BIOINFORMATICS REPORT

Francavilla Ruggiero, Monopoli Tommaso

July 2021

1 Introduction

Multi-omics clustering is the application of clustering techniques to an integrated view of a multi-view dataset. Different views are provided from different omics, like mRNA, miRNA, methylation, proteome and copy number variation, that are the ones we exploit in this study. Multi-omics integration discovers hidden relationships between omics, which would not be revealed by examining single data types individually: for example, cancer subtypes can be defined based on both gene expression and DNA methylation together. In this study, we introduce a new multi-omics integration and clustering method based on deep learning. All codes are available in `https://github.com/Shargus/Multi-omics-clustering`.

2 Multi-omics datasets

We used three datasets for our experiments. Each of them is described hereafter.

2.1 Synthetic samples

A simple synthetic dataset comprising biological data from 500 subjects (samples). For each subject, three omics are available:

- transcriptome (mRNA): 131 features for each sample, representing the expression value of 131 genes;
- methylome (meth): 367 features for each sample, representing the beta value of 367 methylation sites of the genome;
- proteome (prot): 160 features for each sample, representing the intensity measurement of 160 proteins (which reflect their abundance in a cell).

Therefore, the resulting dataset consists of three files, which can be read as three matrices having 500 rows (samples) and respectively 131, 367, 160 columns (features). No missing values are present.

It's important to notice that this dataset is *very* simple, as the number of features for each omic is very limited with respect to real data acquired from high-throughput methodologies in bioinformatics. For instance, the number of identified genes in the human genome is around 60,000, and certain platforms can analyze up to 450k CpG locations in a genomic sample.

Along with the three omics files, a file containing a cluster label for each subject is available. Subjects are clusterized in a total of 5 distinct clusters.

The three omics are visualized in the first row of images of fig. 1. Since the dataset seems way too simple to cluster, we decided to add salt & pepper gaussian noise to the features, obtaining the noisy omics represented in the second row of images of fig. 1. After adding noise, we minmax-normalized the features to the interval $[0, 1]$.

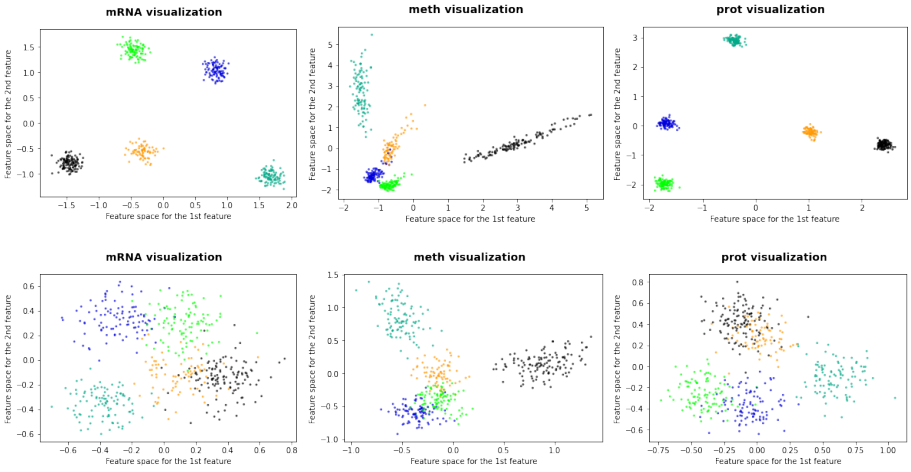

Figure 1: Synthetic omics originally (first row) and after adding salt & pepper noise (second row). The first two principal components for each omic are computed and plotted, to enable 2D visualization of the dataset.

2.2 Lung tumor samples

Retrieval A dataset comprising transcriptome profiling (**mRNA**, **miRNA**), DNA methylation (**meth**) and gene-level copy number variation (**CNV**) data from patients suffering from Lung Adenocarcinoma (TCGA-LUAD) and Lung Squamous Cell Carcinoma (TCGA-LUSC). All the data is freely available on the Genomic Data Commons (GDC) portal, as part of the The Cancer Genome Atlas (TCGA) program.

To build this dataset, we followed these steps:

- first, we queried the GDC repository to retrieve manifest and JSON files associated to TCGA-LUAD and TCGA-LUSC data for each omic of interest;
- then, we discarded patients for which 2+ omic files are present, and kept only the patients for which all of the four omics of interest are available;
- we downloaded the files associated to each selected patient and each omic (by means of the GDC client software);
- finally, we integrated the downloaded files by building one tab-separated file for each omic, having samples as rows and features as columns.

The resulting dataset is composed of 783 samples (429 LUAD and 354 LUSC). The features of interest for each sample are: gene expression level for mRNA (FPKM-UQ normalization applied on HTSeq-counts raw counts of mapped reads to each gene), #reads per million mapped reads (RPM) for miRNA, beta value of Illumina Infinium 450k known CpG sites for meth, copy number of each gene for CNV.

Along with the created dataset, different cluster labels are assigned to LUAD and LUSC samples and stored in a separate file, to later test our multi-omics integration and clustering method.

Pre-processing After having built the lung dataset, the omics (now consisting of $783 \times M$ matrices, where M = #features for each omic) are pre-processed as follows.

First, for every omic, we delete those features containing only missing (NaN) values. Then, we input all the remaining missing values with KNN inputer. The next steps are specific for each omic considered:

- mRNA: keep only protein-coding genes, and delete genes with a zero expression value across all the samples (resulting #features = 19452);
- miRNA: delete sequences with a zero expression value across all the samples, and normalize with log2 normalization (resulting #features = 1623);
- meth: keep only the features/probes in common between Illumina 27k and Illumina 450k methylation arrays (this greatly reduces the dimensionality) (resulting #features = 23381);
- gene CNV: keep only autosomic genes [3], and filter out duplicate columns, so to keep only one copy for each[1] (resulting #features = 26978).

[1]the rationale behind this is that removing those perfectly-correlated columns preserves the distances between samples while greatly reducing the dimensionality

Finally, the resulting matrices are minmax-normalized to the interval [0,1].

The first two principal components of the four omics are visualized in fig. 2, with a different color for each different tumor type.

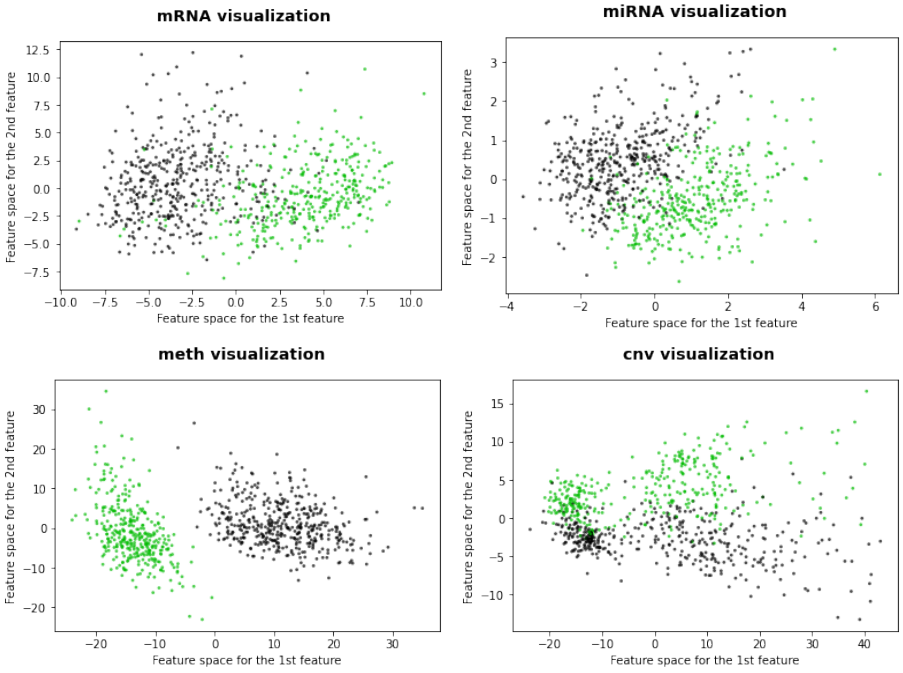

Figure 2: Lung dataset omics after pre-processing

From the visualization, we can see that the meth dataset has two clearly distinguishable clusters, also in two dimensions; we cannot say the same for the remaining omics. As for the CNV, we can spot the possible presence of a batch effect: there is clearly a very dense batch on the left and a sparser one on the right.

We tried to understand whether the two batches were caused by some clinical variables (age, vital status, race, ethnicity, gender, the presence of a synchronous malignancy). This has been done visually, by coloring the 2D CNV points with different colors for different values of each variable; however none of them seemed to explain the batch effect.

Therefore, we decided to accept the presence of a batch effect and tell the two batches apart manually with DBSCAN algorithm. DBSCAN effectively assigned the samples of the denser region of the dataset to a single cluster, and considered the remaining samples as noise points (however, the set of noise points constitutes the sparser batch, so we consider them as part of this second batch).

We correct the batch effect with the pyComBat algorithm [2]. The CNV omic before and after being adjusted for batch effects is visualized in fig. 3.

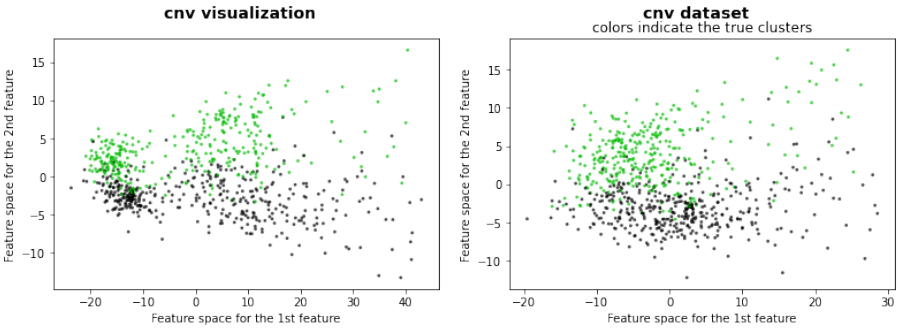

Figure 3: CNV before (left) and after (right) batch effect correction through pyComBat

2.3 Kidney tumor samples

Retrieval A dataset comprising **mRNA**, **miRNA** and **meth** TCGA data from patients suffering from three types of kidney tumor: Kidney Chromophobe Carcinoma (TCGA-KICH), Kidney Clear Cell Carcinoma (TCGA-KIRC) and Kidney Papillary Cell Carcinoma (TCGA-KIRP). All the data is freely available on the GDC portal.

The steps required to build this dataset are the same needed for the Lung dataset previously described.

The resulting dataset is composed of 650 samples (65 for KICH, 312 for KIRC and 273 for KIRP). The features we are interested in for each omic are the same described in the previous paragraph (in this dataset we decided not to include CNV data, as it suffers greatly from batch effect).

Along with the samples, one of three distinct cluster labels is assigned to each of them based on their tumor type, and stored in a separate file.

Pre-processing After having built the kidney dataset, the three omics (now consisting of $650 \times M$ matrices, where $M = \#\text{features}$ for each omic) are pre-processed in the same way as the mRNA, miRNA and meth omics of the lung dataset (section 2.2). The number of resulting features for each omic are: 19446 for mRNA, 1534 for miRNA, 23381 for meth.

The first two principal components of the three omics are visualized in fig. 4, with a different color for each different tumor type.

From the 2D visualization of the omics, we can see that only one of the three clusters (the dark blue one) seems to be fairly distinguishable from the others.

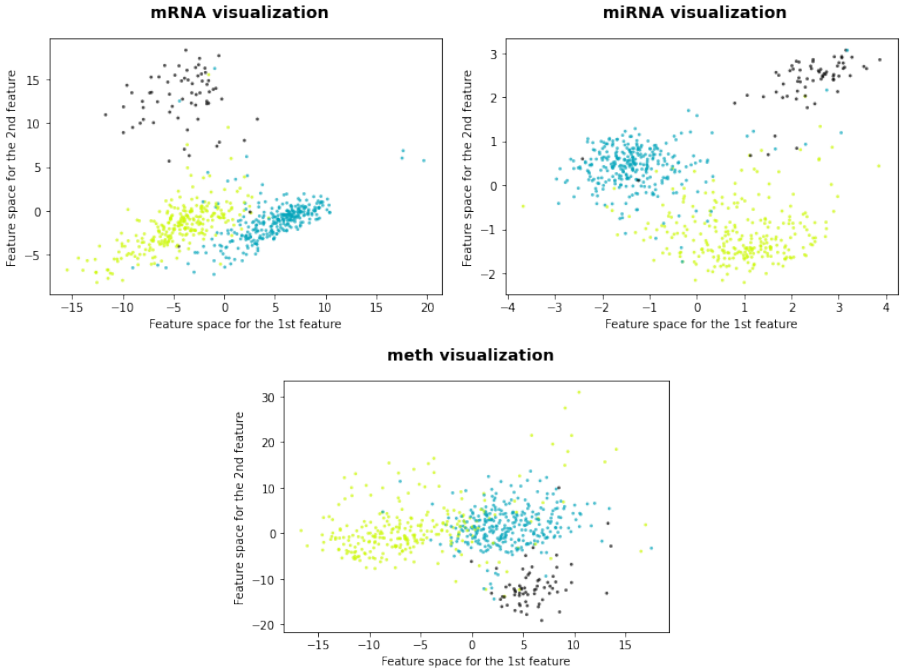

Figure 4: Kidney dataset omics after pre-processing

3 Our proposal

Our method is an unsupervised multi-omics integration and clustering algorithm that uses traditional auto-encoders, artificial neural networks used to learn efficient lower-dimensional representations of (unlabeled) data. For each dataset, an auto-encoder aims at finding a latent space which is a trade off between the most efficient compressed representation of the data (from which the auto-encoder should be able to reconstruct the input signal), and separability between data belonging to different clusters. For each data source taken into account, a different auto-encoder is instantiated, where the architecture is customized according to the complexity of each omic (in terms of n. of features). Distances between points in each latent space are used to build a single, integrated distance matrix, which is representative of the entire multi-omics dataset. After this integration step, a clustering technique (such as spectral clustering, in our method) will be applied on the resulting integrated distance matrix.

A sketch of the integration and clustering pipeline of our method is resumed in fig. 5.

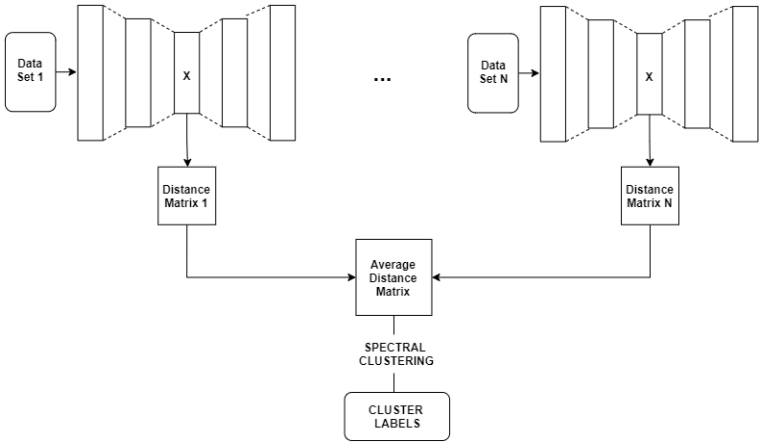

Figure 5: Pipeline of our integration and clustering method

3.1 Pipeline

The pipeline of our methodology consists of different steps.

The first step is the training of the auto-encoders. An auto-encoder is trained for each omic of the dataset. Each training consists of two stages:

- 1^{st} **phase**: traditional autoencoder training with the aim of finding the most representative latent space for the omic (i.e. a non-linear lower-dimensional representation of the data from which we can efficiently reconstruct the input signal). This is done by minimizing a reconstruction loss (see sect. 3.2).

  Once the 1^{st} phase ends and a representative lower-dimensional space is found, K-Means clustering technique is applied with several values of K (ranging from 2 to 10, for example), and different cluster labels are assigned to input samples. The best number of clusters is found combining K-Means with silhouette score: the value of K chosen is the one associated with the highest silhouette.

  Finally, the cluster centers (aka centroids) for each resulting cluster are computed, by averaging the encoded points belonging to each cluster in the latent space.

- 2^{nd} **phase**: introduction of a second loss component with the aim of bringing each encoded sample closer to its nearest cluster center in the latent space. Therefore, at each epoch, the samples in the latent space are reorganized in view of cluster assignments of the previous step.

  After each training epoch, K-means is run again on the dataset in the latent space, with the same K value found at the end of the 1^{st} phase.

The clustering method can return different cluster labels than the current ones: if it occurs that these new assignments allow better cluster separation, in terms of silhouette score, the found cluster labels become current and cluster centers in the latent space are updated; otherwise labels and centroids remain the same. In any case, at the end of the epoch the current silhouette score is updated.

The introduced loss component is discussed in sect. 3.2.

As a second step, after the training of all the autoencoders, each omic is inputed to the encoder of its autoencoder, thus extracting the encoded representation of the omic (aka encoded omic). From this, a matrix of pairwise distances between samples is computed. Different omics are integrated by averaging all these distance matrices. This matrix is the final result of the integration.

Spectral Clustering can now be executed on the average distance matrix, to get the final cluster assignments. The number of clusters K is evaluated by exploiting the silhouette score at values of K between 2 and 10.

The training steps of the autoencoders are resumed in fig. 6.

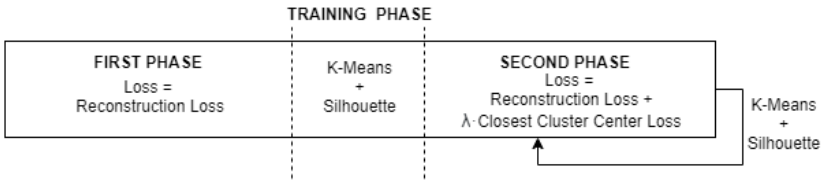

Figure 6: Auto Encoder Training

3.2 Loss function

Two different loss functions characterize the two phases of the training. Both components take the form of a Mean Absolute Error (MAE) loss.

- **Reconstruction Loss**: the "standard" loss that an auto-encoder tries to minimize. It measures the error that the decoder commits in reconstructing the input sample given only its encoded representation in the latent space.

  Given $x^{(i)}$ and $y^{(i)}$, input and reconstructed sample respectively, both having a number of features equal to k, and n mini-batch size, the reconstruction per-minibatch loss (the average of the n per-sample losses) is:

$$rec\_loss(x^{(i)}, y^{(i)}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_k |y_k^{(i)} - x_k^{(i)}|}{k}$$

- **Closest Cluster Center Loss**: the goal of this component is to bring each sample closer to its closest cluster center in the encoded space.

  Given $z^{(i)}$ and $z^{(c)}$, encoded input sample and its closest cluster center in the latent space, l dimension of the latent space and n mini-batch size, the per-minibatch closest cluster center is defined as:

$$ccc\_loss(z^{(i)}, z^{(c)}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\sum_l |z_l^{(i)} - z_l^{(c)}|}{l}$$

In the first phase, the auto-encoder is trained only with reconstruction loss for a certain number of epochs (chosen as a hyperparameter of the training). In the second phase, the auto-encoder is trained with a loss which is the combination of the two components previously defined:

$$loss(x^{(i)}, y^{(i)}, z^{(i)}, z^{(c)}) = rec\_loss(x^{(i)}, y^{(i)}) + \lambda \cdot ccc\_loss(z^{(i)}, z^{(c)})$$

λ is a hyperparameter that weights the contribution of the closest cluster center loss in the sum. After some experiments, we decide to keep this hyperparameter relatively small, to counter an eventual error in the step in which the K-means finds and incorrect number of clusters during the training of the AE.

3.3 Autoencoder architecture

Auto-encoder architecture differs for each input omic. Each auto-encoder is composed of two different fully connected neural networks, symmetrical in this case.

- Encoder: it compresses the input in a lower-dimensional latent space (represented by the bottleneck layer);
- Decoder: given the latent space representation of a sample, its output is the reconstruction of the input.

Among the different approaches for architecture implementation, a possible strategy, which has proven to work really well in our experiments, is the one proposed in paper [4]. Each dense layer of the encoder has a number of parameters ten times smaller than the number of parameters of the previous layer, until the dimension of the latent space (a hyperparameter of the model) is reached.

The dimension of the latent space is chosen as a tradeoff between being the smaller possible and at the same time being the most representative of input source (we want an efficient compression of the input that allows an efficient reconstruction). A strategy to choose this value is by exploiting the scree plot of each omic.

All the layers are followed by the softsign activation function (as done in [4]), except for the bottleneck layer and for the output layer, for which a sigmoid

function is used. Each layer is then followed by a Batch Normalization layer (except for the bottleneck and for the output layer). The number of parameters of input layer of the Encoder and output layer of the Decoder are equal, by definition, to the number of features of the omic.

Take for example the mRNA omic in the lung dataset 4.2, with 783 samples and 19452 features, and suppose that you want your bottleneck layer to have 64 neurons (this will be dimension of the latent space). The resulting customized auto-encoder architecture is reported in the table 1.

Encoder			
Layer	**Input shape**	**Output shape**	**Activation Function**
Dense	$(?, 19452)$	$(?, 1945)$	softsign
BatchNorm			
Dense	$(?, 1945)$	$(?, 194)$	softsign
BatchNorm			
Dense	$(?, 194)$	$(?, 64)$	sigmoid
Decoder			
Layer	**Input shape**	**Output shape**	**Activation Function**
Dense	$(?, 64)$	$(?, 194)$	softsign
BatchNorm			
Dense	$(?, 194)$	$(?, 1945)$	softsign
BatchNorm			
Dense	$(?, 1945)$	$(?, 19452)$	sigmoid

Table 1: mRNA (lung dataset) autoencoder architecture; the '?' corresponds to the minibatch size

Note however that this customized approach to the AE architecture was not always respected faithfully in our experiments: we wanted our encoder and decoder to have 3 dense layers each, but the customized number of neurons was adopted only when the number of features of the omic was above 10,000. For example, the second and second-last dense layer of the AE for the miRNA omic of the lung dataset (which has 1623 features) is chosen to have 100 neurons and not 16 neurons.

3.4 Experiments

Our auto-encoders need hyperparameter tuning. We manually finetune the autoencoders architecture at training phase, ending up with 3 dense layers for

both the encoder and decoder. Batch Normalization is introduced after each layer in order to regularize the network.

A 3-fold cross validation grid search is exploited to find the optimal batch size and learning rate, for each autoencoder to be trained. Early stopping regularization (based on the evolution of the validation reconstruction loss) is used to avoid overfitting and tune the number of epochs of the two training phases.

4 Results

In this section we discuss the results of our method on the three presented datasets.

We ran our method on the three datasets, and recorded the Adjusted Mutual Information (AMI) and Adjusted Rand Index (ARI) scores between the true cluster labels and the labels predicted by the spectral clustering performed on the integrated (average) distance matrix. For a matter of visualization, we also computed the confusion matrix between the true cluster labels and the predicted cluster assignments; note that in general there is no accordance between a specific true cluster label and a specific predicted cluster label: this association is arbitrary, and is up to permutations of the cluster labels. For example, from the confusion matrix we may read that "samples having true label 0 are (almost) all associated to predicted cluster label 3", and this would be perfectly fine.

We also evaluated the clusterings obtained with some literature methods on the three datasets: Multiple Factor Analysis (MFA) [1], iCluster+ [5] and Similarity Network Fusion (SNF) [7].

4.1 Synthetic dataset

The first dataset to which we applied our method is the synthetic one, at which we have added gaussian salt & pepper noise, due to being very separable even at two dimensions (see sect. 2.1). This dataset has 5 distinct clusters. The dimension of the latent space of each autoencoder is 64; Since the number of features for each omic is fairly small, we could not think of using the custom architecture proposed in sect. 3.3; we opted for autoencoders consisting of an encoder with two dense layers of respectively 512 and 256 neurons, a bottleneck layer of 64 neurons and a decoder with two dense layers of respectively 256 and 512 neurons and the final output layer with the same n. of features of the considered omic.

First, we tried to evaluate a simple, naive type of integration: we concatenated the three pre-processed and noisy omics and ran K-means on the resulting dataset. This is known as early integration. Despite the added noise, the K-means algorithm on the early integrated dataset is able to correctly cluster all

the samples. We visualize the resulting dataset (reduced to two dimensions with PCA) and the confusion matrix between the true and predicted clusterings in fig. 7.

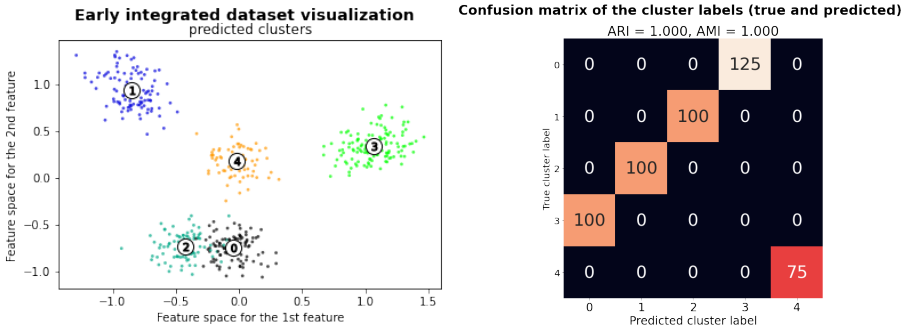

Figure 7: Early integrated synthetic dataset visualization (first two principal components) and confusion matrix of the true and predicted clusterings

Our method is able to obtain the same results of K-means on the early integrated dataset, since it correctly found K=5 clusters (the one which yields the highest silhouette, given the avg. distance matrix and the 5-means cluster labels) and clustering all the samples in these 5 clusters in the same way of K-means on early integrated dataset (up to arbitrary permutations of cluster labels). We visualize the integrated distance matrix by means of multidimensional scaling to two dimensions as well as the confusion matrix of the resulting clustering in fig. 8.

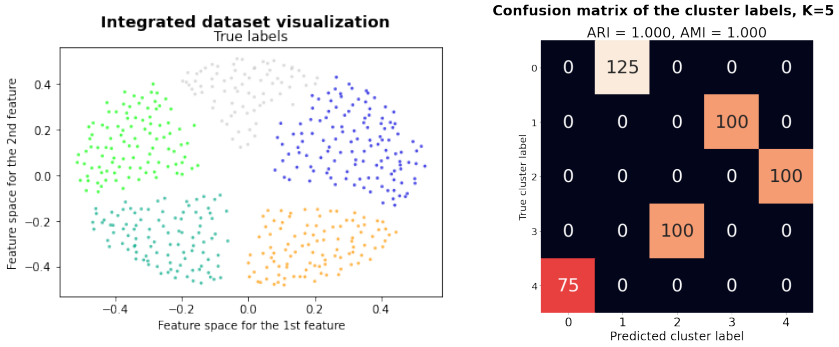

Figure 8: Our method; Synthetic dataset visualization (multidim. scaling applied on the integrated average distance matrix) and confusion matrix of the true and predicted clusterings

We compare our results with the other integration methods we cited previously.

First, we evaluated SNF. We first performed SNF's fusion process and re-

trieved the fused overall status matrix P_t^c; the optimal number of clusters, based on P_t^c, is estimated via an eigengap approach, and the result is five, the correct one. Spectral clustering with K=5 is performed on the fused network matrix. SNF with spectral clustering is able to achieve the same AMI=1 and ARI=1 achieved also by early integration with K-means and our method, thus giving a perfect result.

The second method we evaluated is iCluster+. A problem with iCluster+ is that its computational cost becomes unbearingly high if the omics have a high number of features. Therefore, we first reduced the number of features of the synthetic omics to 128 with PCA. Upon obtaining the latent variable matrix Z (with 64 latent variables, for the sake of comparison with our method), we ran K-means on it, trying several values for K; based on the silhouette score, the optimal number of clusters is 4, which is incorrect. The confusion matrix for K=4 is reported in fig. 9. We also report the confusion matrix for K=5, used in the hypothetical case in which we know a priori the number of clusters in the dataset.

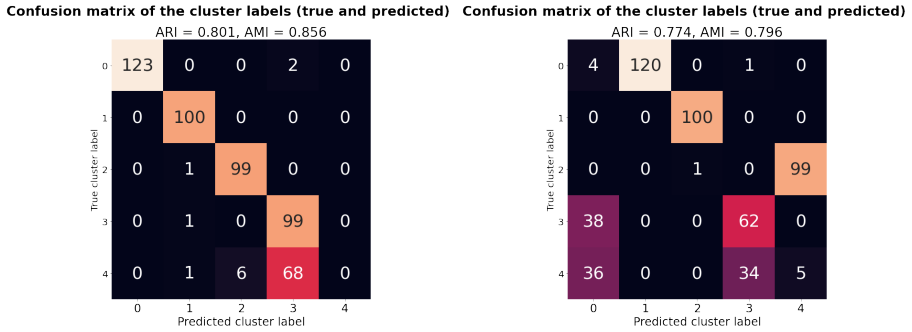

Figure 9: Confusion matrices for iCluster+ with K-means; K=4 (left) and K=5 (right)

We argue that this incorrect result by iCluster+ is due to the fact that this method is not very robust to noise in the data. To verify this assumption, we evaluated iClusterPlus+ with K-means on the original synthetic dataset, without noise. The best result, based on the silhouette score, is indeed K=5, and the AMI and ARI are both 1 (perfect clustering).

Finally, we evaluated the MFA integration method. We chose a number of latent variables equal to 64 (for the sake of comparison with our method) and retrieved the latent variable matrix H; then performed K-means on it with several values of K, and recorded the silhouette score for each. The best silhouette is obtained for K=5 (correct number of clusters); the cluster labels predicted by K-means result in a AMI and ARI scores equal to 1. This perfect result was expected, since MFA is a less naive extension of PCA, which takes into account also the differences between the scales of the features of the different omics of a dataset.

Table 2 recaps the obtained results.

Integr. method	K found	ARI	AMI
Early integration	5	1	1
SNF	5	1	1
iCluster+	4	0.801	0.856
MFA	5	1	1
Ours	5	1	1

Table 2: Synthetic dataset results

4.2 Lung dataset

We tested our method on the Lung dataset. This dataset has 2 distinct clusters. We chose 64 as dimension of our latent space for all the four autoencoders.

As a baseline, we perform early integration of the four omics, followed by K-means, with several values of K. The maximum silhouette score was obtained for K=2, the correct number of clusters. We visualize the early integrated dataset (reduced to 2 dimensions with PCA) with the predicted cluster labels as colors and the associated confusion matrix in fig. 10.

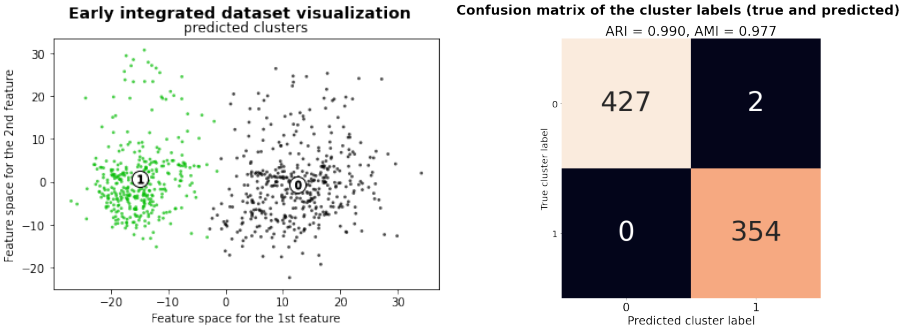

Figure 10: Early integrated lung dataset visualization (first two principal components) and confusion matrix of the true and predicted clusterings

We evaluated our method on the lung dataset, and verified that it correctly finds K=2 clusters and achieves a perfect AMI and ARI score (both equal to 1), therefore performing fairly better than early integration and clustering. We chose 64 as the dimension of the latent space for each autoencoder, and used the customized AE architecture described in sect. 3.3 except for the miRNA omic, for which the encoder has two hidden layers of 162 and 100 neurons respectively. The visualization of the integrated average distance matrix (scaled down to 2 dimensions with Multidimensional scaling) and the confusion matrix are reported in fig. 11.

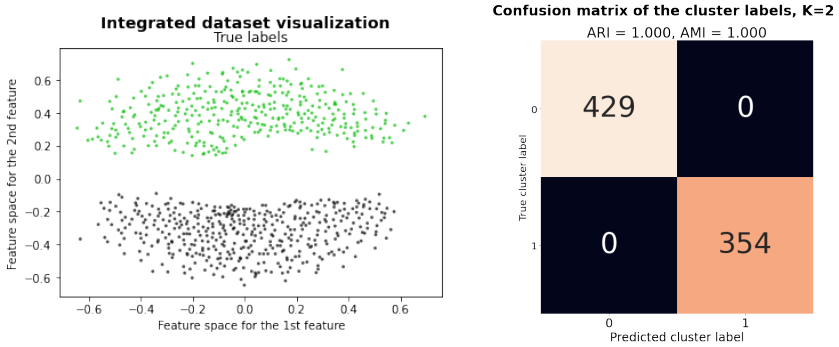

Figure 11: Integrated dataset (visualized through multidim. scaling) and confusion matrix of the true and predicted clusterings for the Lung dataset

We now compare our results with the other three integration methods introduced in the previous section.

First, we evaluated SNF. The optimal number of clusters returned by the eigengap approach of SNF is two, which is correct. We apply spectral clustering with K=2 on the overall status matrix, and visualize the obtained confusion matrix in fig. 12.

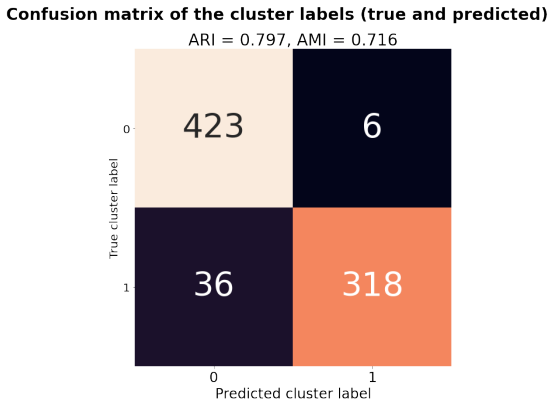

Figure 12: Confusion matrix of SNF with spectral clustering K=2 applied on the lung dataset

Second, we evaluated iCluster+. After having pre-processed the omics, we reduced their feature dimensionality to 256 with PCA, because of the high computational cost of iCluster+. Upon obtaining the latent variable matrix Z with 64 latent variables, we ran K-means on it, trying several values for K; based on the silhouette score, the optimal number of clusters is 2. The confusion matrix is reported in fig. 13

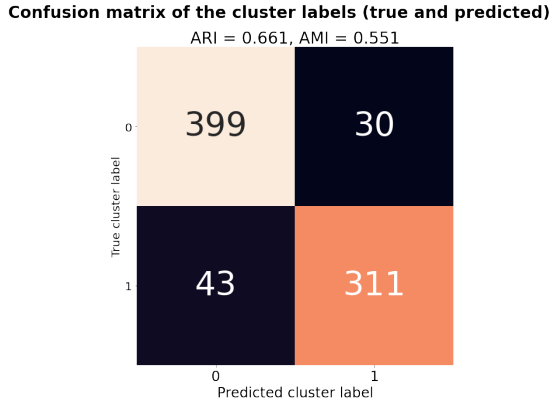

Figure 13: Confusion matrix for iCluster+ with K-means (K=2) applied on the lung dataset

Lastly, we evaluated MFA. We chose 64 as number of latent variables for the method. After having retrieved the latent variable matrix H, we performed K-means on it with several values for K. The best silhouette was obtained for K=2, as we would have expected. The confusion matrix for K=2 is reported in fig. 14

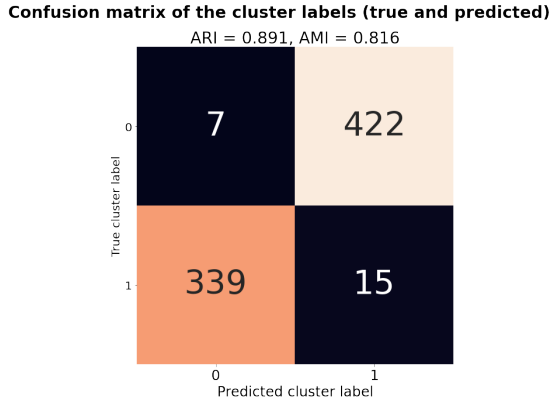

Figure 14: Confusion matrix for MFA with K-means (K=2) applied on the lung dataset

Table 3 recaps the obtained results.

Integr. method	K found	ARI	AMI
Early integration	2	0.990	0.977
SNF	2	0.797	0.716
iCluster+	2	0.661	0.551
MFA	2	0.891	0.816
Ours	2	1	1

Table 3: Lung dataset results

4.3 Kidney dataset

We applied our method on the Kidney dataset. This dataset has 3 distinct clusters. We chose 64 as dimension of our latent space, for all the three autoencoders (this choice was, once again, based on the scree plot of each omic).

First, we perform early integration of the three omics, followed by K-means with several values of K. We recorded the maximum silhouette score, which is obtained in correspondence of K=4 (incorrect). However, three of the four clusters seem to represent the effective three true clusters in our data, as we can see from the early integrated dataset visualization and resulting confusion matrix in fig. 15.

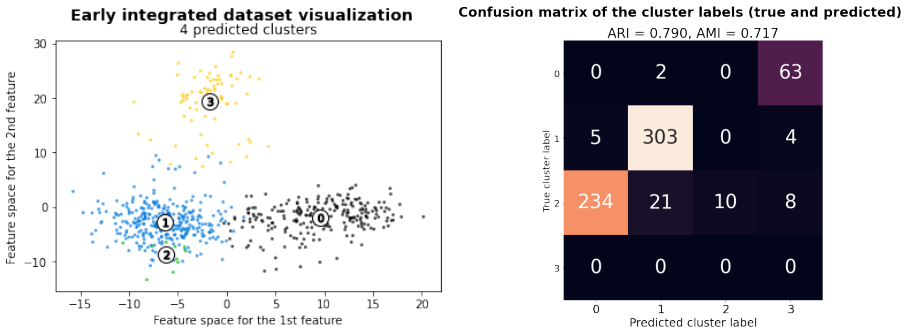

Figure 15: Early integrated kidney dataset visualization (first two principal components) and confusion matrix of the true and predicted clusterings

Our method is instead able to obtain the true number of clusters, since it correctly finds K=3 clusters (the one which yields the highest silhouette, given the avg. distance matrix and the 3-means cluster labels). Moreover, the results in terms of AMI and ARI are better than early integration clustering (see fig. 16)

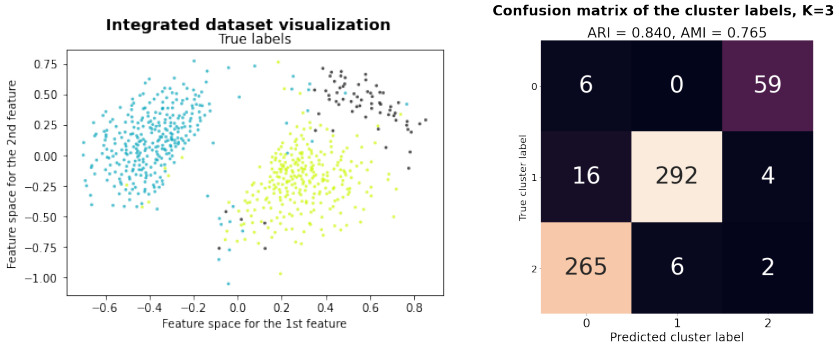

Figure 16: Kidney dataset visualization (multidim. scaling applied on the integrated average distance matrix) and confusion matrix of the true and predicted clusterings

We compare our results with the other integration methods considered.

First, we evaluated SNF. The optimal number of clusters returned by the eigengap approach of SNF is 5 (incorrect). However, spectral clustering with K=5 performed on the fused network matrix (i.e. overall status matrix) yields three out of five clusters which contain almost exclusively samples of the three true clusters (see fig. 17).

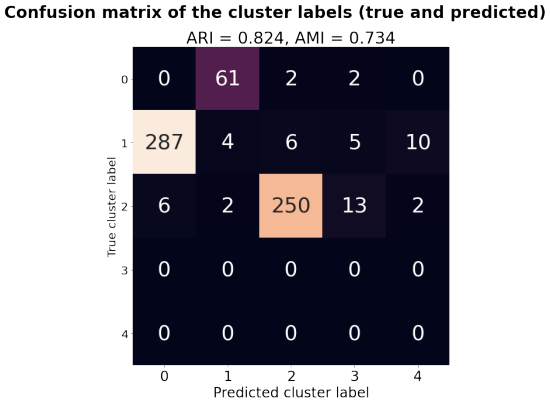

Figure 17: Confusion matrix of SNF with spectral clustering K=5 applied on the kidney dataset

The second method we evaluated is iCluster+. After having pre-processed the omics, we reduced their features to 256 with PCA, for each omic (this is because of the high computational cost of iCluster+). Upon obtaining the latent variable matrix Z with 64 latent variables, we ran K-means on it, trying several values for K; based on the silhouette score, the optimal number of clusters is indeed 3. The confusion matrix for K=3 is reported in fig. 18.

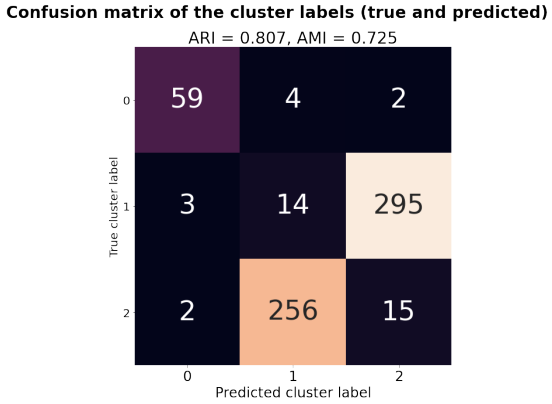

Figure 18: Confusion matrix for iCluster+ with K-means (K=3) applied on the kidney dataset

The last method is MFA. The number of latent variables we chose is again 64. We performed K-means on the latent variable matrix H with several values of K, and recorded the silhouette score for each. The best silhouette is obtained for K=4, which is not consistent with the three actual clusters of the data. The confusion matrix for K=4 is reported in fig. 19. In the same figure, we also report the confusion matrix for K=3, used in the hypothetical case in which we know a priori the number of clusters in the dataset.

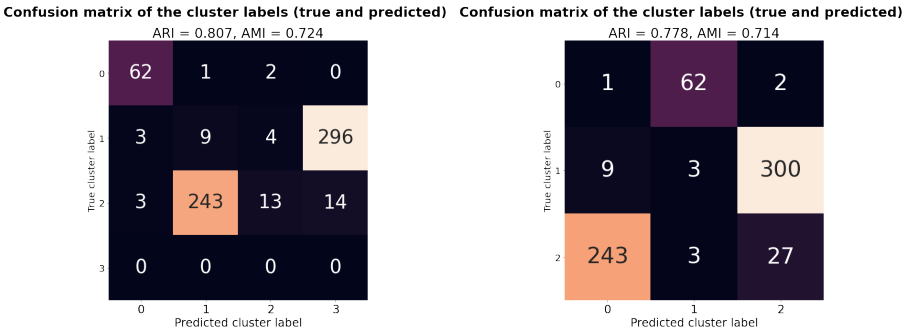

Figure 19: Confusion matrices for MFA with K-means applied on the kidney dataset; K=4 (left) and K=3 (right)

Table 4 recaps the obtained results.

Integr. method	K found	ARI	AMI
Early integration	4	0.790	0.717
SNF	3	0.824	0.734
iCluster+	3	0.807	0.725
MFA	4	0.807	0.724
Ours	3	0.840	0.765

Table 4: Kidney dataset results

5 Conclusions

We presented a new methodology for performing multi-omics integration, which uses traditional autoencoders with a custom loss term for the dimensionality reduction of each omic and an average performed over the distance matrices associated to each encoded omic. From the knowledge of this average distance matrix, we can perform spectral clustering on it.

This integration method resembles a "kernel method": we map each point to a multidimensional non-linear space, and for each couple of points we know the pairwise distances between them in this new space, but not their absolute coordinates.

This method shows some similarities and some differences with respect to the other evaluated methodologies:

- our method is similar to MFA and iCluster+ in that they find a lower-dimensional representation of each omic;
- our method is similar to SNF in that the output of the integration process is a num. samples $\times$ num. samples matrix for both (similarity for SNF, distance matrix i.e. dissimilarity matrix for ours), for which the application of spectral clustering is particularly recommended to perform.

Our approach has some advantages in comparison with other literature methods:

- our method uses deep learning and a custom loss function to perform a non-linear dimensionality reduction, differently from the other cited methods; few other methodologies exist in literature which make use of neural networks for the integration of multi-view datasets;
- our method is robust to noise in the data (see sect. 4.1)
- the combination of reconstruction loss and closest cluster center loss makes our method reliable in spite of the combined application of K-means + silhouette at training time, which may yield a wrong number of clusters with respect to the real one;

- our method has the advantage of being reasonably fast even when the dimensionality of the omics is very high (vs iCluster+).

In the end, we verified that our method outperforms the other cited methods, which are very commonly used in the context of multi-omics integration [6].

References

[1] Hervé Abdi, Lynne J. Williams, and Domininique Valentin. "Multiple factor analysis: principal component analysis for multitable and multiblock data sets". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 5.2 (Feb. 2013), pp. 149–179. DOI: 10.1002/wics.1246. URL: https://doi.org/10.1002/wics.1246.

[2] Abdelkader Behdenna et al. "pyComBat, a Python tool for batch effects correction in high-throughput molecular data using empirical Bayes methods". In: (Mar. 2020). DOI: 10.1101/2020.03.17.995431. URL: https://doi.org/10.1101/2020.03.17.995431.

[3] *Detecting the presence of batch effects*. URL: https://www.goldenhelix.com/learning/knowledge-base/svs-microarray-cnv-quality-assurance-tutorial/#b-detecting-the-presence-of-batch-effects.

[4] Margherita Francescatto et al. "Multi-omics integration for neuroblastoma clinical endpoint prediction". In: *Biology Direct* 13.1 (Jan. 2018). DOI: 10.1186/s13062-018-0207-8. URL: https://doi.org/10.1186/s13062-018-0207-8.

[5] Qianxing Mo et al. "Pattern discovery and cancer gene identification in integrated cancer genomic data". In: *Proceedings of the National Academy of Sciences* 110.11 (Feb. 2013), pp. 4245–4250. DOI: 10.1073/pnas.1208949110. URL: https://doi.org/10.1073/pnas.1208949110.

[6] Indhupriya Subramanian et al. "Multi-omics Data Integration, Interpretation, and Its Application". In: *Bioinformatics and Biology Insights* 14 (Jan. 2020), p. 117793221989905. DOI: 10.1177/1177932219899051. URL: https://doi.org/10.1177/1177932219899051.

[7] Bo Wang et al. "Similarity network fusion for aggregating data types on a genomic scale". In: *Nature Methods* 11.3 (Jan. 2014), pp. 333–337. DOI: 10.1038/nmeth.2810. URL: https://doi.org/10.1038/nmeth.2810.