INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

LaBRI

Exploration de méthodes d'apprentissage profond pour l'analyse comparative de données de cellules uniques en vue d'une médecine personnalisée

Théo Berthet

Stage de fin de 4e année

Du 18/04/2024 au 26/07/2024

Professeur référent :
Sergio Peignier

Encadré par :
Victoria Bourgeais
Patricia Thébault
Raluca Uricaru

Remerciements

Tout d'abord, je tiens à remercier Victoria Bourgeais. Merci de m'avoir accueilli dans l'équipe BKB, merci pour vos conseils et votre soutien tout au long du stage. Votre encadrement fut particulièrement fructueux pour la réalisation de ce stage. Je remercie également Patricia Thébault et Raluca Uricaru pour leurs questionnements, leur réflexion et leur aide qui m'ont également permis d'énormément progresser.

Je ne voudrais pas oublier de remercier Brice Vergnou, mon partenaire de projet. Merci pour ta collaboration et tes questions toujours intéressantes et stimulantes ainsi que de ton investissement dans ce projet.

Ce stage n'aurait pas été possible sans l'aide et le soutien de toutes ces personnes et je leur en suis infiniment reconnaissant.

Table des matières

Introduction

Dans le cadre de mon stage de 4e année à l'INSA, j'ai rejoint le Laboratoire Bordelais de Recherche Informatique dans l'équipe BKB afin de me pencher sur un sujet me paraissant particulièrement intéressant. Il s'agit de l'utilisation de méthodes d'apprentissage profond dans le cadre de l'analyse de données de cellules uniques (single cell RNA-seq / scRNA-seq). L'amélioration de l'analyse de ce type de données, présentant de nombreux challenges, s'avère être un enjeu crucial avec des applications notamment en médecine personnalisée. Dans ce cadre, l'implication de modèles d'apprentissage profond, technique qui a su faire ses preuves dans de nombreux domaines, semble être prometteur [6].

Ainsi, après une brève présentation du laboratoire et une mise en contexte de la mission, une présentation des méthodes employées sera réalisée dont notamment scDeepCluster et Contrastive-sc. L'étude de la reproductibilité et une validation de mon implémentation sera effectuée avant de se pencher, avec un autre jeu de données plus déséquilibré, sur les hyperparamètres des différentes techniques pour finir sur une comparaison entre elles et avec une méthode de référence la PCA (Analyse en Composantes Principales).

1 Le laboratoire

1.1 Présentation du laboratoire

Le LaBRI ou Laboratoire Bordelais de Recherche Informatique est une unité de recherche associée au CNRS (UMR 5800), à l'université de Bordeaux, à Bordeaux INP et partenaire de l'Inria depuis 2002.

La structure réunit environ 300 personnes dont 110 enseignants-chercheurs et 40 chercheurs. Trois axes majeurs constituent les missions du LaBRI, la recherche par celle menée au LaBRI qu'elle soit fondamentale ou finalisée dans le domaine de l'informatique au sens large, la formation, essentiellement via les doctorants ainsi que des actions auprès de plus jeunes publics et le transfert par la collaboration avec des industriels et la création de start-up.

Le laboratoire comprend 14 équipes de recherches regroupées en cinq départements de recherche :

- Combinatoire et Algorithmique (CombAlgo)
- Image et Son (I&S)
- Méthodes et Modèles Formels (M2F)
- Supports et AlgoriThmes pour les Applications Numériques hAutes performanceS (SATANAS)
- Systèmes et données (SeD)

1.2 Présentation de l'équipe

Ce dernier département comprend deux équipes : Progress et BKB (Bench to Knowledge and Beyond). L'équipe BKB dont je faisais partie a pour objectif d'améliorer la récupération, le traitement, la modélisation, l'analyse et la visualisation des données. Les principaux défis auxquels sont confrontées l'équipe sont le volume et la variété des données. En effet, l'une des particularités de l'équipe est qu'elle s'intéresse principalement à des données et à des problèmes concrets. Ainsi, il s'agit de recherches multidisciplinaires en collaboration avec des experts des données en question que ce soit en santé, en sciences humaines et sociales ou bien même en géographie.

1.2.1 Réunions de projet

Chaque semaine, une réunion en petit comité a lieu permettant de partager les recherches et résultats de la semaine, d'en discuter et d'approfondir les analyses. Il s'agit d'un temps de partage constituant également l'occasion d'établir des pistes de recherches pour la suite et donnant la possibilité aux stagiaires d'obtenir l'avis de ses superviseurs et plus généralement de personnes plus expérimentées.

1.2.2 GT et séminaires

Chaque jeudi, à 10h, ont lieu les GT ("Groupes de Travail"). Ainsi, chaque semaine, quelqu'un, généralement un élève de master ou de doctorat, présente son travail devant des membres de l'équipe BKB, l'occasion d'obtenir des feedbacks, des pistes d'exploration et d'amélioration quant au travail réalisé mais également de s'entraîner par exemple à une soutenance.

Certaines semaines, les GT sont suivis de séminaires axés santé numérique, de présentation de travaux de chercheurs du LaBRI ou de l'extérieur. Cela donne un aperçu des méthodes de travail d'un chercheur et permet de se tenir au courant des nouvelles avancées et d'en apprendre plus sur certains domaines.

2 Contexte de la mission

Les données de séquençage sur cellule unique (scRNA-seq) est une technologie permettant de profiler l'expression des gènes à l'échelle d'une seule cellule. Pour obtenir ces données, les cellules d'un tissu sont isolées (par des méthodes de microfluidique par exemple), puis lysées afin de libérer l'ARN qui sera amplifié afin d'en obtenir une quantité suffisante pour le séquençage. Chacune de ces séquences comporte notamment un code barre correspondant à la cellule d'origine. Après séquençage et alignement, les reads provenant d'une même cellule sont regroupés grâce au code barre permettant d'établir une matrice de comptage, c'est-à-dire une table comportant en ligne les gènes et en colonne les

cellules. Chaque valeur dans la matrice correspond au nombre de lectures d'un gène dans une cellule. Les données transcriptomiques de single cell RNA-seq se caractérisent par un nombre important de variables (les gènes) en comparaison au nombre d'observations (les cellules) et en cela ce type de données est soumis un problème connu sous le nom de "curse of dimensionality" ou en français "fléau de la dimension". Le nombre élevé de variables rend plus complexe la visualisation et surtout dans notre cas, le clustering. Différentes approches existent pour essayer de gérer ce problème notamment en réduisant le nombre de dimensions par une projection des données dans des espaces réduits.

Le but du clustering, ici, est de regrouper les cellules selon leur type cellulaire. Lorsque ces clusters sont définis, une étude plus approfondie peut permettre d'apposer des labels (le type cellulaire) sur chacun des clusters en s'intéressant par exemple aux gènes marqueurs exprimés dans les clusters. Le clustering est une étape importante de l'analyse de données de scRNA-seq qui permet, notamment, dans le cadre de la médecine personnalisée de caractériser avec précision les tissus d'un patient afin de mettre en place un traitement personnalisé en fonction des profils cellulaires identifiés. Traditionnellement, le clustering est opéré sur les données après réduction de dimension par PCA (Analyse en Composantes Principales) en utilisant des algorithmes de clustering comme K-means, Louvain ou Leiden.

Une autre manière de s'atteler au problème de la dimensionnalité des données est d'employer des techniques d'apprentissage profond (deep learning). En effet, l'apprentissage profond est une avancée majeure de l'intelligence artificielle de ces dernières années qui s'est imposée dans de nombreux domaines en présentant des résultats bien supérieurs aux méthodes précédentes notamment pour la représentation de données de grandes dimensions d'où l'émergence de ces méthodes en transcriptomique [6]. Une des possibilités est d'utiliser un auto-encodeur (AE), représenté en figure 1, permettant de projeter les données dans un espace réduit de manière non linéaire grâce à un encodeur, puis de reconstruire les données d'entrée depuis l'espace réduit avec un décodeur. Il s'agit d'une méthode d'apprentissage non-supervisée car les labels ne sont pas utilisés dans le calcul de la fonction de perte.

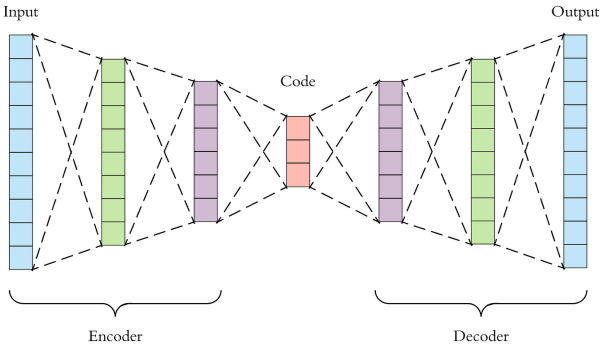


FIGURE 1 – Schéma d'un auto-encodeur

Différentes méthodes utilisant des AE pour le clustering de données de scRNA-seq ont déjà été mises en place. DCA (Deep Count Autoencoder) utilise un auto-encodeur en proposant de remplacer la fonction de perte MSE (Mean Square Error) par une fonction de perte reposant sur une loi ZINB (Zero-Inflated Negative Binomial) qui semble mieux correspondre aux données de scRNA-seq en permettant notamment l'imputation de valeurs pour les zéros techniques de la matrice de comptage aussi appelés "dropout" [4]. En effet, un des autres challenges des données de scRNA-seq est la présence de nombreux zéros dans la matrice de comptage : il s'agit d'une matrice creuse. Cependant parmi ces zéros certains ont une origine biologique et d'autres une origine technique. La méthode de séquençage est ici prône à obtenir de "faux" zéros de par la faible profondeur de séquençage par cellule [9]. La méthode apporte alors une solution efficace à la discrimination et l'imputation de valeur pour ces zéros techniques.

Cette méthode a été reprise en utilisant un auto-encodeur avec la fonction de perte ZINB mais en y associant le clustering à l'auto-encodeur avec une fonction de perte de clustering visant ainsi à optimiser le clustering tout en réalisant la réduction de dimension. Il s'agit de la méthode scDeepCluster [9] qui a donc ajouté une tâche de clustering sur l'espace latent en employant la divergence de Kullback-Leibler (KLD) suivant l'algorithme de "Deep Embedding Clustering" (DEC) [11].

D'autres manières de réduire la dimension des données de scRNA-seq tout en considérant les phénomènes de "dropout" se retrouvent dans la méthode Contrastive-sc [3] et la méthode scMAE [5]. Pour Contrastive-sc, il n'est plus question d'auto-encodeur mais d'encodeur uniquement. Cette technique s'inspire de travaux effectuer sur des images [2] en les adaptant pour l'étude de données transcriptomiques. Pour chaque cellule, deux vues différentes sont créées en appliquant un dropout qui va de manière aléatoire fixer à zéro la valeur de comptage de certains gènes. Ces deux vues différentes sont données au même encodeur et une fonction de perte contrastive permet de rapprocher les vues d'une même cellule et de les éloigner si elles viennent d'une cellule différente. Pour scMAE, il s'agit d'un auto-encodeur associé à un masque. Dans les deux cas, le clustering s'applique a posteriori sur l'espace latent.

Enfin, la méthode scDeepCluster a elle-même été reprise par scACE [7] et scziDesk [1]. La méthode scziDesk propose d'ajouter une troisième fonction de perte permettant d'optimiser le clustering avec un soft K-means sur l'espace latent. scACE, quant à lui, emploie pour l'initialisation des clusters, l'algorithme de Leiden avec une forte résolution pour obtenir un nombre conséquent de clusters mais en favorisant leur pureté pour ensuite fusionner certains clusters durant l'entraînement.

Ainsi, le projet consiste à étudier certaines de ces avancées notamment scDeepCluster, une méthode avec clustering combiné à l'entraînement de l'auto-encodeur, et Contrastive-sc, une méthode impliquant un clustering a posteriori de l'entraînement, mais également à

évaluer la pertinence de ces méthodes d'apprentissage profond par rapport aux méthodes traditionnelles en les comparant notamment à la PCA. Différentes questions se sont alors posées : Comment effectuer le pré-processing des données ? Comment évaluer le clustering ? Comment comparer les méthodes entre-elles ? Quels paramètres sont importants pour chaque méthode ? Est-ce qu'une méthode se démarque des autres ?

J'ai donc mené une étude de la reproductibilité des résultats (partie 4.1), l'exploration d'hyperparamètres importants (fine-tuning) (parties 4.2.1, 4.2.2 et 4.2.3) ainsi que la comparaison de méthodes (partie 4.2.4) suivant la méthode décrite sur la figure 2. Ce travail a été réalisé en collaboration avec Brice Vergnou, étudiant en fin de deuxième année de classes préparatoires qui s'est principalement penché sur le pré-processing des données de scRNA-seq.

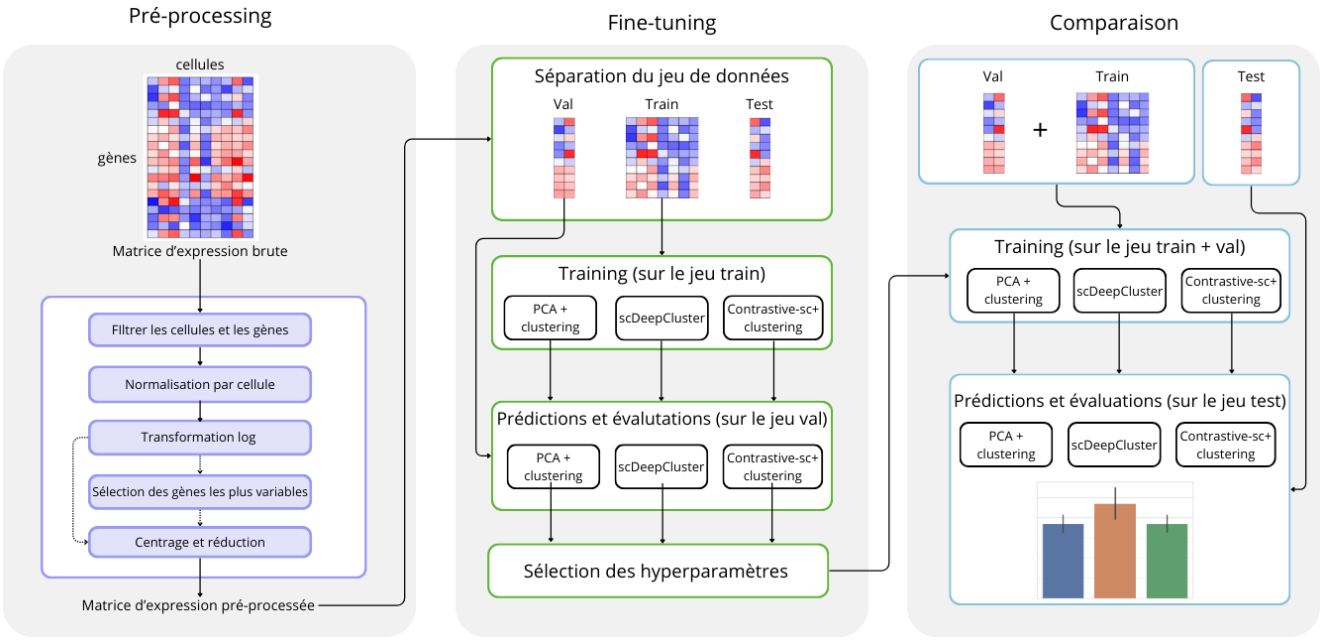


FIGURE 2 – Workflow des expériences menées (sans l'étude de reproductibilité)

3 Matériels et méthodes

3.1 Les données

Les données utilisées sont des matrices de comptage brute des gènes par cellule. Il s'agit de données de scRNA-seq. Les jeux de données utilisés ici sont publiques et comportent les annotations (vérité terrain) qui permettront de vérifier la qualité du clustering lors de l'évaluation des méthodes.

3.1.1 Le jeu de données PBMC 10X 4k

Le jeu de données PBMC 4K est issu du séquençage de cellules mononucléaires sanguines d'un individu sain. Dans notre projet, il s'agit de données prétraitées disponibles directement grâce au papier de scDeepCluster [9]. Les labels ont donc été définis en amont et la répartition des cellules dans chaque label est représentée dans la figure 3. Pour ce jeu de données, nous n'avons pas accès au type cellulaire correspondant à chaque label. Le jeu fourni a déjà été filtré et contient au total 4 271 cellules et 16 653 gènes. A l'origine, il comportait 4 340 cellules et 33 694 gènes.

label	
1	1292
2	702
3	606
4	459
5	450
6	332
7	295
8	135

FIGURE 3 – Répartition des types cellulaires de PBMC 10X 4k

3.1.2 Le jeu de données Baron

Le jeu de données Baron (disponible depuis la librairie R scRNAseq `https://www.bioconductor.org/packages/release/data/experiment/html/scRNAseq.html`) comporte les données obtenues à partir de cellules pancréatiques humaines provenant de quatre donneurs. Le nombre de gènes est de 20 125 et le nombre de cellules est de 8569. La répartition des cellules entre les différents types cellulaires est représentée dans la figure 4 et démontre une très forte asymétrie.

label	
beta	2525
alpha	2326
ductal	1077
acinar	958
delta	601
activated_stellate	284
gamma	255
endothelial	252
quiescent_stellate	173
macrophage	55
mast	25
epsilon	18
schwann	13
t_cell	7

FIGURE 4 – Répartition des types cellulaires de Baron

3.2 Le pré-processing

Les données ont été traitées en utilisant le module Python Scanpy et en suivant la même manière de procéder que dans le cas de scDeepCluster [9]. D'abord, la matrice de comptage brute a été filtrée pour ne garder que les cellules avec au moins un gène exprimé et les gènes exprimés dans au moins une cellule. Ensuite, une normalisation par cellule est effectuée suivie d'une transformation logarithmique et d'une étape de centrage-réduction. Pour le jeu Baron, une étape de sélection des gènes présentant la plus grande variabilité a été appliquée. Dans ce cas, les 2000 gènes les plus variables ont été retenus car ce choix semblait être optimal après plusieurs essais.

3.3 Les méthodes d'apprentissage profond

Les méthodes d'apprentissage profond scDeepCluster et Contrastive-sc ont toutes les deux été réimplémentées en Python 3.10, Keras 3.3.3 et Tensorflow 2.16.1 (bibliothèques

permettant l'implémentation de méthodes d'apprentissage profond).

3.3.1 L'implémentation de scDeepCluster

Originellement, scDeepCluster a été implémenté en Python 3.6. J'ai réimplémenté cette méthode en Python 3.10 ce qui m'a permis de m'approprier le code et la méthode en l'adaptant sur une version plus récente de Python. Le principe de cette technique est le suivant :

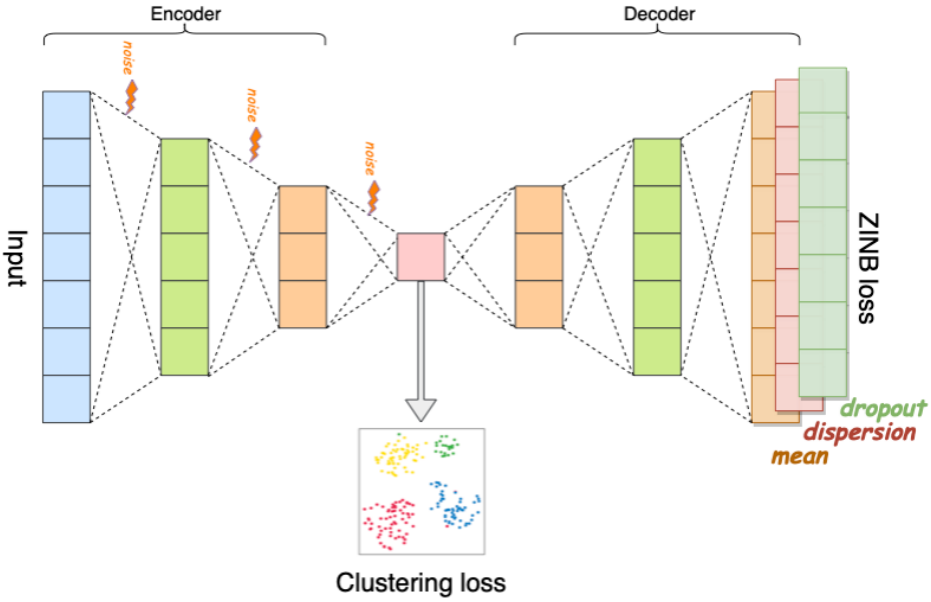


FIGURE 5 – Schéma de l'architecture de scDeepCluster (source : [9])

Comme il est visible sur la figure 5, il s'agit d'un auto-encodeur avec deux fonctions de perte. La méthode suit trois grandes étapes :

- **Pré-entraînement de l'auto-encodeur avec la fonction de perte ZINB uniquement :**

La partie encodeur comprend l'insertion de bruit gaussien entre les différentes couches. Il s'agit donc également d'un "denoising" auto-encodeur ou auto-encodeur débruiteur dont l'objectif est d'être plus robuste aux données bruitées. Le décodeur renvoie trois couches, chacune de la même taille de que la donnée d'entrée. Il s'agit des couches M la couche de moyenne (mean), Θ la couche de dispersion et Π la couche de probabilité de "dropout". Le couche M et la couche Θ ont comme fonction d'activation la fonction exponentielle car la moyenne et la dispersion sont des valeurs positives. La couche Π utilise quant à elle la fonction sigmoïde adaptée dans ce cas car il s'agit de probabilité compris entre 0 et 1. Ces couches permettent ainsi d'obtenir les paramètres nécessaires au calcul de la fonction de

perte ZINB. Cette fonction de perte est décrite comme permettant de discriminer et d'imputer des valeurs aux "zéros techniques".

$$M = diag(s_i) \times exp(W_\mu D)$$

$$\Theta = exp(W_\theta D)$$

$$\Pi = sigmoid(W_r D)$$

Avec D la dernière couche cachée du décodeur, s_i le facteur de taille calculée en amont et donné en entrée. M, Θ et Π sont les matrices des estimations de moyenne, de dispersion et de la probabilité de dropout.

La fonction de perte ZINB vise à maximiser la log-vraisemblance d'une distribution ZINB :

$$L_{ZINB} = -log(ZINB(X^{count}|\pi, \mu, \theta))$$

$$ZINB(X^{count}|\pi, \mu, \theta) = \pi\delta_0(X^{count}) + (1-\pi)NB(X^{count}|\mu, \theta)$$

$$NB(X^{count}|\mu, \theta) = \frac{\Gamma(X^{count}+\theta)}{X^{count}!\Gamma(\theta)}(\frac{\theta}{\theta+\mu})^\theta(\frac{\mu}{\theta+\mu})^{X^{count}}$$

Avec μ la moyenne, θ la dispersion, π la probabilité de dropout, X^{count} la matrice brute de comptage et δ_0, la fonction delta de Dirac qui prend la valeur 1 quand $X^{count} = 0$ et 0 sinon. La distribution ZINB correspond donc à une distribution binomiale négative (NB) sur laquelle une inflation de 0 est réalisée suivant un processus binaire selon la probabilité de dropout π.

- **Initialisation des clusters :**

Après le pré-entraînement avec la fonction de perte ZINB, une couche de clustering est initialisée en prenant comme taille le nombre de clusters défini et en poids le centre de ces clusters. Initialement, les clusters étaient définies avec la méthode K-means permettant d'obtenir facilement les centres des clusters. Nous avons également expérimenté une initialisation par Leiden en déterminant des centroïdes par le calcul de la moyenne par cluster, cette méthode est issue de la méthode scACE [7].

- **Entraînement de l'auto-encodeur avec fonction de perte ZINB + fonction de perte de clustering :**

Finalement, le modèle est entraîné en combinant la fonction de perte ZINB et une fonction de perte de clustering. Cette dernière étant la suivante :

$$L_c = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{qij}$$

$$q_{ij} = \frac{(1+\|z_i-\mu_j\|^2)^{-1}}{\sum_{j'}(1+\|z_i-\mu_{j'}\|^2)^{-1}}$$

$$p_{ij} = \frac{\frac{q_{ij}^2}{\sum_j q_{ij}}}{\sum_{j'}(\frac{q_{ij}^2}{\sum_{j'} q_{ij'}})}$$

La fonction de perte est donc la divergence de Kullback-Leibler entre la distribution P et la distribution Q. Q correspond à la distribution des soft labels (probabilité pour une cellule d'appartenir à chacun des clusters) suivant une loi de Student, q_{ij} est calculé pour un point z_i correspondant à la similarité entre le point z_i et le centre du cluster μ_j selon la distribution t de Student. La distribution P est la distribution cible calculée à partir de Q obtenue en élevant au carré et en normalisant par la fréquence par cluster.

La fonction de perte totale de l'auto-encodeur est donc la suivante :

$$L = L_{ZINB} + \gamma L_c$$

Avec γ un paramètre permettant de contrôler le poids relatif des deux fonctions de perte. Par défaut sa valeur est de 1 dans la méthode de scDeepCluster.

Le modèle est donc entraîné suivant cette nouvelle fonction de perte totale jusqu'à convergence en prenant comme critère d'arrêt que le pourcentage de points ayant changés de clusters soit inférieur à une valeur seuil dont la valeur par défaut est de 0.1%. Le nombre de clusters ne varie pas. Cela permet de raffiner les clusters et d'améliorer l'espace latent.

3.3.2 L'implémentation de Contrastive-sc

La méthode Contrastive-sc est originellement implémentée en Pytorch. Une implémentation en Keras/Tensorflow a été effectuée pour obtenir une implémentation dans les mêmes librairies que l'on peut comparer et, par la même occasion, m'approprier la méthode.

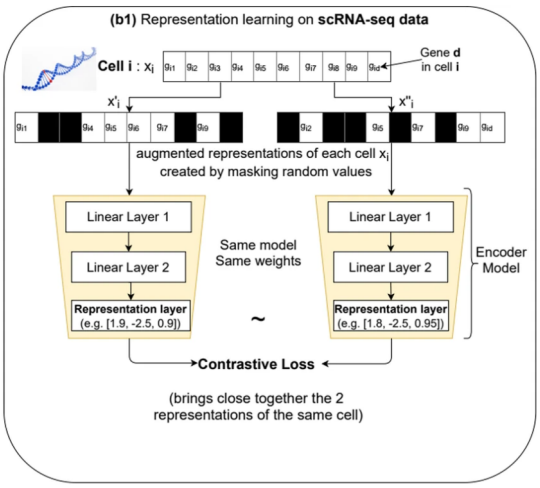


FIGURE 6 – Schéma du principe général de Contrastive-sc (source : [3])

Le principe général s'inspire de travaux provenant de la vision par ordinateur : l'apprentissage contrastif. Différentes versions de chaque image sont créées en transformant l'image de base par rotation, translation, changement de couleur, etc. Dans le cas des données scRNA-seq, cette méthode consiste à masquer un groupe de gènes aléatoires. Cela est réalisé en utilisant une couche "dropout" sur les données en entrée, assignant des valeurs nulles pour certains gènes aléatoirement. Dans la méthode Contrastive-sc, deux dropout aléatoires sont effectués pour chaque cellule permettant d'obtenir deux vues différentes des comptages pour chaque cellule. Les deux vues sont traitées par le même encodeur. Il peut également être souligné que la méthode Contrastive-sc permettait d'obtenir des résultats robustes au changement d'architecture.

La fonction de perte contrastive compare ensuite les différentes représentations obtenues en sortie en essayant de rapprocher les paires provenant d'une même cellule et d'écarter celles qui viennent de cellules différentes. La formule de la fonction de perte étant la suivante :

$$l_{i,j} = -log \frac{exp(z_i.z_j/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} exp(z_i.z_k/\tau)}$$

Avec $\mathbb{1}_{[k \neq i]}$ la fonction indicatrice valant 1 si $k \neq i$, τ, la paramètre température, z_i étant la projection de l'échantillon i et z_j et la projection de la paire associée c'est-à-dire provenant de la même cellule d'origine (paires positives). z_k correspondant à toutes les projections autre que z_i.

3.4 Outils de clustering

Afin de réaliser le clustering, deux méthodes différentes ont été employées. La méthode K-means qui permet de former n clusters et nécessite de prédéfinir le nombre de clusters souhaité ainsi que la méthode Leiden prenant en paramètre la résolution et donc ne nécessitant pas de définir un nombre de clusters explicitement.

3.4.1 La méthode K-means

La méthode K-means nécessite de préciser le nombre de clusters voulu n. Ainsi, n points sont désignés comme centroïdes. Chaque point est assigné au centroïde le plus proche puis chaque centroïde est recalculé. Et ainsi de suite jusqu'à ce que les centroïdes ne changent plus. La méthode K-means est une technique de clustering inductive, c'est-à-dire que le clustering une fois effectué peut être appliqué sur un nouveau jeu de données afin de prédire l'appartenance aux différents clusters.

3.4.2 La méthode Leiden

La méthode de clustering de Leiden est une version améliorée de la méthode de Louvain qui pouvait conduire à la formation de clusters de communautés peu ou pas connectées.

Cette méthode est privilégiée pour le traitement des données scRNA-seq.

L'algorithme emploie un graphe des k plus proches voisins. Dans notre cas, le graphe possède des noeuds correspondants aux cellules et à partir de la matrice de distance chaque cellule est reliée à ses k plus proches voisins. Au départ, chaque noeud définit une communauté. Puis des partitions sont créées en déplaçant les noeuds d'une communauté à une autre dans le but d'optimiser la modularité, c'est-à-dire le nombre d'arêtes dans une communauté comparé au nombre d'arêtes attendu dans les données. Les communautés sont ensuite affinées puis agrégées. Les noeuds formés après agrégation sont déplacés suivant le même principe jusqu'à ce que l'affinement ne change plus la partition.

La méthode de Leiden prend comme paramètre la résolution qui permet de déterminer l'échelle de partition des clusters et donc la grossièreté du clustering. Plus la résolution sera élevée plus il y aura de clusters et inversement. A l'origine, cette méthode de clustering n'a pas été implémenté de manière inductive (dans le module Scanpy) dans le sens où elle ne pouvait pas prédire l'appartenance de nouveaux points aux clusters précédemment déterminés. Cependant, en s'inspirant des travaux de scAce [7], nous avons pu adapter le clustering de Leiden pour le rendre inductif. Pour cela, la moyenne par cluster est calculée et chaque nouveau point est associé au cluster dont la moyenne est la plus proche.

3.5 Métriques d'évaluation du clustering

Afin de mesurer l'adéquation des clusters avec la vérité terrain constituée par les labels des cellules, trois métriques différentes ont été employées qui prennent en compte cette vérité terrain :

- La NMI (Normalized Mutual Information) de valeur comprise entre 0 et 1, correspond à l'information mutuelle entre clusterings divisée par leur entropie. Une valeur de 1 signifie que les deux clusterings comportent exactement la même information. La formule de la NMI est la suivante :

$$NMI = \frac{\sum_{p=1}^{C_U}\sum_{q=1}^{C_V}|U_p \cap V_p| log\frac{n|U_p \cap V_q|}{|U_p||V_q|}}{max(-\sum_{p=1}^{C_U}|U_p| log\frac{|U_p|}{n}, -\sum_{q=1}^{C_V}|V_q| log\frac{|V_q|}{n})}$$

  Avec U et V deux ensembles de clusters d'un set de données à n points avec respectivement C_U et C_V clusters chacun.

- L' ARI (Adjusted Rand index), comprise entre -1 et 1, reflète l'accord entre deux clustering avec une valeur de 1 signifiant un accord total entre les deux clusterings. Il s'agit d'une version ajustée du calcul du Rand Index visant à le rendre moins sensible à la chance. Sa formule étant :

$$ARI = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]}$$

Le calcul de ARI prend en compte :

- a : le nombre de paires de deux objets étant dans le même cluster dans le clustering U et dans le clustering V (vrai positif)
- b : le nombre de paires de deux objets dans le même cluster dans U mais dans différents clusters dans V (faux positif)
- c : le nombre de paires de deux objets dans différents clusters dans U mais dans le même dans V (faux négatif)
- d : le nombre de paires de deux objets dans différents clusters dans U et dans V (vrai négatif)

- La CA (clustering accuracy) représente le meilleur match entre l'affectation d'un cluster et la vérité terrain. La valeur de CA est comprise entre 0 et 1 avec 1 étant un match parfait entre la vérité terrain et le clustering. Sa formule étant :

$$CA = max_m \frac{\sum_{i=1}^{n} 1\{l_i = m(u_i)\}}{n}$$

Avec i un point des données, l_i le label de la vérité terrain et u_i l'affectation selon l'algorithme de clustering, n le nombres de points et m toutes les possibilités une à une d'association entre l'affectation des clusters et les vrais labels.

Enfin, une métrique ne prenant pas en compte la vérité terrain a aussi été considérée, notamment pour la PCA. Il s'agit du score de silhouette qui mesure la cohésion et la séparation des clusters formés selon la formule suivante pour un point i :

$$s_{sil}(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

avec $a(i) = \frac{1}{|I_k|-1} \sum_{j \in I_k, j \neq i} d(x^i, x^j)$, la distance moyenne du point aux autres points de son groupe (cohésion) et $b(i) = min_{k' \neq k} \frac{1}{|I-k|} \sum_{i' \in I_{k'}} d(x^i, x^{i'})$, la distance moyenne du point aux points du clusters voisins (séparation).
Ce qui peut être moyenné sur l'ensemble du clustering :

$$S_{sil} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|I_k|} \sum_{i \in I_k} s_{sil}(i)$$

Le score de silhouette est compris entre -1 et 1, plus le score est élevé, plus la séparation et la cohésion sont bonnes.

Le choix de ces métriques a été motivés par le fait qu'il s'agit de métriques couramment employées dans l'évaluation de méthode de clustering [10, 9, 3].

Afin de réaliser le traitement des données et surtout l'entraînement de mes modèles, j'ai employé les clusters de calcul de l'IFB (Institut Français de Bioinformatique) permettant de calculer des tâches plus rapidement. Aussi, conscient des enjeux environnementaux et énergétiques, j'ai veillé à l'optimisation des ressources informatiques allouées.

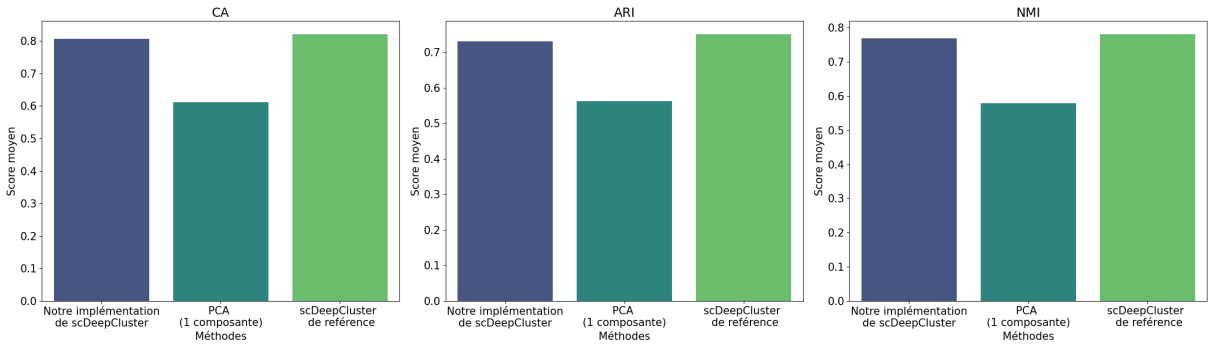


FIGURE 7 – Comparaison des résultats de notre implémentation (moyenne sur 5 runs (une run correspondant à une nouvelle réalisation de la méthode dans son intégralité)), de la PCA et de scDeepCluster

4 Résultats et Discussions

J'ai mené différentes expériences afin d'étudier la reproductibilité des résultats de scDeepCluster, de PCA et de Contrastive-sc avec mes implémentations, de tester celles-ci sur un autre jeu de données plus déséquilibré, le jeu de données Baron et d'étudier l'impact de différents hyperparamètres avant de réaliser une comparaison des différentes méthodes entre elles.

4.1 Étude de la reproductibilité et validation des implémentations

4.1.1 Etude de la reproductibilité de scDeepCluster

En premier lieu, j'ai mené une étude de la reproductibilité des résultats obtenus dans l'article de scDeepCluster [9] ainsi qu'une validation de mon implémentation. Dans ce cadre, j'ai choisi de reproduire les résultats de PCA + Kmeans et de scDeepCluster mais avec mon implémentation. J'ai conservé les mêmes paramètres, c'est-à-dire que tous les gènes ont été conservés. Les données ont été normalisées, transformées par un log, non réduites pour la PCA. Le jeu de données employé est bien sûr le même soit le jeu PBMC 10X 4k, l'algorithme de clustering utilisé étant K-means à 8 clusters (suivant la vérité terrain) et l'architecture de l'auto-encoder est composée de trois couches pour l'encodeur de taille 256, 64 et 32. Les résultats obtenus ont été résumés sur la figure 7.

Il peut être remarqué concernant les résultats de mon implémentation de scDeepCluster qu'ils sont cohérents avec ceux présentés dans l'article. Cela permet de valider mon implémentation. Concernant la PCA, nous avons obtenu des valeurs similaires avec les résultats de l'article en ne conservant qu'une composante principale. Ce choix est soutenu par le score de silhouette mais ne permet de ne capturer que 8% de la variance totale des données. Le choix du nombre de composantes principales de la PCA peut cependant

grandement influencé les résultats sur les trois métriques retenues. Ainsi, une étude plus poussée de la PCA a été effectuée sur un autre jeu de données et est présentée plus bas.

4.1.2 Étude de la reproductibilité de Contrastive-sc

Les résultats de mon implémentation de Contrastive-sc (avec la même architecture que la méthode d'origine soit des couches de taille 200, 40, 60) ont été comparés à ceux décrits dans la publication d'origine [3] sur le jeu PBMC 10X 4k (seuls les résultats des métriques ARI et NMI étaient accessibles). Ces résultats présentés sur la figure 8 permettent de souligner la similarité des scores obtenus et donc de valider mon implémentation.

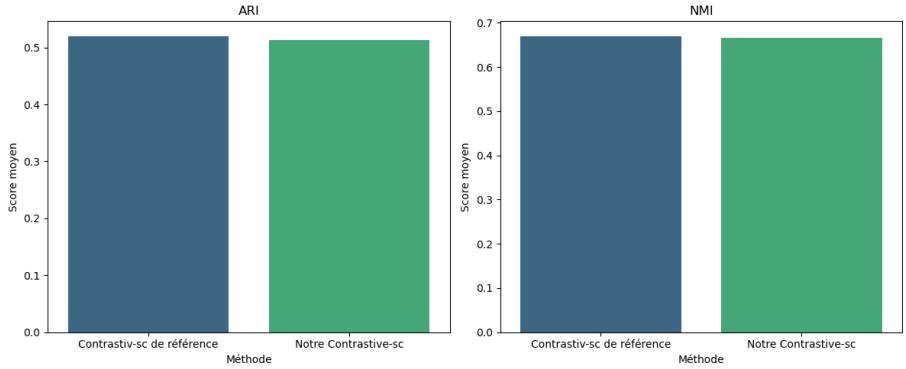


FIGURE 8 – Comparaison des résultats de deux métriques obtenues avec notre implémentation ("Notre Contrastive-sc", moyenne obtenue sur 5 runs) et les résultats de la méthode d'origine ("Contrastive-sc de référence")

4.2 Étude avec un nouveau jeu de données : le jeu Baron

Après avoir étudié la reproductibilité des résultats de scDeepCluster et de Contrastive-sc, j'ai voulu tester ces méthodes avec une autre jeu de données un peu plus complexe car assez déséquilibré, le jeu de données Baron. Pour cela, le jeu de données a été séparé en trois jeux différents : un jeu d'entraînement (70% soit 5998 cellules), un jeu de validation (15% soit 1285 cellules) et un jeu de test (15% soit 1286 cellules). Les modèles sont entraînés sur le jeu d'entraînement et les hyperparamètres sont déterminés grâce aux performances sur le jeu de validation. Enfin une fois les hyperparamètres fixés, les méthodes peuvent être comparées sur le jeu de test (comme schématisé précédemment dans la figure 2).

4.2.1 Étude de la PCA sur les données Baron

J'ai mené une étude de la PCA sur les données Baron, conjointement avec Brice Vergnou, un autre stagiaire, afin de connaître les paramètres importants permettant d'optimiser les résultats de la PCA. Ce collègue a montré que considérer les gènes avec une haute

variabilité permettait d'augmenter les résultats de la PCA avec 2000 gènes comme valeur optimale de gènes sélectionnés.

J'ai effectué une comparaison des méthodes de clustering K-means et Leiden. Le nombre de clusters pour K-means ou la résolution pour Leiden a été sélectionné afin de maximiser l'ARI, la NMI et la CA. Les résultats sont présentés dans la figure 9. Il apparaît alors que suivre le score de silhouette, une métrique qui ne s'appuie pas sur la vérité terrain, ne s'accorde pas avec la maximisation des métriques ARI, NMI et CA. En effet, si l'on souhaite maximiser le score de silhouette, il faudrait conserver deux composantes principales et faire un clustering selon la méthode K-means. Or, dans ce cas les valeurs de ARI, NMI et CA sont plutôt faibles (respectivement 0.4, 0.6, 0.55). Si l'on veut maximiser ces métriques, il faut conserver davantage de composantes principales et utiliser la méthode de Leiden qui semble être supérieure à la méthode K-means. Dans ce cas, le score des différentes métriques se stabilise à partir de 12 composantes conservées (avec comme valeur respective : 0.929, 0.897, 0.912).

Ainsi, la PCA doit être réalisée sur la matrice filtrée pour ne conserver que les 2 000 gènes les plus variables après le pré-processing. Les douze premières composantes seront sélectionnées pour effectuer un clustering avec la méthode de Leiden.

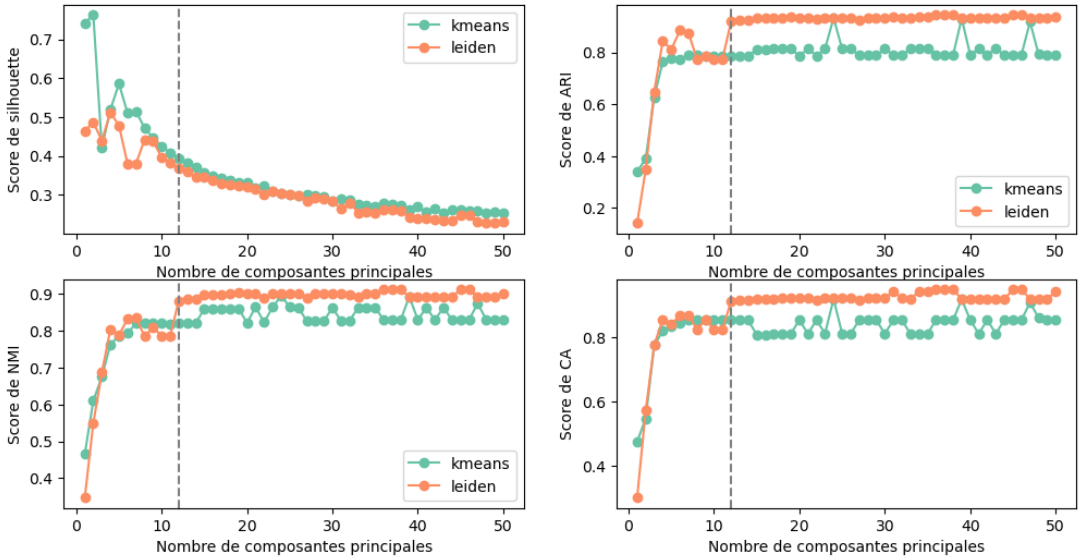


FIGURE 9 – Représentation des scores des métriques sur le jeu de validation selon le nombre de composantes de la PCA (la ligne en pointillée représente le nombre de composantes principales sélectionné)

4.2.2 Impact des hyperparamètres sur la méthode scDeepCluster

Puisque que le jeu de données a été divisé en train et val, un early stopping a pu être mis en place permettant de limiter l'overfitting en arrêtant le pré-entraînement de l'auto-encodeur lorsque la fonction de perte augmente pour le jeu de validation. De plus par comparaison

avec la PCA, l'utilisation uniquement des 2000 gènes les plus variables a également été testée. Les résultats des métriques se sont montrés légèrement meilleurs en utilisant un early stopping. L'utilisation uniquement des 2000 gènes les plus variables apportent une amélioration conséquente aux résultats de scDeepCluster. Ces résultats sont visibles en annexe 15.

Par la suite, j'ai étudié la méthode scDeepCluster sur le jeu Baron. Par soucis de comparaison avec la PCA, la méthode de clustering utilisée pour l'initialisation des clusters est la méthode Leiden. Ensuite l'influence de trois paramètres a été étudiée : l'architecture, la résolution de la méthode de Leiden et le paramètre γ du poids relatif des fonctions de perte.

L'étude de l'architecture présentée avec la figure 12 révèle que l'architecture de base présente d'assez bons résultats. La suppression d'une des couches 64 ou 32 n'entraîne pas de diminution de capacité et la conservation uniquement de l'encodeur pour l'entraînement, et donc l'utilisation juste de la fonction de perte de clustering n'entraîne pas de modification des résultats. Des résultats plus discutables ont été obtenus en ajoutant une couche, diminuant ainsi la taille de l'espace latent ou en ajoutant une couche en entrée de taille supérieure. Ainsi, nous retiendrons l'architecture de base.

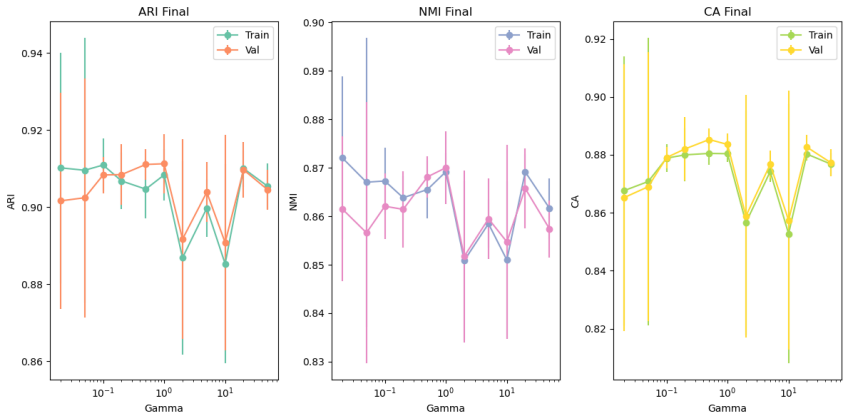


FIGURE 10 – Graphiques du score final moyen sur 5 runs des métriques selon la valeur du paramètre γ (les barres d'erreurs correspondent aux écarts types)

Sur la figure 11, il peut être remarqué qu'une résolution de 0.07 semble permettre d'obtenir les meilleurs scores de métriques. Cela permet de discriminer 6 à 7 clusters. De plus la différence de score entre le jeu d'entraînement et de validation semble diminuer à la fin de l'entraînement. Il s'agit d'un résultat encourageant quant à la généralisation des résultats. J'ai également observé l'évolution de l'espace latent ce qui m'a permis de constater qu'au cours de l'entraînement, les clusters définis à l'initialisation semblent se densifier lors de l'entraînement (annexe 16). De plus, la capacité de cet auto-encodeur à débruiter le jeu de données peut être observée grâce aux matrices présentées en annexe 17.

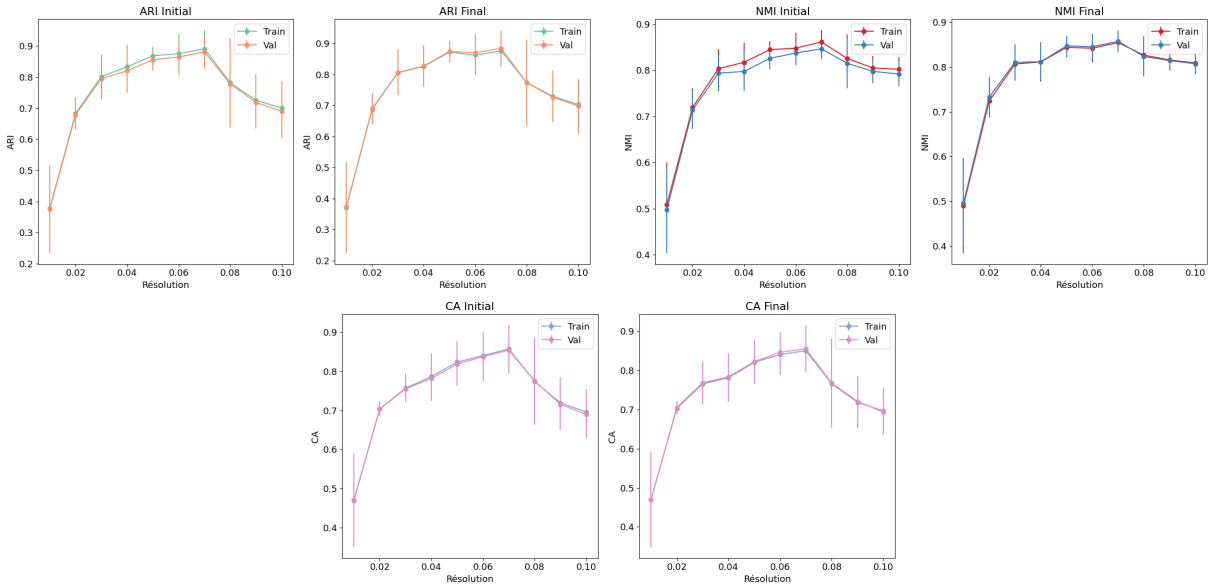
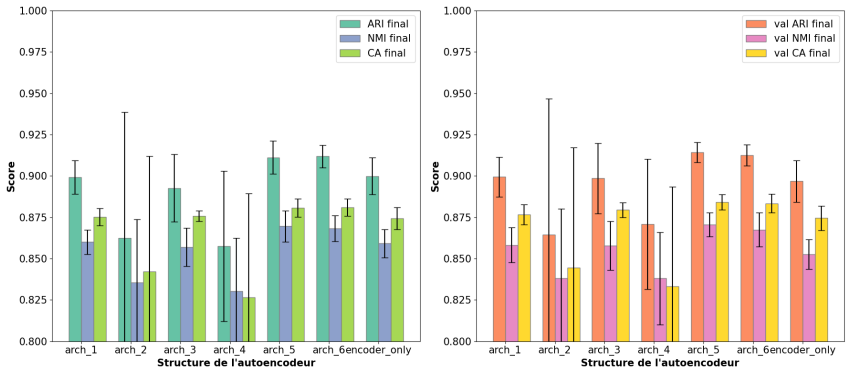


FIGURE 11 – Graphiques du score moyen d'ARI, de NMI et de CA initial (après le pré-entraînement) et final (après entraînement) pour le jeu train et val selon la résolution pour le clustering de Leiden sur 5 runs (les barres d'erreur correspondent aux écarts types)



arch_name	arch
arch_1	[256, 64, 32]
arch_2	[256, 64, 32, 16]
arch_3	[512, 256, 64]
arch_4	[512, 256, 64, 32]
arch_5	[256, 32]
arch_6	[256, 64]
encoder_only	[256, 64, 32]

FIGURE 12 – Graphique du score moyen d'ARI, de NMI et de CA final (après entraînement) pour le jeu train et val selon l'architecture de l'auto-encodeur (encoder_only signifie qu'uniquement la partie encodeur a été conservée après le pré-entraïnement, la fonction de perte est donc uniquement la fonction de perte de clustering) sur 5 runs (les barres d'erreur correspondent aux écarts types)

L'étude de l'influence du poids n'a pas révélé de tendance particulière, mise à part une grande variabilité des résultats pour des valeurs très faibles. La valeur de 1 semble tout de même la plus adaptée au vu des résultats pour les trois métriques visibles sur la figure 10.

4.2.3 Impact des hyperparamètres sur la méthode Contrastive-sc

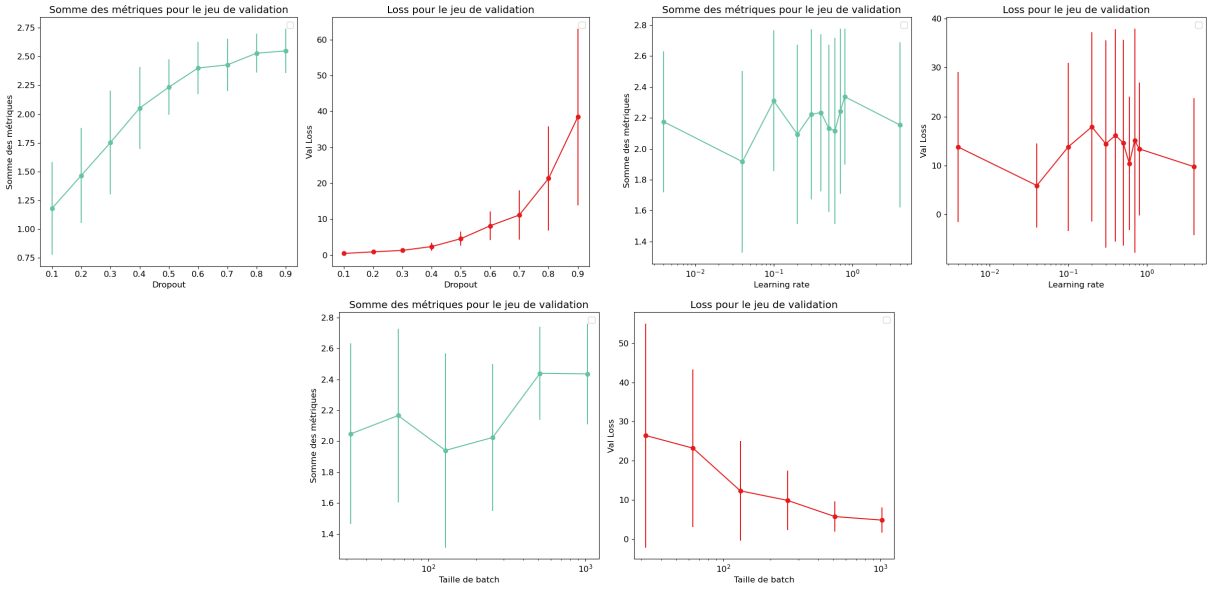


FIGURE 13 – Graphiques de la somme des métriques et de la valeur de la fonction de perte finale pour le jeu de validation selon le taux de dropout, le taux d'apprentissage et le taille de batch

J'ai étudié l'impact de différents hyperparamètres sur la méthode Contrastive-sc en employant un random search. Il s'agit d'une méthode d'optimisation consistant à échantillonner de manière aléatoire des combinaisons d'hyperparamètres dans l'espace de recherche afin d'établir une combinaison optimale. Dans ce cas, les hyperparamètres variants ont été le taux de dropout, la taille de batch et le taux d'apprentissage. Les résultats ont été résumés sur la figure 13. Pour ces expériences, j'ai décidé de conserver la même architecture d'encodeur que pour scDeepCluster (soit 256, 64, 32) car cette architecture semblait adaptée selon les tests précédents.

Le résultat le plus flagrant est l'augmentation des métriques avec le taux de dropout. Ce résultat confirme ceux obtenus dans la publication d'origine [3]. Il apparaît aussi que la valeur de la fonction de perte augmente avec le taux de dropout. Le taux de dropout de 0.9 permet d'obtenir les meilleurs résultats. Concernant le taux d'apprentissage, aucune tendance particulière ne semble ressortir de nos résultats. Dans ce cas, la valeur d'origine de 0.4 sera retenue. Enfin, le score des métriques semble augmenter avec la taille des batchs notamment pour une taille de 512 et 1024 qui obtiennent des scores similaires. En parallèle, la valeur de la fonction de perte diminue avec la taille des batchs. Les résultats ne sont pas aussi clairs que pour le taux de dropout, néanmoins on retiendra la valeur de 512 comme taille de batchs.

4.2.4 Comparaison des méthodes

A présent que les hyperparamètres optimaux ont été définis, j'ai pu comparer les trois méthodes entre elles en les entraînant sur le jeu d'entraînement et de validation afin d'obtenir les prédictions sur le jeu test qui sont présentées dans la figure 14.

Les trois méthodes permettent d'obtenir des résultats globalement similaires. La méthode scDeepCluster semble pour autant un peu inférieure aux deux autres méthodes. Tandis que la méthode Contrastive-sc et la PCA rivalisent l'une avec l'autre. Ces résultats peuvent paraître un peu surprenant notamment au vue de ceux obtenus pour la reproductibilité de scDeepCluster qui semblait obtenir de bien meilleur résultat que la PCA mais il faut préciser qu'ici la méthode d'exécution de la PCA est tout à fait différente car la sélection du nombre de composantes principales a été motivée par la maximisation des métriques utilisées pour la comparaison des méthodes et non par le score de silhouette. Or, comme démontré précédemment, les deux ne s'accordent pas. Cela vient donc souligner l'importance du choix des paramètres de la PCA servant comme point de comparaison pour l'évaluation de nouvelles méthodes.

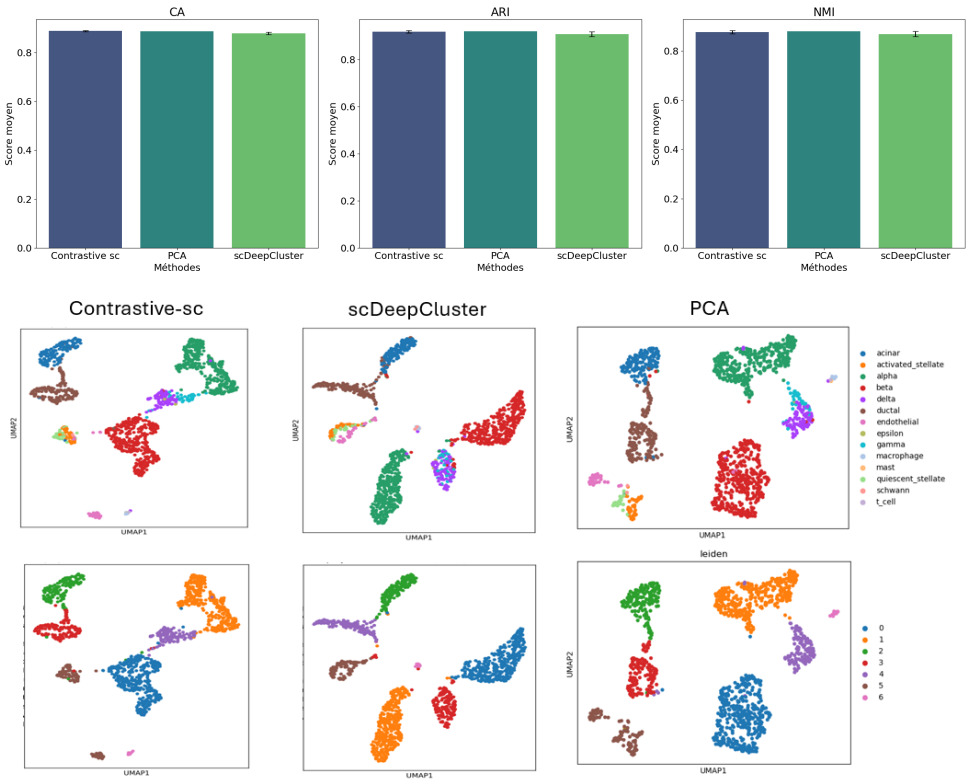


FIGURE 14 – Graphiques des scores moyens pour les trois méthodes (sur 5 runs) et représentations avec une projection UMAP des données depuis l'espace réduit avec les trois méthodes avec la vérité terrain (ligne du haut) et les clusters obtenus (ligne du bas)

Conclusion

L'étude de la PCA, de scDeepCluster et de Contrastive-sc a permis de mettre en relief des hyperparamètres importants dans ces méthodes, notamment l'architecture (fig 12) et l'initialisation des clusters selon la résolution pour scDeepCluster (fig 11), le taux de dropout pour Contrastive-sc (fig 13) et le nombre de composantes principales pour la PCA (fig 9). Sur le jeu de données Baron, Contrastive-sc et la PCA se sont révélées supérieures à scDeepCluster (fig 14). Cependant, notre étude souligne l'importance du choix des hyperparamètres lors de la comparaison des méthodes, notamment avec la PCA. En effet, nous avons rencontré une contradiction entre le score de silhouette et les autres métriques s'appuyant sur la vérité terrain. En s'appuyant uniquement sur le score de silhouette pour choisir le nombre de composantes principales, Contrastive-sc et scDeepCluster obtiennent des résultats bien supérieurs à la PCA, ce qui est traditionnellement fait. Mais si un nombre suffisant de composantes est conservé alors la PCA peut rivaliser avec des méthodes d'apprentissage profond. Cela ne vient que rappeler la sensibilité à la sélection des hyperparamètres des méthodes utilisées pour la réduction de dimension et le clustering des données scRNA-seq ainsi qu'à l'absence de règles explicites [8].

Il reste tout de même des pistes d'amélioration pour ces méthodes d'intelligence artificielle, notamment des modifications de scDeepCluster ont déjà été programmées comme scziDesk [1] ou scAce [7] ou bien des méthodes alternatives comme scMAE [5] qui utilise des masques d'une manière différente de Contrastive-sc et qui emploie cette fois un autoencodeur. Comme autre perspective, la méthode Contrastive-sc pourrait également être améliorée en s'intéressant de plus près à la création des différentes vues ou bien même en combinant une méthode de clustering avec l'entraînement de l'encodeur à la manière de scDeepCluster.

Il est tout de même bon de rappeler qu'il s'agit d'un domaine en pleine évolution et que ces techniques sont toutes assez récentes. Ainsi, de nombreuses pistes sont encore ouvertes quant à l'utilisation de méthodes d'apprentissage profond pour l'analyse de données de cellules uniques offrant la possibilité d'un avenir prometteur.

Références

[1] Liang Chen, Weinan Wang, Yuyao Zhai, and Minghua Deng. Deep soft K-means clustering with self-training for single-cell RNA sequence data. *NAR Genomics and Bioinformatics*, 2(2) :lqaa039, 05 2020.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv*, 2020. PMLR 119 :1597–1607.

[3] Madalina Ciortan and Matthieu Defrance. Contrastive self-supervised clustering of scrna-seq data. *BMC Bioinformatics*, 22(1) :280, 2021.

[4] Gökcen Eraslan, Lukas M. Simon, Maria Mircea, Nikola S. Mueller, and Fabian J. Theis. Single-cell rna-seq denoising using a deep count autoencoder. *Nature Communications*, 10(1) :390, 2019.

[5] Zhaoyu Fang, Ruiqing Zheng, and Min Li. scMAE : a masked autoencoder for single-cell RNA-seq clustering. *Bioinformatics*, 40(1) :btae020, 01 2024.

[6] M Flores, Z Liu, T Zhang, MM Hasib, YC Chiu, Z Ye, K Paniagua, S Jo, J Zhang, SJ Gao, YF Jin, Y Chen, and Y Huang. Deep learning tackles single-cell analysis-a survey of deep learning for scrna-seq analysis. *Brief Bioinform*, 23(1) :bbab531, 2022.

[7] Xinwei He, Kun Qian, Ziqian Wang, Shirou Zeng, Hongwei Li, and Wei Vivian Li. scAce : an adaptive embedding and clustering method for single-cell gene expression data. *Bioinformatics*, 39(9) :btad546, 09 2023.

[8] M Krzak, Y Raykov, A Boukouvalas, L Cutillo, and C Angelini. Benchmark and parameter sensitivity analysis of single-cell rna sequencing clustering methods. *Frontiers in Genetics*, 10 :1253, 2019.

[9] Tian Tian, Ji Wan, Qi Song, and Zhi Wei. Clustering single-cell rna-seq data with a model-based deep learning approach. *Nature Machine Intelligence*, 1(4) :191–198, 2019.

[10] Zhijin Wu and Hao Wu. Accounting for cell type hierarchy in evaluating single cell rna-seq clustering. *Genome Biology*, 21(1) :123, May 2020.

[11] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. *ArXiv*, abs/1511.06335, 2015.

Annexes

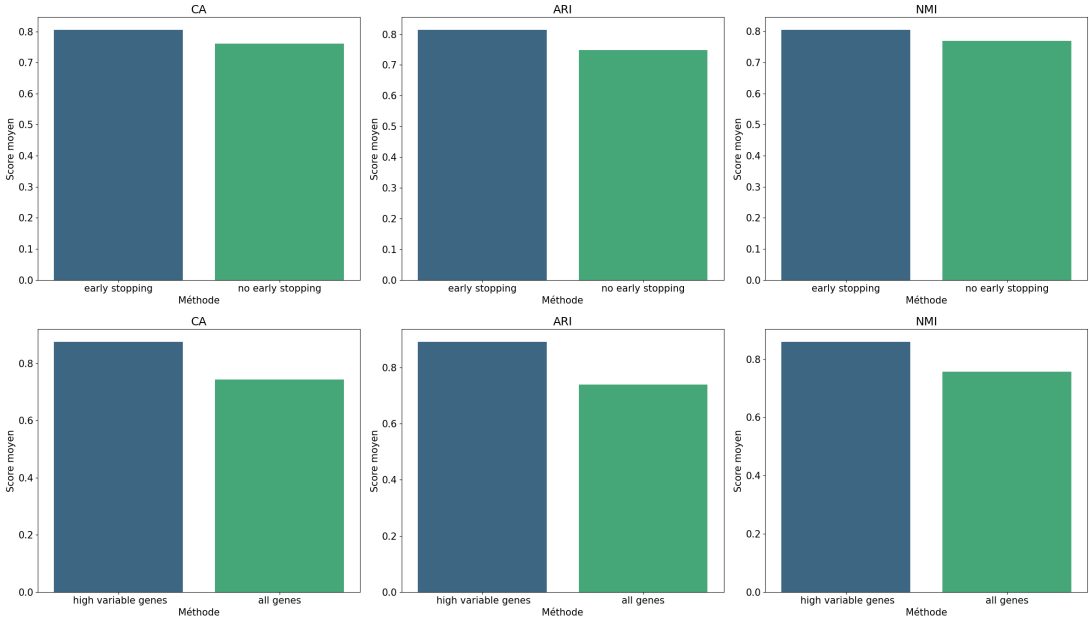


FIGURE 15 – Score moyen des métriques avec early stopping ou sans (ligne du haut) et avec tous les gènes ou uniquement les 2000 gènes les plus variables (lignes du bas)

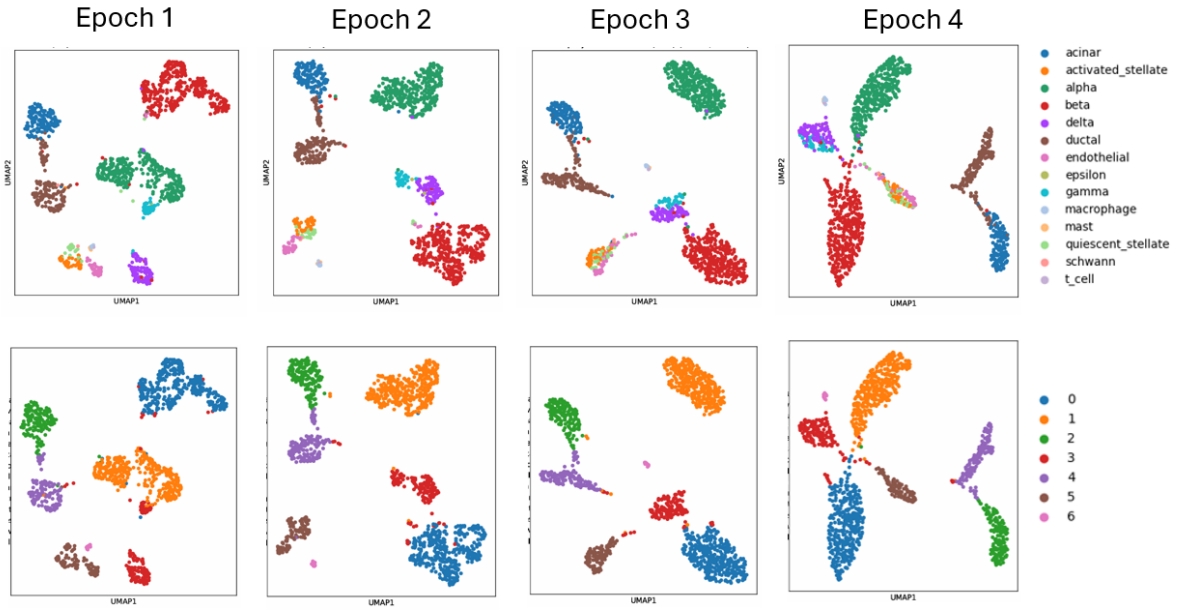


FIGURE 16 – Evolution des représentations des données de l'espace réduit en projection UMAP au cours des premières époch d'entraïnement de scDeepCluster

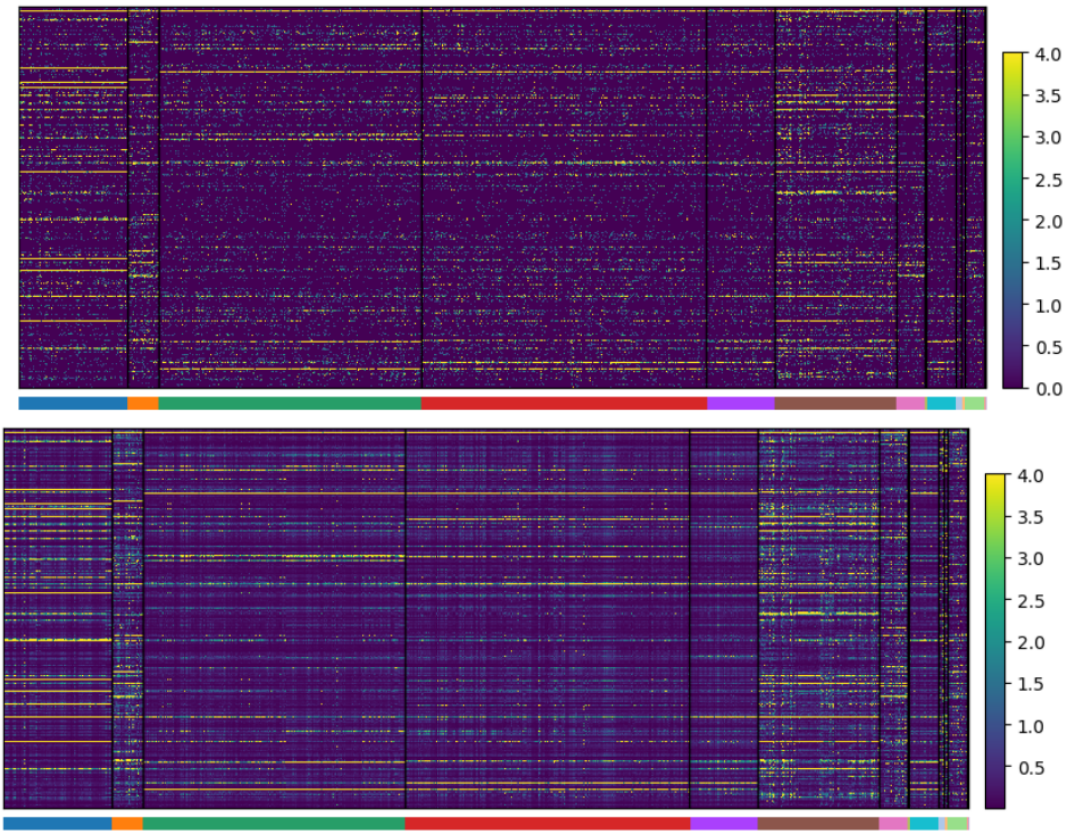


FIGURE 17 – Représentation des matrices brute (en haut) et prédite (en dessous) par l'auto-encodeur avec une fonction de perte ZINB