Project 6: Multiple Omics Deep learning models

Deep learning is a subfield of machine learning that uses deep neural networks to solve complex problems. It is increasingly used in personalized medicine to improve disease prediction, detection and diagnosis. Recent advances in data processing and high performance computing have enabled the implementation of increasingly powerful deep learning models, providing new opportunities for personalized medicine.

Deep learning models for personalized medicine can also be used to combine different sources of "Omics" data such as genetic, proteomic and methylation data. These models, called "Multiple Omics Deep learning models", can help to better understand the mechanisms underlying disease and identify biomarkers relevant for diagnosis and prognosis. These models can also be used to assess the interactions between different sources of omics data and to evaluate how these interactions may influence the response to treatments. These models can also be used to identify potential therapeutic targets and to evaluate response to treatments.

The Cancer Genome Atlas - Bladder Urothelial Carcinoma (TGCA-BLCA) is an online database containing genetic, epigenetic, transcriptomic and proteomic data for urothelial carcinoma of the bladder. These data are obtained from different high-throughput techniques, such as high-throughput sequencing, package proteomics and phosphoproteomics. This database was created by The Cancer Genome Atlas (TCGA) initiative, a large-scale project that aims to map the genomes of different types of cancer. The data in the TGCA-BLCA database are publicly available and are updated regularly.

Urothelial carcinoma of the bladder, or bladder cancer, is the most common type of bladder cancer. It develops in the upper layers (or urothelium) of the bladder. There were 82,290 new cases and 16,710 deaths from bladder cancer in 2022.

We can use this database to create "Multiple Omics" deep learning models to study urothelial bladder carcinomas. These models will be able to combine different sources of omics data to better understand the underlying mechanisms of the disease and identify relevant biomarkers for diagnosis and prognosis.

After a preprocessing of the data, the genetic, proteomic and methylation datasets will be combined to create a multi-omics dataset. We will then use different deep learning algorithms to try to best predict the type of bladder cancer and build a consistent model. Finally, we will use cross-validation to evaluate the performance of the models and to identify the hyperparameters that give the best results.

1. Description of the data

This analysis brings together 3 datasets from 3 omics.

1. Gene expression dataset

The gene expression database contains gene expression data that has been quantified using the Illumina HiSeq platform. The data has been normalized, meaning that it has been adjusted to account for variations in measurement or other factors so that it is more comparable across samples.

The dataset contains 427 samples for 20532 gene expression variables listed.

2. Proteomic dataset

This database contains protein expression data that has been generated using a technique called Reverse Phase Protein Array (RPPA). RPPA is a high-throughput technique that allows for the simultaneous measurement of the levels of multiple proteins in a single sample. The data in this database has been generated by the MDA RPPA Core.

The data set contains 343 samples for 246 listed protein level variables.

3. Methylation dataset

This database contains DNA methylation data that has been generated using the Illumina Human Methylation 450 platform. DNA methylation is a process by which methyl groups are added to the DNA molecule, which can affect gene expression and regulation. The Illumina Human Methylation 450 platform is a widely used technology for measuring DNA methylation on a genome-wide scale. It allows for the simultaneous assessment of methylation status at over 480,000 CpG sites in the genome.

This database is the most exhaustive with 440 samples and 485578 variables but also the largest (3Gb) making it very complicated to manipulate.

In order to reduce the size of this database, the least informative variables were removed using the Anova method. For this, the methylation dataset was divided into 12 sub-datasets (of around 440 by 30001). A python script then used ANOVA to reduce the size of these datasets to 8000 columns. Finally, the 12 reduced sub-datasets were combined into a final "Methylation" dataset of 440 by 96001.

4. Class datasets

A dataset containing the actual classes of the Gene expression and Methylation datasets is also used. For proteomics, we used the samples in common with the two other classes in order to deduce these own classes.

2. Normalization step

1. Merging the data

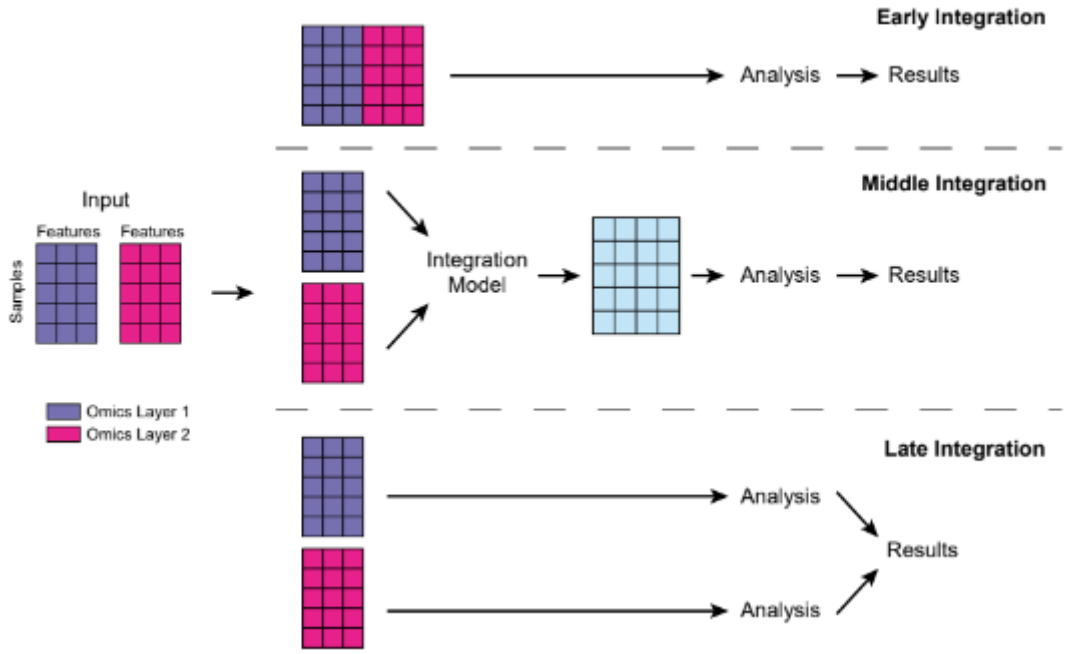


Fig 1 : Illustration of early, middle, and late integration for merging data matrices generated by different omics. (Cai et al. 2022)

For our multi-omics model, we chose an early integration method to merge the different omics datasets.

All steps :

- For each of the omics, the barcodes are normalized in order to merge the datasets. The barcode elements are separated by "." and the first four elements are selected to correspond to the codes of the samples by individuals. Then we add the classes associated to each sample for the methylation and gene expression dataset.

- Next, the 3 datasets are merged according to the barcode. Indeed, the same sample could be used for several omics analysis. We thus obtain a dataset of 425 by 116780. We then check that for the same sample, the classes coming from the gene expression and methylation data are identical and we remove one of the two columns that is duplicated.

- We observe the number of individuals per class:

Transitional cell carcinoma	356
Papillary transitional cell carcinoma	66
Squamous cell carcinoma, NOS	1
Papillary adenocarcinoma, NOS	1
Carcinoma, NOS	1

- We notice that the classes Squamous cell carcinoma NOS, Papillary adenocarcinoma NOS and Carcinoma NOS are under represented (only 1 sample) , so we remove them from the dataset.

Finally, we separate the data from the classes.

2. Reduction of the amount of information

In order to normalize the data, we will replace the empty values by the median of the column.

Moreover, an ANOVA is performed on the dataset in order to select the most informative variables by selecting 40000 variables for a final dataset of 422 by 40000.

Finally, the class dataset is normalized into binary variables.

3. Learning model and cross validations

1. Multi-Layer Perceptron (MLP)

An Multi-Layer Perceptron (MLP) is composed of multiple layers of artificial neurons, or nodes, that are connected by directed edges. The input layer receives the input data, and the output layer produces the final output. The layers in between are called hidden layers, and they transform the input data into a form that can be understood by the output layer. Each layer is fully connected to the next, meaning that each neuron in a layer is connected to all the neurons in the next layer.

We start by separating the dataset into train and test datasets.

Our MLP model uses 5 hidden layers using the Keras library. It starts by importing various necessary libraries, such as the EarlyStopping library for early stopping, the Dropout library for preventing overfitting, and the regularization library for adding regularization to the model.

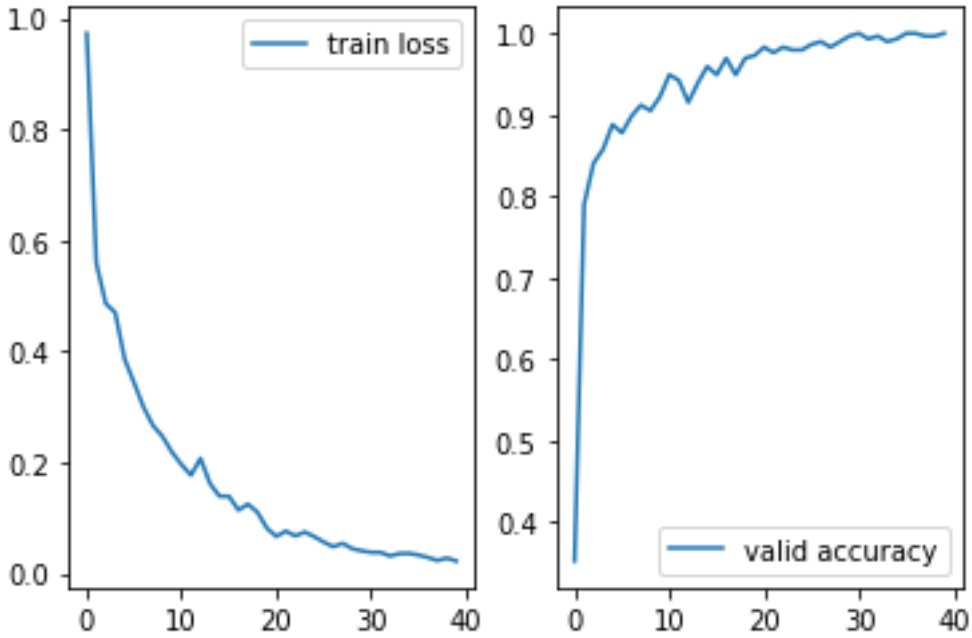
Next, the model's parameters are defined, such as the learning rate, optimizer, dropout rates for each hidden layer, L2 regularization, batch size, and number of epochs. The number of features used to train the model is also defined.

The model itself is created using the Keras Sequential class, which allows for creating a model by stacking layers. The model includes five hidden layers, all using a relu activation and L2 regularization. The first layer has 2000 neurons with an input dimension of nb_features. There is also a dropout layer for each hidden layer to help prevent overfitting. The last layer is an output layer with 2 neurons and a softmax activation function for a multi-class classification problem.

Finally, the model is compiled using the defined optimizer, a categorical cross-entropy loss function, and an accuracy metric. Early stopping is also added to stop training when validation accuracy no longer improves. The model is then trained on the provided training data, with validation data to evaluate the model's performance during training.

After evaluating the model, we notice that the accuracy score is 85.04%

And we have the following graph :



We notice that train loss decreases with time until it reaches a level close to 0 and that valid accuracy increases with time and seems to tend to 1. We notice that the curves are not perfectly smooth. This is probably due to the dropout which helps to avoid overfitting.

We can conclude that the model is valid.

Then, cross validation was used to test the model by varying the hyperparameters. The data is splitted using kfold cross validation. The kfold.split function is used to split the training data into k number of folds. The variable train contains the indices of the training data for the current fold, and test contains the indices of the validation data for the current fold.

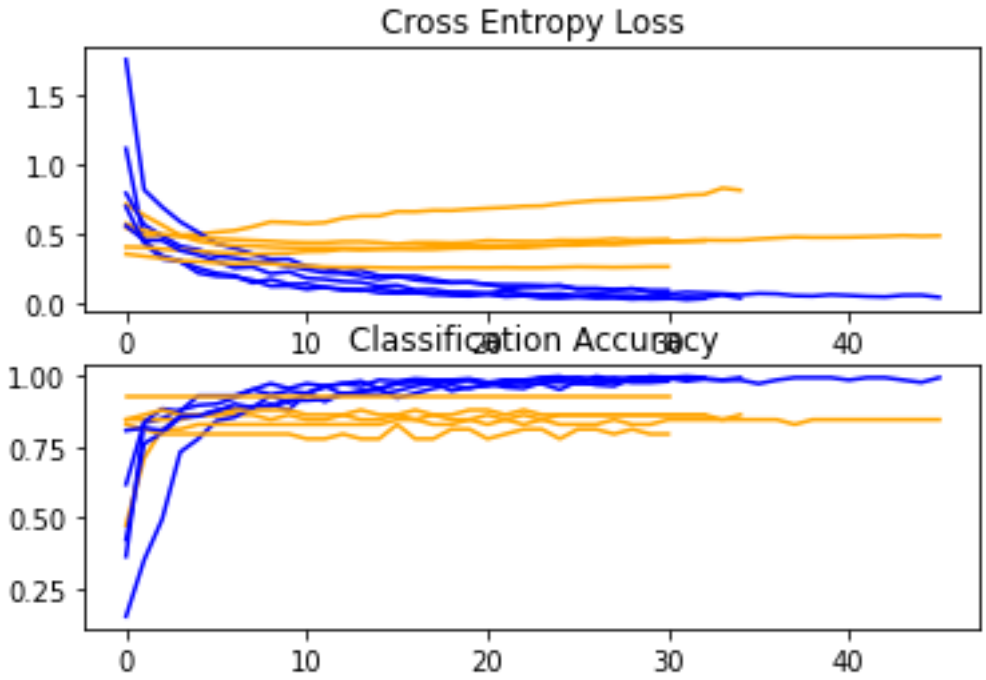
For each iteration of the for loop, a new instance of the Sequential model is created and a series of layers are added to the model as described in the previous answer.

After evaluating the model, we notice that the mean accuracy score is 86.102%

Graph of entropy loss and accuracy for train and test values after cross validation of the MLP model :

Blue -> train

Orange -> test



We notice that for the cross Entropy Loss and Classification Accuracy, the blue and orange lines (corresponding respectively to the trainings and the tests) seem to be close in general. However, for Entropy Loss, it seems that some tests diverge from the train, so it is possible that there is some overfitting.

But in general, we can conclude, after variation of the hyperparameters, that the model seems correct.

2. Convolutional Neural Network (CNN)

A CNN (Convolutional Neural Network) is a type of neural network that uses multiple layers of filters to extract increasingly complex features as the data is processed. Deep learning CNNs for omics data such as gene expression, methylation, and proteomics can be used for tasks such as gene classification, prediction of gene regulation, or identification of relevant biomolecular markers for disease.

For gene expression data, convolution techniques can be used on gene sequences to detect relevant genetic patterns. For methylation data, convolution techniques can be used on methylation sequences to identify relevant methylation sites. For proteomics data, convolution techniques can be used on protein sequences to identify relevant proteins and protein regions.

Our code is using a k-fold cross validation method to train and evaluate a CNN model for a classification task. The data is first split into training and testing sets (X_train, X_test, y_train, y_test) with a test size of 0.3. Then, kfold is initialized with 5 splits, shuffle set to True, and random_state set to 42.

For each iteration of the cross validation, it creates a sequential model with a 1D convolutional layer, followed by a max pooling layer.Then, it adds a 1D convolutional layer, followed by a max pooling layer and a dropout layer. After, it adds another 1D convolutional layer, followed by another max pooling layer. Then it flattens the output and adds two dense layers, one with 100 neurons and one with 2 neurons, with the last one having a softmax activation function for multi-class classification.

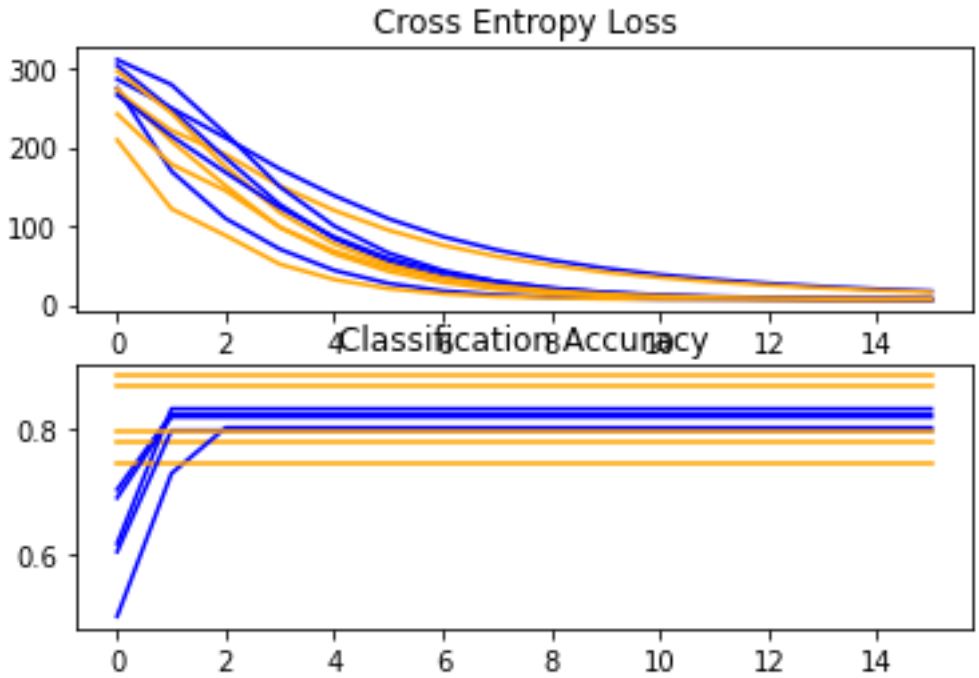
The model is then compiled with the Stochastic Gradient Descent optimizer, categorical cross-entropy loss, and accuracy metric. The model is then fit to the training data and evaluated on the test data. The train loss and validation accuracy are plotted after each iteration. Finally, the code prints the accuracy mean and standard deviation and creates a box plot of the results.

After evaluating the model, we notice that the mean accuracy score is 81.356%

Graph of entropy loss and accuracy for train and test values after cross validation of the CNN model :

Blue -> train

Orange -> test



Comparing the train and test curve for accuracy and entropy loss. We can see that the curves merge and follow the same pattern. We can conclude that the CNN model does not seem to suffer from overfitting which is a good thing. Moreover, the average accuracy of 81.36% shows that the model makes good predictions.

We can conclude that this CNN model is correct in avoiding overfitting and that it correctly predicts the cancer classes.

4. Conclusion

The goal of using all available information, including gene expression, methylation, and protein expression, is to accurately classify each subtype of bladder cancer.

In conclusion, utilizing multiple omics informations is crucial in accurately classifying the subtypes of bladder cancer. This approach can lead to improved diagnostic accuracy and personalized treatment plans, ultimately leading to better patient outcomes. By utilizing these multiple layers of information, we can gain a deeper understanding of the underlying biology of bladder cancer and pave the way for more effective therapeutic strategies in the future.

Bibliography :

Cai, Zhaoxiang, Rebecca C. Poulos, Jia Liu, and Qing Zhong. 2022. “Machine Learning for Multi-Omics Data Integration in Cancer.” *IScience* 25 (2): 103798.