M. (2022). Iˆ 2r-net : Intra-and inter-human relation network for multi-person pose estimation. *arXiv preprint arXiv :2206.10892*.

Ding, Z. and Fu, Y. (2017). Deep domain generalization with structured low-rank constraint. *IEEE Transactions on Image Processing*, 27(1) :304–313.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words : Transformers for image recognition at scale. *arXiv preprint arXiv :2010.11929*.

Druzhkov, P. and Kustikova, V. (2016). A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26 :9–15.

Dubey, A., Ramanathan, V., Pentland, A., and Mahajan, D. (2021). Adaptive methods for real-world domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14340–14349.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Eichner, M., Marin-Jimenez, M., Zisserman, A., and Ferrari, V. (2012). 2d articulated human pose estimation and retrieval in (almost) unconstrained still images. *International Journal of Computer Vision*, 99(2) :190–214.

Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., and Twombly, X. (2007). Vision-based hand pose estimation : A review. *Computer Vision and Image Understanding*, 108(1-2) :52–73.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88 :303–338.

Farahani, A., Voghoei, S., Rasheed, K., and Arabnia, H. R. (2021). A brief review of domain adaptation. *Advances in data science and information engineering : proceedings from ICDATA 2020 and IKE 2020*, pages 877–894.

Feld, H., Mirbach, B., Katrolia, J. S., Selim, M., Wasenmüller, O., and Stricker, D. (2020). Dfki cabin simulator : A test platform for visual in-cabin monitoring functions. In *Proceedings of the 6th Commercial Vehicle Technology Symposium (CVT), 6th International*, University of Kaiserslautern. University of Kaiserslautern, Springer.

Ferguson, M., Ak, R., Lee, Y.-T. T., and Law, K. H. (2017). Automatic localization of casting defects with convolutional neural networks. In *2017 IEEE international conference on big data (big data)*, pages 1726–1735. IEEE.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

F.R.C. (2022). Le neurone - fédération pour la recherche sur le cerveau. `https://www.frcneurodon.org/comprendre-le-cerveau/a-la-decouverte-du-cerveau/le-neurone/`. Dernier accès : 2022-11-01.

Fukushima, K. (1980). Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4) :193–202.

Geng, Z., Sun, K., Xiao, B., Zhang, Z., and Wang, J. (2021). Bottom-up human pose estimation via disentangled keypoint regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14676–14686.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv :1406.2661*.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. (2006). A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19.

Guesdon, R., Crispim-Junior, C., and Tougne, L. (2021). Dripe : A dataset for human pose estimation in real-world driving settings. In *Workshop Autonomous Vehicle Vision (AVVision), Proceedings of the IEEE International Conference on Computer Vision (ICCV).*, pages 2865–2874.

Guesdon, R., Crispim-Junior, C., and Tougne, L. (2022). Multitask metamodel for keypoint visibility prediction in human pose estimation. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5 : VISAPP*, pages 428–436. INSTICC, SciTePress.

Guesdon, R., Crispim-Junior, C., and Tougne Rodet, L. (2023). Synthetic driver image generation for human pose-related tasks. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2023) - Volume 5 : VISAPP*, pages 762–769. INSTICC, SciTePress.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in neural information processing systems*, 30.

Halin, A., Verly, J. G., and Van Droogenbroeck, M. (2021). Survey and synthesis of state of the art in driver monitoring. *Sensors*, 21(16) :5558.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1) :1–12.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8) :2.

Hu, P. and Ramanan, D. (2016). Bottom-up and top-down reasoning with hierarchical rectified gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5600–5609.

Huang, S., Xiong, H., Cheng, Z.-Q., Wang, Q., Zhou, X., Wen, B., Huan, J., and Dou, D. (2020). Generating person images with appearance-aware pose stylizer. In *IJCAI*, `https://github.com/siyuhuang/PoseStylizer`.

Intel RealSense (2022). Intel realsense d435i. `https://www.intelrealsense.com/depth-camera-d435i/`. Dernier accès : 2023-02-28.

Ioffe, S. and Szegedy, C. (2015). Batch normalization : Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134.

iTaSC (2022). itasc algorithm. `https://wiki.blender.org/wiki/Source/Animation/IK`. Accessed : 2022-11-01.

Jackson, P. T., Abarghouei, A. A., Bonner, S., Breckon, T. P., and Obara, B. (2019). Style augmentation : data augmentation via style randomization. In *Workshop on Deep Vision, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) workshops*, volume 6, pages 10–11, `https://github.com/philipjackson/style-augmentation`.

Jagtap, A. D. and Karniadakis, G. E. (2022). How important are activation functions in regression and classification ? a survey, performance comparison, and future directions. *arXiv preprint arXiv :2209.02681*.

Janai, J., Güney, F., Behl, A., Geiger, A., et al. (2020). Computer vision for autonomous vehicles : Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3) :1–308.

Jegham, I., Khalifa, A. B., Alouani, I., and Mahjoub, M. A. (2019). Mdad : A multimodal and multiview in-vehicle driver action dataset. In *International Conference on Computer Analysis of Images and Patterns*, pages 518–529. Springer.

Jiang, T., Lu, P., Zhang, L., Ma, N., Han, R., Lyu, C., Li, Y., and Chen, K. (2023). Rtmpose : Real-time multi-person pose estimation based on mmpose. *arXiv preprint arXiv :2303.07399*.

Jin, X., Lan, C., Zeng, W., Chen, Z., and Zhang, L. (2020). Style normalization and restitution for generalizable person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3143–3152.

Johnson, S. and Everingham, M. (2010). Clustered pose and nonlinear appearance models for human pose estimation. In *Proceedings of the British Machine Vision Conference*. doi :10.5244/C.24.12.

Joo, H., Simon, T., and Sheikh, Y. (2018). Total capture : A 3d deformation model for tracking faces, hands, and bodies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8320–8329.

Ju, S. X., Black, M. J., and Yacoob, Y. (1996). Cardboard people : A parameterized model of articulated image motion. In *Proceedings of the second international conference on automatic face and gesture recognition*, pages 38–44. IEEE.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv :1710.10196*.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410.

Katrolia, J. S., El-Sherif, A., Feld, H., Mirbach, B., Rambach, J. R., and Stricker, D. (2021). Ticam : A time-of-flight in-car cabin monitoring dataset. In *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*, page 277. BMVA Press.

Kawaguchi, K. (2016). Deep learning without poor local minima. *Advances in neural information processing systems*, 29.

Ke, L., Chang, M.-C., Qi, H., and Lyu, S. (2018). Multi-scale structure-aware network for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning : Generalization gap and sharp minima. *arXiv preprint arXiv :1609.04836*.

Khan, M. Q. and Lee, S. (2019). Gaze and eye tracking : Techniques and applications in adas. *Sensors*, 19(24) :5540.

Kim, J. and Park, H. (2023). Limited discriminator gan using explainable ai model for overfitting problem. *ICT Express*, 9(2) :241–246.

Kingma, D. P. and Ba, J. (2014). Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*.

Kopuklu, O., Zheng, J., Xu, H., and Rigoll, G. (2021). Driver anomaly detection : A dataset and contrastive learning approach. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 91–100.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1) :79–86.

Kumar, A., Marks, T. K., Mou, W., Wang, Y., Jones, M., Cherian, A., Koike-Akino, T., Liu, X., and Feng, C. (2020). Luvli face alignment : Estimating landmarks' location, uncertainty, and visibility likelihood. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8236–8246.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database. http ://yann.lecun.com/exdb/mnist/.

Li, J., Wang, C., Zhu, H., Mao, Y., Fang, H.-S., and Lu, C. (2019a). Crowdpose : Efficient crowded scenes pose estimation and a new benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P. S., and He, L. (2020). A survey on text classification : From shallow to deep learning. *arXiv preprint arXiv :2008.00364*.

Li, W., Wang, Z., Yin, B., Peng, Q., Du, Y., Xiao, T., Yu, G., Lu, H., Wei, Y., and Sun, J. (2019b). Rethinking on multi-stage networks for human pose estimation. arXiv `https://github.com/megvii-detection/MSPN.git`.

Lifshitz, I., Fetaya, E., and Ullman, S. (2016). Human pose estimation using deep consensus voting. In *Computer Vision–ECCV 2016 : 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 246–260. Springer.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco : Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Liu, M.-Y. and Tuzel, O. (2016). Coupled generative adversarial networks. *Advances in neural information processing systems*, 29.

Liu, Q., Dou, Q., Yu, L., and Heng, P. A. (2020). Ms-net : multi-site network for improving prostate segmentation with heterogeneous mri data. *IEEE transactions on medical imaging*, 39(9) :2713–2724.

Liu, Z., Luo, P., Qiu, S., Wang, X., and Tang, X. (2016). Deepfashion : Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1096–1104.

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. (2015). Smpl : A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6) :1–16.

Ma, F., Xia, G., and Liu, Q. (2023). Human pose transfer via shape-aware partial flow prediction network. *Multimedia Systems*, pages 1–14.

Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., and Van Gool, L. (2017). Pose guided person image generation. *Advances in neural information processing systems*, 30.

Maik, V., Paik, D., Lim, J., Park, K., and Paik, J. (2010). Hierarchical pose classification based on human physiology for behaviour analysis. *Computer Vision, IET*, 4 :12 – 24.

MakeHuman (2022). Makehuman community. `http://www.makehumancommunity.org/`. Accessed : 2022-11-01.

Mancini, M., Bulo, S. R., Caputo, B., and Ricci, E. (2018). Best sources forward : domain generalization through source-specific nets. In *2018 25th IEEE international conference on image processing (ICIP)*, pages 1353–1357. IEEE.

Martin, M., Roitberg, A., Haurilet, M., Horne, M., Reiß, S., Voit, M., and Stiefelhagen, R. (2019). Drive&act : A multi-modal dataset for fine-grained driver behavior recognition in autonomous vehicles. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*.

Martinez, J., Hossain, R., Romero, J., and Little, J. J. (2017). A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2640–2649.

Morris, D. and Rehg, J. (1998). Singularity analysis for articulated object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 289–289. IEEE Computer Society.

Munea, T. L., Jembre, Y. Z., Weldegebriel, H. T., Chen, L., Huang, C., and Yang, C. (2020). The progress of human pose estimation : A survey and taxonomy of models applied in 2d human pose estimation. *IEEE Access*, 8 :133330–133348.

Narvekar, N. D. and Karam, L. J. (2011). A no-reference image blur metric based on the cumulative probability of blur detection (cpbd). *IEEE Transactions on Image Processing*, 20(9) :2678–2683.

Navya (2023). Les expérimentations des navettes autonomes navya. `hhttps://www.navya.tech/fr/performance-navette/`. Dernier accès : 2023-03-29.

Newell, A., Huang, Z., and Deng, J. (2017). Associative embedding : End-to-end learning for joint detection and grouping. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 2277–2287. Curran Associates, Inc.

Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 483–499. Springer.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan : Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29.

Ortega, J. D., Kose, N., Cañas, P., Chao, M.-A., Unnervik, A., Nieto, M., Otaegui, O., and Salgado, L. (2020). Dmd : A large-scale multi-modal driver monitoring dataset for

attention and alertness analysis. In *Workshop Assistive Computer Vision and Robotics (ACVR), Proceedings of the European Conference on Computer Vision (ECCV)*, pages 387–405. Springer.

Parekh, D., Poddar, N., Rajpurkar, A., Chahal, M., Kumar, N., Joshi, G. P., and Cho, W. (2022). A review on autonomous vehicles : Progress, methods and challenges. *Electronics*, 11(14) :2162.

Park, T., Efros, A. A., Zhang, R., and Zhu, J.-Y. (2020). Contrastive learning for unpaired image-to-image translation. In *Computer Vision–ECCV 2020 : 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 319–345. Springer.

Pascual-Hernández, D., de Frutos, N. O., Mora-Jiménez, I., and Canas-Plaza, J. M. (2022). Efficient 3d human pose estimation from rgbd sensors. *Displays*, 74 :102225.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch : An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., and Sun, J. (2018). Megdet : A large mini-batch object detector. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 6181–6189.

Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P. V., and Schiele, B. (2016). Deepcut : Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4929–4937.

Pons-Moll, G., Fleet, D. J., and Rosenhahn, B. (2014). Posebits for monocular human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2337–2344.

Pumarola, A., Agudo, A., Sanfeliu, A., and Moreno-Noguer, F. (2018). Unsupervised person image synthesis in arbitrary poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8620–8628.

Pytorch metrics (2022). Pytorch implementation of common gan metrics. `https://github.com/w86763777/pytorch-gan-metrics`. Accessed : 2022-11-01.

Qin, X., Wang, J., Chen, Y., Lu, W., and Jiang, X. (2022). Domain generalization for activity recognition via adaptive feature fusion. *ACM Transactions on Intelligent Systems and Technology*, 14(1) :1–21.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv :1511.06434*.

Ramzan, M., Khan, H. U., Awan, S. M., Ismail, A., Ilyas, M., and Mahmood, A. (2019). A survey on state-of-the-art drowsiness detection techniques. *IEEE Access*, 7 :61904–61919.

Redmon, J. and Farhadi, A. (2018). Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn : Towards real-time object detection with region proposal networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc.

Ren, Y., Fan, X., Li, G., Liu, S., and Li, T. H. (2022). Neural texture extraction and distribution for controllable person image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13535–13544.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net : Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015 : 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer.

Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386.

Ruggero Ronchi, M. and Perona, P. (2017). Benchmarking and error diagnosis in multi-instance pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 369–378.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

SAE International (2023). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles j3016_202104. `https://www.sae.org/standards/content/j3016_202104`. Dernier accès : 2023-03-29.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. *Advances in neural information processing systems*, 29.

Sandfort, V., Yan, K., Pickhardt, P. J., and Summers, R. M. (2019). Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. *Scientific reports*, 9(1) :16884.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2 : Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520.

Sapp, B. and Taskar, B. (2013). Modec : Multimodal decomposable models for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1) :1–48.

Si, C., Chen, W., Wang, W., Wang, L., and Tan, T. (2019). An attention enhanced graph convolutional lstm network for skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1227–1236.

Siarohin, A., Sangineto, E., Lathuiliere, S., and Sebe, N. (2018). Deformable gans for pose-based human image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3408–3416.

Sidenbladh, H., De la Torre, F., and Black, M. J. (2000). A framework for modeling the appearance of 3d articulated figures. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 368–375. IEEE.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations*, pages 1–14.

Sobel, I. (2014). An isotropic 3x3 image gradient operator. *Presentation at Stanford A.I. Project 1968*.

Song, S., Zhang, W., Liu, J., and Mei, T. (2019). Unsupervised person image generation with semantic parsing transformation. In *PProceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2357–2366.

Stoffl, L., Vidal, M., and Mathis, A. (2021). End-to-end trainable multi-instance pose estimation with transformers. *arXiv preprint arXiv :2103.12115*.

Sun, K., Xiao, B., Liu, D., and Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Tan, M. and Le, Q. (2019). Efficientnet : Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR.

Tang, H., Bai, S., Zhang, L., Torr, P. H., and Sebe, N. (2020). Xinggan for person image generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 717–734. Springer.

Tang, W. and Wu, Y. (2019). Does learning specific features for related parts help human pose estimation ? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tang, W., Yu, P., and Wu, Y. (2018). Deeply learned compositional models for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Tesla (2023). Tesla - autopilot et capacité de conduite entièrement autonome. `https://www.tesla.com/fr_fr/support/autopilot`. Dernier accès : 2023-03-29.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656.

Tompson, J. J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. *Advances in neural information processing systems*, 27.

Toshev, A. and Szegedy, C. (2014). Deeppose : Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018). Training deep networks with synthetic data : Bridging the reality gap by domain randomization. In *Workshop on Autonomous Driving, Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7167–7176.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization : The missing ingredient for fast stylization. *arXiv preprint arXiv :1607.08022*.

Unity (2022). Unity asset store. `https://assetstore.unity.com/`. Accessed : 2022-11-01.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vijendra, S., Vasudeva, N., and Parashar, H. J. (2016). Recognition of text image using multilayer perceptron. *arXiv preprint arXiv :1612.00625*.

Wang, Y., Qi, L., Shi, Y., and Gao, Y. (2022). Feature-based style randomization for domain generalization. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(8) :5495–5509.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment : from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4) :600–612.

Waymo (2023). Waymo - self driving cars. `https://waymo.com/`. Dernier accès : 2023-09-29.

Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4732.

Xia, H., Zhao, H., and Ding, Z. (2021). Adaptive adversarial network for source-free domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9010–9019.

Xiao, B., Wu, H., and Wei, Y. (2018). Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, `https://github.com/microsoft/human-pose-estimation.pytorch.git`.

Xu, Y., Zhang, J., Zhang, Q., and Tao, D. (2022a). Vitpose : Simple vision transformer baselines for human pose estimation. *Advances in Neural Information Processing Systems*, 35 :38571–38584.

Xu, Y., Zhang, J., Zhang, Q., and Tao, D. (2022b). Vitpose+ : Vision transformer foundation model for generic body pose estimation. *arXiv preprint arXiv :2212.04246*.

Yang, D., Dai, R., Wang, Y., Mallick, R., Minciullo, L., Francesca, G., and Bremond, F. (2021). Selective spatio-temporal aggregation based pose refinement system : Towards understanding human activities in real-world videos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2363–2372.

Yang, J., Zeng, A., Liu, S., Li, F., Zhang, R., and Zhang, L. (2023). Explicit box detection unifies end-to-end multi-person pose estimation. *arXiv preprint arXiv :2302.01593*.

Yang, Y. and Ramanan, D. (2011). Articulated pose estimation with flexible mixtures-of-parts. In *CVPR 2011*, pages 1385–1392. IEEE.

Yang, Y. and Ramanan, D. (2012). Articulated human detection with flexible mixtures of parts. *IEEE transactions on pattern analysis and machine intelligence*, 35(12) :2878–2890.

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2019). Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR.

Zhang, J., Li, K., Lai, Y.-K., and Yang, J. (2021). Pise : Person image synthesis and editing with decoupled gan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7982–7990.

Zhao, M., Beurier, G., Wang, H., and Wang, X. (2020). A pipeline for creating in-vehicle posture database for developing driver posture monitoring systems. In *DHM2020*, pages 187–196. IOS Press.

Zhao, R., Wang, K., Su, H., and Ji, Q. (2019). Bayesian graph convolution lstm for skeleton based action recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 6882–6892.

Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition : A literature survey. *ACM computing surveys (CSUR)*, 35(4) :399–458.

Zheng, C., Zhu, S., Mendieta, M., Yang, T., Chen, C., and Ding, Z. (2021). 3d human pose estimation with spatial and temporal transformers. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 11656–11665.

Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., and Tian, Q. (2015). Scalable person re-identification : A benchmark. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Zhou, K., Liu, Z., Qiao, Y., Xiang, T., and Loy, C. C. (2022). Domain generalization : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhou, K., Yang, Y., Qiao, Y., and Xiang, T. (2021). Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 30 :8008–8018.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2232.

Zhu, Z., Huang, T., Shi, B., Yu, M., Wang, B., and Bai, X. (2019). Progressive pose attention transfer for person image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2347–2356.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1) :43–76.

Zimmermann, C., Welschehold, T., Dornhege, C., Burgard, W., and Brox, T. (2018). 3d human pose estimation in rgbd images for robotic task learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1986–1992. IEEE.

Table des figures

Liste des tableaux