# Apprentissage Contrastif et Segmentation : Exploitation des cartes de profondeur pour améliorer l'apprentissage de représentations

Ahmed BEN SAAD[1], Axel DAVY[1], Bastien HELL[2], Guillaume ERHARD[2], Gabriele FACCIOLO[1]

[1]Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, 91190, Gif-sur-Yvette, France

[2]NamR, 4 Rue Foucault, 75116 Paris, France

{ahmed.ben_saad,axel.davy,gabriele.facciolo}@ens-paris-saclay.fr
{bastienh,guillaumee}@namr.com

**Résumé –** L'apprentissage non-supervisé basé sur l'apprentissage contrastif (Contrastive Learning) a suscité un grand intérêt récemment. Ceci est dû aux excellents résultats obtenus sur une variété de tâches ultérieures (en particulier la classification) sur des ensembles de données de référence (ImageNet, CIFAR-10, etc.) sans avoir besoin d'une grande quantité d'échantillons labelisés. Cependant, la plupart des algorithmes d'apprentissage contrastif de référence (SimCLR, MoCo, BYOL) ne sont pas bien adaptés aux tâches de segmentation. Dans cet article, nous étudions une variante de BYOL récemment proposée (appelée PixelCL), qui est mieux adaptée à la segmentation et nous proposons de l'améliorer en incorporant des informations sémantiques provenant de la profondeur de l'image dans le pré-entraînement. Nos résultats montrent que PixelCL est capable d'utiliser cette nouvelle information pour améliorer les représentations apprises de manière non-supervisée.

**Abstract –** Unsupervised learning based on Contrastive Learning has received a lot of attention recently. This is due to the excellent results obtained on a variety of subsequent tasks (in particular classification) on reference datasets (ImageNet, CIFAR-10, etc.) without the need for a large amount of labeled samples. However, most reference contrastive learning algorithms (SimCLR, MoCo, BYOL) are not adapted to downstream segmentation tasks. Here, we study a recently proposed variation of BYOL (called PixelCL), which is better adapted to segmentation and we propose an improvement to it by incorporating semantic information coming from the image depth in the pretraining. Our results show that PixelCL is able to use this new information, leading to better learned representations in an unsupervised manner.

## 1 Introduction

L'apprentissage contrastif non supervisé fait l'objet de nombreux articles de recherche depuis les dernières années surtout avec l'introduction de SimCLR [1], BYOL [3], MoCo [4] et autres méthodes très performantes. L'idée est de tirer parti de la vaste quantité de données non étiquetées et plus facilement accessibles afin d'apprendre des représentations d'images qui peuvent être utilisées efficacement dans diverses tâches. Cette technique peut être utilisée avec succès comme étape de pré-entraînement, puisqu'elle peut apprendre des représentations qui maximisent l'information mutuelle entre l'entrée et les représentations [10]. Par construction, ces méthodes nous permettent d'apprendre des représentations de manière auto-supervisée de sorte qu'elles soient invariantes par rapport à l'ensemble de transformations utilisé. Il s'agit de transformations géométriques (rotation, zoom, crop, ...) ou de couleur / textures (floutage, changement de luminosité, contraste,...). On obtient donc des représentations très performantes sur des tâches globales comme la classification mais qui sont moins utiles pour des tâches qui nécessitent de l'information locale ou par pixels comme la segmentation. Ce problème a été mis en évidence par les auteurs de [11] qui, pour le résoudre, ont proposé la méthode PixelCL (*Pixel Contrastive Learning*). La

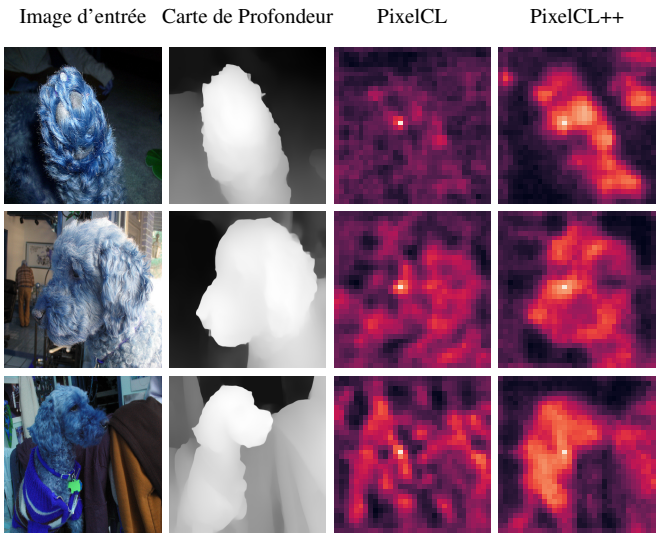


FIGURE 1 – Cartes de similarités de vecteurs de représentations obtenues avec l'algorithme PixelCL [11] et la méthode proposée PixelCL++. Le vecteur de référence est indiqué en blanc sur les similarités. La carte de profondeur est donnée à titre comparatif, elle n'est pas fournie a l'encodeur.

méthode est une version modifiée de BYOL [3] avec l'ajout d'une contrainte qui force les représentations de points spatialement proches dans les images à avoir des représentations plus proches. Ils montrent ainsi qu'elle dépasse la référence supervisée en termes de scores sur de différentes tâches de segmentation. Dans ce travail nous allons montrer que PixelCL présente une limite qui peut nuire à la qualité de ces représentations surtout sur les bords d'objets et nous proposons PixelCL++, une modification de la procédure d'entraînement de PixelCL qui permet de résoudre ce problème en utilisant l'information de profondeur monoculaire. Nous allons ensuite montrer via des expériences que grâce à PixelCL++, nous obtenons des représentations qui conduisent à des meilleurs résultats dans deux tâches de segmentation sémantique : Cityscapes [2] et la segmentation de pans de toits. La Figure 1 illustre la cohérence spatiale des représentations apprises.

# 2 Apprentissage de représentations intégrant l'information de profondeur

Dans cette partie nous allons d'abord décrire la succession des travaux connexes qui mènent à notre contribution PixelCL++. Puis nous allons présenter cette méthode et expliquer pourquoi elle résulte à des représentations de qualité meilleure que PixelCL.

## 2.1 Travaux connexes

L'apprentissage contrastif a suscité un grand intérêt les années dernières surtout avec l'apparition de BYOL [3]. L'objectif étant d'apprendre de manière auto-supervisée des représentations d'images transférables vers de différentes autres tâches supervisées, les auteurs partent du principe que ces représentations doivent être invariantes par rapport à un ensemble de transformations choisi au préalable. Pour cela, ils commencent par générer deux versions transformées $T(x)$ et $T'(x)$ d'une image d'entrée $x$. La première représentation $z_\theta$ est obtenu en faisant passer $T(x)$ par un encodeur $f_\theta$ suivi d'un projecteur MLP $g_\theta$. $T'(x)$ passe par un $momentum\ encoder$ $f_\xi$ suivi d'un projecteur $g_\xi$. Le $momentum\ encoder$ est un réseau de même architecture que l'encodeur $f_\theta$ mais dont les poids sont obtenus en faisant la moyenne glissante exponentielle des différents versions de $f_\theta$ au cours de l'entraînement.

$$z_\theta = g_\theta(f_\theta(T(x))) \tag{1}$$
$$z'_\xi = g_\xi(f_\xi(T'(x))) \tag{2}$$
$$\xi \leftarrow \tau\theta + (1-\tau)\xi. \tag{3}$$

Afin d'assurer que les représentations de $T(x)$ et $T'(x)$ soient similaires sans qu'il y ait de mode collapse, les auteurs introduisent un predicteur $q_\theta$. Ce prédicteur doit être capable de prédire $z_\xi(x)$ à partir de $z_\theta(x)$. La fonction de coût utilisée est la suivante :

$$\mathcal{L}_{BYOL} = ||\bar{q_\theta}(z_\theta) - \bar{z'_\xi}^2||_2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi \rangle}{\left\|q_(z_\theta)\right\|_2 \cdot \left\|z'_\xi\right\|_2}. \tag{4}$$

Cette fonction permet donc de rapprocher l'angle entre la sortie du prédicteur $q_\theta(z_\theta)$ et la sortie de l'encodeur $z_\theta$. Les auteurs prouvent que BYOL est très performant et fait mieux que la référence supervisée pour des différentes tâches de classification. Mais de sa conception, cette méthode apprend des représentations globales qui incorporent de l'information sur toute l'image. Elle détruit aussi toute donnée spatiale présente sur l'image du fait que $z_\theta$ est un vecteur unidimensionnel. Ce qui n'est pas utile pour la segmentation puisqu'on a intérêt que le réseau de neurones se concentre sur les propriétés locales des objets dans une image ainsi que leurs positions.Chaque pixel des cartes de représentations est ensuite "remappé" à l'espace de l'image originale. La distance spatiale normalisée entre chaque paire de pixels des cartes de représentations est calculée puis les distances normalisées sont utilisées pour générer les pairs de pixels positifs et négatifs en utilisant un seuil $\mathcal{T}$ :

$$A(i,j) = \begin{cases} 1, & \text{si } dist(i,j) \leq \mathcal{T} \\ 0, & \text{si } dist(i,j) > \mathcal{T} \end{cases}. \tag{5}$$

Pour assurer la régularité spatiale des représentations apprises, les auteurs ont ajouté un module d'attention alimenté par les cartes de représentations de sortie. Cette propagation a un effet de lissage sur ces cartes, ce qui conduit à des solutions plus cohérentes entre les pixels dans les tâches de prédiction au niveau du pixel. Pour encourager la cohérence de toutes les représentations de tous les encodeurs, les auteurs ont utilisé la fonction de coût suivante :

$$\mathcal{L}_{pixpro} = -cos(y_i, x'_j) - cos(y_j, x'_i), \tag{6}$$

où $i$ et $j$ sont une paire de vecteurs positifs provenant respectivement de deux vues d'augmentation selon la règle d'affectation $A$, $x'_i$ et $y_i$ sont des vecteurs de représentations de pixels des encodeurs $f_\theta$ et $f_\xi$. Cette perte est moyennée sur toutes les paires positives pour chaque image, puis sur les images d'un mini-batch pour conduire la procédure d'apprentissage. Le coût total utilisé est :

$$\mathcal{L}_{total} = \mathcal{L}_{pixpro} + \alpha\mathcal{L}_{BYOL}, \tag{7}$$

où $\alpha$ est un hyperparamètre fixé avant l'apprentissage. On garde donc la branche BYOL avec PixelCL pour ajouter plus d'informations sémantiques aux représentations.

## 2.2 Présentation de PixelCL++

PixelCL [11] étend BYOL en incorporant un module $pixpro$ qui est une sorte de coût contrastif local avec des pairs de crops extraits de la même image. Cela peut engendrer plusieurs problèmes :

— Le seuil (5) est un hyperparamètre fixé à une valeur arbitraire sur la seule base du fait que les pixels voisins appartiendraient au même objet.

— Il peut produire des signaux contradictoires, notamment sur les bords qui séparent deux objets différents dans une image car il ne reflète pas une similarité/différence structurelle dans les patchs recadrés,

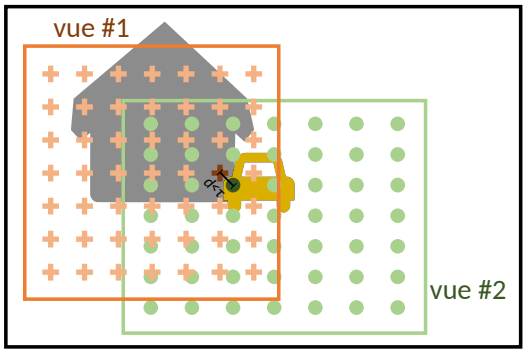


FIGURE 2 – Exemple de cas limite pour PixelCL : les points et les plus représentent respectivement les vecteurs de représentations des vues 1 et 2. Les deux pixels séparés par le segment noir sont proches sur l'image ($d < \tau$) malgré qu'ils appartiennent à deux objets différents.

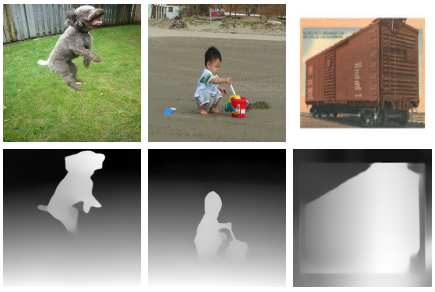

FIGURE 3 – Exemple de cartes de profondeurs générées par Midas sur ImageNet.

Le deuxième problème est important et illustré dans la Figure 2. Les signes plus et les ronds correspondent aux vecteurs de représentations de la première et deuxième vue respectives. Si on s'intéresse aux vecteurs distingués par des couleurs sombres, on remarques qu'ils appartiennent à deux objets différents malgré la proximité sur l'image. PixelCL aura tendance à considérer cette paire de vecteurs comme positive et cela nuit à la qualité des représentations.

Pour résoudre ce problème, nous proposons de modifier la procédure d'affectation des paires de vecteurs positifs ou négatifs de manière à ce qu'elle puisse intégrer certaines informations sémantiques supplémentaires. Les cartes de profondeurs sont une bonne solution car les changements de profondeur peuvent être un indicateur de changement d'objet. Nous avons donc considéré les cartes de profondeur comme second filtre d'affectation.

Nous commençons par produire deux cartes de représentation de deux patchs différents, exactement comme cela est fait pour PixelCL. En même temps, nous calculons la carte de profondeur de l'image originale en utilisant l'algorithme Midas [8] (une méthode d'estimation de la profondeur monoculaire) et nous découpons les parties correspondant aux découpages de l'image d'entrée. Les recadrages de profondeur sont ensuite mis en correspondance avec la taille des cartes de représentations pour obtenir une valeur de profondeur par pixel de carte. Après avoir appliqué la procédure de sélection de l'équation (5), nous filtrons encore les paires positives en éliminant toutes les paires dont la différence de valeur de profondeur est supérieure à un nouveau seuil $\mathcal{T}'$ :

$$A_{depth}(i,j) = \begin{cases} 1, & \text{si } |depth(i) - depth(j)| \leq \mathcal{T}' \\ 0, & \text{si } |depth(i) - depth(j)| > \mathcal{T}' \end{cases}, \quad (8)$$

$$A_{final}(i,j) = A(i,j) \times A_{depth}(i,j). \quad (9)$$

En pratique, on fixe $\mathcal{T} = 0.7$ comme pour PixelCL et on choisit empiriquement $\mathcal{T}' = 0.2$. Nous conservons le même module de propagation des pixels et le même terme de perte que PixelCL. Il est important de noter que les cartes de profondeur ne rentrent pas dans le calcul de gradient et ne sont jamais vues par le réseau au cours de l'entraînement. Elles sont juste utilisées pour la procédure d'affectation de paires positives ou négatives dans le calcul de coût.

# 3 Expériences

Nous avons mené des expériences en utilisant PixelCL++ sur deux tâches de segmentation : celle des pentes de toit ainsi que la segmentation de Cityscapes pour évaluer ses performances et nous l'avons comparé à PixelCL. Le calcul de cartes de profondeur sur des images aériennes est une tâche complexe en soi car elle nécessite d'abondantes données annotées et seuls quelques travaux réussis existent dans la littérature [7, 6]. Nous avons donc décidé de pré-entraîner la nouvelle méthode en utilisant un sous-ensemble d'ImageNet qui se compose des 200 classes présentes dans TinyImageNet [1], mais nous utilisons les images à l'échelle originale. Midas [8] fonctionne très bien sur ces images et produit des cartes de profondeur lisses et plausibles (voir Figure 3). Les deux réseaux pré-entraînés seront mis à la disposition de la communauté.

**Comparaison avec PixelCL** Pour résoudre la tâche de segmentation de pans de toits, on commence par pré-entraîner un ResNet34 [5] qui a été pré-appris sur ImageNet avec PixelCL ou PixelCL++. Les réseaux obtenus sont ensuite utilisés comme encodeurs pour deux U-Nets [9] qui sont affinés avec les données étiquetées. Nous comparons les performances des deux U-Nets dans le Tableau 1. On observe un saut considérable de 0.02 en précision et IoU moyenne sur Cityscapes. Le gain de performance en IoU moyenne monte jusqu'à 0.14 dans le cas de segmentations de pans de toits. La Figure 4 présente quelques exemples qualitatifs où nous pouvons voir que PixelCL++ est beaucoup plus performant que PixelCL. En effet, on observe que les cartes de segmentations obtenues avec PixelCL++ affichent moins d'objets confondus surtout avec l'arrière-plan. Ce qui donne des segmentations de bâtiments plus correctes.

Nous pouvons constater que l'ajout de l'information de profondeur dans la procédure de formation a permis au réseau

1. https ://www.kaggle.com/competitions/tiny-imagenet/data

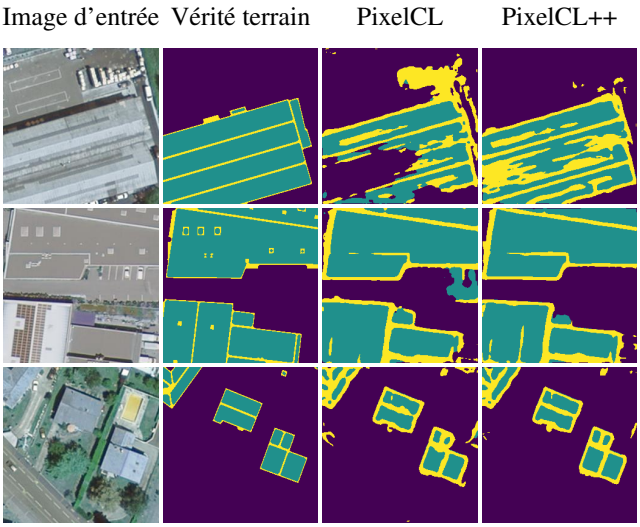


FIGURE 4 – Comparaison qualitative de PixelCL et PixelCL++ sur la tâche de segmentation sémantique de pans de toits. Jaune : bords de toits, Vert : surfaces de toits.

TABLE 1 – Comparaison quantitative de PixelCL et PixelCL++ sur une tâche de segmentation sémantique.

| Base de données | Roof Slopes | | Cityscapes | |
|---|---|---|---|---|
| Méthode | PixelCL | PixelCL++ | PixelCL | PixelCL++ |
| Précision | 0.85 | **0.87** | 0.74 | **0.76** |
| IoU moy. | 0.56 | **0.69** | 0.75 | **0.77** |

d'obtenir de meilleures performances quantitatives et qualitatives. Il convient de répéter ici que, pendant la phase de pré-entraînement, les cartes de profondeur ne sont pas introduites directement dans le réseau. Elles ne sont utilisées que pour la sélection des paires positives et négatives.

**Évaluation de la qualité des représentations apprises** Pour évaluer et comparer les représentations apprises par PixelCL et PixelCL++, nous avons mené l'expérience suivante : On prend une image et on la fait passer par l'encodeur de chaque méthode pour obtenir les représentations respectives. Chaque vecteur de la carte de représentations représente une petite partie de l'image d'entrée. On sélectionne un de ces vecteurs correspondant à un objet (chien, bâtiment,...) et on mesure la similarité avec tous les autres vecteurs des autres bouts de l'image et on les représente à la Figure 1 sous forme de carte de chaleur (les deux dernières colonnes). On remarque que pour PixelCL++, on peut voir que les vecteurs qui correspondent aux patchs d'un même objet sont très similaires, ce qui explique que les formes des objets en question soient visibles sur les cartes de similarités. Ces formes sont moins évidentes pour PixelCL. Cela veut dire que PixelCL++ arrive mieux à faire la part des objets sur l'image grâce à l'information de profondeur utilisée pour sélectionner les paires positives et négatives.

# 4 Conclusion

Les résultats obtenus en incorporant le préalable de profondeur dans le processus d'apprentissage de PixelCL sont très encourageants. Il semble que cette information permette de focaliser les représentations apprises afin qu'elles soient mieux transférées à une tâche de segmentation ultérieure. Ce qui fait que les représentations apprises par PixelCL++ sont plus sémantiquement consistantes et mieux transférables.

**Remerciements** Ces travaux ont bénéficié d'un accès aux moyens de calcul de l'IDRIS au travers de l'allocation de ressources 2022-AD011011801R1 attribuée par GENCI.

# Références

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020.

[2] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset. In *IEEE CVPR Workshop on the Future of Datasets in Vision*, volume 2, 2015.

[3] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*, 33 :21271–21284, 2020.

[4] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE CVPR*, pages 9729–9738, 2020.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] M. Hermann, B. Ruf, M. Weinmann, and S. Hinz. Self-supervised learning for monocular depth estimation from aerial imagery. *ISPRS*, V-2-2020 :357–364, aug 2020.

[7] L. Mou and X. X. Zhu. IM2HEIGHT : Height Estimation from Single Monocular Imagery via Fully Residual Convolutional-Deconvolutional Network. feb 2018.

[8] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation : Mixing datasets for zero-shot cross-dataset transfer. *IEEE PAMI*, 2020.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net : Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[10] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv :1807.03748*, 2018.

[11] Z. Xie, Y. Lin, Z. Zhang, Y. Cao, S. Lin, and H. Hu. Propagate yourself : Exploring pixel-level consistency for unsupervised visual representation learning. In *IEEE CVPR*, pages 16684–16693, 2021.