

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique



Département d'Automatique

Laboratoire de Commande des Processus

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en automatique

Estimation 3D de Pose Humaine

BOUHALI Nouredine

Présenté et soutenu publiquement le (10/07/2019)

Composition du jury :

Président	M. Rachid ILLOUL	M C/A,	ENP, Alger, Algérie
Promoteur	M. Mohamed TADJINE	Professeur,	ENP, Alger, Algérie
Co-encadreur	Abdelhafid ZENATI	Dr.	ENP, Alger, Algérie
Examineur	M. Mohamed Seghir BOUCHERIT	Professeur,	ENP, Alger, Algérie

ENP 2019

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique



Département d'Automatique

Laboratoire de Commande des Processus

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en automatique

Estimation 3D de Pose Humaine

BOUHALI Nouredine

Présenté et soutenu publiquement le (10/07/2019)

Composition du jury :

Président	M. Rachid ILLOUL	M C/A,	ENP, Alger, Algérie
Promoteur	M. Mohamed TADJINE	Professeur,	ENP, Alger, Algérie
Co-encadreur	Abdelhafid ZENATI	Dr.	ENP, Alger, Algérie
Examineur	M. Mohamed Seghir BOUCHERIT	Professeur,	ENP, Alger, Algérie

ENP 2019

Dédicace

*Je dédie ce travail à ma mère, mon père et mes frères,
A mes amis, et à tous ceux qui nous ont aidé dans ce parcours de cinq ans, de
loin ou de proche.*

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de ma thèse et qui m'ont aidée lors de ce travail.

Je voudrais dans un premier temps remercier, mes encadreur de mémoire M. Mohamed TADJINE et Mr Abdelhafid ZENATI, Professeur et Doctorant à l'ENP Alger, pour leur patience, disponibilité et surtout judicieux conseils, qui ont contribué à m'encourager d'aller loin de ce que j'avais en intention.

Je tiens à témoigner toute ma reconnaissance aux personnes suivantes, pour leur aide dans la réalisation de ce mémoire :

Mon frère qui m'a beaucoup appris sur les défis à relever dans le monde de recherche et ingénierie.

Ma mère, pour son soutien constant et encouragements tout au long de ce parcours.
Mes amis Housseme Oularbi, Abdeldjalil Bensaïd, Moukraoui Cherif, Nemdil Ahmed, Abd-nour Hemal, Yacine Benameur, Samir Rebika et toute la promo 2019 d'Automatique. à L'Ecole Nationale Polytechnique d'Alger.

يعد التقدير المرئي لوضعية جسم الإنسان أحد أكثر مشكلات أبحاث الرؤية نشاطًا في مجال الحوسبة البصرية. في عملنا استنادًا DensePose ، نقدم أولاً حالة أحدث تقييم لهذا الميدان ونقدم منهجًا لحل المشكل المدروس متابعة لخوارزمية إلى عملية الالتواء الموسع لتعزيز متانة الخوارزمية ضد النقاط الرئيسية لجسم الانسان المغطاة. يتم تقييم هذه الطريقة جنبًا إلى جنب مع اختبارات المقارنة DensePose الأخيرة بناءً على البيانات المشروحة المقترحة التي وضعها فريق مقابل الخوارزمية الأصلية وبين التكوينات المختلة لنموذجنا. يتم أيضًا اقتراح المناقشات النهائية والاقتراحات لمزيد من تحسين أداء الخوارزميات.

الكلمات المفتاحية

وضعية جسم الإنسان ، التعلم العميق ، تجزئة الصورة ، الرؤية عبر الحاسوب

Abstract

Visual human pose estimation is one of the most highly active vision research problems in the area of Visual Computing. In our work, we first present the state of the art of Human Pose estimation and introduce a follow up approach of the DensePose algorithm based on the dilated convolutions operation to enhance the algorithm's robustness against occluded key points. This latter method is evaluated on the proposed annotated data set by the DensePose team along with comparison tests against the original algorithm and between the chosen configurations of our model. Final discussions and suggestions are also proposed for further performance enhancement of the algorithms.

Keywords : Human Pose, Deep Learning, Image segmentation.

Résumé

L'estimation visuelle de la pose humaine est l'un des problèmes les plus actifs de vision par ordinateur. Dans notre travail, nous présentons d'abord l'état de l'art de l'estimation de la Pose Humaine et introduisons une approche alternative de l'algorithme d'estimation "DensePose" basée sur l'opération des convolutions dilatées, ceci est fait pour améliorer la robustesse de l'algorithme face aux points clés ou partie du corps invisible. On évalue cette méthode sur la base de données proposées par l'équipe de DensePose avec des tests comparatifs par rapport à l'algorithme original et entre les configurations choisies. Des discussions finales et des suggestions sont également proposées pour améliorer encore les performances des algorithmes.

Mots clé : La pose humaine, Algorithme d'apprentissage, Segmentation de l'image.

Table des matières

Table des figures

Liste des abréviations

Introduction	9
1 Méthodes de base et l'état de l'art	18
1.1 L'hierarchie de l'estimation de pose humaine	19
1.1.1 La détection d'objets	19
1.1.2 La segmentation sémantique et segmentation des instances	20
1.2 Les approches existantes de segmentation sémantique	24
1.2.1 Les approche des Encodeur/Décodeur	24
1.2.2 Les approches à base des proposition de régions	24
1.2.3 L'approche des réseaux de neurones entièrement connectés	32
1.3 L'état de l'art de l'estimation de pose humaine	33
1.3.1 L'estimation de la pose 3D à partir des localizations des points clés	33
1.3.2 L'estimation directe de pose 3D	35
1.4 L'algorithme d'estimation de pose dense "DensePose"	37
2 L'estimation de pose humaine en utilisant les réseaux de neurones convolutionnels	39
2.1 Les réseaux de neurones convolutionnels	40
2.1.1 Le modèle de l'architecture des CNN	40
2.1.2 Les types de couche utilisées dans les CNN	42
2.1.3 Les ResNets	46
2.2 Le pipeline d'estimation de la pose humaine	50
2.2.1 L'extraction des <i>features</i>	50
2.2.2 Les Feature Pyramid Networks	51
2.2.3 Les réseaux de proposition des régions (RPN)	54

2.2.4	La détection d'objets	56
2.2.5	La segmentation sémantique de l'image	59
2.3	L'estimation 3D de la pose humaine	61
2.4	Evaluation de la performance des modèles d'estimation de la pose hu-	
	maine	63
2.4.1	Métrique d'évaluation de détection d'objet	63
2.4.2	Métrique d'évaluation des correspondances surfacique	64
2.5	Les bases de données des pose humaine	65
3	Analyse et expérimentation sur l'algorithme DensePose	69
3.1	Analyse du problème visée dans l'algorithme DensePose	70
3.2	Proposition du modèle Dilated DensePose	72
3.2.1	Convolution dilatée	72
3.2.2	Implémentations des convolutions dilatées	74
3.3	Procédure d'entraînement	77
3.3.0.1	Le pas d'apprentissage	78
3.3.1	Les rythme mise à jour des poids adaptatifs	79
3.3.2	Initialisation des poids	80
3.4	La plateforme logicielle d'apprentissage approfondi	81
4	Résultats expérimental	85
4.1	La procédure d'expérimentation sur le Dilated	
	DensePose	86
4.1.1	La variation de la précision	86
4.1.2	L'optimisation des hyperparamètres	89
4.1.3	Les variations de la fonction des coûts	89
4.2	Conclusion	91
	Conclusion générale	92
	Bibliographie	94
	Annexes	97
.1	Installation de Pytorch/DensePose	98
.2	Modèle d'estimation de pose DensePose dilaté	102

Table des figures

1	Le modèle squelette de pose humaine	10
2	Les application de l'estimation de pose humaine	15
1.1	Le flux de données de la détection d'objet dans les CNN	19
1.2	Architecture d'AlexNet entraînée sur deux core GPU	21
1.3	VGG-16	22
1.4	GoogleNet	23
1.5	Le réseaux Resnet 34 comparé avec un CNN de 34 couches et VGG-19	23
1.6	Les architectures des CNN basé sur les proposition de région.	24
1.7	L(architecture des R-CNN	26
1.8	Fast R-CNN	28
1.9	Détection des boîtes de délimitation : cls symbolise couche de classifica- tion, reg c'est la couche de régression des coordonnées des boites.	29
1.10	Faster R-CNN	30
1.11	Region-Fully Convolutional Networks	31
1.12	Segmentation par Mask R-CNN	32
1.13	Fully Convolutional Neural Networks	33
1.14	Les champs d'affinité des partie dans l'estimation de pose humaine	34
1.15	Fully Convolutional Neural Networks	36
1.16	Le modèle SMPL du corps humain	37
1.17	Le signal de supervision d'entraînement de DensePose	38
2.1	Un réseau de neurones ANN	41
2.2	Le réseau de neurones CNN à une seul couche	41
2.3	Block résiduel de transformation identité	47
2.4	Les deux type de bloc résiduel.	47
2.5	Les différentes configuration des ResNets	48
2.6	L'erreur Top-5 de la des gagnant de CIFAR 10	49

2.7	Comparaison sur l'effet d'augmentation de la profondeur sur l'erreur des CNN normal et les ResNets	49
2.8	L'effet d'augmentation de profondeur sur l'erreur des ResNets	50
2.9	L'architecture pyramid proposée dans les FPN	51
2.10	Résolution et valeur sémantique à travers les couches des CNN	52
2.11	Le schéma Data flow des FPN	53
2.12	La formation des <i>feature map</i> dans la partie descendante	54
2.13	FPN + Region Proposal Networks	55
2.14	Les proposition de boites de délimitation	56
2.15	Détection d'objet par proposition des régions	57
2.16	Flux des données dans les Fast R-CNN	58
2.17	Flux des données de Faster R-CNN à base des FPN	59
2.18	L'opération RoI Pooling et RoIAlign	60
2.19	L'architecture du Mask R-CNN globale	61
2.20		62
2.21	Le régression dense de pose 3D	62
2.22	Object Keypoint Similarity	64
2.23	Exemples tirés de HumanEva II	66
2.24	La segmentation et régression dans DensePose	68
2.25	Exemple tirée de dataset DensePoseCOCO	68
3.1	Visualisation de cartes de caractéristiques (extraites pour une tâche de classification) à différents niveaux de la couche cachée dans le réseau : les niveaux de caractéristiques plus élevés représentent une valeur sémantique et une complexité de représentation des données plus élevées, au lieu d'une représentation à une échelle supérieur.	72
3.2	Le champ réceptif d'une convolution dilatée à pas 1, 6, 24	73
3.3	La perte d'information dans les convolution usuel en comparaison avec les convolution dilatée	74
3.4	L'algorithme Atrous Spatial Pyramid Pooling qui utilise la dilatation sur le même niveau de couche convolutionnelle	76
3.5	L'effet du rythme d'apprentissage sur la précision	79
3.6	La plateforme de développement COLAB	82
3.8	L'interface des commande de COLAB	82

3.9	Le résultat temps réel de l'entraînement du réseau de neurones sur CO-	
	LAB.	84
4.1	Les résultat de teste de précision (AP) simulé de l'algorithme DensePose	87
4.2	Les résultats de teste de précision originaux de l'algorithme DensePose	
	[12]	87
4.3	Les fonctions de coût des configuration C1 et C2 en terme de nombre	
	des itérations	90

Liste des abréviations

ANN	Artificial Neural Networks
ASPP	Atrous Spacial Pyramid Pooling
CNN	Convolutional Neural Networks
EM	Expectation Maximization
EQM	Erreur Quadratique Moyenne
FCN	Fully Convolutional Networks
FC	Fully Connected
FPN	Feature Pyramid Networks
GD	Gradient Descent
GPS	Geodesic Point Similarity
GPU	Graphical Processing Unit
HOG	Histogram of Oriented Gradients
IHR	Intéraction Homme Machine
MLP	Multi-Layer Perceptron
MPPE	Multi-Person Pose Estimation
NMS	Non Maximum Suppression
OKS	Object Keypoint Similarity
PCK	Percentage of Correct Keypoints
PDJ	
R-CNN	Region based Convolutional Neural Networks
R-FCN	Region based Fully Convolutional Neural Networks
RCP	Ratio of Correct Keypoints
RPN	Region Proposal Networks

RoI	Region of Interest
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SMPL	Skinned Multi-Person Linear Model
SPPE	Single person Pose Estimation
SVM	Support Vector Machine
mAP	mean Average Precision

Introduction

Qu'est-ce que l'estimation de pose humaine

L'estimation de la pose humaine est considérée comme l'une des tâches fondamentales dans le domaine de la vision par ordinateur et l'un des problèmes de recherche les plus actifs dans la littérature du traitement des images. Il s'agit d'extraire la position géométrique et l'orientation du corps humain en utilisant une ou plusieurs images. Elle peut être aussi définie comme étant le problème de la localisation des articulations humaines (aussi appelées points clé ou "keypoints" comme les coudes, poignets ... etc) à partir d'une images (pixels), pour sortir au final par un squelette humain désignant l'orientation de chaque segment du corps.

Un squelette humain représente l'orientation d'une personne dans un format graphique. Essentiellement, il s'agit d'un ensemble de coordonnées qui peuvent être reliées pour décrire la pose de la personne. Chaque coordonnée dans le squelette est connue comme une partie du corps. Une connexion valide entre deux parties est connue sous le nom de paire (ou membre). Notez que toutes les combinaisons de pièces ne donnent pas lieu à des paires valides.

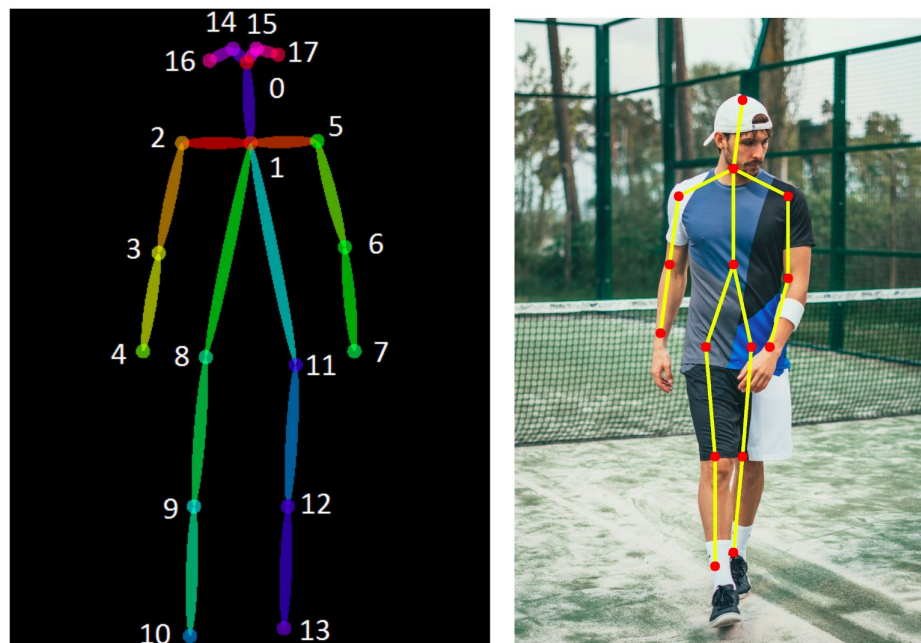


FIGURE 1 – Le modèle squelette de pose humaine

Estimation de pose 2D - Estimez les coordonnées de pose 2D (x,y) pour chaque articulation à partir d'une image RGB.

Estimation de pose 3D - Estimez une pose 3D , ou (x,y,z) sont les coordonnées incluant l'information de la profondeur.

L'estimation de la pose humaine est un domaine qui fait l'objet de nombreuses recherches, tant en termes de profondeur que de largeur. Le principal problème avec l'estimation de la pose humaine dans ce contexte est le fait qu'il y a un écart sémantique entre l'entrée et la sortie, réduisant le problème à une question de représentation. En particulier, les images ne sont qu'un tableau de nombres qui symbolise l'intensité de la lumière dans un pixel spécifique. Cependant, il s'agit d'extraire de cette grille des informations significatives, consistant à trouver la position de chaque pixel de chaque point clé et à attribuer un nom spécifique à cette articulation (dans le corps humain). Par conséquent, le problème est devenue un problème de représentation et traitement de signal/image et non pas un problème d'estimation mathématique.

Les contraintes et critères

L'estimation de pose humaine est une tâche qui englobe plusieurs aspects de travail selon des critères d'application ou contraintes d'utilisation (ex. utilisation sur des image ou vidéo). Pour cela, on représente ci-dessous les contraintes optées qui définissent le type de tâche visée dans l'estimation de pose humaine :

1. Estimation de la pose pour une ou plusieurs personnes :

Selon le nombre de personnes suivies, l'estimation de la pose peut être classée en estimation de la pose pour une seule personne et pour plusieurs personnes. L'estimation de la pose pour une seule personne (Single Person Pose Estimation) est la plus facile des deux, avec une contrainte d'existence d'une seule personne dans le cadre de l'image. D'autre part, l'estimation de la pose par plusieurs personnes (Multi Person Pose Estimation), présente plus de difficultés car l'emplacement et le nombre de personnes sur une image sont inconnus, ainsi que la nécessité de traiter les problèmes supplémentaires des occlusions entre personnes. Les approches initiales dans l'estimation des poses étaient principalement axées sur le SPPE, mais avec la disponibilité d'énormes ensembles de base de données multi-personnes, le problème du MPPE a récemment fait l'objet d'une attention accrue.

2. Estimation de pose par correspondance surfacique ou par détection des points clés :

La modalité de la sortie de l'algorithme joue un rôle important dans l'applicabilité de ce dernier. Typiquement, le résultat de l'estimation de la pose humaine est la localisation de tous les points clés (articulations) dans l'espace 2D ou 3D. D'autre part, Les techniques basé sur la correspondance surfacique (aussi appelé «Dense Pose estimation» ou « Human mesh recovery») vise à cartographier tous les pixels appartenant au corps humain d'une image RGB à la surface 3D du modèle paramétrique utilisé du corps humain.

3. Vue simple ou vue multiple :

Une grande partie de la recherche consiste à résoudre le problème de l'estimation de la pose à l'aide d'une caméra à image unique. Cependant, il existe certains algorithmes qui tentent d'utiliser des données provenant de plusieurs points de vue/caméras, les combinant pour générer des poses plus précises et mieux gérer les occlusions entre points clés ou personnes. La recherche sur l'estimation de la pose multi-caméras est actuellement quelque peu limitée, principalement en raison du manque de bons ensembles de base de données.

4. La nature de l'entrée :

Dans ce cas, la modalité fait référence aux différents types d'entrées disponibles. Les Images RGB (dans la nature) ou images des capteur de profondeur ou de mouvement (RGB-D, MoCap) avec un réglage et acquisition des image contrôlé

au laboratoire. Compte tenu de la facilité d'acquisition, il existe trois formes principales de données d'entrée :

(a) Image Rouge-Vert-Bleu (RGB) :

Les images que nous voyons autour de nous sur une base quotidienne, et le type d'entrée le plus commun pour l'estimation de la pose. Les modèles utilisant une entrée uniquement RGB ont un énorme avantage sur les autres en termes de mobilité de la source d'entrée. Ceci est dû à la facilité de disponibilité des caméras courantes (qui capturent des images RGB), ce qui en fait les modèles pouvant être utilisés sur un grand nombre d'appareils.

(b) Image à profondeur (temps de vol) :

Dans une image de profondeur, la valeur d'un pixel correspond à la distance par rapport à la caméra, mesurée par rapport au temps de vol. L'introduction et la popularité des appareils peu coûteux comme Microsoft Kinect ont facilité l'obtention de données de profondeur. L'image en profondeur peut compléter l'image RGB pour créer des modèles de vision par ordinateur plus complexes et plus précis.

(c) Image infrarouge (IR) :

Dans une image IR, la valeur d'un pixel est déterminée par la quantité de lumière infrarouge réfléchiée par la caméra. L'expérimentation dans le domaine de vision par ordinateur basée sur des images IR est minimale par rapport aux images RGB ou image en profondeur. Microsoft Kinect fournit également des images infrarouges pendant l'enregistrement. Cependant, il n'existe actuellement aucun ensemble de base de données contenant des images infrarouges.

5. Estimation de la position 2D ou 3D :

Selon la dimension de sortie requise, le problème de l'estimation de la pose peut être classé en deux dimensions : estimation de la pose 2D et estimation de la pose 3D. L'estimation de la position 2D permet de prédire l'emplacement des articulations du corps dans l'image (en termes de coordonnées des pixels). D'autre part, l'estimation de pose 3D prédit une disposition spatiale tridimensionnelle de toutes les articulations du corps comme résultat final.

La plupart des modèles d'estimation de pose 3D prédisent d'abord la pose 2D, puis essaient de l'élever à la pose 3D. Cependant, il existe aussi des techniques

d'estimation de pose 3D « end-to-end » qui permettent de prédire directement la pose en 3D.

6. Modèle du corps :

Chaque algorithme d'estimation de pose s'accorde au préalable sur un modèle corporel. Il permet à l'algorithme de formaliser le problème de l'estimation de la pose humaine en celui de l'estimation des paramètres du modèle corporel. Au début, la plupart des algorithmes utilisent un modèle simple de squelette cinématique rigide de N articulations (N est généralement compris entre 13 et 30) comme résultat final. Formellement, les modèles cinématiques peuvent être représentés sous forme de graphe, où chaque sommet V représente une articulation. Les bords E peuvent coder des contraintes ou des hypothèses antérieures sur la structure du modèle corporel qui représente des poses singulières non existantes.

Un tel modèle suffit pour la plupart des applications. Cependant, pour de nombreuses autres applications telles que l'animation des caractères, un modèle plus élaboré est nécessaire. Certaines techniques ont considéré un modèle de maillage très détaillé, représentant l'ensemble du corps avec un nuage de points, le dernier modèle de corps développé de ce type est celui présenté par F. Bogo et al dans [1].

Un autre modèle corporel plutôt primitif qui a été utilisé dans les pipelines de l'estimation de pose antérieurs est un modèle corporel basé sur la forme. Dans les modèles basés sur la forme, les parties du corps humain sont approximées à l'aide de formes géométriques telles que rectangles, cylindres, coniques, etc.

Les challenges et difficultés du problème.

Les défis de l'analyse visuelle de la pose humaine se situent à différents niveaux. Puisque cette tâche de reconnaissance d'un concept visuel (ex. la pose humaine) est relativement insignifiante pour un être humain, il vaut la peine de considérer les défis impliqués du point de vue d'un algorithme de vision par ordinateur. La liste des défis présentée ci-dessous est encore inachevée, sachant que la représentation brute des images est sous la forme d'un tableau tridimensionnel des valeurs de luminosité des pixels :

- **Variation du point de vue.** Une seule instance d'un objet peut être orientée de plusieurs façons par rapport à la caméra, et en particulier pour un objet complexe comme le corps humain.

- **Variation d'échelle.** Les classes visuelles présentent souvent des variations de taille (taille dans le monde réel, pas seulement en termes d'extension dans le plan de l'image) en plus des variations de forme corporelle et de vêtements.
- **Déformation.** Les humains ne sont pas des corps rigides et peuvent présenter des diverses déformations de manière extrême par rapport à l'espace des pose représenté ou abordés dans un algorithme de reconnaissance.
- **Occlusion.** Les objets d'intérêt peuvent être occultés ou invisible. Parfois, seule une petite partie d'un objet (aussi peu que quelques pixels) pouvait être visible. Cela peut se manifester par une occlusion de soi ou d'environnement.
- **Conditions d'éclairage.** Les effets de l'éclairage et luminosité sont drastiques au niveau des pixels.
- **L'arrière-plan est encombré.** Les objets d'intérêt peuvent se fondre dans leur environnement, ce qui les rend difficiles à identifier.

D'autres considérations apparaissent également dans les approches des algorithmes d'apprentissage liées à la variation des bases de données d'apprentissage qui empêchent l'algorithme de généraliser les résultats de son processus d'apprentissage. La plupart des données disponibles sont prise dans la nature ou « in -the-wild », avec une gamme de variation limitée de poses, de formes et d'apparences humaines, ce qui rend encore la généralisation plus difficile par rapport à des poses qui n'existaient pas auparavant ou comme leur dans des environnements non restreints et bondés.

Les application pratique

Les applications de pose humaine couvrent un large spectre de domaines incluant, mais sans s'y limiter, l'indexation et la surveillance vidéo (pour l'analyse du comportement et la reconnaissance d'action), la sécurité automobile, la robotique d'assistance (ex : contrôle du robot par geste à distance) en tant d'un sous-domaine de l'Interaction Homme-Robot ou Interaction Homme-Machine (IHR /HMI), l'analyse des performances sportives et les applications de réalité virtuelle .



FIGURE 2 – Les application de l'estimation de pose humaine

Les différentes approches pour l'estimation de pose humaine

Le problème de l'estimation de la pose humaine a longtemps été discuté dans la littérature de vision par ordinateur avant les progrès des algorithmes d'apprentissage profond et le développement des Processus (CPU et GPU) à gamme de fréquence élevée. En effet, un bon nombre des premières approches basées sur le graphisme d'ordinateur (Computer Graphics) utilisaient des algorithmes de calcul lourds pour estimer la pose humaine à partir des images. Plusieurs approches de l'estimation de la posture humaine ont été introduites au fil des ans. Les méthodes les plus anciennes (et les plus lentes) estiment généralement la pose d'une seule personne dans une image qui n'avait qu'une seule personne au départ. Ces méthodes permettent souvent d'identifier d'abord les membre individuelles, puis d'établir des liens entre elles pour créer la pose.

Les approches classiques

Dans cette catégorie, c'est la majorité des approches graphiques antérieures qui utilisaient des d'algorithmes conçus préalablement et manuellement dans l'extraction et la description de caractéristiques des image (« features ») comme les algorithmes du SIFT et HOG [20], [6] traitant les pixel de l'image comme étant des fonction continue représentant les surface des objet. L'intersection des surface ou le soudain changement d'intensité de pixel signifie l'existence des bord, et en s'y basant, Il construit des vecteur de gradient de l'image utilisée ultérieurement dans la détection des bords du corps humain en passant par un modèle paramétrique :

1. L'approche classique de l'estimation de la pose articulée est l'utilisation du cadre

des structures picturales. L'idée de base est ici de représenter un objet par un ensemble de "pièces" disposées dans une configuration déformable (non rigide). Une "pièce" est un modèle d'apparence qui est assorti dans une image. Lorsque les pièces sont paramétrées en fonction de l'emplacement et de l'orientation des pixels, la structure résultante peut modéliser l'articulation, ce qui est très pertinent pour l'estimation de la pose. (Une tâche de prédiction structurée).

2. La méthode ci-dessus, cependant, vient avec la limitation d'avoir un modèle de pose ne dépendant pas des données de l'image. En conséquence, la recherche s'est concentrée sur l'enrichissement du pouvoir de représentation des modèles.
3. **Modèles de pièces déformables** Yang et Ramanan [30] utilisent un modèle de mélange de pièces qui exprime des relations articulaires complexes. Les modèles de pièces déformables sont une collection de modèles disposés dans une configuration déformable et chaque modèle possède un modèle global + des modèles de pièces. Ces modèles sont appariés dans une image pour reconnaître/détecter un objet. Le modèle basé sur les pièces peut bien modéliser les articulations sans prendre en compte le contexte globale de l'image.

Les approches fondées sur l'apprentissage profond

Le pipeline classique des méthodes antérieurs a ses limites et l'estimation de la pose a été grandement remodelée par les réseaux de neurones convolutionnels (Convolutional Neural Networks). Avec l'introduction de "DeepPose" [25] par Toshev et al, la recherche sur l'estimation de la pose humaine a commencé à passer des approches classiques aux approches du Deep Learning. La plupart des systèmes récents d'estimation de pose ont universellement adopté les CNN comme leur principal élément constitutif, remplaçant largement les caractéristiques artisanales et les modèles graphiques ; cette stratégie a permis d'améliorer considérablement les repères standard. En règle générale, le pipeline de ces méthodes comporte deux approches :

1. Approches descendantes : ces méthodes impliquent une étape de segmentation au début, où chaque humain est d'abord segmenté dans une boîte de délimitation, suivie d'une estimation de pose effectuée individuellement sur chaque boîte de délimitation. L'estimation descendante de la pose peut être classée selon des approches fondées sur un modèle corporel génératif et/ou sur l'apprentissage profond.
2. L'algorithme prédit d'abord toutes les parties du corps/points clés présentes dans

l'image. Ceci est généralement suivi par la formulation d'un graphe, basé sur le modèle corporel, qui relie les articulations appartenant à la même personne. Cette méthode est connue sous le nom de « approche ascendante ».

La structure du rapport

La thèse est organisée comme suit :

1. Le premier chapitre définit l'hierarchie de l'estimation de pose humaine, ses niveau représente les domaine et tâche qui sont incorporée dans le pipeline de l'estimation de pose. L'état de l'art de chaque niveau a été présenté
2. Dans le chapitre 2, on commence par définir les notion de base du pipeline de l'estimation de pose humain à partir des Réseaux de neurones convolutionnels jusqu'à la fin de l'estimation pour aboutir à l'algorithme DensePose de l'estimation dense 3D de pose humain.
3. Dans le chapitre 3, on présente notre travail sur l'algorithme DensePose à base des opération de convolution dilaté et on rapporte le code source en Annexe.
4. Dans le chapitre 4, on rapporte les résultat de simulation et d'entraînement des notre architecture de Dilated DensePose et la précision de chaque configuration prise.
5. On conclut par une analyse de nos résultat de travail et propose des aspects futures pour la recherche dans ce thème.

Chapitre 1

Méthodes de base et l'état de l'art

1.1 L'hierarchie de l'estimation de pose humaine

Avant d'entamer à l'état de l'art du domaine d'estimation de pose, et dans le cadre de ce travail, nous expliquerons plus en détail les tâches abstraites de bas niveau qui constituent la base de référence pour comprendre le pipeline général de l'algorithme d'estimation de la pose humaine. Dans la recherche en vision par ordinateur, les trois tâches principales s'inscrivent dans l'hierarchie suivante :

- La tâche fondamentale concerne la **classification**, qui consiste à faire une prédiction de classe (ou probabilité d'appartenance aux classes possibles) pour une entrée globale.
- La tâche qui se situe au niveau supérieur suivant est la **localisation/détection**, qui fournit non seulement les classes mais aussi des informations supplémentaires concernant la localisation spatiale de ces classes, ce qui inclut la tâche d'estimation de la pose.
- Enfin, la **segmentation sémantique** permet d'obtenir une inférence fine en faisant des prédictions denses et en déduisant des étiquettes pour chaque pixel, de sorte que chaque pixel est assorti par la classe de sa région et de l'objet qui l'enveloppe.

1.1.1 La détection d'objets

La résolution du problème de détection d'objet consiste à placer une boîte de délimitation serrée (« Bounding Box ») autour de ces objets et à associer la catégorie d'objet correcte à chaque boîte de délimitation, l'apprentissage de l'emplacement de la boîte de délimitation peut naturellement être modélisé comme un problème de régression.

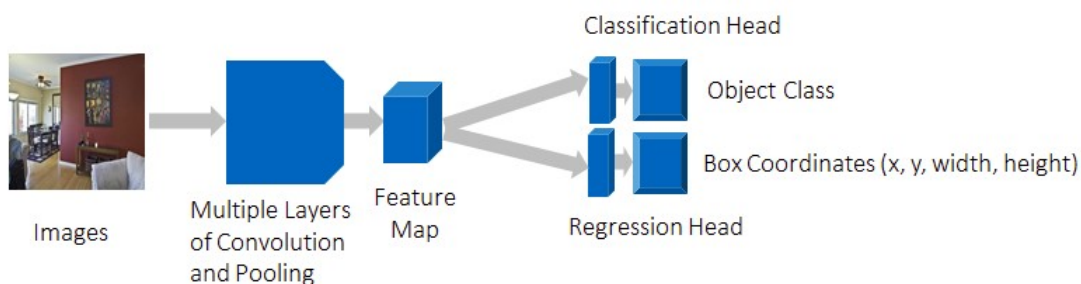


FIGURE 1.1 – Le flux de données de la détection d'objet dans les CNN

Il y a deux approches pour trouver ce sous-ensemble de fenêtres, qui mènent à deux catégories différentes d'algorithmes de détection d'objets.

- La première catégorie d'algorithmes forme d'abord des propositions de régions. Cela signifie que les régions très susceptibles de contenir un objet sont sélectionnées

soit à l'aide des techniques traditionnelles de vision par ordinateur (comme l'algorithme Selective Search), soit à l'aide d'un réseau de propositions régionales (Region Proposal Network) basé sur l'apprentissage profond. Une fois que nous avons rassemblé les fenêtres candidates, nous pouvons formuler un nombre défini de modèles de régression et de modèles de classification pour résoudre le problème de détection des objets. Cette catégorie inclut des algorithmes comme Faster R-CNN [24]. Les algorithmes de cette catégorie sont généralement appelés Two-Stage-Methods. Elles sont généralement plus précises, mais plus lentes que la méthode en une seule étape que nous présentons ci-dessous.

- La deuxième catégorie d'algorithme recherche des objets à emplacements fixes et des tailles fixes. Ces emplacements et ces tailles sont choisis de façon stratégique afin que la plupart des scénarios soient couverts. Ces algorithmes séparent généralement les images originales en zones de grille de taille fixe. Pour chaque région, ces algorithmes tentent de prédire un nombre fixe d'objets de certaines formes et tailles prédéterminées (ancres). Les algorithmes appartenant à cette catégorie sont appelés One-Stage-Methods. YOLO[7], SSD[8] et RetinaNet[9] sont des exemples de ce dernier. Les algorithmes de cette catégorie sont généralement plus rapides mais moins précis. Ce type d'algorithme est souvent utilisé pour des applications nécessitant une détection en temps réel.

1.1.2 La segmentation sémantique et segmentation des instances

La segmentation sémantique est une forme de détection d'objet de niveau supérieur. Il s'agit du processus de partitionnement d'une image en segments multiples cohérents et en contours fermés, généralement utilisé pour localiser les objets et leurs limites dans le plan des images. Pour ce qui concerne la segmentation d'instance, c'est un cas agnostique de segmentation d'image, où chaque objet de la même classe doit avoir une sortie unique.

La segmentation sémantique est une étape naturelle dans la progression de la tâche de prédiction visuelle brute envers la prédiction fine. Il est également utile de passer en revue quelques réseaux de neurones convolutionnel standard qui ont apporté des contributions significatives au domaine de la vision par ordinateur, car ils sont souvent utilisés comme modèle de base des systèmes de segmentation sémantique :

1. **AlexNet** Le pionnier du réseau CNN profond de Toronto, qui a remporté le

concours de classification ImageNet 2012 avec une précision de 84,6 %. Il se compose de 5 couches de convolution, Max-Pooling, et ReLU comme non-linéarités, avec 3 couches entièrement convolutionnelles à la sortie. Selon la figure suivante, l'architecture a été séparé en deux dans l'article original à cause des limitation du Hardware (carte graphique à RAM faible) et a été entraînée sur deux core GPU en parallèle.

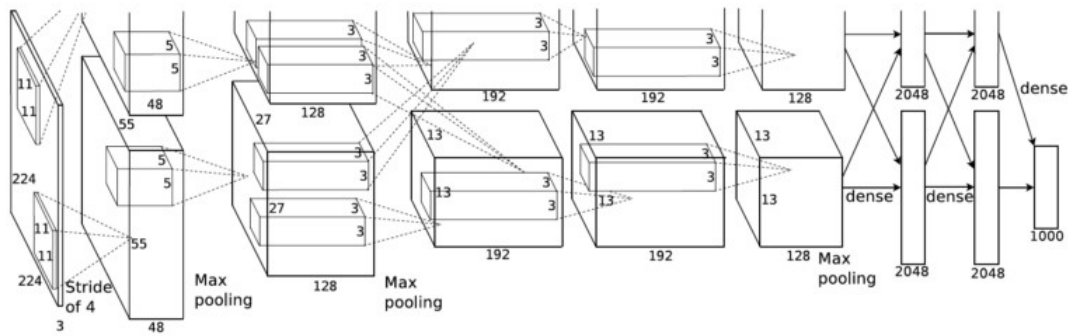


FIGURE 1.2 – Architecture d'AlexNet entraînée sur deux core GPU

2. **VGG-16** Ce modèle d'Oxford a remporté le concours ImageNet 2013 avec une précision de 92,7 %. Il utilise une pile de couches de convolution avec de petits champs réceptifs dans les premières couches au lieu de quelques couches à champs réceptifs grand.

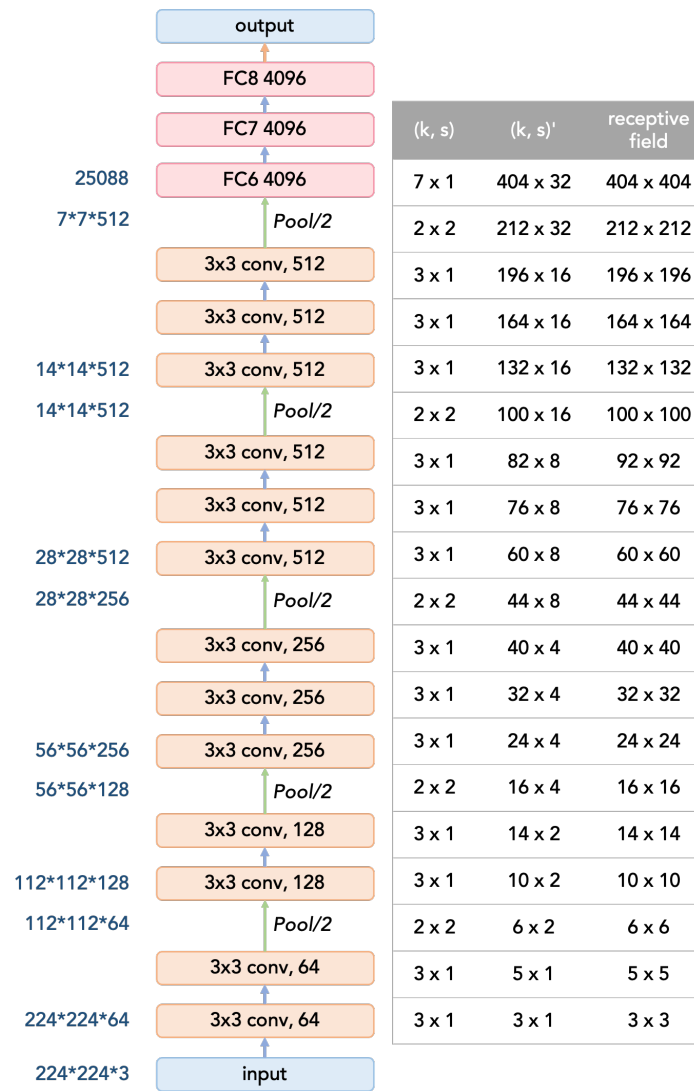


FIGURE 1.3 – VGG-16

3. **GoogleNet** Ce réseau conçu par Google a remporté le concours ImageNet 2014 avec une précision de 93.3% . Il est composé de 22 couches et d'un nouvel élément de base appelé module d'Inception. Le module se compose d'une couche Network-in-Network, d'une opération Pooling, d'une couche de convolution de grande taille et d'une couche de convolution de petite taille. Les modules de démarrage mettent l'accent sur la largeur de la couche, et sont utilisés 9 fois dans la première version du réseau de démarrage.

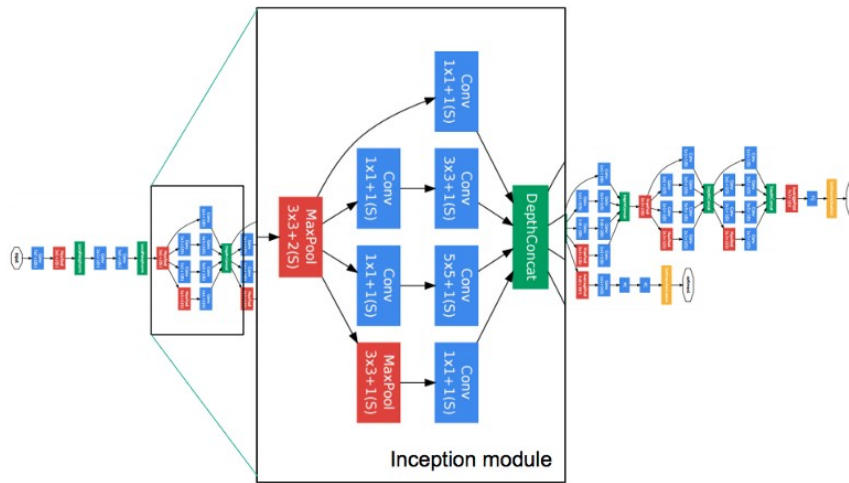


FIGURE 1.4 – GoogleNet

4. **ResNet** Ce modèle de Microsoft a remporté le concours ImageNet 2016 avec une précision de 96,4 % . Il est connu pour ses variantes (ResNet 34, 50, 101, 152) et la profondeur maximale de ses 152 couches et l'introduction des blocs résiduels. Les blocs résiduels abordent le problème de la formation d'une architecture vraiment profonde en introduisant des connexions de saut d'identité (ou à convolution de champs réceptive faible afin que les couches puissent copier leurs entrées sur la couche suivante. Plus de détail sont présenté dans le chapitre suivant.

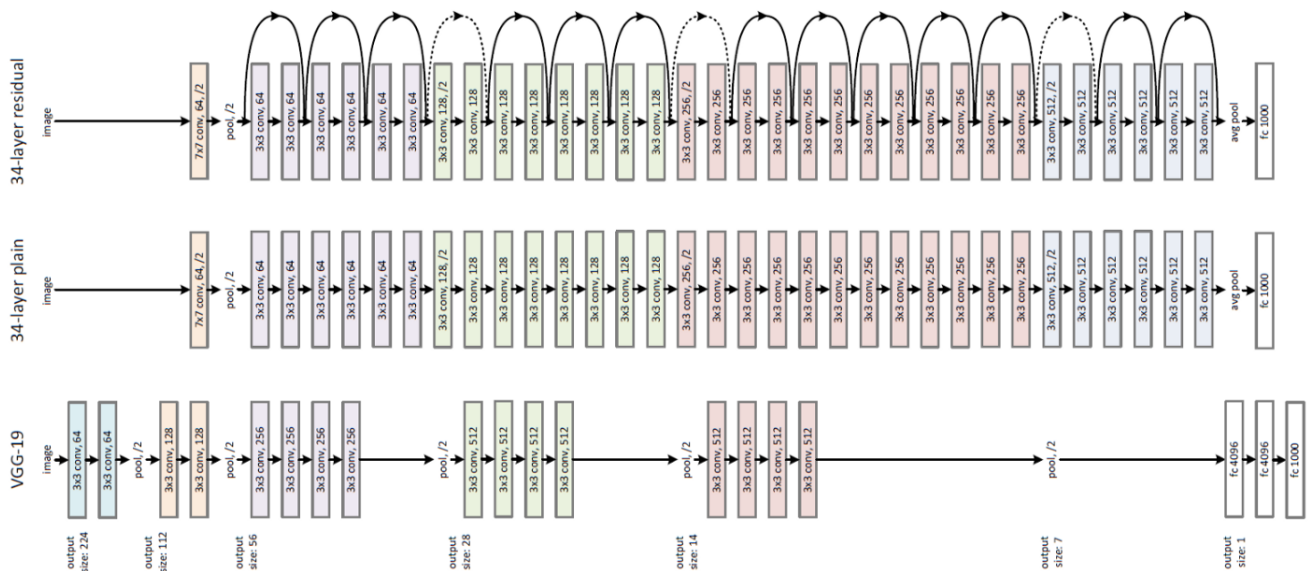


FIGURE 1.5 – Le réseaux Resnet 34 comparé avec un CNN de 34 couches et VGG-19

1.2 Les approches existantes de segmentation sémantique

1.2.1 Les approche des Encodeur/Décodeur

Une architecture générale de segmentation sémantique peut être généralement considérée comme un réseau d'**encodeurs** suivi d'un réseau de **décodeurs** :

Le **codeur** est généralement un réseau de classification préformé comme VGG/-ResNet suivi d'un réseau de décodeurs.

La tâche du **décodeur** est de projeter sémantiquement les caractéristiques discriminantes (résolution inférieure) apprise sur l'espace pixel (résolution supérieure) pour obtenir une classification dense.

Contrairement à la classification où le résultat final du réseau de neurones profond est la seule chose importante, la segmentation sémantique exige non seulement une discrimination au niveau du pixel, mais aussi un mécanisme pour projeter les caractéristiques discriminantes apprises aux différentes étapes de l'encodeur sur l'espace pixel. Différentes approches utilisent différents mécanismes dans le cadre du mécanisme de décodage.

1.2.2 Les approches à base des proposition de régions

Les méthodes basées sur les régions suivent généralement le pipeline "segmentation par reconnaissance", qui extrait d'abord les régions de forme libre d'une image et les décrit, suivis par une classification de région. Au moment du test, les prédictions basées sur les régions sont transformées en classification de pixels, généralement en étiquetant un pixel en fonction de la région la plus élevée qui le contient.

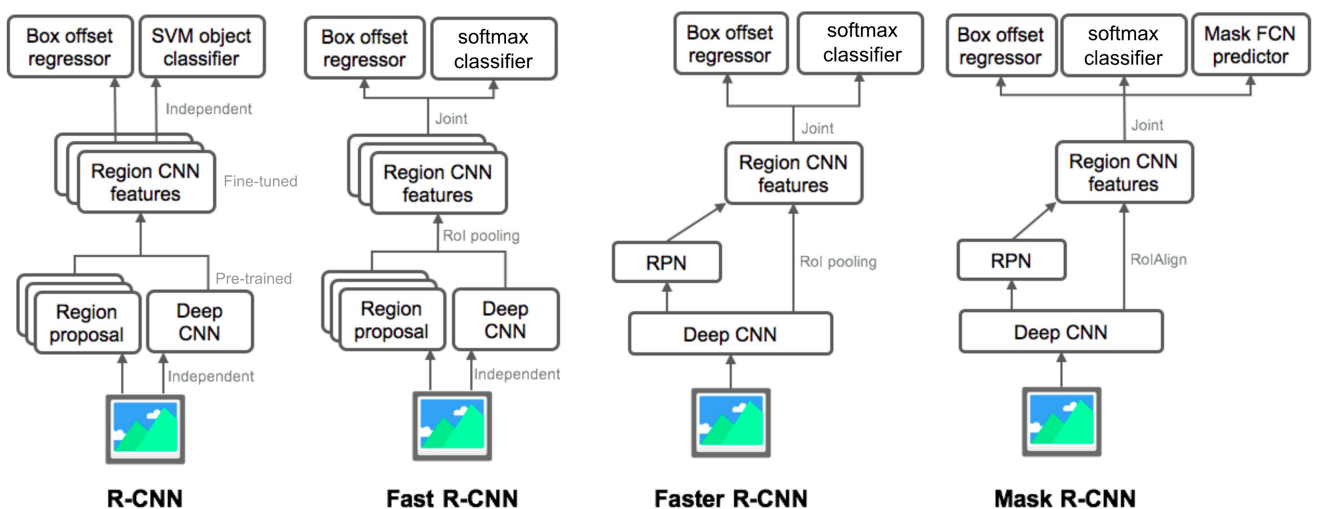


FIGURE 1.6 – Les architectures des CNN basé sur les proposition de région.

La figure ci-dessus représente les algorithmes de segmentation à base des région ; ou on a les block composants de ces modèles sont, de bas en haut :

1. L'entrée est une image RGB, et le Deep CNN est un réseau utilisé pour l'extraction de caractéristiques d'image à travers une succession de fonctions de Convolution, Pooling et fonction d'activation (généralement par une élimination par seuil des valeurs positives seulement "ReLU" qui présente beaucoup d'avantage par rapport au fonction d'activation utilisées antérieurement, comme le Tanh et le Sigmoid).
2. Les réseaux de proposition de la région (RPN) sont des réseaux entièrement convolutionnel formés pour proposer des ensembles de « features » qui ont été extraits par le CNN de base.
3. Dans les sorties, le Box Offset est les coordonnées du pixel supérieur gauche de la boîte de délimitation.
4. Le classificateur d'objets SVM est une fonction mathématique utilisée comme exemple de classificateur de la boîte de délimitation pour détecter la présence des motifs de caractéristiques humaines. Et durant la phase d'entraînement du CNN, l'algorithme du gradient est utilisé pour minimiser cette fonction de coût calculée total sur le groupe d'images utilisée dans chaque itération de l'optimisation.
5. Le dernier élément du Masque R-CNN est utilisé pour la classification des pixels qui appartiennent à une partie du corps humain visible dans l'image.

R-CNN (Régions avec fonctionnalité CNN)

Ce type d'architecture est le premier travail incluant les méthodes régionales. Il effectue la segmentation sémantique basée sur les résultats de la détection d'objet. R-CNN utilise d'abord l'algorithme de recherche sélective [26] pour extraire une grande quantité (environ 2000) de propositions d'objets et calcule ensuite les « features » du CNN de base pour chacune d'elles. En d'autre termes, le couche des « features » du CNN est calculée 2000 fois!. Enfin, il classe chaque région à l'aide des fonction SVM linéaires propres à la classe. R-CNN peut être construit sur n'importe quelle structure de référence CNN, comme AlexNet, VGG, GoogLeNet et ResNet.

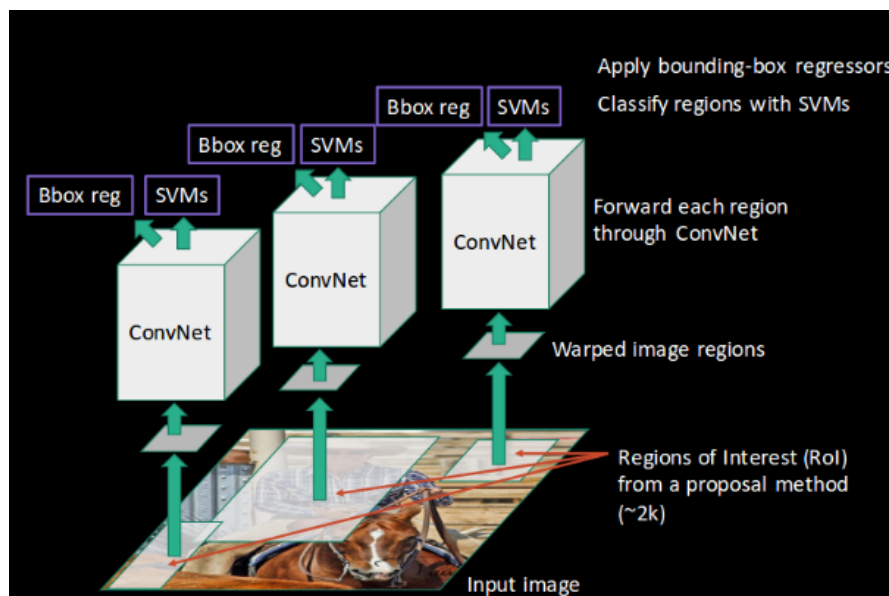


FIGURE 1.7 – L(architecture des R-CNN

Pour la tâche de segmentation d'image, R-CNN a extrait 2 types de caractéristiques pour chaque région : la caractéristique de région complète et la caractéristique du premier plan, et a constaté que cela pourrait conduire à une meilleure performance en les concaténant ensemble comme la caractéristique de région. R-CNN a réalisé d'importantes améliorations de performance grâce à l'utilisation des fonctionnalités hautement discriminantes du CNN. Cependant, elle présente également quelques inconvénients pour la tâche de segmentation :

La fonction n'est pas compatible avec la tâche de segmentation.

La caractéristique ne contient pas suffisamment d'information spatiale pour la génération de limites précises.

La génération de propositions prend du temps et affecterait grandement la performance finale surtout pour une utilisation real-time. En raison de ces goulots d'étranglement, des recherches récentes ont été proposées pour résoudre les problèmes, notamment le Fast R-CNN [9] et Faster R-CNN [24].

Fast Region based CNN

L'objectif des Fast Region-based CNN est de réduire la consommation de temps liée au nombre élevé de modèles nécessaires à l'analyse de toutes les propositions régionales.

Un CNN principal est utilisé avec plusieurs couches de convolution qui prend l'image entière en entrée au lieu d'utiliser un CNN pour chaque proposition de région

(R-CNN). Les **régions d'intérêt** (Regions of Interest RoI) sont détectées à l'aide de l'algorithme de recherche sélective appliquée sur les grille de caractéristiques « feature maps » produites. Formellement, la taille des feature maps est réduite à l'aide d'une couche RoI-pooling pour obtenir une région d'intérêts valide avec une hauteur et une largeur fixes comme des hyperparamètres. Chaque couche RoI alimente des couches entièrement connectées (couche de réseaux de neurones classique), créant ainsi un vecteur de caractéristiques. Le vecteur est utilisé pour prédire l'objet observé à l'aide d'un classificateur SOFTMAX qui a été choisis en revanche du choix de la fonction SVM des R-CNN, et pour adapter la localisation des boîtes de délimitation avec un régresseur linéaire.

On définit les termes précision et rappel des algorithmes d'apprentissage comme étant :

La précision : mesure la précision des prévisions. c'est-à-dire que le pourcentage des prédictions est correct.

Le rappel : mesure à quel point l'algorithme trouve tous les points positifs et correcte. Par exemple, nous pouvons trouver 80% des cas positifs possibles dans les K principales prédictions.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$TP = TruePositive$$

$$TN = TrueNegative$$

$$FP = FalsePositive$$

$$FN = FalseNegative$$

Les meilleurs modèles des Fast R-CNN ont atteint des scores mAP (mean Average Precision) de 70,0% pour l'ensemble de données du test PASCAL 2007, 68,8% pour l'ensemble de données du test PASCAL 2010 et 68,4% pour l'ensemble de données du test PASCAL 2012 [7]. Un bon score mAP de détection d'objet doit approcher 100. Le score Average precision calcule la précision moyenne pour une valeur de Recall entre 0 et 1.

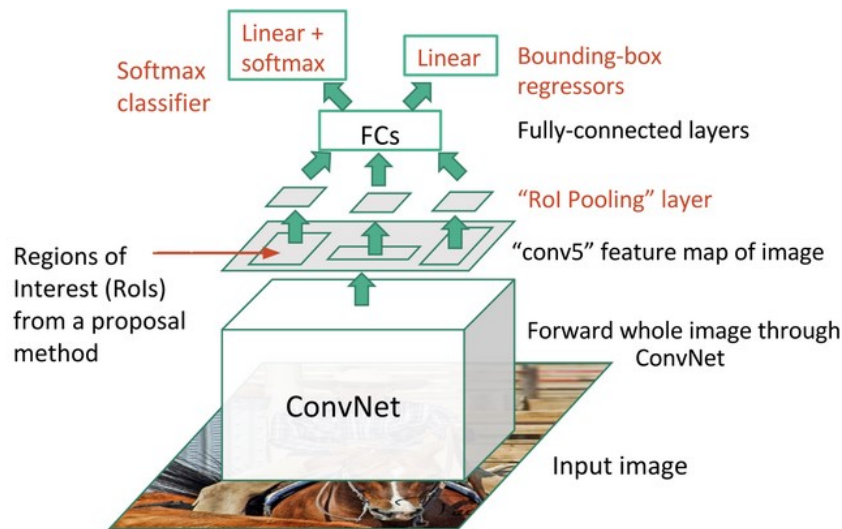


FIGURE 1.8 – Fast R-CNN

L'image d'entrée alimente un réseau de base CNN qui extrait les régions d'intérêt RoI des feature maps. Chaque région est séparée à l'aide d'une couche RoI Pooling et alimente des couches de convolution entièrement connectées. Ce vecteur est utilisé par un classificateur softmax pour détecter l'objet et par un régresseur linéaire pour modifier les coordonnées de la zone limite.

Faster Région based CNN

Les propositions de région RoI détectées à l'aide de la méthode de recherche sélective étaient coûteuses sur le plan informatique. S.Ren et al. [24] ont mis en place un **réseau de propositions de région** (RPN) pour générer directement les RoI, prédire des boîtes de délimitation et détecter des objets. Le Faster R-CNN est une combinaison entre le modèle RPN et le modèle Fast R-CNN.

Un modèle CNN prend en entrée l'image entière et produit les feature-maps. Une fenêtre de taille 3x3 permet de faire défiler toutes les feature-maps et de donner un vecteur de features lié à deux couches de convolution entièrement connectées (le réseau RPN), une pour la régression de boîtes de délimitation et une pour leur classification. Plusieurs propositions de régions sont prédites par les couches entièrement connectées. Le RPN fournit $4*k$ valeurs pour k propositions de régions (coordonnées et longueurs des boîtes de délimitation) et la sortie de la couche de classification des boîtes est de taille $2*k$ (probabilité de la présence de l'objet avec un seuil maintenu à 0.5 habituellement).

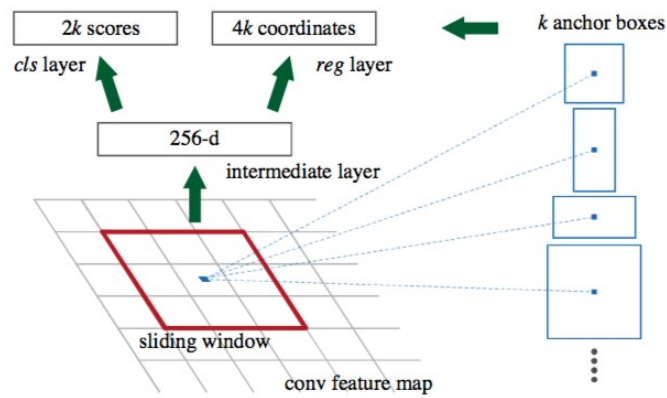


FIGURE 1.9 – Détection des boîtes de délimitation : cls symbolise couche de classification, reg c'est la couche de régression des coordonnées des boites.

Détection des boîtes de délimitation : cls symbolise couche de classification, reg c'est la couche de régression des coordonnées des boites.

Faster R-CNN utilise RPN pour éviter la méthode de recherche sélective, il accélère les processus de formation et de test, et améliore les performances. L'RPN utilise un modèle préformé sur la base de données ImageNet pour la classification. Ensuite, les propositions de régions générées avec l'RPN sont utilisées comme entrée de Fast R-CNN.

Les meilleurs modèle du FasterR-CNN ont obtenu des scores mAP de 78,8% pour l'ensemble de données du test de la compétition PASCAL COV 2007 et de 75,9% pour l'ensemble de données du test de PASCAL COV 2012.

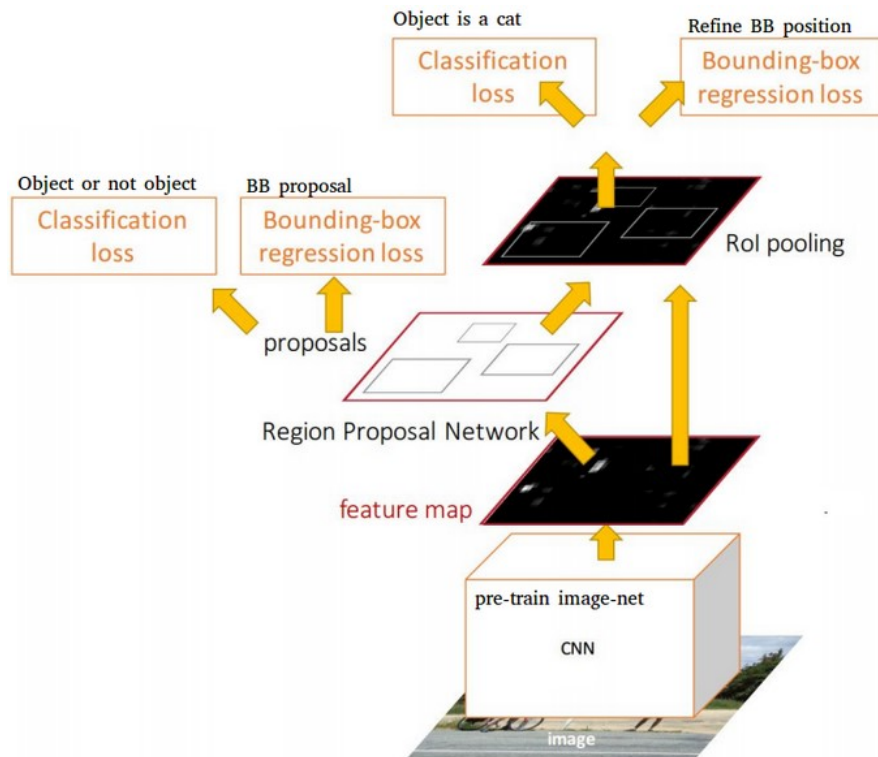


FIGURE 1.10 – Faster R-CNN

L'image d'entrée alimente un modèle CNN pour produire des boîtes d'ancrage en tant que propositions de région avec une confiance (probabilité) de contenir un objet. Un Fast R-CNN est utilisé en prenant en entrée les cartes feature-maps et les propositions de région d'intérêt RoI. Pour chaque boîte, il produit des probabilités de détecter chaque objet et corrige l'emplacement de la boîte.

Region based Fully Convolutional Networks (R-FCN)

Les méthodologies Fast et Faster R-CNN consistent à détecter les propositions de région RoI et à reconnaître un objet dans chaque région. Le réseau entièrement convolutionnel basé sur les proposition de région (R-FCN) publié par J.Dai et al[5] est un modèle avec seulement des couches convolutionnel permettant une rétropropagation complète pour la formation et la déduction. Les auteurs ont fusionné les deux étapes de base en un seul modèle pour prendre en compte simultanément la détection de l'objet (invariant d'emplacement) et sa position.

Un modèle ResNet-101[15] prend l'image initiale en entrée. Les sorties de la dernière couche comportent des cartes appelées **cartes de score sensible à la position** (Position-sensitive feature maps) plutôt que des sorties de variables discrettes, chacune étant spécialisée dans la détection d'une catégorie (classe d'objets) dans un endroit donné. La dernière couche est donc constituée de cartes de score $k*k*(c+1)$ où " k " est

la taille de la carte de score, et " c " le nombre de classes. Toutes ces cartes forment la banque de points. Fondamentalement, l'algorithme crée des ensemble qui peuvent reconnaître une partie d'un objet. Par exemple, pour $k=3$, on peut reconnaître 3×3 parties d'un objet.

En parallèle, un RPN est utilisé pour générer une région d'intérêt (RoI). Et chaque RoI est divisé en bacs qui sont vérifiés par rapport au banque de score. Si un nombre suffisant de ces parties sont activées, alors l'ensemble vote "oui" à l'existence d'un objet détecté.

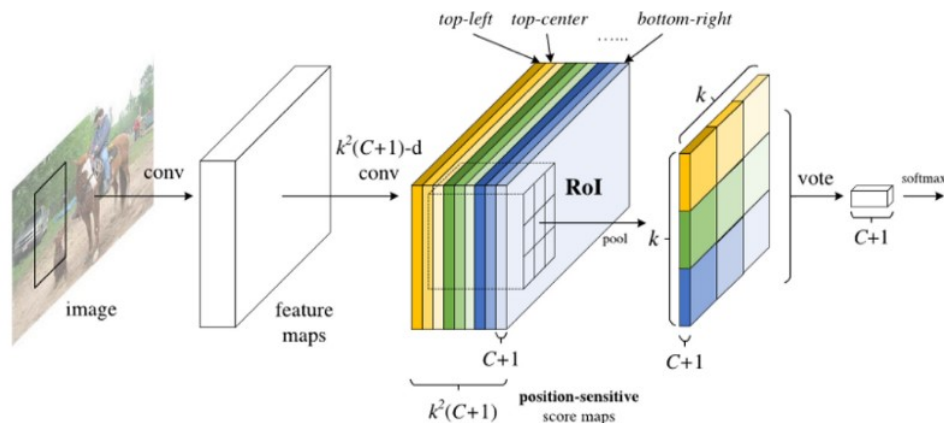


FIGURE 1.11 – Region-Fully Convolutional Networks

L'image d'entrée alimente un modèle ResNet pour produire des feature maps. Un modèle RPN détecte la région d'intérêt avec la probabilité d'existence de l'objet.

Les meilleurs R-FCNs ont atteint des scores mAP de 83,6% pour l'ensemble de données du test PASCAL 2007 et 82,0% . Les auteurs ont remarqué que le R-FCN est 2,5 à 20 fois plus rapide que son contre-part Faster R-CNN .

Mask Region based CNN(Mask R-CNN)

Une autre extension du modèle Faster R-CNN a été publiée par K.He et al [14] ajoutant une branche parallèle à la détection des boîtes de délimitation afin de prédire le masque de l'objet. Le masque est la segmentation par pixel du corps humain (ou d'autre objet en général) dans une image. Ce modèle surpasse l'état de l'art dans les quatre défis COCO : la segmentation des instances, la détection des boîtes de délimitation, la détection des objets et la détection des points clés du corps humain.

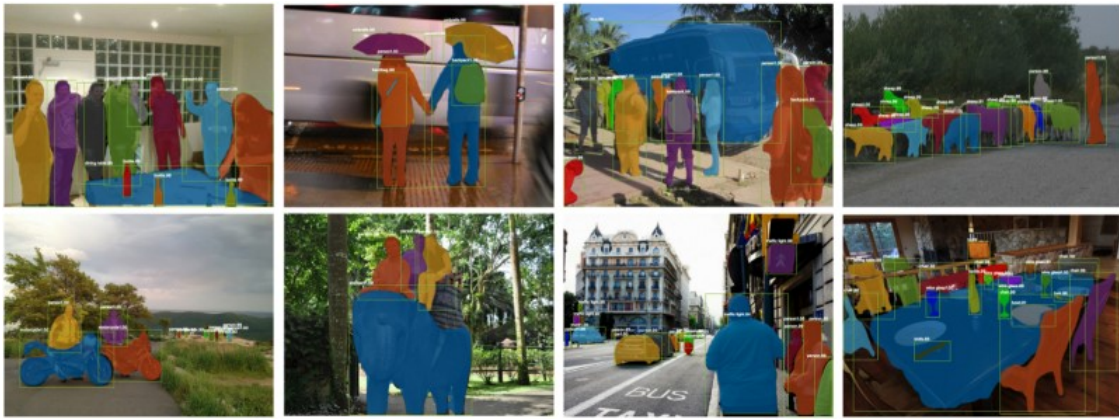


FIGURE 1.12 – Segmentation par Mask R-CNN

Exemples d'application du masque R-CNN sur l'ensemble de données de test COCO. Le modèle détecte chaque objet d'une image, sa localisation et sa segmentation précise par pixel.

Le Mask R-CNN utilise le pipeline de Faster R-CNN et utilise les Region Proposal Network (RPN) pour générer des propositions de boîtes de délimitation et produit en même temps pour chaque Région d'intérêt (RoI) une sortie supplémentaire de classification au niveau des pixels pour la segmentation d'objet.

Le modèle prend une image en entrée et alimente un réseau ResNet de 101 couches et détecte les RoI. Ensuite, une branche du réseau est reliée à une couche entièrement connectée (Fully connected layer) pour calculer les coordonnées des boîtes de délimitation et les probabilités associées à la présence d'objets. L'autre branche est liée à deux couches convolutionnelles, la dernière calcule le masque de l'objet détecté. Mask R-CNN permet également d'obtenir des résultats compétitifs dans l'estimation de la pose humaine comme indiqué dans [14], l'astuce consiste à considérer chaque point clé du corps humain comme une seule classe pour une tâche de classification achevée par le Mask R-CNN.

Le Mask R-CNN a atteint des scores mAP de 62,3% et 39,8% pour la métrique officielle sur l'ensemble de données du COCO de 2016.

1.2.3 L'approche des réseaux de neurones entièrement connectés

Le réseau FCN (Fully Convolutional Network) [19] original apprend un mappage de pixels en pixels, sans extraire les propositions de la région. Son idée principale est de faire en sorte que la CNN classique prenne en entrée des images de taille arbitraire. La restriction imposée aux CNN d'accepter et de produire des étiquettes uniquement pour des entrées de taille spécifique provient des couches entièrement connectées qui

sont fixes et qui ont été remplacées par des couches convolutionnelle et les couche de type Pooling dans les FCN leur donnant la possibilité de faire des prédictions sur des entrées de taille arbitraire, avec des prédictions de sortie dense (sortie à dimensions augmentées). Cependant, ces carte de caractéristique générées sont généralement de faible résolution en raison de convolutions et les opération Pooling multiples qui entraînent des limites de segmentation floues.

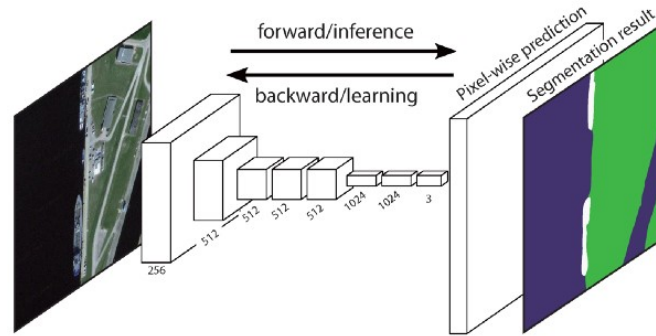


FIGURE 1.13 – Fully Convolutional Neural Networks

Diverses approches plus avancées fondées sur la FCN ont été proposées pour résoudre ce problème, notamment DeepLab-CRF[3] et son opération de convolution dilatée qui seront examinées plus en détail dans notre travail expérimental.

1.3 L'état de l'art de l'estimation de pose humaine

1.3.1 L'estimation de la pose 3D à partir des localisations des points clés

Dans ce type d'algorithme, c'est l'estimation de la pose 2D suivie d'une fonction Hypothèse ou d'un réseau en cascade pour prédire les informations de profondeur à partir de la localisation des points clés :

Les méthodes de ce type prédisant les coordonnées absolues peuvent aussi estimer une pose relative à la racine du corps (le torse) puis calculer la translation via une tâche d'optimisation secondaire, ou estimer les joints dans un système de coordonnées centré par la caméra [27]. Dans le cadre de la localisation de points clés en 2D, certaines méthodes prédisent un ensemble de cartes de confiance, chaque carte représentant une partie particulière du squelette de la pose humaine (image-b de la figure suivante), et un second ensemble de champs d'affinité des membres (Part Affinity Fields, image-c de la figure suivante) qui représente le degré d'association entre les points clés détectés [2].

En utilisant les cartes de confiance des points clés, des graphes-bipartite sont formés entre les paires de points clés (comme le montre l'image-d ci-dessous).

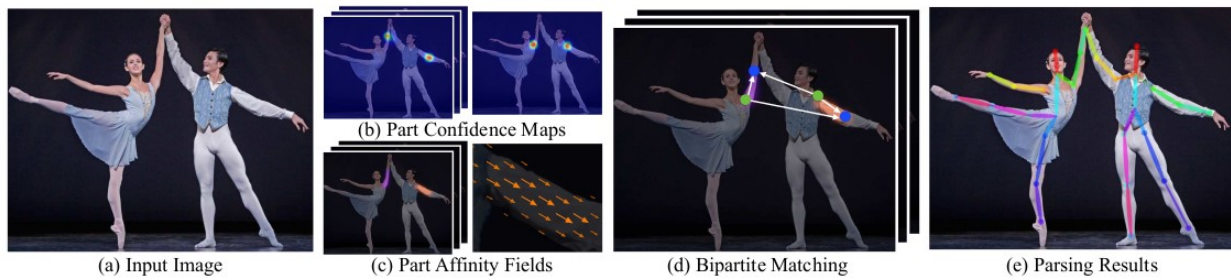


FIGURE 1.14 – Les champs d'affinité des partie dans l'estimation de pose humaine

Les auteurs de [8] ont proposé un réseau de neurones de transformation spatial symétriques pour extraire une région de haute qualité pour une seule personne d'une boîte de délimitation inexacte avec un estimateur de pose pour une seule personne (SPPE) pour estimer le squelette humain pour cette personne. Un réseau de dé-transformation spatial est utilisé pour remapper la pose humaine estimée vers le système de coordonnées d'image original, en plus d'une technique de la non-suppression maximale (Non Maximal Suppression) appliqué à la pose paramétrique qui est utilisée pour traiter la question des déductions de pose redondantes.

L'étape suivante consiste à estimer la pose 3D en trouvant des fonctions d'hypothèses 2D à 3D. Alors que les approches d'apprentissage profond existantes utilisent une minimisation de l'erreur quadratique moyenne basée sur une distribution gaussienne unimodale ou sur les réseaux multimodaux de densité de mélange [17], dans cet article de W. Bastian et R. Bodo [28], les auteurs proposent de faire une inférence à travers un réseau de reprojection (RepNet) qui apprend une cartographie d'une distribution des positions 2D vers une distribution des positions 3D utilisant une approche contrastée avec une autre partie du réseau qui estime la caméra.

Cependant, l'estimation de la pose 3D à partir de joints 2D est plus difficile car les données de pose 2D contiennent moins d'informations ou la première étape a engendré une réduction d'information globale du contexte de l'images, donc plus d'ambiguïtés.

Un algorithme récent que nous avons déjà présenté dans la section de segmentation d'image ci-dessus, qui est le Mask R-CNN, a également montré des performances significatives dans l'estimation de la pose humaine, Mask R-CNN a été étendu pour l'estimation de la pose humaine 2D par le raisonnement suivant :

Supposons qu'un objet dans notre image puisse appartenir à l'une des classes K . La branche de segmentation de Mask R-CNN sort K masque binaires de taille $m \times m$,

où chaque masque binaire représente tous les objets appartenant à cette classe unique. Les points clés du corps humain appartenant à chaque personne dans l'image peuvent être extraits en modélisant chaque type de point clé comme une classe distincte (épaule, hanche, poignet...) et en traitant ceci comme un problème de segmentation. Parallèlement, l'algorithme de détection d'objet peut être formé pour identifier l'emplacement des personnes. En combinant les informations de localisation de la personne ainsi que l'ensemble de ses points clés, on obtient un squelette de pose humain pour chaque personne de l'image.

Cette méthode ressemble presque l'approche descendante, mais l'étape de détection des personnes est réalisée en parallèle avec l'étape de détection des points clés. En d'autres termes, l'étape de détection du point clé et l'étape de détection de la personne sont indépendantes l'une de l'autre.

1.3.2 L'estimation directe de pose 3D

Dans cette catégorie, nous formons des réseaux de neurones convolutionnels profonds de prédiction end-to-end pour estimer les poses humaines en 3D directement à partir des images d'entrée. Zhou et al [32] utilisent une représentation clairsemée pour les poses 3D et prédisent la pose 3D avec un algorithme de maximisation de l'espérance (Expectation Maximization). Les poses 2D sont considérées comme une variable cachée dans l'algorithme EM pour supprimer le besoin de données 2D-3D synchronisées. Parket al [22] améliore les CNNs conventionnels en concaténant l'estimation de pose 2D ainsi que les informations sur les positions relatives par rapport à plusieurs articulations. Pavlakos et al [23] utilisent la représentation volumétrique pour représenter les poses 3D et adoptent le réseau de sabliers empilés [21], qui est à l'origine conçu pour l'estimation des poses 2D, pour prédire les cartes de chaleur volumétriques 3D.

Une méthode hybride d'apprentissage par transfert faiblement supervisé a été proposée par X. Zhou et al [31] qui utilise des étiquettes mixtes 2D et 3D. Le sous-réseau d'estimation de pose 2D et le sous-réseau de régression de profondeur 3D partagent les mêmes caractéristiques (feature-maps), de sorte que les étiquettes de pose 3D pour les environnements intérieurs peuvent être transférées sur des images in-the-wild qui n'étaient pas prises dans un environnement de laboratoire. L'approche directe bénéficie de la richesse de l'information contenue dans les images, comme l'orientation avant-arrière des membres. Cependant, elle sera également affectée par un certain nombre de facteurs tels que le fond, l'éclairage, les vêtements, etc. Un réseau formé

sur un ensemble de données ne peut pas être bien généralisé aux autres ensembles de données avec des environnements différents, par exemple de l'environnement intérieur (laboratoire) et extérieur (image in-the-wild).

Dans le travail de X. Bin et al [5], la structure du réseau est assez simple et consiste en un ResNet et quelques couches de déconvolution à la fin. Mais au lieu d'utiliser le suréchantillonnage (Upsampling) comme dans le réseau de modèle sablier (Stacked Hourglass Network) dans [21] pour augmenter la résolution de la carte de caractéristiques, cette méthode combine des blocs de paramètres convolutifs comme couches déconvolutionnelles.

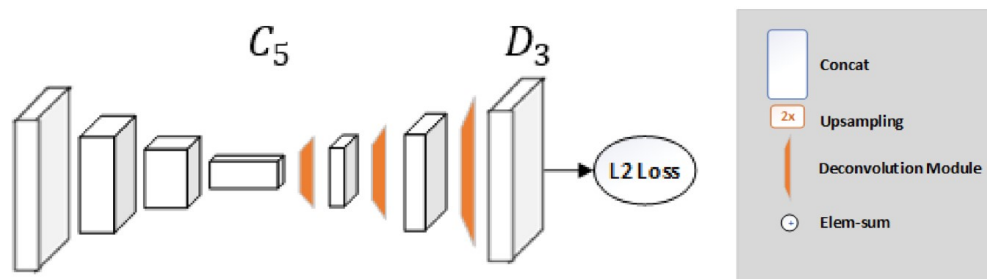


FIGURE 1.15 – Fully Convolutional Neural Networks

L'erreur quadratique moyenne (EQM) est utilisée comme fonction de coût entre les cartes thermiques (heatmaps) prévues et les cartes thermiques ciblées. La carte thermique ciblée pour le joint k est générée en appliquant un modèle de fonction Gaussienne 2D centré sur la position d'un point clés d'indice k avec un écart-type de 1 pixel.

Le modèle HRNet (High-Resolution Network) proposé par [29] a obtenu des résultats assez précise dans l'ensemble de base de données de la compétition COCO. La plupart des méthodes précédentes font varier la résolution des cartes de caractéristique (feature-maps) d'une représentation élevée à une représentation faible à une représentation élevée au finale. HRNet maintient une représentation à haute résolution tout au long du processus. L'architecture part d'un sous-réseau à haute résolution comme première étape, et ajoute progressivement des sous-réseaux à haute et basse résolution un par un pour former plus d'étapes et connecter les sous-réseaux multirésolution en parallèle. Les cartes thermiques sont régressées à l'aide d'une fonction de coût EQM.

1.4 L'algorithme d'estimation de pose dense "DensePose"

Le laboratoire FAIR (Facebook's AI research) a publié l'un des algorithmes les plus efficaces en termes de temps d'inférence et de précision de pose humaine, appelé DensePose [12] développé dans le cadre d'une plateforme de développement de l'apprentissage profond appelé Detectron. L'algorithme utilise le logiciel Pytorch comme son back-end qui intègre les outils d'inférence, de formation et de modélisation pour les réseaux de neurones. Le réseau DensePose a trois canaux de sortie, formés pour fournir une affectation des pixels à des parties du corps humain, et des coordonnées U-V. L'architecture résultante est effectivement aussi rapide que MaskRCNN. Ainsi, l'algorithme établit des correspondances denses d'une image 2D à un modèle 3D constituant une représentation en surface du corps humain.

L'entraînement du réseau a incorporé un modèle du corps humain appelé SMPL [?] représenté dans la figure suivante :



FIGURE 1.16 – Le modèle SMPL du corps humain

Le DensePose a été entraîné sur un ensemble de base de données appelé DensePose-COCO, fourni par les mêmes auteurs. Le signal de supervision de l'entraînement des CNN a été formé par l'annotation des images en utilisant le modèle paramétrique SMPL qui visualise le corps humain comme un nuage de points spatiaux. Chaque point du corps humain se voit attribuer une coordonnée pixel s'il est visible dans l'image pendant la phase d'annotation.

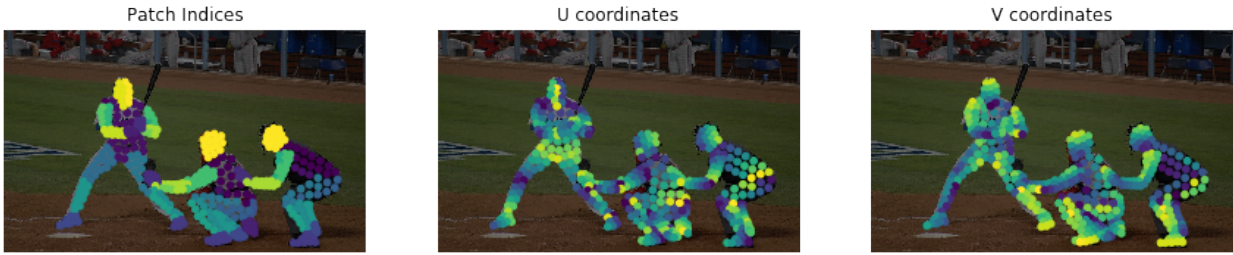


FIGURE 1.17 – Le signal de supervision d’entraînement de DensePose

Chapitre 2

L'estimation de pose humaine en utilisant les réseaux de neurones convolutionnels

Introduction

Ce chapitre présente une brève explication sur l'architecture des réseaux de neurones convolutionnels, ainsi qu'une analyse détaillée du pipeline de l'algorithme DensePose. Chaque sous-architecture de l'algorithme est illustrée dans une section autonome unique.

2.1 Les réseaux de neurones convolutionnels

Les réseaux de neurones convolutionnels (CNNs/ConvNets) sont une extension des réseaux de neurones artificiels ANN ordinaires également appelés Multi-Layer Perceptrons (MLP). Ils sont composés de neurones inspirés par les synapses et les axones des neurones du cerveau humain. Chaque neurone reçoit quelques entrées, effectue un produit de points entre l'entrée et ses paramètres précédemment calculés dans la phase d'apprentissage ou "entraînement" (des poids) et suit ce le résultat de ce produit par une éventuelle fonction de non-linéarité. Le réseau exprime une seule fonction de score différentiable : des pixels de l'image brute d'un côté aux scores des classes de l'autre.

Afin d'apprendre les paramètres appropriés du neurone, une fonction de perte est ajoutée (par exemple SVM/Softmax) sur la dernière couche (généralement une couche entièrement connectée FC) avec les mêmes bases d'apprentissage que celles qui s'appliquent aux MLP.

Les architectures CNN supposent explicitement que les entrées sont des images, ce qui les permet d'encoder certaines propriétés dans l'architecture, et rend la fonction d'inférence dans l'étape d'inférence direct (forward Pass) plus efficace à mettre en œuvre et réduit considérablement le nombre de paramètres dans le réseau.

2.1.1 Le modèle de l'architecture des CNN

Les réseaux de neurones reçoivent une entrée (un seul vecteur), et la transforment à travers une série de couches cachées (Hidden Layers). Chaque couche cachée est constituée d'un ensemble de neurones, où chaque neurone est entièrement connecté à tous les neurones de la couche précédente, indépendamment des neurones de la même couche. La dernière couche entièrement connectée est appelée "couche de sortie" et dans les cas de classification, elle représente les scores des classes. Il en résulte un grand nombre de paramètres d'apprentissage, même dans le cas des images à faible ré-

solution. De plus, la connectivité total engendre un gaspillage de mémoire et le grand nombre de paramètres entraînerait rapidement un sur-apprentissage. Le surapprentissage est un phénomène qui apparaît dans les réseaux de neurones qui sont entraînés sur une base de donnée et qui donne des résultat de précision faible quand ils sont utilisée sur des point de donnée non existant ultérieurement dans le base de donnée initial. En d'autre mot, la généralisation des résultat d'entraînement n'est pas fiable.

Volumes 3D de neurones : les réseaux de neurones convolutionnels profitent du fait que l'entrée est constituée d'images et ils contraignent l'architecture d'une manière plus sensible. En particulier, à la différence des ANN, les couches d'un CNN ont des neurones disposés en 3 dimensions : **largeur, hauteur, profondeur**, où la profondeur d'une seule couche symbolise le nombre de opération de convolutions utilisées, la hauteur et la largeur se rapportent à la taille de la fenêtre convolution.

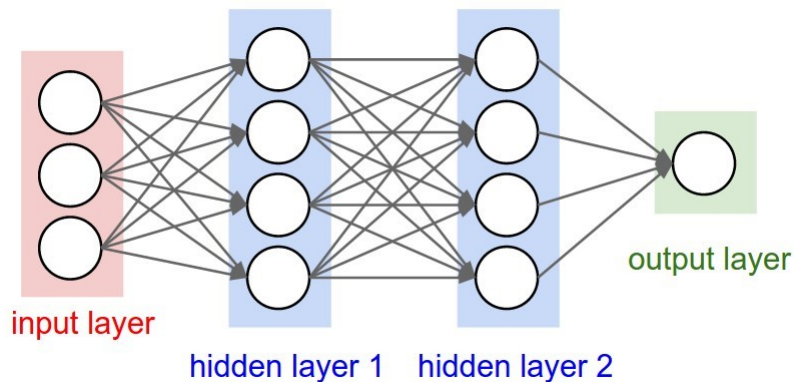


FIGURE 2.1 – Un réseau de neurones ANN

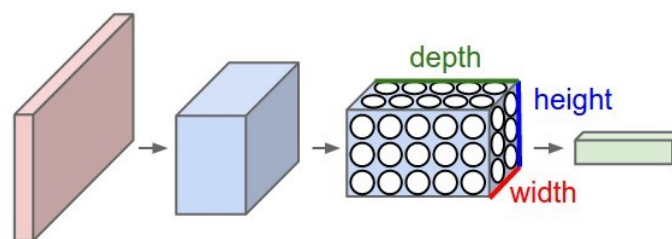


FIGURE 2.2 – Le réseau de neurones CNN à une seul couche

Comme il est illustré ci dessus, en haut : Un réseau de neurones régulier à 3 couches ; en bas : un CNN qui dispose d'un volume de neurones en trois dimensions (largeur, hauteur, profondeur) dans chaque couche . Chaque couche d'un CNN transforme le volume d'entrée 3D en volume de sortie 3D des activations de neurones.

2.1.2 Les types de couche utilisées dans les CNN

Il existe trois principaux types de couches pour construire des architectures CNN : **Couche de convolution CONV**, **couche de Pooling** et **couche entièrement connectée FC**. Exemple d'architecture : Vue d'ensemble. Un simple CNN pour la classification des ensembles de données de la base de données des caractères écrits à la main de CIFAR-10 pourrait avoir l'architecture [INPUT - CONV - RELU - POOL - FC]. En plus de détails :

1. INPUT[32x32x3] contiendra les valeurs brutes des pixels de l'image, dans ce cas une image de largeur 32, hauteur 32, et avec trois canaux couleur R,G,B. La couche CONV calcule la sortie des neurones qui sont connectés aux régions locales dans l'entrée, en calculant un produit point entre leur poids et une petite région dans le volume d'entrée. En elle se fait glissé sur tout l'image par un pas spécifié (stride). Cela peut donner un volume tel que [32x32x12] si nous avons décidé d'utiliser 12 filtres avec un pas de 1.
2. La couche RELU appliquera une fonction d'activation (non linéarité) par élément, telle que : $\max(0,x)$. seuil à zéro. La taille du volume reste inchangée ([32x32x12]).
3. La couche POOL effectuera une opération de sous-échantillonnage le long des dimensions spatiales (largeur, hauteur), ce qui donnera un volume tel que [16x16x12]. Une couche de Pooling est la couche réalisant une opération d'extraire le maximum des neurone existant dans une fenêtre glissante sur toute les le volume des sortie de neurone de la couche précédente par un pas spécifié.
4. FC (c.-à-d. entièrement connecté) calcule les scores de chaque classe, ce qui donne un volume de taille[1x1x10], où chacun des 10 nombres correspond à un score de classe. Comme avec les réseaux de neurones ordinaires et comme son nom l'indique, chaque neurone de cette couche sera connecté à tous les sortie du volume précédent.

Notez que certaines couches contiennent des paramètres (fixés avec l'algorithme de gradient pendant la phase d'entraînement) tels que CONV et FC et d'autres non ayant une fonction implémentée fixe telle que les couches POOL/RELU.

La couche de convolution

La couche CONV est l'élément de base d'un CNN qui effectue la plupart des opérations de calcul de charges lourdes. Les paramètres de la couche CONV sont consti-

tués d'un ensemble de filtres apprentissables. Chaque filtre est petit dans l'espace (largeur et hauteur), mais s'étend sur toute la profondeur du volume d'entrée. Dans le cas de l'entraînement ou de l'inférence, nous calculons la sortie (scores de classe) du réseau en faisant une passe en avant (forward pass), où nous passons l'entrée à travers les couches du réseau en calculant chaque sortie et en l'injectant dans la couche suivante jusqu'à obtenir les scores de classe.

Dans la couche CONV, nous répartissons chaque filtre sur la largeur et la hauteur du volume d'entrée et calculons les produits entre les entrées du filtre et l'entrée à n'importe quelle position. Le résultat de cette dernière opération est une carte d'activation bidimensionnelle qui donne les réponses de ce filtre à chaque position spatiale.

Chaque couche de CONV aurait un ensemble de filtres de convolution (p. ex. 12 filtres), et chacun d'eux produirait une carte d'activation bidimensionnelle distincte. Ces cartes d'activation sont empilées le long de la dimension de profondeur pour produire le volume de sortie appelé "*Feature Maps*".

La forme la plus courante d'une architecture CNN empile quelques couches CONV-RELU, les suit avec des couches POOL, et répète ce motif jusqu'à ce que l'image ait été fusionnée spatialement à une petite taille. À un moment donné, il est courant de passer à des couches entièrement connectées. La dernière couche entièrement connectée contient la sortie, telle que les scores de classe. En d'autres termes, l'architecture CNN la plus courante suit le modèle :

INPUT -> [[CONV -> RELU]*N -> POOL ?]*M -> [FC -> RELU]*K -> FC

Où l'astérisque * indique une répétition. De plus, $N \geq 0$ (et habituellement $N \leq 3$), $M \geq 0$, $K \geq 0$ ($K \leq 3$).

Connectivité locale : Lorsqu'il s'agit d'entrées de grande dimension telles que des images, chaque neurone a une région locale du volume d'entrée. L'étendue spatiale de cette connectivité est un hyperparamètre appelé champ récepteur (réceptive field) du neurone (ce qui correspond à la taille du filtre). L'étendue de la connectivité le long de l'axe de profondeur est toujours égale à la profondeur du volume d'entrée : les connexions sont locales dans l'espace (largeur et hauteur), mais toujours complètes sur toute la profondeur du volume d'entrée.

Arrangement spatial : Trois hyperparamètres contrôlent la taille du volume de sortie : **la profondeur**, **le pas** (stride) et le **zero-padding**. Nous en discuterons plus loin :

Tout d'abord, la profondeur du volume de sortie est un hyperparamètre : elle correspond au nombre de filtres que l'on souhaite utiliser, chacun apprenant à chercher un

certain feature de différent dans l'entrée.

Deuxièmement, nous devons spécifier le pas avec laquelle nous glissons le filtre. Lorsque il est 1, nous déplaçons les filtres un pixel à la fois.

Comme nous le verrons bientôt, il sera parfois pratique de remplir le volume d'entrée avec des zéros autour de la frontière. La taille de ce zero-padding est un hyperparamètre qui nous permettra de contrôler la taille spatiale des volumes de sortie.

Nous pouvons calculer la taille spatiale du volume de sortie en fonction de la taille W du volume d'entrée, de la taille F du champ réceptif des neurones de la couche de CONV, du Pas S avec laquelle ils sont appliqués, et de la quantité de remplissage P de zéro-padding utilisée.

$$O = 1 + (W - F + 2P)/S$$

Remarque : En général, le réglage du zéro de remplissage est $P=(F-1)/2$, lorsque le pas est de 1 assure que le volume d'entrée et le volume de sortie auront la même taille dans l'espace. Il est très courant d'utiliser le zero-padding de cette façon pour avoir les cartes de caractéristiques avec la même résolution spatiale que l'entrée représentant des caractéristiques particulières appelées "bords", ceci et utilisé dans notre travail d'expérimentation au chapitre 4 et sera expliqué plus rigoureusement dans le cas de l'opération de convolution dilatée.

Contraintes sur le pas : Notez encore une fois que les hyperparamètres d'arrangement spatial ont des contraintes mutuelles. Par exemple, lorsque l'entrée a une taille $W=10$, si l'on n'utilise pas de zero-padding $P=0$, et que la taille du filtre est $F = 3$, alors il serait impossible d'utiliser la foulée $S = 2$, puisque $(W - F + 2P)/S + 1 = 4,5$, c'est-à-dire la dimension ne sera pas un entier, indiquant que les neurones ne "s'ajustent" pas proprement et de façon symétrique sur l'entrée. Par conséquent, ce paramètre des hyperparamètres est considéré comme invalide, et une bibliothèque CNN pourrait lancer une exception ou un pad zéro pour le reste des calculs.

La rétropropagation ("Backpropagation") : Dans le processus d'entraînement, nous utilisons une passe en arrière (Backward pass), après chaque passe en avant pour calculer le gradient de la fonction de perte par rapport au poids des neurones. Cette opération nous permet d'optimiser la fonction de perte dans l'entraînement en utilisant l'algorithme du gradient pour aboutir à une configuration des poids de neurones qui mémorise ou « reconnaît » les information encodée dans les image. H. Geoffrey y et al [4] propose la première implémentation de la rétropropagation dans les ANN.

Convolutions dilatées : Un développement récent consiste à introduire un hyperparamètre de plus dans la couche CONV appelée dilatation. Plus précisément, les filtres qui ont des espaces entre chaque cellule, sont appelés convolutions dilatées ou "à trous". Cette particularité est très intéressante car elle nous permet de fusionner l'information spatiale à travers les entrées de manière beaucoup plus agressive avec moins de couches. Par exemple, si on empile deux couches 3x3 CONV l'une sur l'autre, les neurones de la 2ème couche sont une fonction de la fenêtre 5x5 de l'entrée (on dirait que le champ réceptif effectif de ces neurones est 5x5). Si nous utilisons des convolutions dilatées, ce champ réceptif efficace se développerait beaucoup plus rapidement. Nous expliquerons plus en détail la convolution dilatée dans le chapitre de notre modèle.

Couche de Pooling

Il est courant d'insérer périodiquement une couche Pooling entre les couches CONV successives dans une architecture CNN. Sa fonction est de réduire progressivement la taille spatiale de la représentation pour réduire la quantité de paramètres et de calculs dans le réseau, et donc aussi de contrôler le surajustement qui est le processus des réseaux de neurones pour mémoriser les modèles d'entrée qui n'existent que dans l'ensemble de données d'apprentissage et ne pas être capable de bien généraliser sur de nouvelles données.

La couche Pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée et la redimensionne dans l'espace, en utilisant l'opération MAX. Ainsi, le Max pooling prend le maximum des valeurs des neurones dans une petite fenêtre qui est glissée sur le volume d'entrée. La forme la plus courante est une couche de pooling avec des filtres de taille 2x2 appliqués avec un pas de 2 sous-échantillons à chaque tranche de profondeur dans l'entrée par 2 sur la largeur et la longueur. La dimension de profondeur reste inchangée. Plus généralement, la sortie de la couche Pooling est de dimensions :

$$W_2 = 1 + \frac{W_1 - F}{S}$$

$$H_2 = 1 + \frac{H_1 - F}{S}$$

$$D_2 = D_1$$

Il est intéressant de noter qu'il n'y a que deux variations courantes de la couche Pooling max que l'on trouve dans la pratique : Une couche avec $F = 3, S = 2$ (aussi appelée regroupement en chevauchement), et plus communément $F = 2, S = 2$. Les tailles de Pooling avec des champs réceptifs plus grands sont trop destructrices et engendrent une perte d'information importante.

General Pooling. En plus du Max Pooling, les unités de Pooling peuvent également exécuter d'autres fonctions, telles que le "Average Pooling" ou même le " L_2 Norme Pooling".

2.1.3 Les ResNets

Après le développement de la recherche sur les CNN, différentes propositions d'architecture sont apparues, chacune transcendant son passé. Les ResNets sont actuellement de loin les modèles de réseaux de neurones convolutionnels les plus modernes et sont le choix par défaut pour utiliser les CNN dans la pratique avec des réseaux très profonds utilisant des connexions résiduelles. Les blocs résiduels dans les réseaux de neurones convolutionnels ont été utilisés à la place de chaque simple couche de convolution. Ce type particulier de neurones utilise une connexion de saut de la couche précédente à la couche suivante, ce qui entraîne une succession d'ajouts de caractéristiques (features) des couches précédentes.

La raison pour laquelle la connexion par saut est utilisée est due au fait qu'en augmentant la profondeur des réseaux traditionnels, il y a une augmentation initiale de la précision (diminution de l'erreur de validation) qui se stabilise après un certain nombre d'itération de la phase d'apprentissage. Et si la profondeur est encore augmentée, le point dans lequel la précision se stabilise diminuera excessivement, c'est-à-dire que le modèle est trop adapté à l'ensemble de données et ne peut pas généraliser ses résultats (c.à.d Overfitting).

Les réseaux résiduels ResNet [4] proposé par He et al en 2015 résolvent ce problème en ajoutant les raccourcis de connexion. De plus, les connexions de saut permettent un chemin où le gradient ne disparaît pas (le problème du Vanishing Gradient ou le gradient de la fonction de coût s'approche de zéro dans les couches dernières du réseau et entrave les couches premières de s'entraîner sur les nouveaux points de données). La cause de gradient disparaissant est due à la fonction d'activation des neurones comme les

sigmoid ou RELU qui compresse l'espace d'entre sur chaque couche (RELU ignore les valeur négative et sigmoid donne des sortie dans la plage [0,1]). En plus, l'entraînement de ce type de réseaux est facile et efficace car les connexions de saut n'engendre aucun changement dans la rétropropagation.

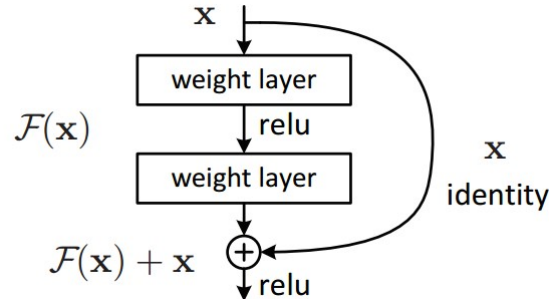


FIGURE 2.3 – Block résiduel de transformation identité

Il existe deux types de connexions résiduelles :

Les raccourcis d'identité (x) peuvent être utilisés directement lorsque l'entrée et la sortie ont les mêmes dimensions :

$$y = F(x, W_i) + x$$

Quand les dimensions changent :

Le raccourci effectue toujours le mappage d'identité, avec des entrées de zéro supplémentaires, complétées par l'augmentation de la dimension.

Le raccourci de projection est utilisé pour faire correspondre la dimension (fait par conv 1x1) en utilisant la formule suivante

$$y = F(x, W_i) + W_s x$$

Le premier cas n'ajoute aucun paramètre supplémentaire, le second les ajoute sous forme de W qui doivent être mise à jour dans la phase d'entraînement .

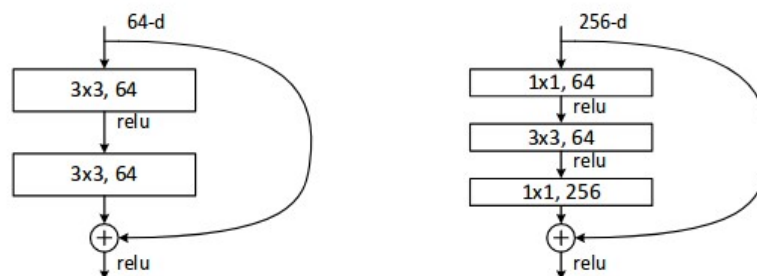


FIGURE 2.4 – Les deux type de bloc résiduel.

A titre de comparaison, nous présentons la précision des CNN usuel (Alexnet ou VGGnet) et des ResNets de 18 et 34 couches. Même si le réseau à 18 couches n'est que le sous-espace dans le réseau à 34 couches, il fonctionne encore mieux. ResNet surpasse de beaucoup les performances de ResNet au cas où le réseau serait plus profond. Dans le tableau suivant, on présente une comparaison de l'erreur de la compétition ImageNet

Le nombre de couche \ l'architecture	CNN	ResNet
18	27.94	27.88
34	28.54	25.03

TABLE 2.1 – Tableau des précision des architecture ResNet et CNN pour les même nombres de couches

L'article des ResNet [15] présente également d'autres études sur les propositions d'architecture en variant le nombre de couche et les type de convolution. Chacun des réseaux présentés dans le tableau ci-dessous a la même première couche (Conv1), ajoute quatre autres couches de Conv (2-5) où nous répétons chaque couche un certain nombre de fois (le nombre de multiplication).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

FIGURE 2.5 – Les différentes configuration des ResNets

Chaque bloc ResNet a une profondeur de 2 couches (utilisé dans les petits réseaux comme ResNet 18, 34) ou de 3 couches (ResNet 50, 101, 152). Il existe différents types de ResNets où la profondeur du réseau est un facteur important. Le choix de la profondeur appropriée dépend de la complexité et de la taille de l'ensemble de base de données. Ce facteur détermine sa précision dans des tâches telles que la classification où ResNet obtient le meilleur résultat. Un exemple de référence CIFAR 10 de ResNet avec différentes profondeurs est présenté ci-dessous :

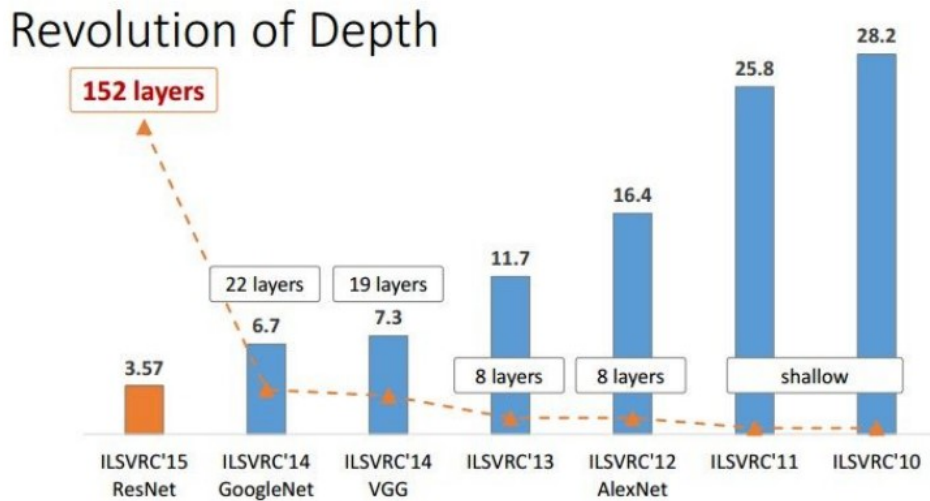


FIGURE 2.6 – L'erreur Top-5 de la des gagnant de CIFAR 10

De plus, les figures suivantes présentent les cinq principales erreurs de la classification ImageNet 2015 :

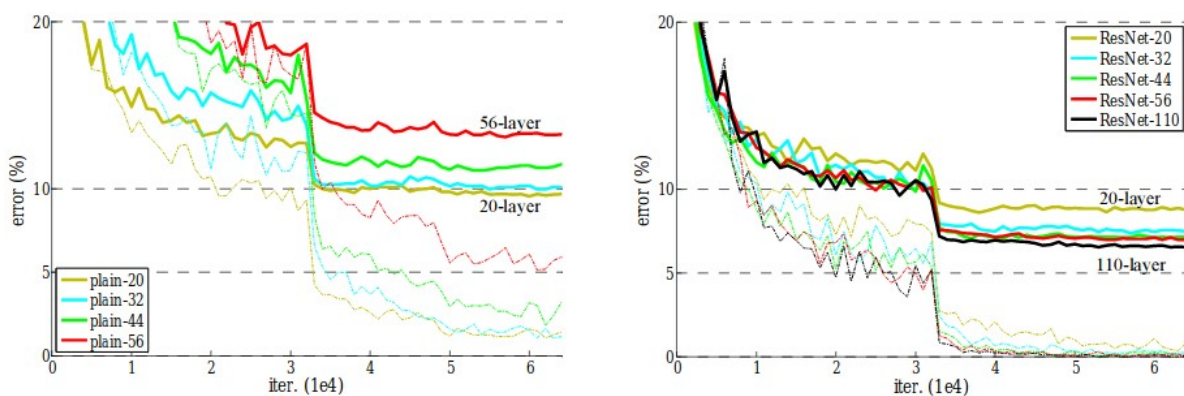


FIGURE 2.7 – Comparaison sur l'effet d'augmentation de la profondeur sur l'erreur des CNN normal et les ResNets

Dans la figure ci-dessus, nous remarquons que l'augmentation de la profondeur d'un réseau ResNet diminue considérablement l'erreur de classification par rapport au réseau de même nombre de couches dans un réseau convolutionnel simple. Cepen-

nant, un nombre élevé de couches est très découragé pour deux raisons principales :

1. Le réseau surajusterait les données et ne généraliserait pas bien aux variantes et aux nouvelles poses humaines ou dans des environnements occlusifs. Ainsi, des facteurs tels que la taille de l'ensemble de données et les spécifications telles que les données d'environnement de laboratoire ou les données annotées in-the-wild (complexité de donnée) sont les critères pour décider du nombre de couches appropriées pour l'application.
2. La grande taille des paramètres du réseau à apprendre pendant l'entraînement ainsi que le critère du temps d'entraînement /inférence qui représentent des paramètres essentiels dans l'évaluation du réseau.
3. Et enfin, la précision du réseau cesse de diminuer après une certaine profondeur.

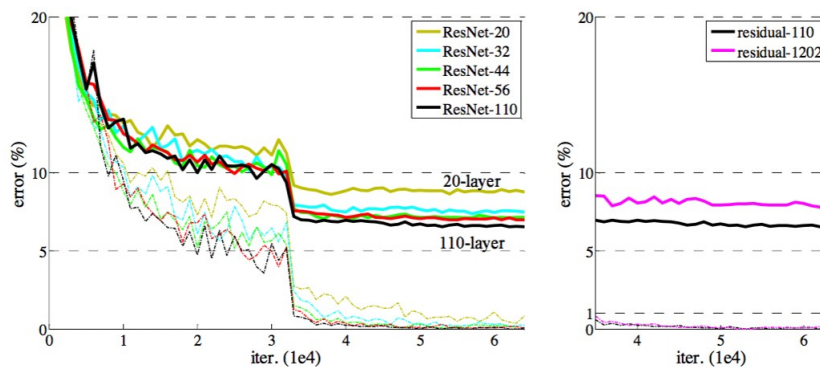


FIGURE 2.8 – L'effet d'augmentation de profondeur sur l'erreur des ResNets

A ce stade, l'estimation de la pose reste une boîte noire avec des paramètres internes qui n'ont pas encore été présentés. Dans ce qui suit, nous expliquerons le pipeline et les paramètres d'évaluation utilisés pour la pose humaine et illustrerons les ensembles de Dataset utilisés récemment par les chercheurs.

2.2 Le pipeline d'estimation de la pose humaine

2.2.1 L'extraction des *features*

L'extraction des caractéristiques dans le domaine de Machine Learning se réfère à la création des données dérivées à partir de données brutes (telles qu'une image ou des vidéos), qui peuvent être utilisées comme entrées dans un algorithme d'apprentissage. Les caractéristiques peuvent être explicites ou implicites. Les fonctions explicites incluent des fonctions conventionnelles basées sur la vision par ordinateur comme l'histogramme des gradients orientés (HoG) [6] et la transformation des caractéristiques in-

variantes à l'échelle (SIFT) [20]. Ces caractéristiques sont calculées explicitement avant d'alimenter l'algorithme d'apprentissage suivant.

Les caractéristiques implicites se réfèrent à des feature maps basées sur l'apprentissage profond comme les sorties de réseaux de neurones convolutionnel profonds complexes. Ces cartes ne sont jamais créées explicitement, mais font partie d'un pipeline complet formé de bout en bout. Dans l'estimation de la pose humaine, les algorithmes basés sur l'apprentissage tels que DensePose utilisent une architecture particulière construite sur des réseaux CNN normaux, appelés Feature Pyramid Networks (FPN) [18].

2.2.2 Les Feature Pyramid Networks

La détection d'objets à différentes échelles est un défi, en particulier pour les petits objets : utiliser une pyramide de la même image à différentes échelles pour détecter des objets peut être une solution valable. Cependant, le traitement d'images à échelles multiples prend beaucoup de temps et la demande de mémoire est trop élevée pour être formée de bout en bout simultanément. Ainsi, les chercheurs qui ont développé les FPN ont proposé de créer une pyramide de caractéristiques et de les utiliser pour la détection d'objets. Cependant, les cartes d'entités plus proches de la couche d'image sont composées de structures de bas niveau qui ne sont pas efficaces pour une détection précise des objets.

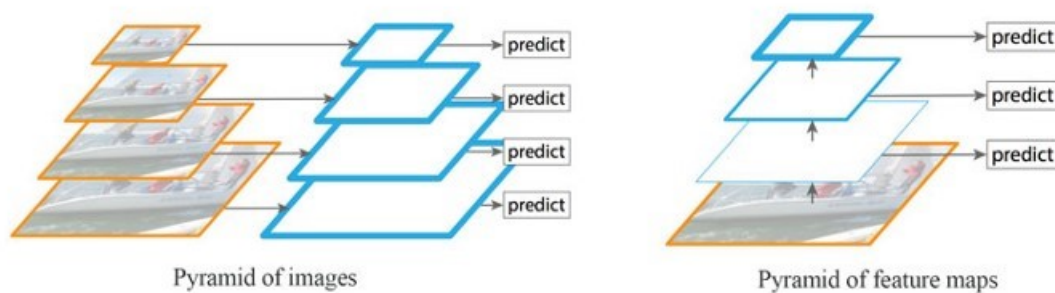


FIGURE 2.9 – L'architecture pyramid proposée dans les FPN

Feature Pyramid Network (FPN) est un extracteur de caractéristiques conçu pour ce concept de pyramide avec précision et rapidité en tête. Il s'agit d'une carte multi-éléments (**Multiscale feature maps**) avec des informations de meilleure qualité que la pyramide de caractéristiques régulière pour la détection d'objets.

Flux de données. Le FPN se compose d'une voie ascendante et d'une voie descendante. La voie ascendante est le réseau convolutionnel habituel pour l'extraction

d'entités. Au fur et à mesure que nous montons, la résolution spatiale diminue. Avec plus de structures de haut niveau détectées, la valeur sémantique de chaque couche augmente.

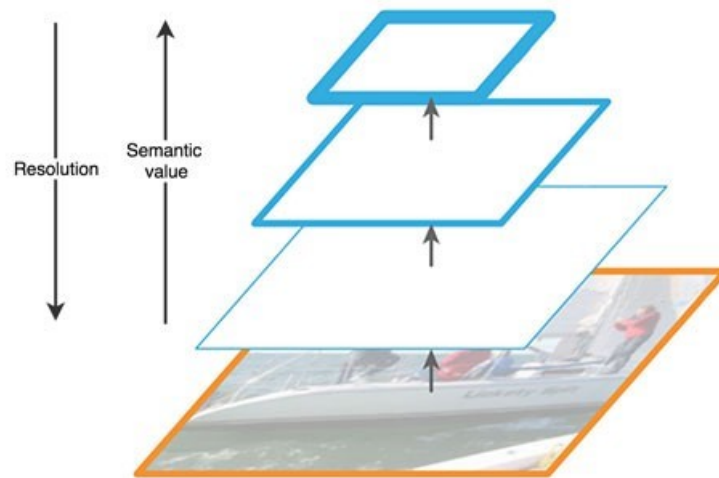


FIGURE 2.10 – Résolution et valeur sémantique à travers les couches des CNN

En général, les couches inférieures ne sont pas sélectionnées pour la détection d'objets. Ils sont en haute résolution mais la valeur sémantique n'est pas assez élevée pour justifier leur utilisation car le ralentissement de la vitesse est significatif. Le FPN fournit une voie descendante pour construire des couches à plus haute résolution à partir d'une couche riche en sémantique. Alors que les couches reconstruites sont sémantiquement fortes mais les emplacements des objets ne sont pas précis après tout le downsampling (échantillonnage) et le upsampling (suréchantillonnage). Nous ajoutons des connexions latérales entre les couches reconstruites et les cartes de caractéristiques correspondantes pour aider le détecteur à prédire les meilleurs emplacements. Il sert également de connexion de saut pour faciliter l'entraînement (semblable à ce que fait ResNet).

Partie ascendante (Bottom-up Pathway)

La partie ascendant utilise ResNet pour construire le passe ascendant. Elle se compose de plusieurs modules de convolution ($CONV_i$ pour i est égale 1 à 5) qui ont chacun plusieurs couches de convolution en fonction de la version ResNet (50, 101...). Au fur et à mesure que l'on monte, la dimension spatiale est réduite de moitié (c'est-à-dire le divisant sur le pas). La sortie de chaque module de convolution est étiquetée comme C_i et utilisée plus tard dans la voie descendante.

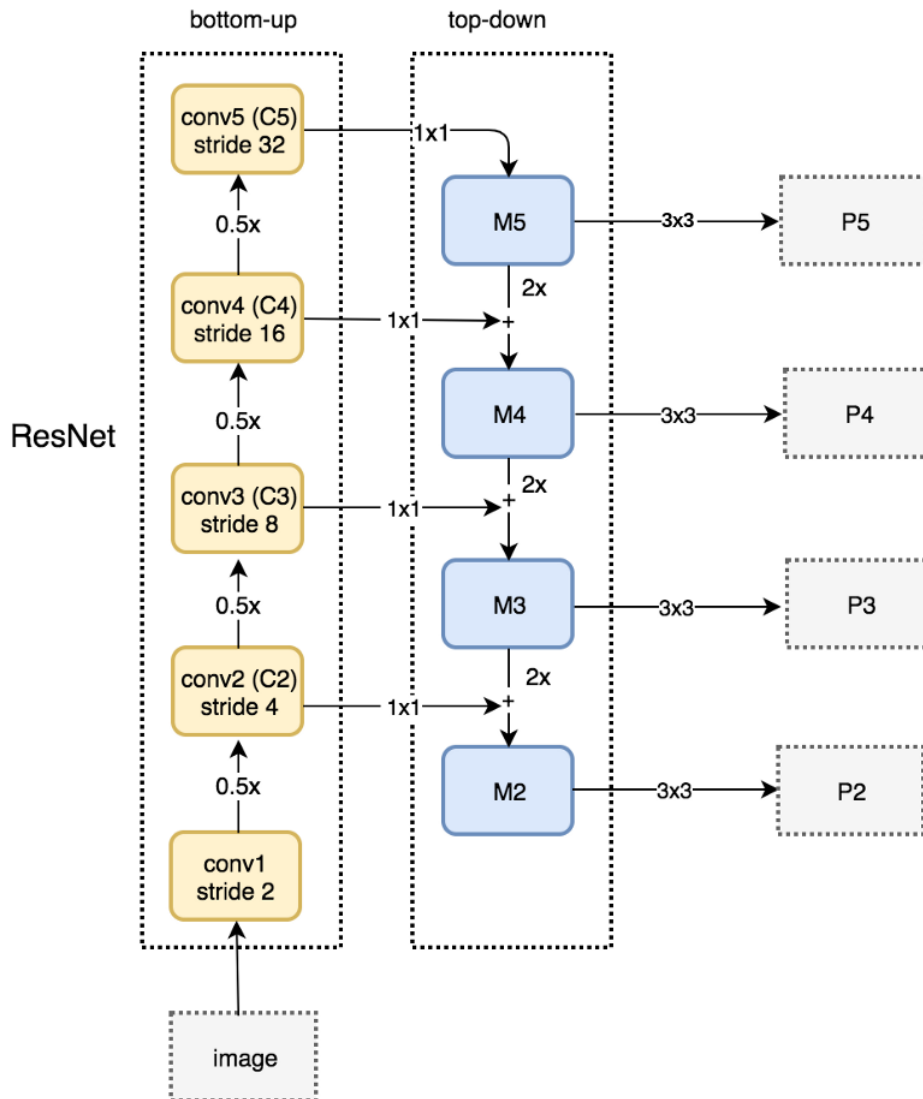


FIGURE 2.11 – Le schéma Datas flow des FPN

La partie descendante (Top down pathway)

Nous appliquons un filtre de convolution 1x1 pour réduire la profondeur du canal C5 à 256-d afin de créer la carte de caractéristiques M5. Ceci devient la première couche de carte de caractéristiques utilisée pour la prédiction d'objets.

Au fur et à mesure que nous descendons le deuxième chemin, nous suréchantillonons la couche précédente de 2 en utilisant le suréchantillonnage des voisins les plus proches (nearest neighbor upsampling) qui interpolent les nouvelles valeurs de nœuds de caractéristiques de pixels entre eux. Nous appliquons à nouveau une convolution de 1x1 aux cartes de caractéristiques correspondantes dans le cheminement ascendant. Ensuite, nous les ajoutons élément par élément et aboutir au carte Pi. Nous appliquons une convolution de 3x3 à toutes les couches fusionnées. Ce filtre réduit l'effet d'échantillonnage lors de la fusion avec la couche suréchantillonnée.

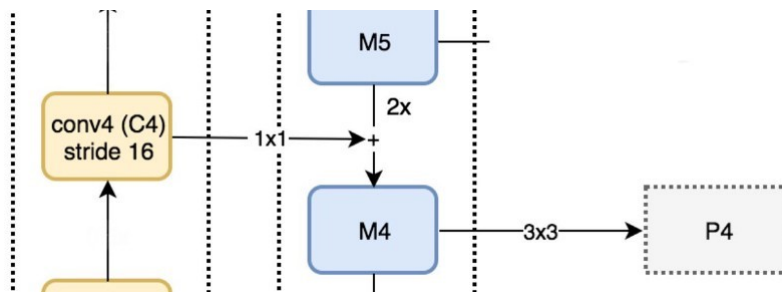


FIGURE 2.12 – La formation des *feature map* dans la partie descendante

Nous répétons le même processus pour P3 et P2. Cependant, nous nous arrêtons à P2 parce que la dimension spatiale de C1 est trop grande. Et sinon, cela ralentira trop le processus. Toutes les cartes pyramidales (P5, P4, P3 et P2) ont 256 canaux de sortie.

2.2.3 Les réseaux de proposition des régions (RPN)

Dans le réseau de base, le FPN n'est pas un détecteur d'objets en soi. Il s'agit d'un extracteur de caractéristiques qui fonctionne avec les détecteurs d'objets. Par exemple, nous extrayons plusieurs couches de cartes d'entités avec FPN et les introduisons dans un RPN (un détecteur d'objets utilisant des convolutions et des ancres) pour détecter des objets. L'RPN applique des convolutions de 3x3 sur les cartes de caractéristiques, suivies d'une convolution séparée de 1x1 pour la régression des RoI. Ces couches convolutionnelles 3x3 et 1x1 sont appelées tête RPN. La même tête est appliquée à toutes les cartes d'entités.

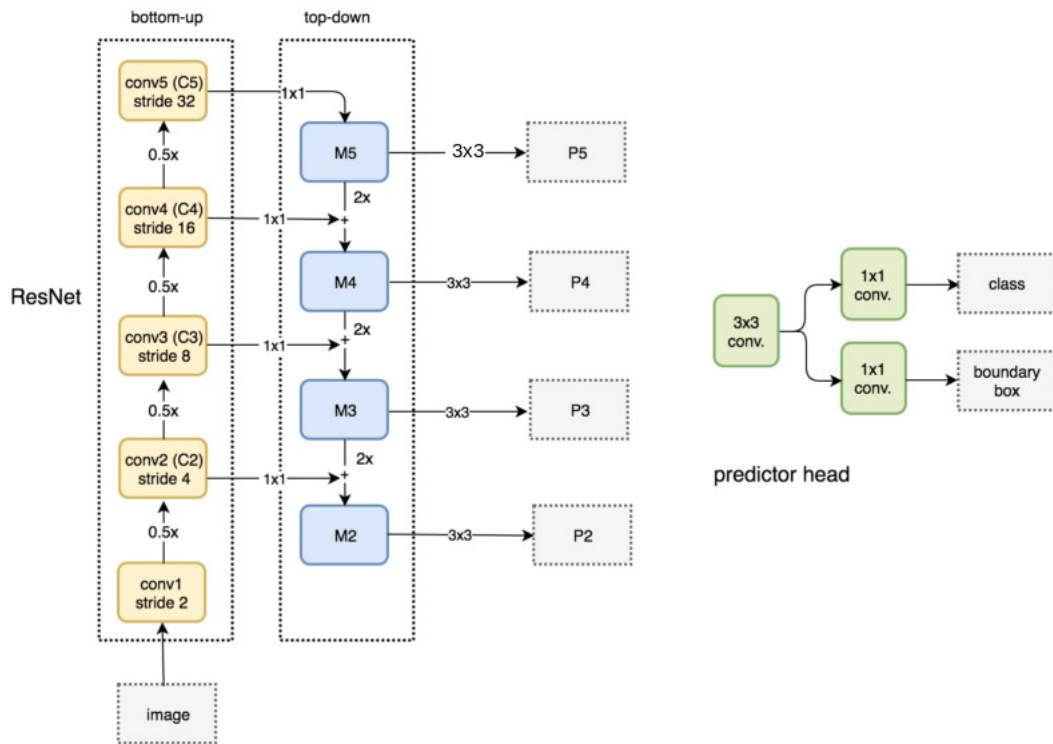


FIGURE 2.13 – FPN + Region Proposal Networks

La sortie des RPN peut être de différentes résolutions et formes, et afin de fixer la même taille de RoI, l'opération de RoI Pooling est utilisée dans le processus. **RoI Pooling**. Le Pooling des régions d'intérêt est une opération largement utilisée dans les tâches de détection d'objets à l'aide de réseaux de neurones convolutionnel. Par exemple, pour détecter plusieurs voitures et piétons dans une seule image. Son but est d'effectuer une Pooling maximale sur les entrées de tailles non uniformes afin d'obtenir des feature maps de taille fixe (ex. 7x7). Dans la tâche de détection d'objet, nous cherchons habituellement à sortir des boîtes de délimitation autour de n'importe quel objet à partir d'un ensemble précédemment spécifié de catégories et d'attribuer une classe à chacun d'eux. Cette tâche est plus difficile que les tâches de classification comme celles de MNIST ou de CIFAR. Sur chaque image ou vidéo, il peut y avoir plusieurs objets, certains se chevauchant, d'autres mal visibles ou occlus. De plus, pour un tel algorithme, la performance peut être une question clé. En particulier pour les voitures à conduite autonome, nous devons traiter des dizaines de frames per second fps. La couche prend deux entrées :

Une carte des donnée d'activation (feature maps) de taille fixe obtenue à partir de la phase d'extraction de caractéristiques à l'aide d'un réseau convolutionnel profond avec plusieurs convolutions et des couches de Pooling maximales.

Une matrice de $4*k$ représentant une liste de régions d'intérêt des RPN, où k est un nombre de RoI. La première colonne représente l'index de l'image et les quatre autres sont les coordonnées des coins supérieur gauche et inférieur droit de la région ou, dans d'autres cas, les coordonnées supérieures gauche et la longueur et la largeur de la boîte.

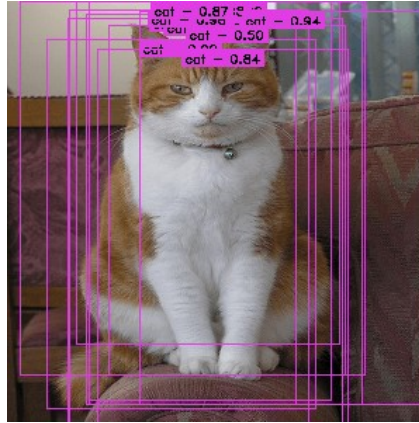


FIGURE 2.14 – Les proposition de boites de délimitation

Pour chaque région d'intérêt de la liste d'entrée, l'opération de Pooling du RoI prend une section de la carte des caractéristiques d'entrée qui lui correspond et la met à l'échelle selon une taille prédéfinie (p. ex., 7×7). La mise à l'échelle se fait par :

- Diviser la proposition de la région en sections de taille égale (dont le nombre est le même que la dimension de la production).
- Trouver la valeur la plus élevée dans chaque section.
- Copier ces valeurs maximales dans le tampon de sortie.

Le résultat est qu'à partir d'une liste de rectangles de différentes tailles, nous pouvons rapidement obtenir une liste des cartes de caractéristiques correspondantes avec une taille fixe. Notez que la dimension du résultat de la Pooling des RoI ne dépend pas réellement de la taille de la carte des caractéristiques d'entrée ni de la taille des propositions de la région. Elle est déterminée uniquement par le nombre d'articles que nous divisons la proposition en sections. Dans ce contexte, la vitesse de traitement augmente considérablement.

2.2.4 La détection d'objets

Dans l'architecture de détection d'objet, le processus est généralement décomposé en deux étapes :

- Proposition de la région : Avec une image d'entrée, trouvez tous les endroits possibles

où les objets peuvent être localisés. Le résultat de cette étape devrait être une liste de cases délimitant les positions probables des objets. Il s'agit souvent de propositions de régions ou de régions d'intérêt.

- Classement final : pour chaque proposition de région de l'étape précédente, décider si elle appartient à l'une des classes cibles ou à l'arrière-plan.

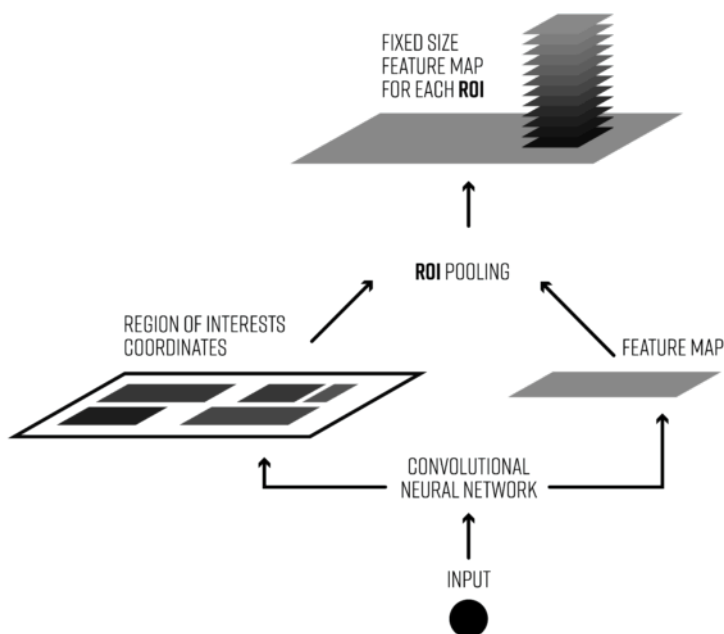


FIGURE 2.15 – Détection d'objet par proposition des régions

Dans la première phase, l'algorithme génère habituellement des propositions de région qui doivent avoir un rappel élevé (pourcentage de propositions positives de toutes les régions de propositions positives qui existent dans l'image et qui contiennent une classe d'objets valide) car si un objet n'est pas détecté pendant la première phase (proposition de région), il n'y a aucun moyen de la classer correctement dans la deuxième phase ou d'estimer la position de l'humain dans cette région s'il en existe un. Et cela est possible grâce aux RPNs ou à des algorithmes spéciaux de proposition de région prêts à l'emploi tels que la recherche sélective générant un très grand nombre de propositions. La plupart d'entre elles seront classées en arrière-plan dans la deuxième phase de l'algorithme de détection ou à l'aide d'RPN. Certains problèmes avec cette dernière architecture sont :

La génération d'un grand nombre de régions d'intérêt peut entraîner des problèmes de performance. Cela rendrait la détection d'objets en temps réel difficile à mettre en œuvre.

C'est sous-optimal en termes de vitesse de traitement.

Impossibilité d'effectuer une formation de bout en bout de l'algorithme, c'est-à-dire que nous ne pouvons pas former tous les composants du système en une seule phase de formation, notamment parce que les systèmes formés de bout en bout sont plus performants que les méthodes de traitement post-hoc. Pour réaliser la tâche des propositions de région à l'aide de réseaux de neurones, l'opération de RoI Pooling des régions d'intérêt est généralement mise en œuvre après les RPN dans le pipeline de l'architecture. Et pour éviter les doublons pour le même objet, certains réseaux utilisent la Suppression Non-Maximale (NMS) par classe. Cette opération regroupe les cases de la tâche de proposition de la région qui se chevauchent fortement pour la même classe et sélectionne uniquement la prévision la plus fiable par un seuil de confiance.

Deuxièmement, pour réaliser la tâche de classification si la zone de délimitation contient une classe d'objets valide (humain), la détection d'objets utilise diverses architectures de réseau telles que Faster R-CNN et YOLO.

Le flux de données Fast-CNN et Faster R-CNN est présenté ci-dessous. Il fonctionne avec une couche de carte de caractéristiques pour créer des RoI. Nous utilisons les RoI et les feature maps pour créer des ensemble de caractéristiques local à intégrer dans le RoI Pooling.

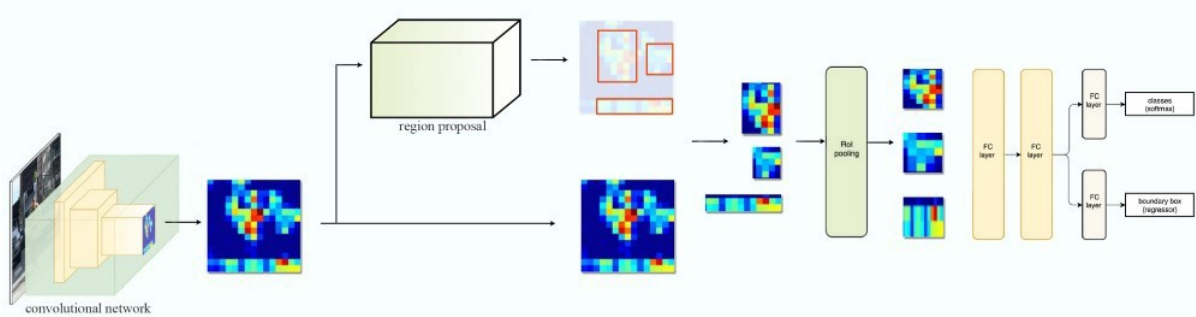


FIGURE 2.16 – Flux des données dans les Fast R-CNN

Dans l'algorithme de DensPose, on utilise plutôt les RPN lorsque l'on génère une pyramide des feature maps. Nous appliquons le RPN pour générer des RoI. En fonction de la taille du RoI et nous sélectionnons la couche des feature maps à l'échelle la plus appropriée pour extraire les correctifs des caractéristiques.

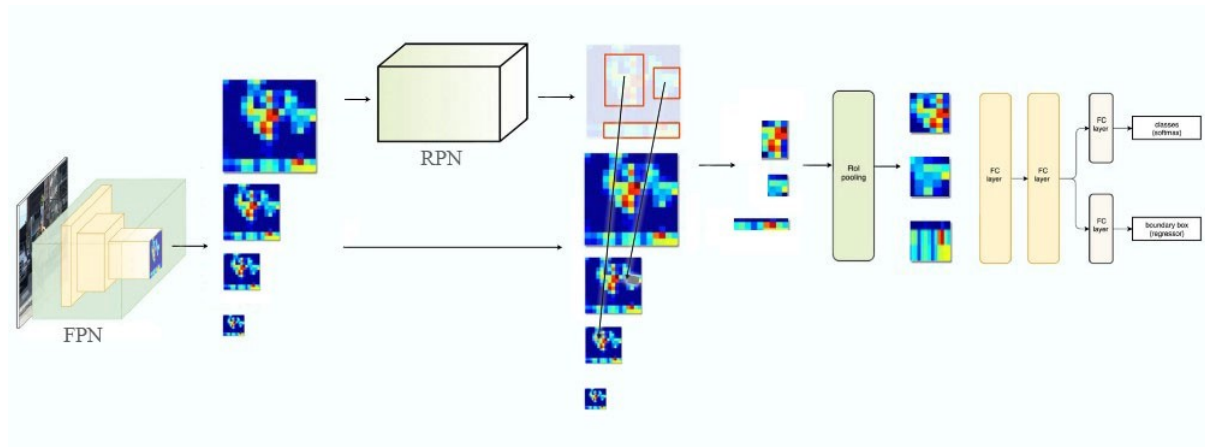


FIGURE 2.17 – Flux des données de Faster R-CNN à base des FPN

La formule pour choisir une carte de caractéristiques précise est basée sur la largeur w et la hauteur h du RoI.

$$k = k_0 + \log_2(\sqrt{wh}/224)$$

Avec k la couche numéros k des carte P du FPN et $k_0=4$. Donc, si $k = 3$, nous sélectionnons P3 comme cartes de caractéristiques. Nous appliquons le Pooling du RoI et envoyons le résultat à la tête de Fast R-CNN (Fast R-CNN et Faster R-CNN ont la même tête) pour terminer la prédiction.

2.2.5 La segmentation sémantique de l'image

Dans la tâche d'estimation de la pose humaine, des algorithmes tels que DensePose utilisent Mask R-CNN pour une segmentation complète du corps humain afin d'obtenir de meilleurs résultats, en ajoutant un masque de sortie séparé sur seulement les pixels qui font partie du corps humain, en plus de l'introduction de l'opération RoIAlign due au fait que les candidats des boîtes de délimitation peuvent être de différentes tailles. RoIAlign est utilisé pour réduire la taille de la caractéristique extraite de sorte qu'ils soient tous de la même taille et effectuent le calcul plus efficacement que RoI Pooling.

RoIAlign.

L'opération RoIAlign améliore considérablement la précision de la segmentation. La sélection des limites des cellules à partir des régions d'intérêt dans les cartes de caractéristiques est habituellement digitalisée et forcée de s'aligner de nouveau sur les limites des cartes d'entrée dans le cas de la Pooling du RoI. Par conséquent, chaque cellule cible peut ne pas avoir la même taille (diagramme en bas à gauche dans l'image suivante). ROI Align ne digitalise pas la limite des cellules (en haut à droite) et fait

en sorte que chaque cellule cible ait la même taille (en bas à droite). Il applique également l'interpolation entre les pixels les plus proches, telle qu'elle est présentée dans la figure de droite, afin de mieux calculer les valeurs de la carte des caractéristiques à l'intérieur de la cellule. Par exemple, en appliquant l'interpolation, la valeur maximale de la caractéristique en haut à gauche passe de 0,8 à 0,88 maintenant.

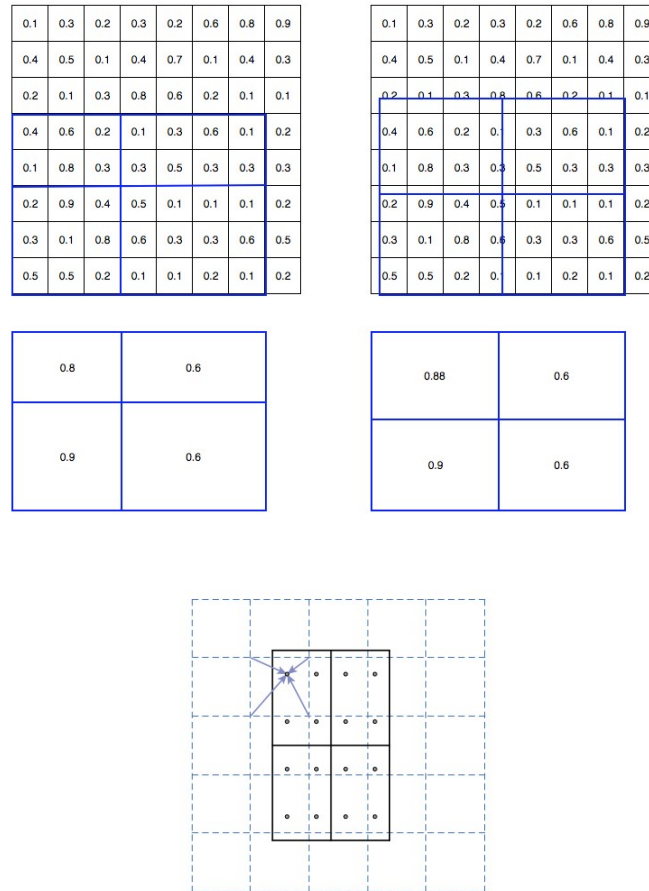


FIGURE 2.18 – L'opération ROI Pooling et ROIAlign

Quant à la tête de segmentation, le Faster R-CNN construit tous les travaux de base pour les extractions de caractéristiques et les propositions de ROI. À première vue, la segmentation de l'image peut nécessiter une analyse plus détaillée pour colorer les segments d'image. Cependant, le travail supplémentaire requis est simplement effectué dans Mask R-CNN après l'opération ROIAlign où nous ajoutons 2 couches de convolution supplémentaires pour construire le masque.

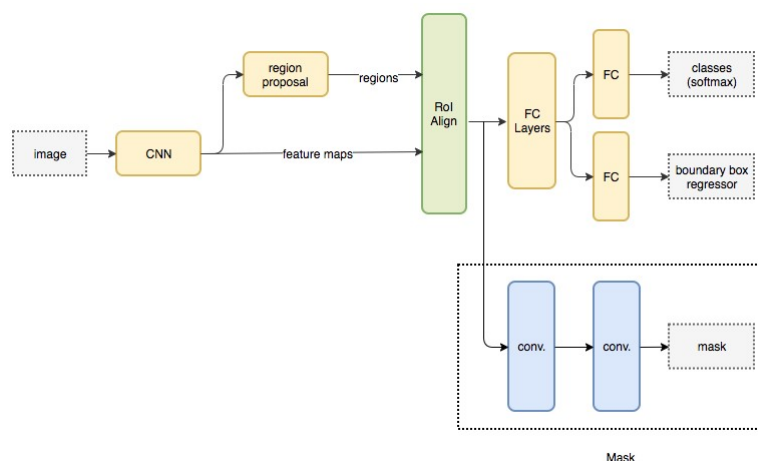


FIGURE 2.19 – L’architecture du Mask R-CNN globale

Dans la figure ci-dessus, le premier réseau convolutionnel est dédié à la tâche d’extraction d’entités comme mentionné précédemment, ce qui donne les prochaines cartes de caractéristiques. Le deuxième réseau convolutionnel est dédié aux propositions de régions pour extraire les régions caractéristiques susceptibles d’inclure un corps humain (objet). Le Pooling des RoI (ou RoIAlign dans le cas de Mask R-CNN) est utilisée pour redimensionner les régions d’intérêt selon des formes similaires. Les derniers couches entièrement connectés sont utilisés pour la prédiction des sorties :

- La boîte de délimitation.
- La classification de l’objet à l’intérieur de la région caractéristique.
- La classification de chaque pixel s’il est inclus dans l’objet ou l’arrière-plan.

2.3 L’estimation 3D de la pose humaine

En plus de la sortie du Mask R-CNN, Densepose incorpore un réseau entièrement convolutionnel séparé est utilisé pour la tâche d’estimation du point clé en plus de la sortie RoIAlign. Et afin d’établir des correspondances 3D entre la surface du corps humain et la partie segmentée de l’image ou de la carte fonctionnelle, l’algorithme DensePose utilise un réseau de neurones entièrement convolutionnel pour la régression de forme dense afin de calculer le paramétrage bidimensionnel, U-V de la surface du corps humain basé sur l’algorithme DenseReg [13].

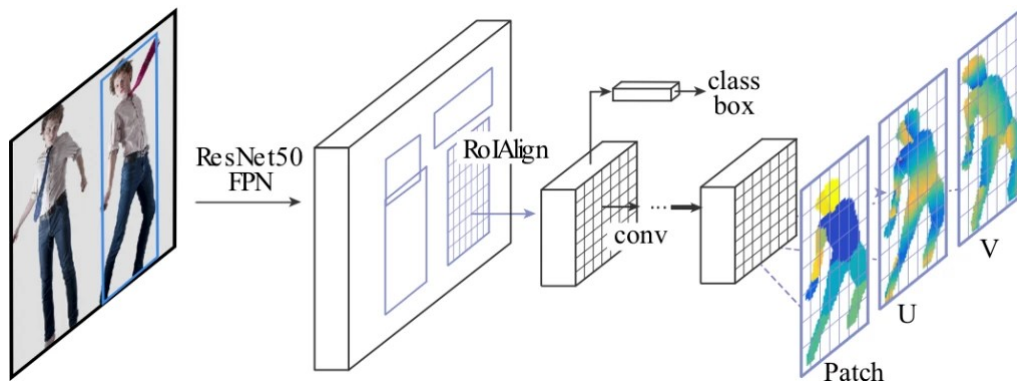


FIGURE 2.20

En effet, dans le calcul coordonné 3D, des travaux de recherche antérieurs ont montré que les CNNs peuvent fournir des prédictions remarquablement précises lorsqu'ils sont entraînés à prédire des variables catégorielles, indiquant en position la partie du corps ou le repère correspondant à chaque pixel. En s'appuyant sur ce fait, le DenseReg présente une méthode hybride qui combine une classification avec un problème de régression. En fait, la tête de réseau trouve d'abord une correspondance dense en divisant la surface. Et pour chaque pixel, déterminez :

- à quelle partie de la surface il appartient,
- où sur le plan paramétrique 2D (une version cylindrée du modèle SMPL) de la pièce à laquelle elle correspond.

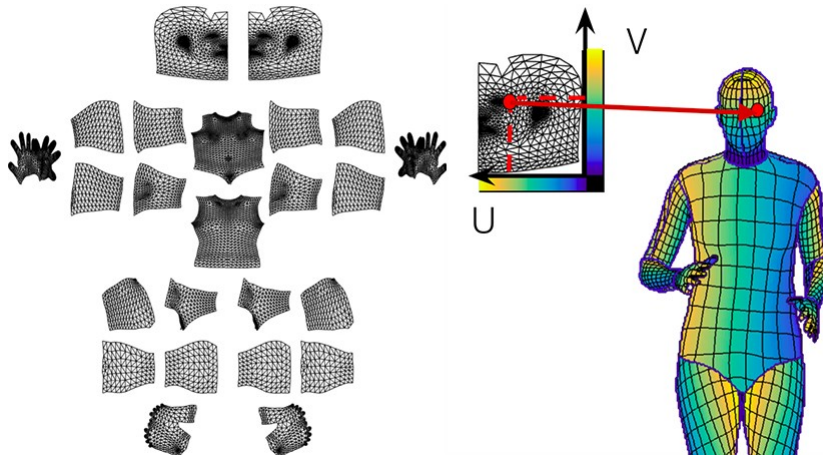


FIGURE 2.21 – Le régression dense de pose 3D

La modèle de surface du corps humain est divisée par une grille cartésienne, en quantifiant uniformément et séparément les coordonnées horizontales et verticales U dans la grille divisé en K partie, où K est un paramètre de conception. Pour toute image mise en correspondance avec le domaine du modèle, cela induit un étiquetage discret, qui peut être récupéré en formant un CNN pour la classification. Pour une

valeur suffisamment grande de K , même un simple résultat de classification pourrait fournir une estimation raisonnable du champ de correspondance du pixel. Ensuite, nous calculons l'erreur résiduelle entre les coordonnées $U - V$ désirées et quantifiées et ajoutons un module séparé qui essaie de régresser un point plus précis.

2.4 Evaluation de la performance des modèles d'estimation de la pose humaine

2.4.1 Métrique d'évaluation de détection d'objet

Les méthodes de détection d'objet et estimation de pose développées auparavant utilisent un certain nombre de fonctions pour calculer et évaluer la précision des sorties de CNN, on cite les critères suivants : Pourcentage de pièces correctes - PCK : Un membre ou point clé est considéré comme détecté (une pièce correcte) si la distance entre les deux emplacements prévus de l'articulation et les emplacements réels de l'articulation du membre est inférieure à la moitié de la longueur du membre (communément désignée comme PCK@0.5). Plus le PCK est élevé, plus la précision du modèle est élevée.

1. Pourcentage de points clés corrects - PCK : Une articulation détectée est considérée comme correcte si la distance entre l'articulation prévue et l'articulation réelle (dans le signal de supervision) se situe dans un certain seuil. Le seuil peut être l'un ou l'autre :

$PCK_h@0.5$ est quand le seuil = 50% du lien osseux de la tête

$PCK@0.2$ = Distance entre l'articulation prévue et l'articulation vraie $< 0,2 * \text{diamètre du torse}$

Parfois, on prend 150 mm comme seuil.

2. Pourcentage des articulations détectées - PDJ : Une articulation détectée est considérée comme correcte si la distance entre l'articulation prévue et l'articulation réelle se situe dans une certaine fraction du diamètre du torse. $PDJ@0.2$ = distance entre l'articulation prévue et l'articulation vraie $< 0,2 * \text{diamètre du torse}$.
3. Object Keypoint Similarity (OKS) basé sur mAP : Communément utilisé dans le cadre du défi des points clés de la COCO.

Où on a :

- d_i est la distance euclidienne entre le point clé détecté et point correct du Dataset.

- v_i est le drapeau de visibilité de la vérité de terrain.

- s est l'échelle des objets.

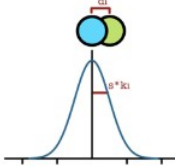
$$OKS = \frac{\sum_i e^{-\frac{d_i^2}{2s^2k_i^2}} \delta(v_i > 0)}{\sum_i \delta(v_i > 0)}$$


FIGURE 2.22 – Object Keypoint Similarity

-k est une constante par point clé qui contrôle la perte.

OKS joue le même rôle que l'IoU (Intersection de l'Union IoU entre les boîtes de délimitation prédit et correct) dans la détection des objets. Elle est calculée à partir de la distance entre les points prévus et les points de vérité normalisés par l'échelle de la personne.

2.4.2 Métrique d'évaluation des correspondances surfacique

Dans le cadre de l'estimation de la pose à base de correspondance surfacique, deux façons différentes sont utilisées pour estimer l'exactitude de la correspondance sur l'ensemble du corps humain :

Évaluation ponctuelle (Per-point evaluation). Cette approche permet d'évaluer l'exactitude de la correspondance sur toute l'image à l'aide du rapport Ratio of Correct Points (RCP), où une correspondance est déclarée correcte si la distance géodésique est inférieure à un certain seuil. Comme le seuil t varie, on obtient une courbe $f(t)$, dont l'aire nous fournit une indication scalaire de la précision de la correspondance. Pour n'importe quelle image donnée, nous avons un ensemble variable de points venant avec des signaux de vérité de la base de données. Nous résumons la performance sur l'ensemble de ces points, rassemblés à travers des images. Nous évaluons l'aire sous la courbe pour deux valeurs différentes de limite de courbe, $a = 10$ cm et 30 cm où la précision diminue quand a augmente. Cette mesure de performance est facilement applicable aux scénarios pour une ou plusieurs personnes et peut fournir des valeurs directement comparables.

Évaluation par cas (Per-instance evaluation). En utilisant le paramètre de similarité des points géodésiques (GPS) :

$$GPS_j = \frac{1}{|P_j|} \sum \exp \frac{-g(i_p, \hat{i}_p)}{2k^2}$$

Où $g(,)$ est la distance géodésique entre deux points du modèle surfacique. P_j est l'ensemble des points de vérité annotés sur la personne d'instance j , i_p est le sommet estimé

par un modèle au point p , \hat{p} est le vertex de vérité p et k est un paramètre normaliseur. Nous réglons $k=0.255$ de sorte qu'un point unique ait une valeur GPS de 0.5 si sa distance géodésique par rapport au point correcte est égale à la demi-taille moyenne d'un segment de corps. D'ou un GPS de 0.5 est dans un cas idéal, tout en allant au-delà, cela nécessite également une localisation plus précise d'un point sur la surface.

2.5 Les bases de données des pose humaine

Les ensembles de données les plus courantes en vision par ordinateur pour la détection d'objets et les tâches d'estimation de pose sont :

MPII : un ensemble de données d'estimation de la position 2D de plusieurs personnes comprenant près de 500 activités humaines différentes, recueillies à partir de vidéos Youtube. MPII a été le premier ensemble de données à contenir une gamme aussi diversifiée de poses et le premier ensemble de données à lancer un défi d'estimation 2D Pose en 2014.

COCO : L'ensemble de données des points clés du COCO est un ensemble de données d'estimation de la position 2D de plusieurs personnes avec des images recueillies sur Flickr. Le COCO est le plus grand ensemble de données d'estimation de poses 2D à ce jour, et envisage un point de référence pour tester les algorithmes d'estimation de poses 2D. D'autres ensembles de données importants pour l'estimation de la pose humaine comprennent soit la un signal de supervision avec une squelette 3D complète à l'aide de capteur MoCap, soit des images de pose générées synthétiquement par des réseaux de neurones type encodeur-decodeur , soit une annotation en surface de la pose vrai manuellement :

Human3.6m : Ensembles de données à grande échelle pour la détection 3D humaine dans les environnements naturels (2014). Un ensemble de données de 11 personnes faisant 17 poses communes dans un environnement intérieur, soit un total de 3,6 millions d'images. Les mesures suivantes sont incluses :

Vues RGB : 4 projection, dont un avec profondeur

Positions exactes des articulations 3D à partir du MoCap

Étiquettes de partie de corps au niveau des pixel pour un sous-ensemble de données scans laser 3D des acteurs (réalisés une seule fois, pas tout au long des vidéos) Cet ensemble de données comprend principalement des poses de tous les jours, dans des scénarios naturels. Ils incluent également une certaine augmentation de la réalité mixte pour l'ensemble de données.

HumanEva (2009) : Cet ensemble de données est semblable à l'ensemble de données de Human3,6 m, mais il est plus limité puisqu'il est composé de deux ensembles de données, HumanEva I et HumanEva II, tous deux dans un environnement intérieur contrôlé. L'ensemble de données HumanEva I se compose de :

- Vues RGB : 3 standards.
- Positions des articulations 3D à 60 Hz de MoCap (Motion Capture).

Cet ensembles de données ont des vêtements plus naturels, mais cela vient avec le coût du déplacement des capteurs et des données MoCap moins précises.

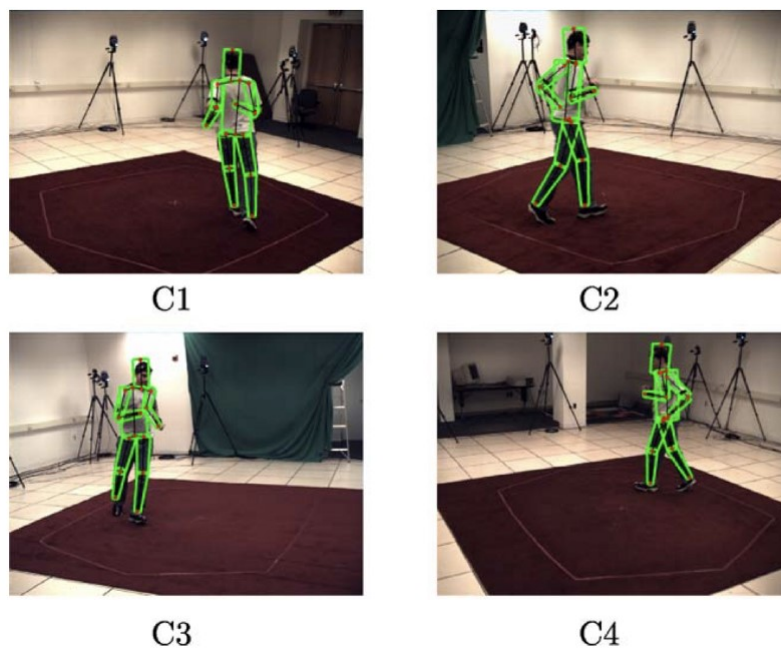


FIGURE 2.23 – Exemples tirés de HumanEva II

Capture totale : L'ensemble de données Total Capture englobe l'estimation de la posture humaine en 3D par fusion des vidéo et des capteurs inertiels IMU. Les vidéos sont multi-vues intérieure, l'IMU et la mocap avec 1.9 Million d'image, et un Environnement très contraint. Cet ensemble de données est utile pour comparer l'estimation de la pose 3D à l'aide de caméras IMU et multi-vues au donnée de signal de supervision à l'aide de mocap, mais encore une fois, il est assez limité du fait d'être dans un environnement contrôlé.

3D poses in-the-wild : Dans cet ensemble de données, 60 vidéos dans la nature avec annotation de pose 2D et 3D sont incluses, avec la pose de l'appareil photo et les modèles scannés avec différentes variations de vêtements. Les poses sont estimées à l'aide d'IMU et d'une vidéo 2d portable.

Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision 2016 :

Cet ensemble de données utilise un système multi-caméra sans les marqueurs, enregistrant la pose 3D dans un écran vert pour changer les arrière-plans. Il comprend 8 acteurs, 8 caméras, plus de 1.3M images. C'est encore à l'intérieur dans un environnement relativement contrôlé, donc ce n'est pas vraiment à l'état sauvage.

SURREAL 2017 :

Nombre d'images : 6.5M

Nombre de sujets : 145

SURREAL (Synthetic Humans for REAL tasks) est un nouvel ensemble de données à grande échelle avec des images synthétiques mais réalistes de personnes rendues à partir de séquences 3D de données humaines de capture de mouvements. Il comprend plus de 6 millions d'images accompagnées de la pose de vérité au sol, de cartes de profondeur et de masques de segmentation. Les images de SURREAL sont rendues à partir de séquences 3D de données MoCap. Pour assurer le réalisme des corps humains dans cet ensemble de données, les chercheurs ont décidé de créer des corps synthétiques en utilisant le modèle de corps SMPL. De plus, les créateurs de l'ensemble de données SURREAL ont assuré une grande variété de points de vue, de vêtements et d'éclairage.

UP-3D 2017

Nombre de sujets : 5,569

Nombre d'images : 5569 images d'entraînement et 1208 images de test. UP-3D est un ensemble de données qui "unit les gens" de différents ensembles de données pour des tâches multiples. En particulier, en utilisant l'algorithme SMPLify, les chercheurs obtiennent des modèles corporels 3D de haute qualité pour plusieurs ensembles de données de pose humaine. Les annotateurs humains ne trient que les bons et les mauvais ajustements. Il combine deux ensembles de données LSP (11000 images d'entraînement et 1000 images de test) et la partie individuelle de l'ensemble de données MPII-HumanPose (13030 images d'entraînement et 2622 images de test). Les ajustements validés formaient l'ensemble de données UP-3D initial avec 5569 images d'entraînement et 1208 images de test.

DensePose 2018

Nombre d'images : 50K

Nombre de correspondances annotées : 5M

Il s'agit d'un ensemble de données de vérité au sol à grande échelle avec des correspondances annotées manuellement sur des images COCO 50K en utilisant un pipeline d'annotations spécialement développé .

Comme illustré ci-dessous, dans la première étape, les annotateurs définissent les ré-

gions correspondant aux parties visibles et sémantiquement définies du corps. Dans un deuxième temps, chaque région est échantillonnée avec un ensemble de points à peu près équidistants et les annotateurs sont priés d'apporter ces points en correspondance avec la surface.

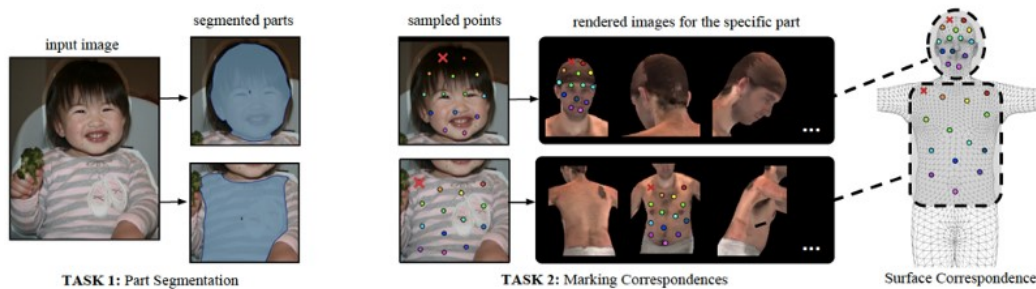


FIGURE 2.24 – La segmentation et régression dans DensePose

Vous trouverez ci-dessous des visualisations d'annotations sur les images de l'ensemble de validation Valeurs d'image (gauche), Pose 3D des deux ca U (milieu) et V (droite) pour les points collectés.



FIGURE 2.25 – Exemple tirée de dataset DensePoseCOCO

DensePose est le premier ensemble de données de pose 3D collectées manuellement pour l'estimation de la pose humaine dense.

Chapitre 3

Analyse et expérimentation sur l'algorithme DensePose

Introduction

Dans le but d'améliorer les performance et analyser les manque en performance, on présente dans ce chapitre notre énoncé du probleme pour expliquer l'inefficacité des algorithmes d'estimation de pose humaine basé sur les FPN. On propose une architecture basé sur les convolution dilatées et la procédure d'entraînement de cet modèle.

3.1 Analyse du problème visée dans l'algorithme DensePose

Après avoir conduit des testes qualitatifs de l'algorithme DensePose sur une variété d'images (et non celles présentées par les auteurs), nous constatons que la phase de détection et de la segmentation humaine présentent encore quelques lacunes majeures en termes de performance. Le masque de segmentation et la boîte de délimitation du corps semblent manquer de précision dans le cas d'existence d'une seule personne et surtout de plusieurs personnes. Les cas où les images incluent des personnes à une partie visible du corps humain, ou lorsqu'il existe un nombre élevé d'humains sont, malgré la complexité du modèle, clairement un obstacle majeur en raison de la diminution remarquable de la tâche de segmentation des parties de corps.

Dans notre travail, nous abordons ce dernier problème, et nous inspectons la source possible du problème, modifions l'architecture du réseaux CNN back-end de DensePose afin de surmonter ces obstacles ou au moins analysons et améliorons le benchmark actuel de l'algorithme.

En effet, malgré la complexité du réseau et le grand nombre de paramètres, l'occlusion et les parties invisibles du corps empêchent l'algorithme d'obtenir des résultats satisfaisants, même si celui-ci est l'un des plus avancés et des plus récents algorithmes dans le domaine. D'une autre part, l'échelle de vue de la personne à l'intérieur de l'image présente également un problème dans l'estimation des poses humaines, malgré le fait que le réseau a été entraîné sur des images en différentes résolutions. Au fait, chaque humain dans une image occupe un espace encore plus limité dans le plan de l'image si l'échelle diminue ou le nombre de personnes existant augmente. Avec ce dernier point, l'extracteur des *feature maps* proposé dans l'architecture DensePose qui est basé sur les Feature Pyramid Networks (FPN) apporte une solution possible au problème en inspectant différents niveaux en forme de pyramide des caractéristiques de l'image. Cependant, deux inconvénients découlent de cette architecture réseau :

L'architecture ne résout pas le deuxième le problème visé puisque les couches supé-

rieures de l'hierarchie du réseau ont un niveau de représentation plus élevée avec une complexité accrue des données et une résolution spatiale réduite, et non un niveau de représentation des objets à des échelles supérieures. Les auteurs de [18] supposent que les caractéristiques de différentes couches résoudre un problème dépendant de l'échelle, alors qu'en fait, elles résoudre un problème dépendant de la complexité des données (voir la figure 3.1). De plus, ces caractéristiques ne se fondent pas nécessairement correctement dans l'ensemble du vecteur de caractéristiques qui est construit sur les caractéristiques à partir de diverses valeurs sémantiques. Même la façon de concaténer les caractéristiques présente certaines ambiguïtés qui n'ont pas été étudiées dans [18]. Par conséquent, soit une étude appropriée doit être menée sur l'effet de diverses stratégies de concaténation de vecteurs qui peuvent en déduire la meilleure méthodologie de cette concaténation, soit un autre aspect de la résolution doit être inspecté comme nous l'avons fait dans notre travail actuel.

Une complexité accrue des Feature Maps a été développée à des fins de classification des images. Dans cet esprit, les FPN créés pour la phase d'extraction des caractéristiques ne représentaient que les caractéristiques visées dans la classification des images. Plus concrètement, Les tâches liées à la localisation, telles que l'estimation de la pose humaine ou la détection d'objets en général, n'ont toujours pas d'extracteurs de caractéristiques développés spécifiquement pour chaque tâche relative. Cet aspect des CNN dans l'estimation de la pose humaine est fortement négligé dans la littérature de vision par ordinateur où la plupart des chercheurs ont tendance à concentrer leurs études sur la variation et l'amélioration de la tête ou du corps central des CNN pour cette tâche avec les mêmes architectures de base que les réseaux de classification des images, oubliant les différences entre les caractéristiques nécessaires pour une tâche d'estimation humaine et celles nécessaires pour une tâche simple où une couche totalement connectée effectue les calculs essentiels au lieu des architectures "front-end" existantes très complexes dans la détection des objets.

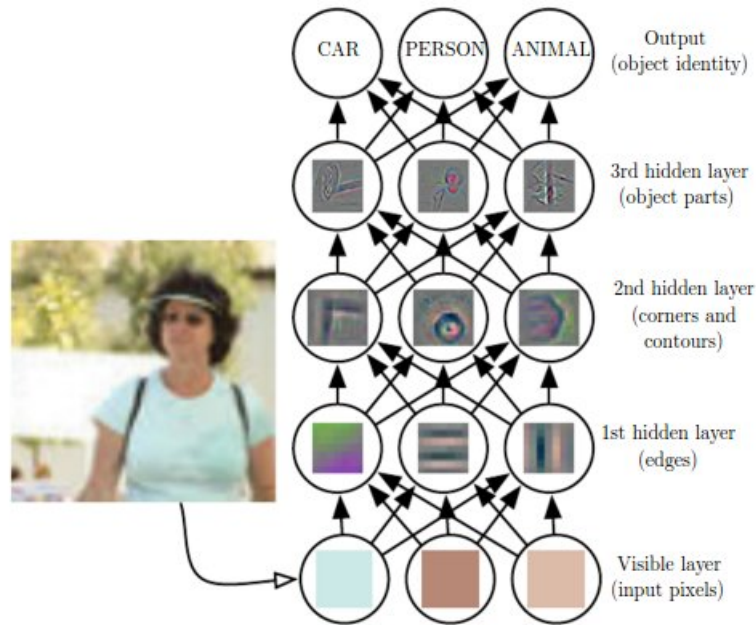


FIGURE 3.1 – Visualisation de cartes de caractéristiques (extraites pour une tâche de classification) à différents niveaux de la couche cachée dans le réseau : les niveaux de caractéristiques plus élevés représentent une valeur sémantique et une complexité de représentation des données plus élevées, au lieu d'une représentation à une échelle supérieur.

3.2 Proposition du modèle Dilated DensePose

Dans le contexte d'un réseau de neurones convolutionnel dédié à une tâche d'estimation de la pose humaine, et afin d'améliorer les performances de la phase d'extraction de caractéristiques, nous présentons d'abord le fonctionnement de "Dilated Convolution", et son rapport avec le problème. Inspiré par la proposition de réseau du modèle DeepLab [3], nous nous engageons dans l'évaluation de l'effet des paramètres de convolution dilatée en entraînant le modèle CNN utilisé dans l'algorithme DensePose avec les modifications relatives.

3.2.1 Convolution dilatée

Pour faire suite à l'ancienne proposition d'architecture du réseau CNN, nous expliquons le principe des convolutions dilatées et leur effet sur le champ réceptif des opérations de convolution à travers les couches du CNN.

Comme indiqué précédemment, les convolutions dilatées sont un type spécial de convolutions qui calcule la sortie des neurones associés en prenant un flux de données discret de la couche précédente (soit la couche d'entrée, soit la couche de neurones pré-

cédente), c'est-à-dire en spécifiant un pas que nous prenons en compte dans les deux dimensions de longueur et largeur de la couche d'entrée pour multiplier par les poids appris de la couche de convolution dilatée. Comme on le voit ci-dessous, le champ réceptif d'une opération normale de convolution est le nombre exact de pixels/neurones pris dans l'entrée, alors qu'il est supérieur dans une fenêtre de convolution dilatée de même dimension (selon le pas "rate").

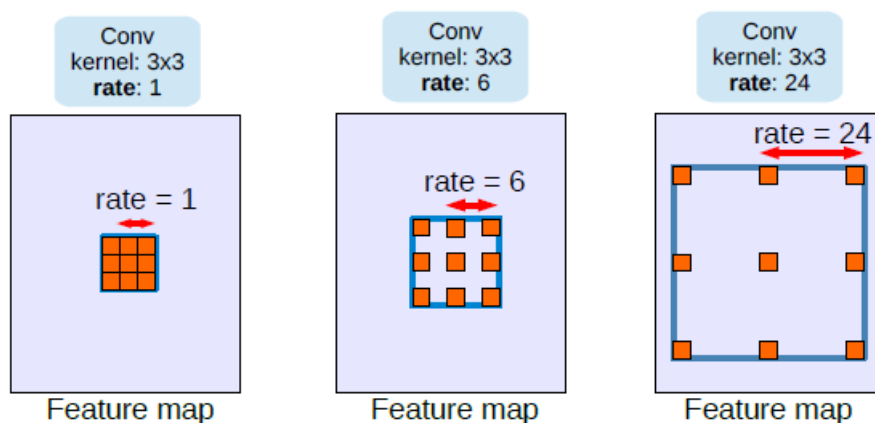


FIGURE 3.2 – Le champ réceptif d'une convolution dilatée à pas 1, 6, 24 .

Quant à la perte d'information à travers les couche du réseau CNN usuel, elle peut sembler faible selon la configuration normale. Cependant, plus on avance en profondeur dans le réseau, plus cette représentation diminue en termes d'efficacité. Plus particulièrement, tant que nous empilons plus de couches dans le réseau, l'information de localisation des caractéristiques dans l'espace de ces couches continue à diminuer. De plus, pour le même nombre de couches, l'entraînement d'une convolution normale n'adapterait pas efficacement les poids puisqu'elle tient compte des pixels/neurones très proches, alors qu'elle adapte plus efficacement les poids de convolution de dilata-tion aux données d'entraînement en prenant en compte un neurone de convolution et la carte des caractéristiques de l'entrée.

Pour illustrer ce dernier point, si nous avons besoin de caractéristiques de la même résolution que l'entrée, nous devrions appliquer une opération de suréchantillonnage qui augmente la résolution de la carte de caractéristiques par un certain taux. Et en visualisant l'effet de cette opération de convolution/suréchantillonnage, nous verrions clairement une perte d'entrée telle que présentée par Liang et tous [3] dans la figure 3.3, ou la convolution dilaté montre une représentation plus dense des caractéristiques lorsqu'une convolution dilatée est appliquée au lieu du bloc extracteur de caractéristiques.

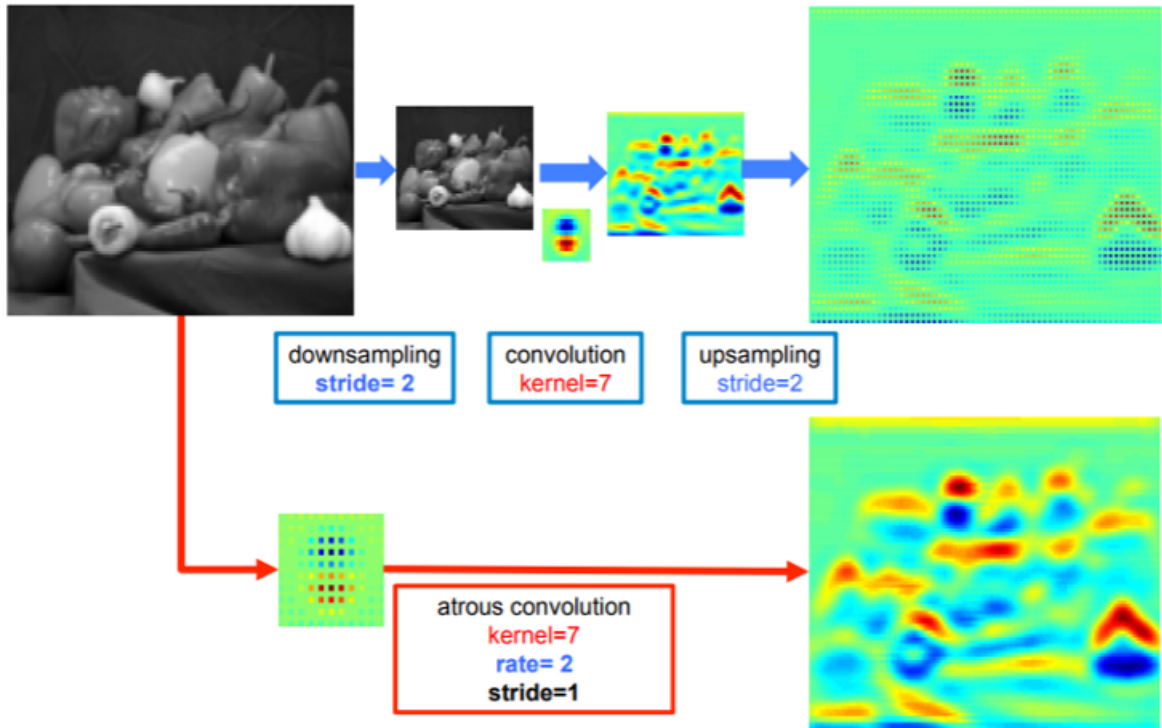


FIGURE 3.3 – La perte d’information dans les convolution usuel en comparaison avec les convolution dilatée

3.2.2 Implémentations des convolutions dilatées

Pour une implémentation mathématique de la convolution dilatée (appelé aussi "atrous"), nous utilisons l’équation de convolution suivante pour calculer la sortie y d’indice i du neurone :

$$y[i] = \sum_{k=1}^K x[i + r.k]w[k]$$

Où x est le pixel d’entrée multiplié par le poids w . Le paramètre de la convolution " r " est le paramètre qui contrôle le pas par laquelle nous nous déplaçons le long de la longueur et de la largeur de la carte des caractéristiques d’entrée. Si " r " augmente jusqu’à qu’il dépasse une certaine valeur, l’effet de la convolution dilatée serait destructeur sur l’homogénéité des cartes des caractéristiques de sortie. Ceci est dû au fait que les neurones de convolution voisins apprendraient dans ce cas-ci, des caractéristiques très dissociées ou décorréelée, résultant en des sorties de neurones qui sont sémantiquement démêlées. Cependant, la sortie des neurones de convolution doit toujours conserver cette homogénéité afin d’avoir une représentation valide de l’information apprise (La pose humaine).

Quant à l'implémentation algorithmique de la convolution dilatée, deux méthodes analogiques existent dans ce cas :

La première est d'utiliser un neurone de convolution avec des zéros insérés entre les poids et d'appliquer un produit de point entre les neurones et une fenêtre de même taille des pixel ou sortie de neurones.

La seconde est d'insérer des zéros dans la carte des caractéristiques en entrant par le même pas et d'appliquer un produit de points avec le neurone de convolution de la même taille.

Les deux approches présentent des avantages et des inconvénients en termes d'utilisation de la mémoire, de temps d'entraînement et d'inférence, mais la première est celle qui est la plus utilisée dans les approches de programmation car elle optimise le processus d'entraînement. En outre, un autre point important dans l'approche algorithmique de la convolution est le remplissage par zéro nécessaire pour appliquer correctement la convolution autour les frontières. La taille de la sortie des convolutions dilatées dépend du taux de dilatation comme suit :

$$O = 1 + \frac{(i - k - (k - 1)(d - 1) + 2p)}{s}$$

Où I est la taille d'entrée, O la taille de sortie, k la taille du neurone de convolution non dilatée, s est le pas, p le paramètre de remplissage et d le paramètre de dilatation. En d'autres termes, la taille effective du neurone est de $k - (k - 1)(d - 1)$. Lorsque le paramètre de dilatation est réglé de $d = 1$, cela correspond à l'opération de convolution usuelle.

L'algorithme DensePose est basé sur le réseau ResNet CNN, qui est dédié à l'étape d'extraction des fonctionnalités. Cette dernière architecture des CNNs est une cascade de blocs résiduels, et chaque bloc résiduel contient trois couches de convolution (deux convolution de largeur 1x1, et une de largeur 3x3) dans le cas des couches ResNet 50, 101 et 152. Chaque bloc résiduel a une connexion de saut d'entrée qui serait ajoutée à la sortie. Nous introduisons la convolution dilatée à la place de la convolution 3x3 dans CONV 2, CONV3 et CONV4, du fait que la convolution de 1x1 est utilisée avec une pas de 2 qui présente une forme d'optimisation des caractéristiques elle-même. Pour une implémentation détaillée du code DensePose, veuillez vous référer à l'Annexe. Nous choisissons la couche CONV 2-4 de DensePose ResNet 50 à des fins d'évaluation, car le temps de entraînement d'un ResNet 101 est très long. La couche CONV1 n'a pas été choisie parce que ce bloc de tige n'implique pas une perte majeure dans la perte de localisation qui commence à partir de la couche suivante uni-

quement à cause des convolutions répétitives à venir. La couche CONV5 n'a pas été choisie dans le processus de convolution dilatée parce que la forme finale des cartes de caractéristiques extraites devait être conservée comme dans le réglage initial de l'algorithme DensePose. En fait, la modification de la dilatation dans le dernier CONV affecterait le signal de sortie pour la phase suivante qui est la création des pyramides des caractéristiques et leurs addition dans les cinq couches CONV.

Liang et tous[] présentent dans leur étude les algorithmes Atrous Spatial Pyramid Pooling pour classifier un pixel central, ASPP exploite des fonctionnalités multi-échelles en utilisant plusieurs filtres parallèles avec des taux différents (6, 12, 18, 24).

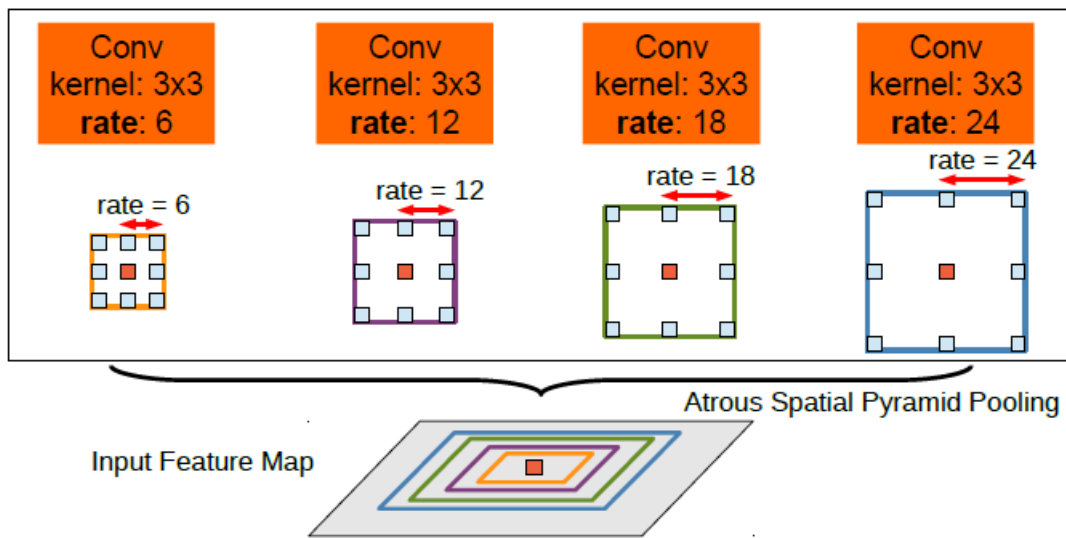


FIGURE 3.4 – L'algorithme Atrous Spatial Pyramid Pooling qui utilise la dilatation sur le même niveau de couche convolutionnelle

Notre approche utilise un principe similaire mais à partir d'un pourcentage différent. Comme ResNets est l'épine dorsale d'un FPN, les multiples caractéristiques parallèles sont tirées de quatre groupes de couches sémantiquement variables avec trois taux de dilatation à la place. Le paramètre de dilatation augmente avec l'augmentation de la profondeur de la couche. Cette dernière est due à l'augmentation à la fois de la perte de la localisation et de la densité de l'information par couche de convolution.

Après analyse du temps d'entraînement estimé des différentes configurations des paramètres de dilatation, on constate que lorsque les paramètres de dilatation dépassent une certaine valeur (8 pour CONV1, 18 CONV2, 24 CONV3), le temps d'entraînement devient peu pratique. De plus, et comme nous l'avons déjà dit, l'effet de la convolution dilatée serait destructeur sur l'homogénéité des cartes des caractéristiques de sortie.

En raison des limitations matérielles, nous avons choisi de former l'architecture DensePose basée sur ResNet50 avec deux configurations de $(d1, d2, d3)$: $(d1, d2, d3) = (2, 6, 24), (4, 8, 16)$

Le choix tient compte d'un temps d'entraînement raisonnable et des valeurs numériques proposées en fonction des paramètres de dilatation DeepLab. De plus, nous prenons le paramètre de Padding égale au paramètre de dilatation pour que la taille de sortie du neurone de Convolution 3x3 qui est réglé avec une pas de 1, ce qui ne change pas de la configuration normale, produisant la même carte des caractéristique à chaque niveau et augmentant le champ récepteur :

$$d = p(=>)O = 1 + I + 2p - k - (p - 1)(k - 1)/s$$

$$k = 3(=>)O = 1 + I + 2p - 3 - 3p - 1 + 3 + p/s$$

$$O = 1 + I - 1/s$$

$$s = 1(=>)O = I$$

Compte tenu de ces considérations, nous serions également en mesure d'étudier plus efficacement l'effet du paramètre de dilatation à partir d'un niveau de programmation plus élevé au lieu de modifier les paramètres internes du réseau et la dimension des cartes de caractéristiques.

3.3 Procédure d'entraînement

Lors de l'entraînement d'un réseau de neurones convolutionnel, différents facteurs doivent être pris en compte tels que la fonction de coût utilisée, comme les fonctions de coût représentées ci dessous, Sofmax et SVM respectivement.

$$\sigma(z)_j = \frac{\exp z_j}{\sum_k = 1^k \exp z_k}$$

$$SVM = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Dans l'algorithme DensePose, la entraînement du CNN utilise la fonction de coût d'entropie croisée EC Pour la perte de classification des partie du corps :

$$EC = - \sum_x p(x) \log(p(x))$$

Ou $p(x)$ est la sortie de la fonction d'activation de la dernière couche qui utilise la fonction sigmoid au lieu de RELU pour but de compresser la sortie dans l'intervalle $[0, 1]$

1], représentant des valeur analogue à des probabilité d'exactitude de classification de la sortie.

Et une perte L1 lisse [9] pour chaque fonction de régression spécifique à une partie du corps. La perte de régression d'une partie n'est prise en compte que pour les pixels occupés par cette partie.

$$Loss = \sum_{x,y,w,h} smooth_{L1}()t_i^u - v_i$$

Ou on a x,y les coordonnées du point haut en gauche de la boite délimitation, et (w,h) les dimension de la boite, et :

$$smooth_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{else} \end{cases}$$

3.3.0.1 Le pas d'apprentissage

Lorsqu'il s'agit des taux d'apprentissage de la procédure d'optimisation, leur effet sur le temps d'entraînement et sur la précision (erreur de perte/validation) est crucial. Les réseaux de neurones en général sont des fonctions d'hypothèses que nous entraînons à accorder en fonction la tâche visée. Il a été démontré que l'espace de ces hypothèses contient des fonctions convexes qui ont nécessairement un minimum global en fonction des données d'entraînement. Afin d'atteindre ce minimum, nous formons le réseau et mettons à jour les paramètres qui représentent la fonction d'hypothèse. Or, si le rythme d'apprentissage de la mise à jour de ces paramètres est trop élevé, le réseau ne convergerait jamais vers le minimum global (ou même divergerait), s'il est trop petit, le temps d'entraînement serait irrémédiablement long. Par conséquent, nous utilisons différents schémas de mise à jour de poids pour ajuster le rythme d'apprentissage afin d'obtenir une valeur ajoutée efficace pendant l'entraînement.

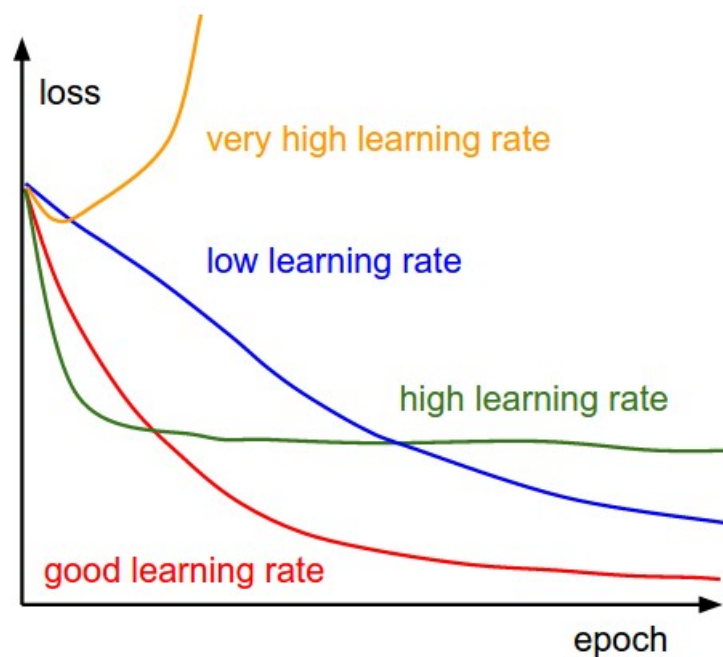


FIGURE 3.5 – L’effet du rythme d’apprentissage sur la précision

le défi de l’utilisation des rythmes d’apprentissage est que leurs hyperparamètres doivent être définis à l’avance et qu’ils dépendent fortement du type de modèle et du problème. Une autre problématique c’est que le même rythme d’apprentissage est appliqué à toutes les mises à jour des paramètres. Si nous avons peu de données, il se peut que nous voulions plutôt mettre à jour les paramètres dans une autre mesure. Le schéma de décroissance du gradient stochastique de Descent est largement utilisé dans la entraînement des CNNs. En fait, il s’agit d’une forme de descente de gradient (GD) qui est utilise en raison de la convergence rapide et temps d’entraînement réduit, comparé avec les autres approches. Par exemple, notre temps de entraînement estimé de ResNet50DensePose est de :

- GD avec une taille de lot de 2 ou 3 : ≈ 9 jours.
- *SGD* : 5 jours.

3.3.1 Les rythme mise à jour des poids adaptatifs

Il existe d’autres algorithmes de descente de gradient adaptatif comme **Adagrad**, **RMSprop** et **Adam** qui fournissent une approche heuristique sans nécessiter un travail coûteux de réglage manuel des hyperparamètres pour la programmation du pas d’apprentissage. En bref, Adagrad effectue des mises à jour plus importantes pour des paramètres plus clairsemés et des mises à jour plus petites pour des paramètres moins clairsemés. Il a de bonnes performances avec des données éparses et un réseau

de neurones d'entraînement à grande échelle. Cependant, son rythme d'apprentissage monotone s'avère généralement trop agressif et arrête d'apprendre trop tôt lors de l'entraînement de réseaux de neurones profonds. RMSprop ajuste la méthode Adagrad d'une manière très simple pour tenter de réduire son taux d'apprentissage agressif et monotone. Adam est une mise à jour de l'optimiseur RMSProp qui est comme RMSprop avec une valeur exprimant le moment d'apprentissage de la fonction de coût.

Dans notre fichier de configuration du processus d'entraînement (voir Annexe), nous utilisons une seule image par itération dans la procédure d'optimisation, et nous mettons à jour nos paramètres à chaque fois. Le paramètre responsable est le "Batch Size" que nous réglons à 1. De plus, pour des raisons de robustesse de l'algorithme et d'invariance d'échelle, nous mettons à jour les paramètres sur 6 échelles différentes de chaque image en entrée (640, 672, 704, 736, 768, 800). Quant à l'étape RPN, le processus d'entraînement génère 512 propositions de région par image lors de la passe en avant de l'entraînement.

Les programmes des taux d'apprentissage comprennent aussi une étape d'échauffement de l'optimisation, et on en distingue deux types : constants et progressifs. Nous utilisons la configuration progressive appelée "Step decay" selon la règle de mise à l'échelle du taux d'apprentissage linéaire décrite dans "Accurate, Large Minibatch SGD : Training ImageNet in 1 Hour" Goyal et al" [11]. Le programme d'apprentissage est réalisé en optimisant la fonction de perte avec un taux d'apprentissage (lr) multiplié par un facteur d'échauffement défini comme un hyperparamètre égal à 0,1.

Le taux d'apprentissage dépend également du nombre de cœurs GPU disponibles sur la machine. Différents schémas d'apprentissage existent pour un nombre variable de cœurs GPU. Nous disposons d'un GPU à un seul cœur et 12Go de RAM fournis par le service Cloud de Google que nous avons utilisé pour former notre réseau de neurones. Le pas d'apprentissage est réglé sur 0,0001 avec un programme d'échauffement de 1000 itérations et un Lr de 0,1.

3.3.2 Initialisation des poids

Le but de l'initialisation des poids est d'empêcher les sorties d'activation des couches d'exploser ou de disparaître au cours d'un passage en avant à travers un réseau de neurones profond. Si l'un ou l'autre se produit, les gradients de perte seront soit trop importants, soit trop faibles pour que le calcul du passe en amont soit bénéf-

fique, et il faudra plus de temps pour que le réseau converge s'il converge en premier temps. Si nous initialisons le réseau avec des poids aléatoires très faibles, cela peut conduire à des distributions non homogènes d'activations entre les couches d'un réseau. Si nous utilisons une initialisation aléatoire de grande valeur, et que la fonction d'activation utilisée est un Softmax, cela conduirait à des neurones saturés avec un gradient de presque zéro empêchant le réseau de s'entraîner. L'initialisation correcte est un domaine de recherche actif, mais la convention actuelle d'initialisation est l'initialisation de He et al [16] :

$$W = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in / 2)$$

Où `np` est la bibliothèque python numpy, `fan_in` est la taille d'entrée du calque précédent, et `fan_out` est la taille de sortie. Les poids aléatoires sont calculés à partir d'une distribution normale de la moyenne 0 et d'une variance de 1, puis, les poids sont multipliés par le facteur développé initialement dans l'initialisation de Xavier [10] sans le facteur 0.5 introduit par He et al [16].

3.4 La plateforme logicielle d'apprentissage profondi

La plateforme de Deep Learning que nous avons utilisée pour installer le logiciel et ses dépendances s'appelle "Google Colab". Il s'agit d'un service fourni par google pour le Cloud Computing dans différents domaines, mais principalement pour Deep Learning. L'interface graphique de ce service est fournie sous la forme d'un notebook Jupyter qui s'ouvre à l'aide d'un navigateur Web normal comme le montre la figure suivante :

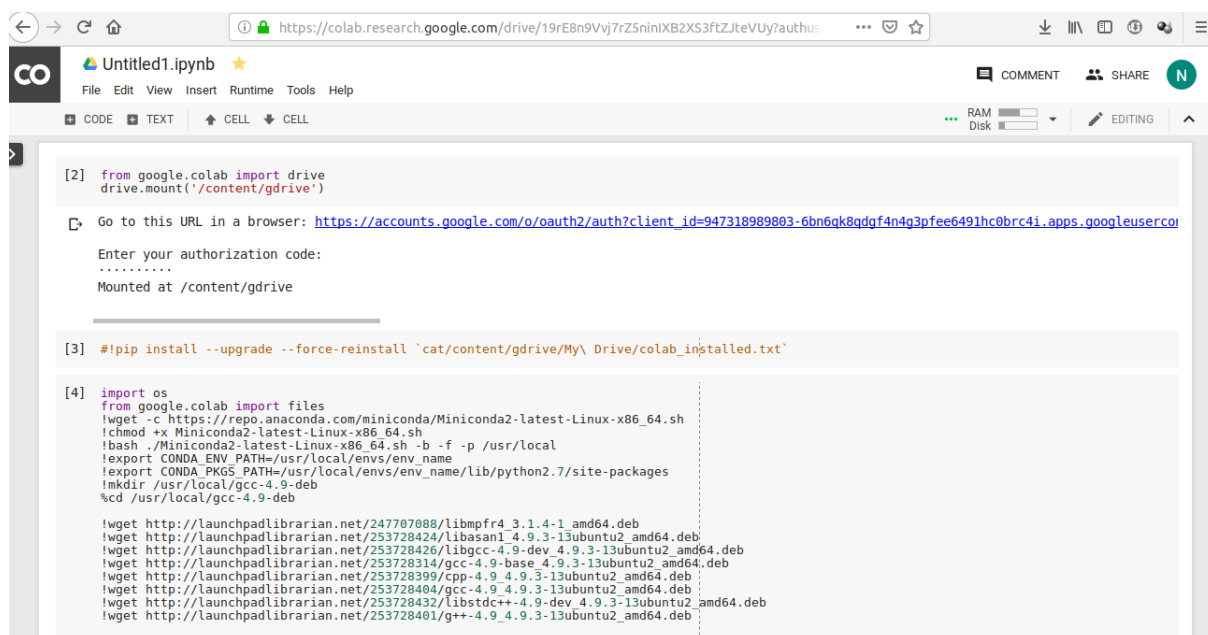


FIGURE 3.6 – La plateforme de développement COLAB

L'interface du notebook permet de créer une liste de cellules, chacune contenant des parties du programme principal. Le back-end de l'interface est une distribution Linux (18.04) avec un GPU NVIDIA Tesla K80 et un espace disque de 40 Go. De plus, le portable peut être utilisé avec deux types de matériel informatique (GPU ou TPU) avec deux versions Python (2.7 et 3.7), les commandes Python et linux peuvent être exécutées dans les cellules Colab (ajouter "!" au début de chaque commande de terminal linux) :

L'ordinateur portable Colab peut également être connecté au lecteur d'un compte google. Nous avons utilisé le service pour télécharger l'ensemble de données COCO afin d'utiliser notre modèle DensePose dans notre compte drive et nous avons utilisé l'authentification Drive dans Colab.

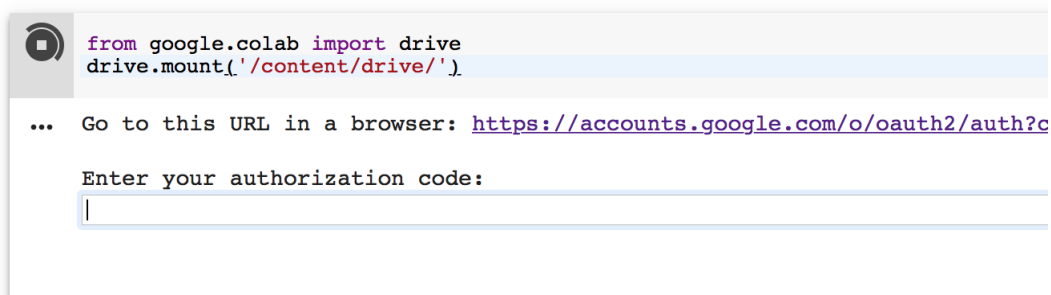


FIGURE 3.8 – L'interface des commande de COLAB.

Google Colab est livré avec un inconvénient majeur et un avantage majeur :

L'inconvénient est que le service peut être utilisé pendant 12 heures consécutives, après quoi le neurone GPU alloué à l'utilisateur est déconnecté et initialisé à zéro, par lequel l'utilisateur perd toutes les données locales et variables temporelles. Une réinitialisation des paquets installés doit être effectuée après cette limite.

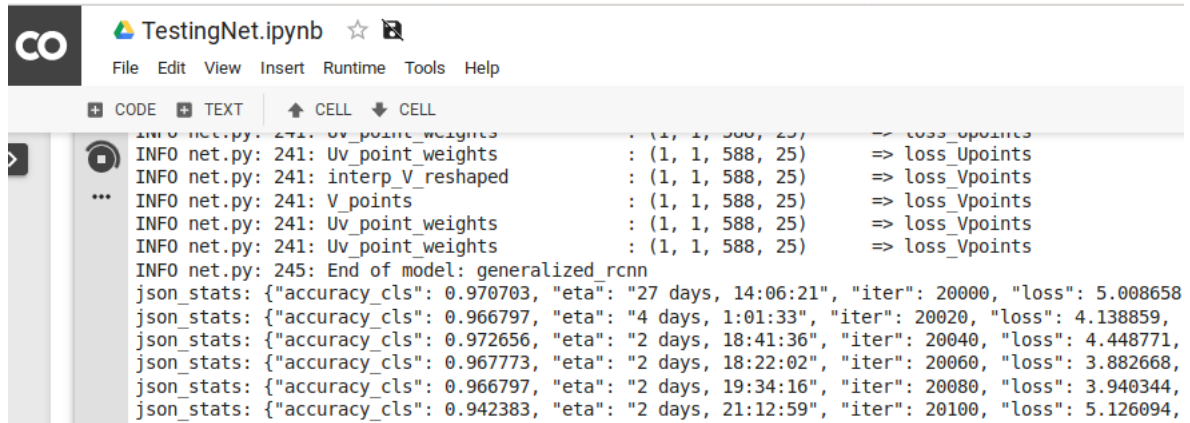
L'avantage de google Colab est que l'utilisateur peut effectuer ses processus informatiques sur plusieurs ordinateurs portables Colab. En économisant ainsi tous les calculs qui ne dépassent pas la limite des 12 heures, qui ont des exigences de calcul limitées et qui peuvent être effectués en parallèle. Bien que tous les ordinateurs portables d'un compte google partagent le même neurone linux et le même neurone GPU. Afin d'effectuer les calculs sur différents coeurs de GPU, on peut utiliser différents comptes Google pour chacun d'eux.

DensePose a été développé à l'aide du logiciel Detectron par la division de recherche en IA de Facebook. Ce logiciel est créé à partir de code utilisant la plateforme d'apprentissage approfondi "Pytorch", et il est responsable de tout le code relatif à la Vision dans Pytorch. Afin de former notre modèle DensePose, nous utilisons Detectron pour effectuer la procédure d'optimisation et établir un lien symbolique Linux (voir annexe) entre la sortie d'entraînement Detectron qui enregistre les poids entraînés en réseau et le vecteur momentum de l'optimiseur, et un fichier de sortie dans le lecteur. L'optimiseur enregistre l'état à l'initialisation et après chaque 20000 itérations, ce qui donne 32 fichiers de points de contrôle pour 720000 itérations.

Nous fournissons en annexe deux cahiers pour l'installation de Pytorch/DensePose dans Colab et pour les modifications du code source de notre modèle. Les commandes d'installation sont basées sur la version Python 2.7 de Detectron, qui est la version stable du logiciel à partir de juillet 2019. L'ensemble de données doit être pré-téléchargé sur le lecteur google, et seule une authentification du compte est nécessaire. Pendant la entraînement, la sortie du script est liée au lecteur google, et il enregistre les poids appris après chaque itération 20000. Le nombre maximum d'itérations pour le GPU unique utilisé est de 72000000 itérations et de 13000000 itérations pour un GPU 8 coeurs. De plus, les blocs-notes peuvent être trouvés dans le référentiel Github référencé dans l'annexe avec le modèle formé à partir de différentes configurations de paramètres de dilatation.

Variations de la fonction de perte : Nous évaluons la fonction de perte du réseau

en échantillonnant la perte d'entraînement après chaque 100 itérations pour les 1000 premières itérations, et sur les 20000 itérations suivantes de chaque point de contrôle d'entraînement, ce qui donne 32 points de données qui sont illustrés sur le graphique suivant. Nous prenons la moyenne des 100 premières itérations à chaque point de contrôle, qui est le nombre d'itérations nécessaire pour que la dynamique de l'optimisation se stabilise à une variance constante.



```
INFO net.py: 241: Uv_point_weights : (1, 1, 588, 25) => loss_Upoints
INFO net.py: 241: Uv_point_weights : (1, 1, 588, 25) => loss_Upoints
INFO net.py: 241: interp_V_resaped : (1, 1, 588, 25) => loss_Vpoints
INFO net.py: 241: V_points : (1, 1, 588, 25) => loss_Vpoints
INFO net.py: 241: Uv_point_weights : (1, 1, 588, 25) => loss_Vpoints
INFO net.py: 241: Uv_point weights : (1, 1, 588, 25) => loss_Vpoints
INFO net.py: 245: End of model: generalized rcnn
json_stats: {"accuracy_cls": 0.970703, "eta": "27 days, 14:06:21", "iter": 20000, "loss": 5.008658}
json_stats: {"accuracy_cls": 0.966797, "eta": "4 days, 1:01:33", "iter": 20020, "loss": 4.138859}
json_stats: {"accuracy_cls": 0.972656, "eta": "2 days, 18:41:36", "iter": 20040, "loss": 4.448771}
json_stats: {"accuracy_cls": 0.967773, "eta": "2 days, 18:22:02", "iter": 20060, "loss": 3.882668}
json_stats: {"accuracy_cls": 0.966797, "eta": "2 days, 19:34:16", "iter": 20080, "loss": 3.940344}
json_stats: {"accuracy_cls": 0.942383, "eta": "2 days, 21:12:59", "iter": 20100, "loss": 5.126094}
```

FIGURE 3.9 – Le résultat temps réel de l'entraînement du réseau de neurones sur COLAB.

Par exemple, au point de contrôle de 20000 itérations, la valeur de la fonction de perte se stabilise dans la moyenne de [5.008658, 4.138859, 4.448771, 3.882668, 3.940344, 5.126084] = 4.24230667.

Chapitre 4

Résultats expérimental

4.1 La procédure d'expérimentation sur le Dilated DensePose

Dans le domaine du Machine Learning et le développement des réseaux de neurones convolutionnel dédiées à différente tâche, il existe plusieurs axes de travail visée pour améliorer les performances de ces derniers. Les trois tâches de calculs sont les suivantes :

1. L'entraînement du modèle à zero, soit en modifiant l'architecture de *back-bone* ou en changeant la base de donnée .
2. L'affinement des réseaux CNN en modifiant que le module de sortie et l'entraînant sans ré-entraîner le "back-end".
3. La génération de sortie, soit sur des entrées individuelles ou sur la partie teste de la base de donnée pour évaluer la précision (en moyenne) et la capacité de généralisation du réseaux de neurones.

La configuration du modèle entraîné

Pour rapporter les résultat d'entrainement du réseaux de neurones convolutionnel de l'algorithme Dilated Densepose, on redéfinit d'abord la configuration prise. Comme il a été mentionnée, les deux architecture entraînée met en oeuvre les deux paramétrages des variables de dilatation suivantes :

$$C1 = (d1, d2, d3) = (4, 6, 16)$$

$$C2 = (d1, d2, d3) = (2, 6, 24)$$

4.1.1 La variation de la précision

Pour estimer la précision des réseaux de neurone, la deuxième phase de développement sert à calculer le paramètre de précision AP introduit au chapitre 2 qui représente le pourcentage des prédictions correctes qui sont vrai dans la base de donnée sur toute les sortie calculé et classée comme une instance positive (correcte). Les AP sont calculée pour des indices IoU (le pourcentage de surface détecté par les boites de délimitation sur toute l'union des surface de boite de délimitation prédite et correcte). Par exemple, AP@.75 veut dire qu'on calcule la précision pour une intersection de 75% de l'union des deux surfaces des boîtes de délimitation. Dans le calcul d'une précision

moyenne mAP, on prend les IoU suivante :

$$mAP = meanAP@(0.5 : 0.05 : 0.95)$$

Ou on a le minimum requis pour la détection d’objet dans une boîte c’est 50% de la surface d’union. L’ AP peut être calculé à partir des OKS présenté au chapitre 2 qui mesure la similarité entre les prédiction de points clés (ou la prédiction dense de DensePose) et les vrai valeur de la base de donnée. Pour le faire, on utilise la formule suivante pour un certain seuil s :

$$AP@s = \frac{\sum_p \sigma(OKS_p > s)}{\sum_p 1}$$

Dans l’entraînement de notre modèle de DensePose, on a établi les tâches suivante :

- On effectue l’inférence du modèle de DensePose originale. Les résultats qualitatives illustre les difficultés rencontrées par l’algorithme pour donner la bonne segmentation des corps humains présents, et par conséquent l’algorithme ne va pas donner une estimation de la pose humaine dans l’étape suivante.
- En plus, on fait le teste de précision de DensePose originale sur la partition des donnée teste de la base de donnée DensePose COCO. Ce qui nous a permis d’avoir des résultats similaire à ceux présentée dans l’article de DensePose [12].

```

('Categories:', [1])
('Final', 1.0, 0.0)
DONE (t=0.27s).
INFO json_dataset_evaluator.py: 217: ~~~~ Mean and per-category AP @ IoU=[0.50,0.95] ~~~~
INFO json_dataset_evaluator.py: 218: 53.8
INFO json_dataset_evaluator.py: 226: 53.8
INFO json_dataset_evaluator.py: 227: ~~~~ Summary metrics ~~~~
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.538
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.821
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.590
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.242
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.515
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.686
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.212
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.564
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.591
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.304
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.569
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.741
INFO json_dataset_evaluator.py: 194: Wrote json eval results to: test/dense_coco_2014_minival/generalized_rcnn/detection_results.pkl
INFO task_evaluation.py: 55: Evaluating bounding boxes is done!
    
```

FIGURE 4.1 – Les résultat de teste de précision (AP) simulé de l’algorithme DensePose

Method	AP	AP ₅₀	AP ₇₅	AP _M	AP _L	AR	AR ₅₀	AP ₇₅	AR _M	AR _L
DensePose (ResNet-50)	51.0	83.5	54.2	39.4	53.1	60.1	88.5	64.5	42.0	61.3
DensePose (ResNet-101)	51.8	83.7	56.3	42.2	53.8	61.1	88.9	66.4	45.3	62.1

FIGURE 4.2 – Les résultats de teste de précision originaux de l’algorithme DensePose [12]

- On a ensuite entraîné deux configuration de Dilated DensePose (C1 et C2) en utilisant Google Colab. Le temps d'entraînement était d'environ 3-4 jours pour chaque configuration. Mais vue que Colab permet ses utilisateurs d'une 12 heures d'utilisation affilée sur son Back-end GPU, on était obligé de faire une reconfiguration qui dure 45 minutes avant de relancer l'entraînement pour chaque 12 heures. Comme il a été mentionnée, le dataset COCO a été téléchargé et préconfigurée dans le google drive une seul fois au début, et le dossier des checkpoints d'entraînement a été mise en relation avec le drive par la commande : `!ln -s [destination de fichier de sortie d'entraînement] [destination de google drive dans colab]`. Ce dernier est pour enregistrer les checkpoint chaque 20000 itération dans google drive.
- D plus, on a évalué la fonction coût pour les deux configuration pendant l'intervalle [40 , 500000] des itérations.
- Dans un dernier point, on effectue des testes de précision sur nos deux modèles entraînée sur DensePose COCO. On a mis les notebook Colab de configuration et d'entraînement en Annex englobant toutes détails technique d'implémentation. Dans le tableau suivant, on rapporte les résultat de précision mAP moyen des checkpoints de chaque 100000 itération, pour les deux configuration C1 et C2 :

La configuration (d1, d2, d3)	Le checkpoint d'entraînement (itérations)					
	100000	200000	300000	400000	500000	600000
C1 =(4, 8, 16)	11.7	16.8	17.4	19.4	22.0	22.7
C2 = (2, 6, 24)	6.4	9	19	22.6	26.7	27.2

Comme il est apparant dans le tableau précédant, le modèle de configuration C2 a achevé la meilleur précision AP entre les deux configuration. Cette configuration présente une dilatation plus faible que C1 au niveau de CONV2, et plus forte au niveau de CONV4. Il est clair que cette augmentation permet de compenser la perte d'information au fur et à mesure qu'on avance plus profondément dans le réseau de neurones. Tandis qu'il faut mettre les paramètres de dilatation faible au niveau des couches CONV première, ceci est fait pour ne pas engendrer des temps d'entraînement irréalisable dans certaines machines.

4.1.2 L'optimisation des hyperparamètres

Dans le contexte de notre travail, nous étions capable d'entraîner les réseaux CNN de l'algorithme DensePose qu'à travers la plateforme de Colab, ce qui a limiter notre capacité de trouver des paramètre de dilatation optimaux qui nous donnent la meilleur performance concernant ce critère et le problème qu'on a introduisé au début de ce chapitre. Au faite, l'une des étape de développement des réseaux CNN, c'est l'optimisation des hyperparamètres.

Les hyperparamètres sont toutes variable qui influe la précision, complexité en temps d'entraînement et de prédiction, complexité en espace et le rappel des réseaux CNN, mais qui ne sont pas appris durant la phase d'entraînement. Par exemple, le nombre de neurones, le nombre de couche de chaque type, les paramètres de dialataion ... ect. L'optimization des hyperparamètres est un processus qui demande beaucoup de puissance de calcule pour trouver les bonne hyperparamètres du réseau de neurones. Ceci explique l'écart entre la précision de notre modèle de Dilated DensePose et le DensePose original(27.3 vs 53). Et vue que nous avons pris en considération le temps d'entraînement en priorité et non pas la précision dans le choix de ces hyperparamètres, on a trouvé cet écart.

L'algorithme de la validation croisée

L'une des approches du domaine de l'optimization des hyperpatamètres, c'est la cross validation. Au faite, Il s'agit d'une technique d'évaluation des modèles de Machine Learning par l'entraînement de plusieurs modèles sur des sous-ensembles des données d'entrée disponibles et leur évaluation sur le sous-ensemble complémentaire des données. Grâce à la validation croisée, il y a de fortes chances que nous puissions détecter facilement les sur-ajustements ou sur-apprentissage qui empêche les réseaux de CNN de généraliser leur résultat et avoir une erreur de teste minimale.

Dans un des type de validation croisée, on a l'algorithme de validation croisée à k partition, ou on divise la base de donnée d'entraînement seulement (les données de testes ne sont jamais modifiées) en k partitions. Une de ces partition est utilisé comme donnée de teste pour l'entraînement répétée qui vise à trouver les meilleurs hyperparamètres.

4.1.3 Les variations de la fonction des coûts

Pendant la phase d'entraînement end-to-end des réseaux de neurones de DensePose, on échantillonne les variations de la fonction de coût (critère d'optimization)

abordées dans les deux architecture implémentées. Les graphes de variation de la fonction de coût sont représentés dans la figure suivante :

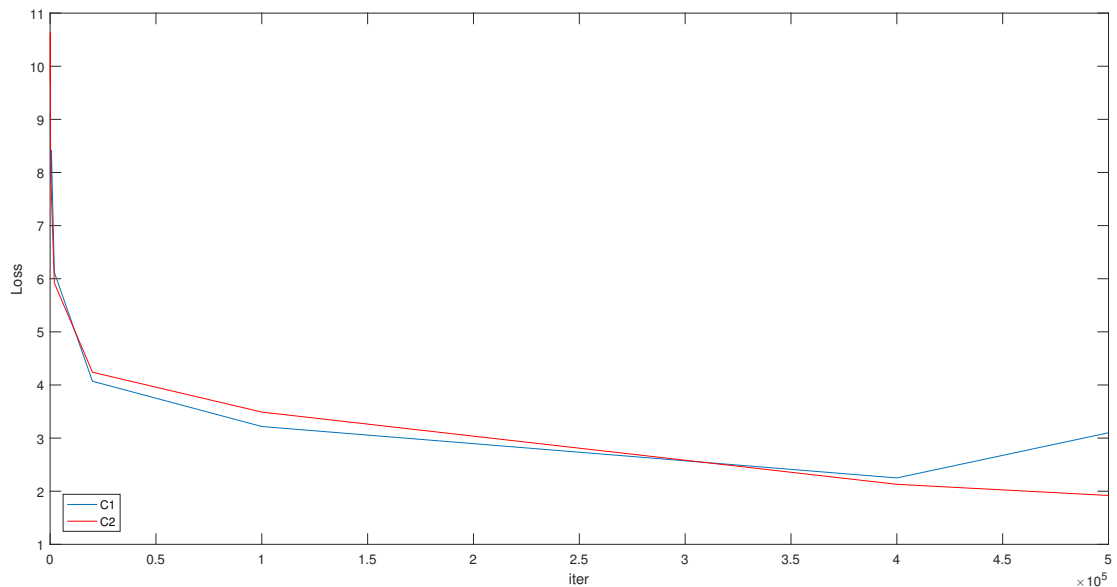


FIGURE 4.3 – Les fonctions de coût des configuration C1 et C2 en terme de nombre des itérations

Interprétation des variation de la fonction de coût

Comme dans tout autres cas d'entraînement des réseaux de neurones, la valeur de la fonction de coût des réseaux commence par une très large valeur au début de l'optimisation à cause de l'initialisation aléatoire des poids des neurones à travers le réseaux. Ce dernier permet d'avoir un point initial indépendant de la base de donnée sur laquelle on adapte les poids de notre réseaux. Il n'y a pas, cependant, une approche systématique pour initialiser les poids différemment pour différent problèmes traités par les CNN. Dans ce qui suit, une diminution brusque de la fonction coût au cours des première centaines d'itération, à cause du pas d'optimisation élevé proposée dans la loi du descente de gradient stochastique. La diminution brusque est aussi généralement associée par des petites augmentations qui ne sont pas apparente à cause de la fréquence de notre échantillonnage, ces augmentations sont due au fait que le pas élevés d'optimisation risque de sortir du point de convexité (minimum) du réseaux de neurones selon la base de donnée associé. Pour la première configuration C1, on remarque que la diminution brusque atteint la valeur 5.26 à l'itération 1000, comparé par une valeur de 5.5 dans la configuration C2. La configuration C1 présente un coût inférieur à celui de C2 avant l'itération 300000, mais au de la, la fonction coût de C2 devienne inférieur à celle de C1. D'où, la précision de C2 est meilleur que celle de C1

car les réseaux CNN de C2 s'adapte plus homogènement que ceux des C1.

4.2 Conclusion

Dans la partie expérimentale de notre travail, on a réussi à modifier l'architecture de l'algorithme DensePose 2018, pour introduire et étudier l'effet de la dilatation des couche de convolution (au niveau de CONV 2, 3, 4) sur la précision du modèle. Après la phase d'entraînement, le modèle à base des convolution dilatées a présenté une erreur supérieur à celle de l'algorithme DensePose sans dilatation. Ceci peut être due au choix des paramètre de dilatation qui peut être destructive comme il peut être avantageux.

L'utilisation de l'optimisation des hyperparamètre des variables de dilatation peuvent être utilisé pour trouvé la dilatation meilleur, en revanche, on a inspectée une comparaison entre deux modèle de DensePose qui utilise deux configuration de dilatation différentes, mais on a aboutit à des résultat qui favorisent des dilatation forte au niveau des couche CONV supérieur et faible au niveau des couche CONV inférieur.

Conclusion générale

L'estimation de pose humaine représente un problème de modélisation dans le domaine de la vision par ordinateur. Au fait, l'écart sémantique entre l'entrée de ce système d'estimation et la sortie change la forme du problème. Il transforme le problème d'une tâche de calcul et recherche de fonction hypothèse en un problème de recherche de modèle de représentation. L'estimation de pose humaine vise une variété de domaines d'application comme la Virtual Reality, l'analyse des performances sportives et même le contrôle des robots à distance.

Les développements récents des réseaux de neurones convolutionnels les ont mis en amont de la chaîne des modèles de représentation. Ce qui explique leur résultat précis dans le domaine de la vision par ordinateur. La tâche d'estimation de pose humaine est donc une partie d'un pipeline des algorithmes de résolution à aspect d'apprentissage approfondi. En effet, les réseaux de neurones convolutionnels ont dépassé les performances des algorithmes à approche classique. Ceci est dû à la disponibilité de bases de données immenses qui sont mises en œuvre pour l'entraînement des CNN dédiés à la tâche de classification, de détection d'objets ou de segmentation sémantique.

Dans le contexte de cette thèse, on a investigué l'estimation de pose humaine à aspect d'apprentissage et on a choisie l'une des architectures les plus performantes et récentes d'estimation de pose humaine, c'est l'algorithme DensePose. Cet algorithme, proposé par Facebook, a montré de meilleures performances en termes de précision et de temps d'utilisation. L'architecture est basée sur l'algorithme de segmentation sémantique Mask R-CNN qui utilise l'approche de segmentation à base de propositions de régions et un CNN de base (Back-Bone) ResNet qui est l'état de l'art des architectures d'extraction des caractéristiques des images. L'algorithme utilise aussi un modèle de corps humain appelé SMPL et fait une correspondance surfacique des pixels de l'image d'entrée avec les points de représentation de chaque partie visible du corps humain. La régression des coordonnées des pixels a donc montré une meilleure précision que les approches à base d'estimation des coordonnées des articulations du corps humain.

Notre travail est inspiré de l'algorithme Deeplab qui introduit l'opération de convolution dilatée. Cette particularité de convolution usuelle augmente leur champ réceptif dans le plan de l'image ou neurones d'entrée. Et en introduisant les modifications inspirées par cet algorithme, on a abordé une analyse de l'effet des paramètres de dilatation sur les performances de ces algorithmes d'estimation de pose humaine. On a utilisé la dilatation dans les trois couches de convolution interne sur le DensePose à base de ResNet50, et on a entraîné deux configurations de ce modèle de Dilated DensePose ou on a remarqué l'augmentation des performances avec l'augmentation de la dilatation dans la dernière couche et sa diminution dans la première couche.

Pour l'aspect futur de ce travail, on propose l'optimisation des hyperparamètres de notre modèle de Dilated DensePose. Pour atteindre le paramètre optimal et les implémenter dans le pipeline des algorithmes de reconnaissance d'action.

Bibliographie

- [1] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter V. Gehler, Javier Romero, and Michael J. Black, *Keep it SMPL : automatic estimation of 3d human pose and shape from a single image*, CoRR **abs/1607.08128** (2016).
- [2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh, *Realtime multi-person 2d pose estimation using part affinity fields*, CoRR **abs/1611.08050** (2016).
- [3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille, *Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs*, CoRR **abs/1606.00915** (2016).
- [4] Yann Le Cun, *A theoretical framework for back-propagation*, (1988).
- [5] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun, *R-FCN : object detection via region-based fully convolutional networks*, CoRR **abs/1605.06409** (2016).
- [6] N. Dalal and B. Triggs, *Histograms of oriented gradients for human detection*, **1** (2005), 886–893 vol. 1.
- [7] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman, *The pascal visual object classes (voc) challenge*, Int. J. Comput. Vision **88** (2010), no. 2, 303–338.
- [8] Haoshu Fang, Shuqin Xie, and Cewu Lu, *RMPE : regional multi-person pose estimation*, CoRR **abs/1612.00137** (2016).
- [9] Ross B. Girshick, *Fast R-CNN*, CoRR **abs/1504.08083** (2015).
- [10] Xavier Glorot and Yoshua Bengio, *Understanding the difficulty of training deep feed-forward neural networks*, **9** (2010), 249–256.
- [11] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He, *Accurate, large minibatch SGD : training imagenet in 1 hour*, CoRR **abs/1706.02677** (2017).
- [12] Riza Alp Güler, Natalia Neverova, and Iasonas Kokkinos, *Densepose : Dense human pose estimation in the wild*, CoRR **abs/1802.00434** (2018).

- [13] Riza Alp Güler, George Trigeorgis, Epameinondas Antonakos, Patrick Snape, Stefanos Zafeiriou, and Iasonas Kokkinos, *Densereg : Fully convolutional dense shape regression in-the-wild*, CoRR **abs/1612.01202** (2016).
- [14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick, *Mask R-CNN*, CoRR **abs/1703.06870** (2017).
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep residual learning for image recognition*, CoRR **abs/1512.03385** (2015).
- [16] ———, *Delving deep into rectifiers : Surpassing human-level performance on imagenet classification*, CoRR **abs/1502.01852** (2015).
- [17] Chen Li and Gim Hee Lee, *Generating multiple hypotheses for 3d human pose estimation with mixture density network*, CoRR **abs/1904.05547** (2019).
- [18] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie, *Feature pyramid networks for object detection*, CoRR **abs/1612.03144** (2016).
- [19] Jonathan Long, Evan Shelhamer, and Trevor Darrell, *Fully convolutional networks for semantic segmentation*, CoRR **abs/1411.4038** (2014).
- [20] David G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision **60** (2004), no. 2, 91–110.
- [21] Alejandro Newell, Kaiyu Yang, and Jia Deng, *Stacked hourglass networks for human pose estimation*, CoRR **abs/1603.06937** (2016).
- [22] Sungheon Park, Jihye Hwang, and Nojun Kwak, *3d human pose estimation using convolutional neural networks with 2d pose information*, CoRR **abs/1608.03075** (2016).
- [23] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G. Derpanis, and Kostas Daniilidis, *Coarse-to-fine volumetric prediction for single-image 3d human pose*, CoRR **abs/1611.07828** (2016).
- [24] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun, *Faster R-CNN : towards real-time object detection with region proposal networks*, CoRR **abs/1506.01497** (2015).
- [25] Alexander Toshev and Christian Szegedy, *Deeppose : Human pose estimation via deep neural networks*, CoRR **abs/1312.4659** (2013).
- [26] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, *Selective search for object recognition*, International Journal of Computer Vision **104** (2013), no. 2, 154–171.

- [27] Márton Véges and András Lörincz, *Absolute human pose estimation with depth prediction network*, CoRR **abs/1904.05947** (2019).
- [28] Bastian Wandt and Bodo Rosenhahn, *RepNet : Weakly supervised training of an adversarial reprojection network for 3d human pose estimation*, CoRR **abs/1902.09868** (2019).
- [29] Bin Xiao, Haiping Wu, and Yichen Wei, *Simple baselines for human pose estimation and tracking*, CoRR **abs/1804.06208** (2018).
- [30] Yi Yang and Deva Ramanan, *Articulated human detection with flexible mixtures of parts*, IEEE Trans. Pattern Anal. Mach. Intell. **35** (2013), no. 12, 2878–2890.
- [31] Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, and Yichen Wei, *Weakly-supervised transfer for 3d human pose estimation in the wild*, CoRR **abs/1704.02447** (2017).
- [32] Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei, *Deep kinematic pose regression*, CoRR **abs/1609.05317** (2016).

Annexes

.1 Installation de Pytorch/DensePose

Les instructions suivantes sont un guide pour l'installation du logiciel Pytorch sur la plate-forme Google Colab, suivi par les configurations nécessaires à l'installation de DensePose en plus du paramétrage de Dataset COCO DensePose.

```

1
2 #Start
3 # Mount the google drive account with Google Colab. The drive must contain a folder
   named #cocodataset with the Train2014/Val2014/Annotations folders from the main
   website of COCO #datasets.
4 from google.colab import drive
5 drive.mount('/content/gdrive')
6 #Install the Miniconda platform for the Python distribution associated with the
   Colab Notebook #(Python 2.7).
7 import os
8 from google.colab import files
9 !wget -c https://repo.anaconda.com/miniconda/Miniconda2-latest-Linux-x86_64.sh
10 !chmod +x Miniconda2-latest-Linux-x86_64.sh
11 !bash ./Miniconda2-latest-Linux-x86_64.sh -b -f -p /usr/local
12 # Install GCC C code Compiler of version 4.9 that s needed for the DensePose make
   operation.
13 !mkdir /usr/local/gcc-4.9-deb
14 %cd /usr/local/gcc-4.9-deb
15 !wget http://launchpadlibrarian.net/247707088/libmpfr4_3.1.4-1_amd64.deb
16 !wget http://launchpadlibrarian.net/253728424/libasan1_4.9.3-13ubuntu2_amd64.deb
17 !wget http://launchpadlibrarian.net/253728426/libgcc-4.9-dev_4.9.3-13ubuntu2_amd64.
   deb
18 !wget http://launchpadlibrarian.net/253728314/gcc-4.9-base_4.9.3-13ubuntu2_amd64.deb
19 !wget http://launchpadlibrarian.net/253728399/cpp-4.9_4.9.3-13ubuntu2_amd64.deb
20 !wget http://launchpadlibrarian.net/253728404/gcc-4.9_4.9.3-13ubuntu2_amd64.deb
21 !wget http://launchpadlibrarian.net/253728432/libstdc++-4.9-dev_4.9.3-13
   ubuntu2_amd64.deb
22 !wget http://launchpadlibrarian.net/253728401/g++-4.9_4.9.3-13ubuntu2_amd64.deb
23
24 !sudo dpkg -i gcc-4.9-base_4.9.3-13ubuntu2_amd64.deb
25 !sudo dpkg -i libmpfr4_3.1.4-1_amd64.deb
26 !sudo dpkg -i libasan1_4.9.3-13ubuntu2_amd64.deb
27 !sudo dpkg -i libgcc-4.9-dev_4.9.3-13ubuntu2_amd64.deb
28 !sudo dpkg -i cpp-4.9_4.9.3-13ubuntu2_amd64.deb
29 !sudo dpkg -i gcc-4.9_4.9.3-13ubuntu2_amd64.deb
30 !sudo dpkg -i libstdc++-4.9-dev_4.9.3-13ubuntu2_amd64.deb
31 !sudo dpkg -i g++-4.9_4.9.3-13ubuntu2_amd64.deb
32 # Install the Pytorch dependencies.

```

```
33 !conda install -y pyyaml=3.13 typing mkl=2019.1 mkl-include=2019.1 pydot networkx
    opencv \ mock scipy h5py cython matplotlib
34 !pip install -U memory_profiler
35 # Set the COCOAPI package for the processing of datasets in COCO format.
36 COCOAPI= '/usr/local/COCOAPI'
37 !git clone https://github.com/cocodataset/cocoapi.git $COCOAPI
38 %cd $COCOAPI/PythonAPI
39 # Install into global site-packages
40 !make install
41 # Install Pytorch
42 !conda install -y pytorch torchvision cudatoolkit=10.0 -c pytorch
43
44
45 #Set the Pytorch path with the Conda environment path so that all instruction from
    Pytorch #packages can be recognized and linked to their appropriate bin files.
46 import os
47 os.environ['TORCH_PATH'] = ":CONDA_PKGS_PATH/torch"
48 os.environ['CAFFE2_INCLUDE_PATH'] = ":TORCH_PATH/include/caffe2"
49 #Install protobuf 3.6.1 t protocol buffer for serialized data transfer used in
    Pytorchs package #management
50 !conda install -y protobuf=3.6.1
51 #Verify the Installation of Pytorch (Caffe2 is a Deeplearning platform that was
    originally #independent but was integrated in Pytorch, and its the basic deep
    learning back-end in Pytorch #responsible for DensePose basic deep learning
    operations) . The output should be 1 Success
52 !python2 -c 'from caffe2.python import workspace; print(workspace.NumCudaDevices())'
53 !python2 -c 'from caffe2.python import core' 2>/dev/null && echo "Success" || echo "
    Failure"
54 #DensePose installation:
55 # Clone the DensePose Repository from Github
56 import os
57 DENSEPOSE='/usr/local/densepose/'
58 !git clone https://github.com/facebookresearch/densepose $DENSEPOSE
59 %cd $DENSEPOSE
60 #Clone the Utils file from the original Detectron Github repository to replace the
    env.py h a t s #corrupted from DensePose.
61 os.chdir('/usr/local/densepose/detectron/utils/')
62 !rm env.py
63 !wget https://raw.githubusercontent.com/facebookresearch/Detectron/master/detectron/
    utils/env.py
64 #Test the DensePose installation.
65 %cd DENSEPOSE
66 !python $DENSEPOSE/detectron/tests/test_spatial_narrow_as_op.py
```



```

67 #Clone the Pytorch Github Repository to replace the Caffe2 files in the original
    Densepose directory that lacks certain file needed for configuration.
68 %cd /usr/local
69 !git clone https://github.com/pytorch/pytorch
70 !cp -avr /usr/local/pytorch/caffe2/utils /usr/local/lib/python2.7/site-packages/
    torch/include/caffe2
71 %cd /usr/local/densepose
72 #Make the following changements to the CmakeLists file after Copying it from the
    original #DensePose Repo or reading it using the Line !cat CmakeLists.txt .
    To write in a file in Colab use #the line: %%writefile CmakeLists.txt followed
    by the whole file instructions.
73 #The + is for newly added instructions, and - for deleting certain
    instructions.
74 #+set(Caffe2_DIR "/usr/local/lib/python2.7/site-packages/torch/share/cmake/Caffe2")
75 #+add_library(libprotobuf STATIC IMPORTED)
76 #+set(PROTOBUF_LIB "/usr/local/lib/libprotobuf.a")
77 #+set_property(TARGET libprotobuf PROPERTY IMPORTED_LOCATION "$ #{PROTOBUF_LIB}")
78 #+include_directories("/usr/local/lib/python2.7/site-packages/torch/lib/include")
79 #+include_directories("/usr/local/lib/python2.7/dist-packages/torch/share/cmake/
    Caffe2/")
80 #+include_directories("/usr/local/lib/python2.7/site-packages/torch/share/cmake/
    Caffe2")
81 #+include_directories("/usr/local/pkgs/pytorch-1.1.0-py2.7_cuda10.0.130_cudnn7.5.1_0
    /lib/#python2.7/site-packages/torch/share/cmake/Caffe2")
82 #-target_link_libraries(caffe2_detectron_custom_ops caffe2_library)
83 #+target_link_libraries(caffe2_detectron_custom_ops caffe2_library libprotobuf)
84 #-target_link_libraries(caffe2_detectron_custom_ops_gpu caffe2_gpu_library)
85 #+target_link_libraries(caffe2_detectron_custom_ops_gpu caffe2_gpu_library
    libprotobuf)
86 %cd /usr/local/densepose/detectron/ops
87 #Make the following changements to the pool_points_interp.h file after Copying it
    from the original #DensePose Repo or reading it using the Line !cat
    pool_points_interp.h . To write in a file in #Colab use #the line: %%
    writefile pool_points_interp.h
88 #- spatial_scale_(OperatorBase::GetSingleArgument<float>("spatial_scale", 1.))
89 #+spatial_scale_(this->template GetSingleArgument<float>(
90     "spatial_scale", 1.))
91
92 %cd /usr/local/densepose
93 !make clean
94 !pip install -r $DENSEPOSE/requirements.txt
95 %cd $DENSEPOSE/build
96 !make CC=gcc-4.9 CPP=g++-4.9 CXX=g++-4.9 LD=g++-4.9
97 !make ops CC=gcc-4.9 CPP=g++-4.9 CXX=g++-4.9 LD=g++-4.9

```

```
98 #Test The densePose operations Library installation:
99 !python $DENSEPOSE/detectron/tests/test_zero_even_op.py
100 #install the DensePose COCO annotations
101 os.chdir('/usr/local/densepose/DensePoseData/')
102 !bash get_densepose_uv.sh
103 !bash get_eval_data.sh
104 !bash get_DensePose_COCO.sh
105 #Link the COCO dataset that s was downloaded to Google drive (use the !wget URL
    instruction #from Colab)with the DensePose datasets directory.
106 %cd /usr/local/densepose
107 !ln -s /content/gdrive/'My Drive'/cocodataset /usr/local/densepose/detectron/
    datasets/data/coco
108 !cp /usr/local/densepose/DensePoseData/DensePose_COCO/densepose_coco_2014_minival.
    json /usr/local/densepose/detectron/datasets/data/coco/annotations
109 !cp /usr/local/densepose/DensePoseData/DensePose_COCO/densepose_coco_2014_train.json
    /usr/local/densepose/detectron/datasets/data/coco/annotations
110 !cp /usr/local/densepose/DensePoseData/DensePose_COCO/
    densepose_coco_2014_valminusminival.json /usr/local/densepose/detectron/datasets
    /data/coco/annotations
111
112 ##### Create an output file in google drive, and link the Detectron output with it,
    in order to save #all checkpoints in drive every 20000 iterations.
113 !mkdir /content/gdrive/'My Drive'/detectron-output
114 %cd /content/gdrive/'My Drive'
115 !ln -sf "$(pwd)/detectron-output" /tmp/detectron-output
116
117 #ENDED
```

Le dernier Colab Notebook configure l'ensemble de l'environnement pour l'entraînement, le test et l'inférence en utilisant l'algorithme d'estimation de la pose humaine "DensePose" de Facebook. La plate-forme utilisée est le service Google Colab de Cloud Computing. Afin de les installer sous Windows ou Linux, un logiciel Anaconda doit être téléchargé et installé, en plus des pilotes GPU, du support CUDA pour la bonne série de GPU (ex Tesla K80 pour Colab) et de la bibliothèque cuDNN pour le support des calculs d'apprentissage profond dans Nvidia. Seuls les GPU Nvidia sont compatibles avec les plates-formes Deep Learning qui sont basées sur les packages CUDA tel que TensorFlow, Pytorch, Caffé et Lasagne.

.2 Modèle d'estimation de pose DensePose dilaté

Après avoir installé les paquets du logiciel Pytorch et DensePose dans Google Colab comme c'était présenté dans le précédent Colab Notebook, et dans un souci de reproductibilité de nos résultats, nous présentons maintenant le code nécessaire pour créer un modèle Dilated DensePose avec la configuration des paramètres (d1, d2, d3) :

```

1
2 #Start
3 #First, enter the appropriate directory for the modeling of the ResNet backbone oin
   DensePose:
4 %cd /usr/local/densepose/detectron/modeling
5 #in the next Colab Cell, execute the following source code modifications for the
   creation of a #Dilated DensePose model
6 %%writefile ResNet.py
7 #####-
8 def add_stage( model, prefix, blob_in, n, dim_in, dim_out,
   dim_inner, dilation, stride_init=2):
9     """Add a ResNet stage to the model by stacking n residual blocks."""
10    # e.g., prefix = res2
11 for i in range(n):
12    blob_in = add_residual_block(model, '{}_{}'.format(prefix,i), blob_in, dim_in,
   dim_out, dim_inner,dilation, stride_init,
13    # Not using inplace for the last block;
14        # it may be fetched externally or used by FPN
15        inplace_sum=i < n - 1 )
16    dim_in = dim_out
17 return blob_in, dim_in
18
19 #####+
20 def add_stage(model,prefix, blob_in, n, dim_in, dim_out, dim_inner, dilation, pad=
   0, stride_init=2):
21    """Add a ResNet stage to the model by stacking n residual blocks."""
22    # e.g., prefix = res2
23    for i in range(n):
24        blob_in = add_residual_block( model, '{}_{}'.format(prefix, i), blob_in,
   dim_in, dim_out, dim_inner, dilation, stride_init,pad, # Not using
   inplace for the last block;
25        # it may be fetched externally or used by FPN
26        inplace_sum=i < n - 1 )
27        dim_in = dim_out
28    return blob_in, dim_in
29 # In the add_ResNet_convX_body, Introduce the following changes to the add_stage
   instructions #of Res2, Res3 and Res4

```

```

30 #+ s, dim_in = add_stage(model, 'res2', p, n1, dim_in, 256, dim_bottleneck,d1,d1)#
    First Dilation #parameter
31     if freeze_at == 2:
32         model.StopGradient(s, s)
33     s, dim_in = add_stage(
34         model, 'res3', s, n2, dim_in, 512, dim_bottleneck * 2, d2,d2) #Second
    Dilation Para eter
35
36     if freeze_at == 3:
37         model.StopGradient(s, s)
38     s, dim_in = add_stage(
39         model, 'res4', s, n3, dim_in, 1024, dim_bottleneck * 4, d3, d3 )#Third
    dilation parameter
40     s, dim_in = add_stage(
41         model, 'res5', s, n4, dim_in, 2048, dim_bottleneck * 8,1,1 )
42
43 #####+
44 def add_residual_block( model, prefix, blob_in, dim_in, dim_out, dim_inner,
    dilation, stride_init=2,pad=0, inplace_sum=False):
45     """Add a residual block to the model."""
46     # prefix = res<stage>_<sub_stage>, e.g., res2_3
47     # Max pooling is performed prior to the first stage (which is uniquely
48     # distinguished by dim_in = 64), thus we keep stride = 1 for the first stage
49     stride = stride_init if ( dim_in != dim_out and dim_in != 64 #and dilation == 1
    ) else 1
50 #The last comment is the instruction responsible for the stride=2 in the Convolution
    1x1 of each #residual block, if i t s not modified as presented the ResNet
    model would build, but an error of the #addition of two matrices with different
    dimensions (one would be the double of the other) in the #Top Down path of the
    FPN construction phase.
51     # transformation blob
52     tr = globals()[cfg.RESNETS.TRANS_FUNC]( model, blob_in, dim_in, dim_out,
    stride, prefix, dim_inner,pad=pad, group=cfg.RESNETS.NUM_GROUPS,
    dilation=dilation )
53     # sum -> ReLU
54     # shortcut function: by default using bn; support gn
55     add_shortcut = globals()[cfg.RESNETS.SHORTCUT_FUNC]
56     sc = add_shortcut(model, prefix, blob_in, dim_in, dim_out, stride)
57     if inplace_sum:
58         s = model.net.Sum([tr, sc], tr)
59     else:
60         s = model.net.Sum([tr, sc], prefix + '_sum')
61
62     return model.Relu(s, s)

```

```

63 #####+
64
65 def bottleneck_transformation( model, blob_in, dim_in, dim_out, stride, prefix,
    dim_inner, pad=0, dilation=1, group=1):
66     """Add a bottleneck transformation to the model."""
67     # In original resnet, stride=2 is on 1x1.
68     # In fb.torch resnet, stride=2 is on 3x3.
69     (str1x1, str3x3) = (stride, 1) if cfg.RESNETS.STRIDE_1X1 else (1, stride)
70
71     # conv 1x1 -> BN -> ReLU
72     cur = model.ConvAffine( blob_in, prefix + '_branch2a', dim_in, dim_inner,
        kernel=1, stride=str1x1, pad=0, inplace=True )
73     cur = model.ReLU(cur, cur)
74
75     # conv 3x3 -> BN -> ReLU
76     cur = model.ConvAffine( cur, prefix + '_branch2b', dim_inner, dim_inner,
        kernel=3, stride=str3x3, pad=pad, dilation=dilation, group=group,
        inplace=True )
77     cur = model.ReLU(cur, cur)
78
79     # conv 1x1 -> BN (no ReLU)
80     # NB: for now this AffineChannel op cannot be in-place due to a bug in C2
81     # gradient computation for graphs like this
82     cur = model.ConvAffine( cur, prefix + '_branch2c', dim_inner, dim_out, kernel
        =1, stride=1, pad=0, inplace=False )
83     return cur
84 ##### If the ResNet model used is based on Affine Channel Convolutions,
    the code #modifications must also be introduced to the functions:
    basic_gn_shortcut and #bottleneck_gn_transformation.
85 #ENDED

```

Désormais, pour former, tester et utiliser le modèle précédent dans Google Colab, les fichiers de configuration doivent également être modifiés afin d'être compatibles avec le core de GPU présent dans le back-end de la plate-forme Google Colab. Le code suivant, est le fichier de configuration exact que nous avons utilisé pour l'entraînement du modèle DensePose Dilaté basé sur le réseau ResNet50_FPN.

```

1
2 %%writefile DensePose_ResNet50_FPN_s1x-e2e.yaml
3
4 MODEL:
5   TYPE: generalized_rcnn
6   CONV_BODY: FPN.add_fpn_ResNet50_conv5_body
7   NUM_CLASSES: 2

```

```
8 FASTER_RCNN: True
9 BODY_UV_ON: True
10 ##### The nesxt Parameter configures the number of GPUs present on the used
    Hardware
11 NUM_GPUS: 1
12 SOLVER:
13 WEIGHT_DECAY: 0.0001
14 LR_POLICY: steps_with_decay
15 GAMMA: 0.1
16 WARM_UP_ITERS: 1000
17 WARM_UP_FACTOR: 0.1
18 # Linear scaling rule:
19 # 1 GPU:
20 BASE_LR: 0.00025
21 MAX_ITER: 720000
22 STEPS: [0, 480000, 640000]
23 # 2 GPUs:
24 #   BASE_LR: 0.0005
25 #   MAX_ITER: 360000
26 #   STEPS: [0, 240000, 320000]
27 # 4 GPUs:
28 #   BASE_LR: 0.001
29 #   MAX_ITER: 180000
30 #   STEPS: [0, 120000, 160000]
31 #8 GPUs:
32 #BASE_LR: 0.002
33 #MAX_ITER: 130000
34 #STEPS: [0, 100000, 120000]
35 FPN:
36 FPN_ON: True
37 MULTILEVEL_ROIS: True
38 MULTILEVEL_RPN: True
39 FAST_RCNN:
40 ROI_BOX_HEAD: fast_rcnn_heads.add_roi_2mlp_head
41 ROI_XFORM_METHOD: RoIAlign
42 ROI_XFORM_RESOLUTION: 7
43 ROI_XFORM_SAMPLING_RATIO: 2
44 #####The next Parameter introduces the Network parameters for the 3D pose
    inference from #####the 2D segmentation output. The Used FCN is of 8 stacked
    fully convolutional layer, of ##### and a heatMap size of 56x56
45 BODY_UV_RCNN:
46 ROI_HEAD: body_uv_rcnn_heads.add_roi_body_uv_head_v1convX
47 NUM_STACKED_CONVS: 8
48 NUM_PATCHES: 24
```

```
49 USE_DECONV_OUTPUT: True
50 CONV_INIT: MSRAFill
51 CONV_HEAD_DIM: 512
52 UP_SCALE: 2
53 HEATMAP_SIZE: 56
54 ROI_XFORM_METHOD: RoIAlign
55 ROI_XFORM_RESOLUTION: 14
56 ROI_XFORM_SAMPLING_RATIO: 2
57 ##
58 # Loss weights for annotation masks.(14 Parts)
59 INDEX_WEIGHTS : 2.0
60 # Loss weights for surface parts. (24 Parts)
61 PART_WEIGHTS : 0.3
62 # Loss weights for UV regression.
63 POINT_REGRESSION_WEIGHTS : 0.1
64 ##
65 BODY_UV_IMS: True
66 TRAIN:
67 ##### We comment the next line in order to use the weights from the last
        checkpoint and not #download the pretrained weights from the official DensePose
        respository that are incompatible #with the dilation configurations.
68 # WEIGHTS: https://dl.fbaipublicfiles.com/detectron/ImageNetPretrained/MSRA/R-50.pkl
        pkl
69 DATASETS: ('dense_coco_2014_train', 'dense_coco_2014_valminusminival')
70 SCALES: (640, 672, 704, 736, 768, 800)
71 MAX_SIZE: 1333
72 IMS_PER_BATCH: 1
73 BATCH_SIZE_PER_IM: 512
74 USE_FLIPPED: True
75 RPN_PRE_NMS_TOP_N: 2000 # Per FPN level
76 TEST:
77 DATASETS: ('dense_coco_2014_minival',)
78 PROPOSAL_LIMIT: 1000
79 SCALE: 800
80 MAX_SIZE: 1333
81 NMS: 0.5
82 FORCE_JSON_DATASET_EVAL: True
83 DETECTIONS_PER_IM: 20
84 RPN_PRE_NMS_TOP_N: 1000 # Per FPN level
85 RPN_POST_NMS_TOP_N: 1000
86 OUTPUT_DIR: ' '
87
88 ##### ENDED
```

Pour entrainer et tester le modèle de Dilated DensePose, nous utilisons les commandes

suivantes :

```
1 ##### Training:
2 !python2 /usr/local/densepose/tools/train_net.py \
3     --cfg /usr/local/densepose/configs/DensePose_ResNet50_FPN_s1x-e2e.yaml \
4     OUTPUT_DIR /tmp/detectron-output
5 ##### Test :
6 python2 tools/test_net.py \
7     --cfg configs/DensePose_ResNet101_FPN_s1x-e2e.yaml \
8     TEST_WEIGHTS /content/gdrive/ My Drive /detectron-output/train/
9     dense_coco_2014_train:dense_coco_2014_valminusminival/generalized_rcnn/
10    model_iter719999.pkl \
11    NUM_GPUS 1
12 ##### ENDED
```

Avec ceci, nous concluons la configuration du code du modèle Dilated DensePose, Les poids de nos réseaux CNN entraînés et les Notebooks ci-dessus peuvent être trouvés dans le dépôt Github : "<https://github.com/NoureddineBouhali>". Le (d1, d2, d3) peut être réglé à chaque phase d'apprentissage, mais le fichier de sortie doit être adapté en conséquence afin d'éviter tout chevauchement avec des valeurs de paramètres différentes.