Estimation de la posture humaine

Karim HOCINE
karim.hocine@etu.sorbonne-universite.fr
Student number 3801631

Anis NEHMAR
anis.nehmar@etu.sorbonne-universite.fr
Student number 3803714

***Abstract*—Dans le contexte de l'interaction homme-machine, l'estimation de la posture humaine représente un enjeu majeur, notre travail consiste à prédire la posture d'un être humain à partir de points clés (`Key Points`) avec différents réseaux de neurones convolutifs. Une évaluation des performances de plusieurs réseaux convolutifs sera menée dans le but d'effectuer une étude comparative et de déterminer quel réseau est le plus adapter pour la résolution de notre problème.**

I. Introduction

Dans le but d'estimer la pose d'un être humain en utilisant des capteurs optiques, On se concentre sur les réseaux de neurones convolutifs et plus précisément l'architecture U-NET pour la prédiction des points clés d'un humain pris dans plusieurs images avec différentes positions. Dans le cadre de ce projet, on utilise un algorithme qui génère aléatoirement des postures différents d'un `stickman` dont on connaît les points clés pour faire l'évaluation du modèle obtenu.

II. Travaux associés

Dans cette section, nous passons en revue les travaux récents sur la conception des architectures de réseaux convolutifs et les développements récents qui traite de l'estimation de la pose humaine.

Les recherches sur les architectures de réseaux sont très actives depuis l'apparition de AlexNet [1] est apparu. Tout d'abord, en utilisant des filtres plus petits, le réseau VGG [2] devient plusieurs fois plus profond que l'AlexNet et permet d'obtenir des résultats plus précis. Ensuite, les réseaux de type Highway [3] ont pu étendre la profondeur à plus de 100 couches grâce aux connexions de raccourci. Plus récemment, le DenseNet [4] surpasse le ResNet grâce à ses connexions denses.

L'architecture U-Net [5] a été proposée pour la segmentation d'images biomédicales. Elle a été utilisée pour la segmentation sémantique [6], l'alignement des visages [7], etc. **Newell et al** [8] utilisent les U-Nets empilés dans l'estimation de la pose humaine. Ils appliquent également le module résiduel [9] dans les U-Nets empilés. Récemment, certains efforts [10, 11] ont tenté d'introduire la connectivité dense [4] dans les U-Nets. Cependant, leurs raccourcis de connexion ne se font que dans un seul U-Net.

Les approches basées sur les CNNs [12, 13, 14, 15] dominent l'estimation et la prédiction de la pose humaine. **Newell et al**[8] appliquent les U-Nets empilés et obtiennent une précision d'estimation élevée. Ces types de réseaux utilisent des modules plus sophistiqués: Des modèles graphiques ou des réseaux antagonistes supplémentaires [16, 17].

III. Méthodologie

Dans cette section nous allons aborder les différentes étapes du projet. Étant donné que le code est fourni pour ce projet, la première étape est de prendre en main ce dernier pour comprendre les différentes fonctions implémentées. Des `parsers` ont été créés pour modifier les paramètres d'exécution de du programme directement à partir de du terminal `Python`, ce qui permet une manipulation plus rapide des paramètres tout en évitant de modifier le code par erreur.

Après la prise en main du code il faut lancer le programme pour effectuer un premier apprentissage avec l'architecture `VGG` mise à disposition. Cette première exécution permet de corriger les erreurs présentes dans le code et voir si l'apprentissage s'effectue sans problème et si le modèle entraîné est sauvegardé.

En apprentissage automatique, pour avoir un modèle mathématique permettant de réaliser des prédictions ou des décisions, les données d'entrée sont souvent divisées en trois jeux de données: Base d'apprentissage, de validation et de test. le code fourni dans ce projet contient les données d'entraînement utilisées pour l'ajustement des paramètres de notre modèle, et le jeux de données de test qui permet l'évaluation du modèle obtenu lors l'apprentissage, mais il manque la base de validation. Cette dernière donne une évaluation impartiale d'un ajustement du modèle sur la base d'apprentissage (par exemple à chaque epoch), et d'avoir une idée sur les performances de notre modèle. Pour créer la base de validation, des images sont choisies aléatoirement dans les quatre bases de tests. Ensuite, pour obtenir une évaluation sur cette base, un calcul de la fonction coût est effectué à la fin de chaque epoch d'entraînement, d'où la possibilité de visualiser la convergence ou la divergence de notre modèle.

La dernière étape consiste à implémenter un Réseau `U-Net`. En suivant la logique de programmation du code fourni, l'architecture se compose de deux parties : Un squelette (`Backbone`) qui est dans notre cas l'architecture du réseau `U-Net` et une tête de prédictions qui prend en entrée la sortie du (`Backbone`) pour pouvoir l'adapter aux prédictions que nous souhaitons réaliser. En suite, le résultat de la prédiction est déchiffré grâce à des cartes de Dans le cas où la prédiction est correcte, on peut vérifier que le modèle utilise les bonnes régions de l'image, et non pas des éléments du décor pour déterminer les points clés. Dans le cas où la prédiction est

mauvaise, on aimerait connaître les éléments de l'image qui ont conduit à ce résultat.

IV. Expériences

Cette section abordera les modifications apportées à l'architecture `U-Net`, puis les résultats obtenus lors de l'apprentissage seront comparés avec d'autres architectures utilisées pour résoudre le même problème.

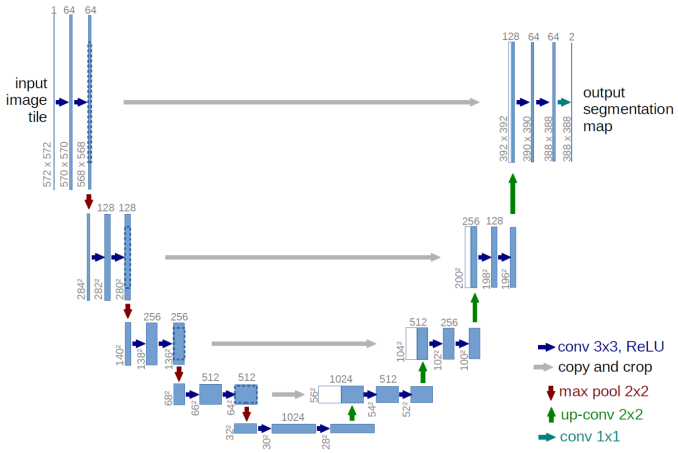

Fig. 1: General architecture of U-Net network (**Source:** https://datascientest.com/u-net).

Le réseau `U-Net` se compose d'une partie contractante et une voie expansive, ce qui lui confère une architecture en forme de U. La partie contractante est constituée d'une série d'opérations de convolution qui, chacune est suivie d'une unité linéaire rectifiée (ReLU) et d'une opération de `pooling maximum`. Pendant la contraction, les informations spatiales sont réduites tandis que les informations sur les caractéristiques (la profondeur) sont augmentées. La voie expansive combine les informations de caractéristiques et spatiales à travers une séquence de convolutions et concaténations ascendantes. Cette partie du réseau permet d'augmenter la résolution de l'image pour avoir en sortie du réseau une image de même résolution que l'image d'entrée. [5].

Ayant des ressources matérielles limitées, entraîner un réseau `U-Net` complet est très compliqué. Pour pallier à ce problème nous avons implémenté une forme réduite du réseau en réduisant le nombre de filtres utilisé lors des convolutions successives d'un facteur k jusqu'à ce que l'on arrive à un nombre de paramètres entraînables inférieurs à 1 million. Cette méthode permet de garder la forme originale du réseau. Pour éviter les problèmes de dimensions lors de la concaténations, nous choisirons des images en entrée de dimensions $h * w$, où "h" et "w" sont des puissances de 2.

Facteur k	Nb de paramètres	Temps estimé (5 epochs)
K = 1	16,988,320	9h:38m:0.00
K = 2	4,648,320	4h:5m:0.00s
k = 4	1,561,072	1h:50m:0.00s
k = 8	788,136	1h:17m:0.00s

Tableau 1 : Nombre de paramètres entraînable en fonction du facteur de réduction du nombre de filtre k..

L'architecture validée est celle avec 788,13de paramètres

Hardware	Type
CPU	intel(R) Core(TM) i7-8565U
GPU	intel(R) UHD Graphics 620
RAM	8,00 Go

Tableau 2 : Ressources matérielles utilisées.

V. Résultats

Pour Un facteur de réduction $k = 8$ Nous entérinons notre architecture sur 5 epochs avec des batch_size de 64. Nous obtenons une loss moyenne 0.226 sur la base d'entraînement et 0.107 sur la base de validation. Quant à l'évaluation sur les bases de test, nous obtenons 4,08. La fonction de coût utilisée par défaut est la `"Mean Per Joint Position Error"` (MPJPE). En analysant les résultats obtenus, on remarque que notre modèle à un problème de généralisation. Ceci dit, Le nombre d'epochs pour laquelle nous avons entraîné notre modèle étant petit, l'évaluation est peu fiable.

VI. Conclusion

En comparant les résultats obtenus par le réseau `U-Net` avec les autres réseaux utilisés, les deux réseaux `U-Net` testés sont classés derniers en matière de précision. On peut alors en déduire que ce type de réseau est peu fiable pour résoudre des problème d'estimation de pose.

VII. RÉFÉRENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for largescale image recognition. arXiv, 2014.

[3] Rupesh K Srivastava, Klaus Greff, and Jürgen S. Training very deep networks. In NIPS,2015.

[4] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In CVPR, 2017

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, 2015.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[7] Xi Peng, Rogerio S Feris, Xiaoyu Wang, and Dimitris N Metaxas. A recurrent encoderdecoder network for sequential face alignment. In ECCV, 2016..

[8] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In ECCV, 2016.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016

[10] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In CVPRW, 2017.

[11] Xiaomeng Li, Hao Chen, Xiaojuan Qi, Qi Dou, Chi-Wing Fu, and Pheng Ann Heng. H-denseunet: Hybrid densely connected unet for liver and liver tumor segmentation from ct volumes. arXiv, 2017.

[12] Ita Lifshitz, Ethan Fetaya, and Shimon Ullman. Human pose estimation using deep consensus voting. In ECCV, 2016.

[13] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern A., Mykhaylo A., Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In CVPR, 2016

[14] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In CVPR, 2016.

[15] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris Metaxas. Learning to forecast and refine residual motion for image-to-video generation. In ECCV, 2018.

[16] Yu Tian, Xi Peng, Long Zhao, Shaoting Zhang, and Dimitris N Metaxas. Cr-gan: Learning complete representations for multi-view generation. IJCAI, 2018.

[17] Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In CVPR, 2018.