Bioinformatics project: active regulatory regions prediction using ML

Irene Mantegazza
895776
irene.mantegazza@studenti.unimi.it

Introduction

The exponential growth of the amount of biological data available raises two problems: on one hand, efficient information storage and management and, on the other hand, the extraction of useful information from these data. The second problem is one of the main challenges in computational biology, which requires the development of tools and methods capable of transforming all these heterogeneous data into biological knowledge. These tools and methods should allow us to provide knowledge in the form of testable machine learning models. There are several biological domains where machine learning techniques are applied for knowledge extraction from data: genomics, proteomics, microarrays, systems biology, evolution and text mining.

Genomics is one of the most important domains in bioinformatics. From genome sequences, we can extract the location and structure of the genes; also,we can identify the active or inactive regulatory regions[1]. In the human genome, less than 2% of the DNA sequence is made up of protein-coding genes; while, the rest of the genome is non-coding and was previously regarded as junk DNA, with no known purpose [2]. Subsequently, scientists realized that some of non-coding regions are essential to the function of cells, especially to the control of gene activity. Regulatory regions can be classified into cis-regulatory, which regulate the expression of the nearby genes or trans-regulatory factors, which control the expression of the distant genes. We will focus on the first class, whose the two most known and important elements are promoters and enhancers.

Promoters are about 100-1000 base pairs long and are adjacent and typically upstream of the transcribed gene[3].
They control the binding of DNA to RNA polymerase which transcribes DNA to mRNA that will be translated into a functional protein. Therefore, promoters function like a switch to turn on or off the transcription of the target gene.
Enhancers, which are about 200 base pairs, can dynamically control the expression level of the target gene through their interaction with promoters, even if they are far away from their target genes in the linear sequence space. Although distal to the promoter which it should bind to, a transcriptionally active enhancer is brought close to its target promoter by DNA looping in 3D nuclear space and then connected to it through some transcription factors (which are proteins), called activators. Once these transcription factors create the link, enhancers enhance or sometimes diminish the transcription rate of the gene. If only active, promoters and enhancers can perform their function of regulating gene expression; hence, it is important to identify the active regulatory region inside a given cell type. The cellular line which has been assigned to me is HEK293, whose complete name is Human embryonic kidney 293[4]. It is a immortalised cell line derived from a spontaneously miscarried or aborted fetus or human embryonic kidney cells grown in tissue culture taken from a female fetus in 1973.
For immortalised cell lines [5] we mean cells that have been manipulated to proliferate indefinitely and can thus be cultured for long periods of time.

Experimental setup

The considered tasks

The aim of the project is to detect if the promoters or enhancers in the assigned cell line are active and so they contribute to the regulation of the gene expression, or inactive.

Datasets

The datasets that have been used are:
- epigenomic dataset from ENCODE
- genomic sequence from UCSC Genome Downloader

Regarding the labels, I retrieved them from FANTOM.

The Encyclopedia of DNA Elements (ENCODE) is a public research project which aims to identify functional elements in the human genome.

ENCODE supports further biomedical research by generating community resources of genomics data, software, tools and methods for genomics data analysis, and products resulting from data analyses and interpretations[6].

The UCSC Genome Browser is an online and downloadable genome browser hosted by the University of California, Santa Cruz (UCSC). It is an interactive website offering access to genome sequence data from a variety of vertebrate and invertebrate species and major model organisms, integrated with a large collection of aligned annotations[7].

FANTOM (Functional ANnoTation Of the Mammalian genome) is an international research consortium focused on functional annotation of mammalian genomes and characterization of transcriptional regulatory networks[8].

In the work, the considered genome is hg38 assembly, the first major revision of the human genome[9] and the chosen window size , which indicates the number of nucleotides in the considered DNA region, is 256.

I extracted the epigenomic data through the *epigenomic_dataset*[10] Python package and the genomic sequence using *ucsc_genomes_downloader* [11] Python package .

The promoters epigenomic dataset is composed of 196 features and 80408 samples; insted, the enhancers epigenomic dataset is a dataframe of 196 features and 63285 rows.

The Figure1 shows the epigenomic data of promoters: expressed in bed file format, the first three columns represent the coordinates of the considered region and the fourth indicates the strand. A strand of DNA is referred as positive sense if its sequence corresponds to the mRNA transcript produced; however, the other strand, called negative sense, is transcribed by the RNA polymerase and it is complementary to the other one. The names of all the other columns are the names of the genes relative to a protein we are measuring the activation of. The values of these columns are obtained by averaging or calculating the maximum of the values inside the given window .

The enhancers data are very similar to the promoters data; the only difference is the absence of values of the strand column. As a matter of fact, enhancers are not next to the gene,but symmetrical on the DNA spiral.

The genomic sequence is represented by an array of long strings composed of A, T, G, C which are the initial letters of the four nucleotides.

The labels are represented through TPM[12], which is a metric with the aim to normalize for sequencing depth and gene length.

Class imbalance

Data are said to suffer the class imbalance problem when the class distributions are highly imbalanced. If a dataset is affected by class imbalance, many classification learning algorithms will have low predictive accuracy for the infrequent class[13]. In our case, this is not a problem because the imbalance is always better than 1/10 (Figure2).

Rate between features and samples

The rate between features and samples must be greater than 1; it means that the number of samples exceeds the number of features and so we don't risk generating a model with overfitting. We have a rate of 509.5 for promoters and 322.8 for enhancers.

Constant features

The epigenomic dataset doesn't present features with constant values.

Data pre-processing

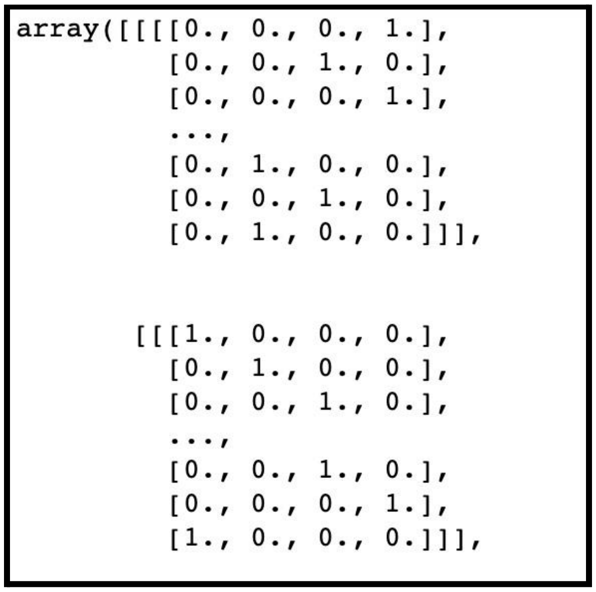
Data pre-processing consists of the following steps:

1. one-hot encoding of sequence data

2. conversion of the labels to make them suitable for binary classification task
3. data imputation
4. feature scaling
5. feature selection

One-hot encoding

I used the KerasBadSequence[13] package to make the one-hot-encoding of the genomic sequence. One-hot encoding is a process by which categorical variables are converted into a form that could be provided to machine learning algorithms to do a better job in prediction. It creates a new binary feature for each possible category and assigns a value of 1 to the feature of each sample that corresponds to its original category. The result is:



```
array([[[[0., 0., 0., 1.],
         [0., 0., 1., 0.],
         [0., 0., 0., 1.],
         ...,
         [0., 1., 0., 0.],
         [0., 0., 1., 0.],
         [0., 1., 0., 0.]]],


       [[[1., 0., 0., 0.],
         [0., 1., 0., 0.],
         [0., 0., 1., 0.],
         ...,
         [0., 0., 1., 0.],
         [0., 0., 0., 1.],
         [1., 0., 0., 0.]]],
```

Figure 3: example of the one-hot encoding of the genomic sequence

Labels conversion

Since we want to make binary classification to detect the active or inactive regions, we have to change the values of the labels so that the inactive regions could be represented by 0 and the active ones by 1. The chosen approach for the conversion obtains a balance between two classes by selecting a value close to 1 as threshold for promoters and a value close to 0 for enhancers. Thus, I decided to set 0 and 1 as thresholds for enhancers and promoters.

Data imputation

Data imputation is the procedure to replace the missing values, or Nan values, with substituted values. Imputation is important because missing values can drastically impact the data analysis and the machine learning model's quality. Luckily, from the Nan detection, it results that ,in the enhancers epigenomic dataset, there is only one missing value. Among all the methods of data imputation, I selected the knn imputation, which uses the k-nearest-neighbor (k-nn) algorithm to predict the Nan value.

Feature scaling

The goal of feature scaling (or normalization) is to transform features to be on a similar scale. If the dataset presents features with large values,these large values can dominate or skew some machine learning algorithms. The result is that the algorithms pay most of their attention to the high values and ignore the variables with smaller values. One of the normalization techniques is z-scoring that represents the number of standard deviations away from the mean. We would use z-scoring to ensure the feature distributions have mean = 0 and standard deviation= 1.[14] Because of the presence of outliers, the calculations of mean and standard deviation could be skewed. Thus, to avoid the influence of the outliers, in this project, I used the robust scaler that removes the median and scales the data according to the quantile range[15]

Feature selection

The goal of feature selection is to find the best set of features that allows building useful models. Indeed, the irrelevant features can decrease the accuracy of the models and make the model learn based on unimportant features.
Boruta is a feature selection method that trains a random forest classifier or regression and computes a feature importance metric to evaluate the importance of each feature.
Because of its computational expensiveness, I didn’t perform Boruta.

Data distributions and correlations

Features distributions

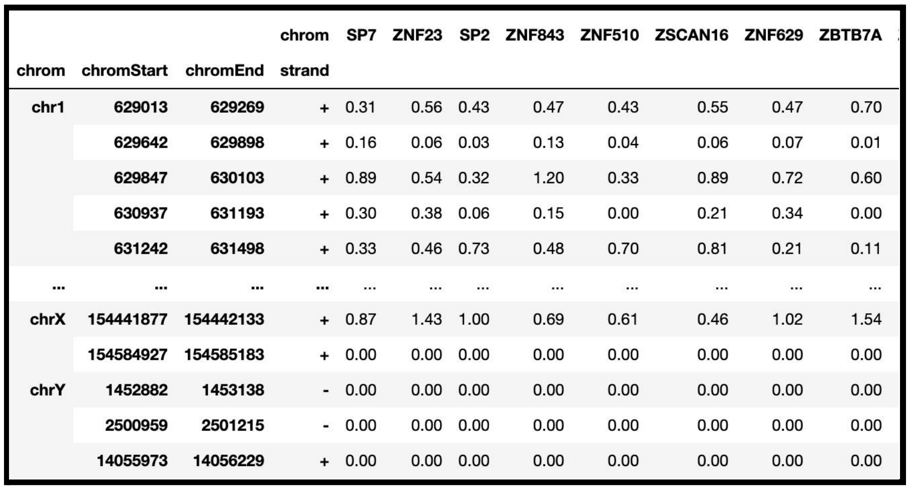
Since the epigenomic dataset is composed of many features, I use only the 5 most different features in the promoters data and the 5 most different features in the enhancers data to show the feature distribution. The metric adopted for the distance is the pairwise euclidean distance and, to reduce the impact of outliers, they are filtered before the 0.01 percentile and the 0.99 percentile. In figure 3, we can see the histograms showing the distributions of the features.

Correlation with output

If a feature is not correlated with the output, it isn’t informative and causes gaussian noise. To check if the features of the epigenomic dataset are correlated with the output, we can use different scores: the Pearson coefficient, the Spearman coefficient and the Maximum information coefficient (MIC).
The Pearson coefficient measures the linear correlation between two datasets, assuming that the two datasets are normally distributed. Spearman score captures the monotonicity of the relationship of two datasets and it doesn’t require that the two datasets are normally distributed. In the end, MIC coefficient is able to measure linear and non linear correlation. Since calculating MIC score can be computationally expensive, I only calculated it for the features that aren’t correlated with the output according to the previous tests.

About Spearman and Pearson tests, we consider a feature and the output correlated if the p-value is lower than 0.01.
P-value represents the significance of the correlation and a small p-value is an indication that the null hypothesis, according to which the two variables are not related, is false.Therefore, we can conclude that the correlation coefficient is different from zero and that a linear relationship exists[16]. On the contrary, there is no correlation and we have to eliminate the feature.
The features non correlated with the output are shown in the following table(table 1).

In the models evaluation, we will find and drop the features non correlated with the output inside the training set for each holdout split.



			chrom	SP7	ZNF23	SP2	ZNF843	ZNF510	ZSCAN16	ZNF629	ZBTB7A
chrom	**chromStart**	**chromEnd**	**strand**								
chr1	**629013**	**629269**	**+**	0.31	0.56	0.43	0.47	0.43	0.55	0.47	0.70
	629642	**629898**	**+**	0.16	0.06	0.03	0.13	0.04	0.06	0.07	0.01
	629847	**630103**	**+**	0.89	0.54	0.32	1.20	0.33	0.89	0.72	0.60
	630937	**631193**	**+**	0.30	0.38	0.06	0.15	0.00	0.21	0.34	0.00
	631242	**631498**	**+**	0.33	0.46	0.73	0.48	0.70	0.81	0.21	0.11
...	**...**	**...**	**...**	...	...	...	...	...	...	...	...
chrX	**154441877**	**154442133**	**+**	0.87	1.43	1.00	0.69	0.61	0.46	1.02	1.54
	154584927	**154585183**	**+**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
chrY	**1452882**	**1453138**	**-**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2500959	**2501215**	**-**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	14055973	**14056229**	**+**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 1: promoters epigenomic data

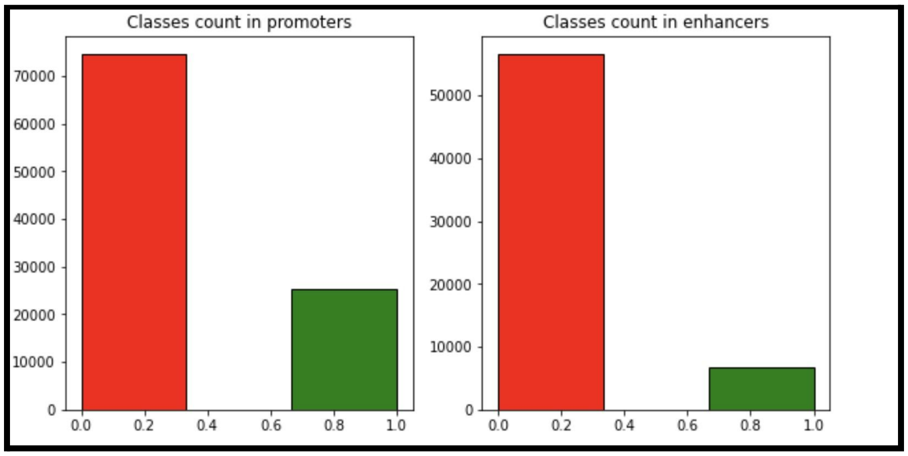


Figure 2: class imbalance

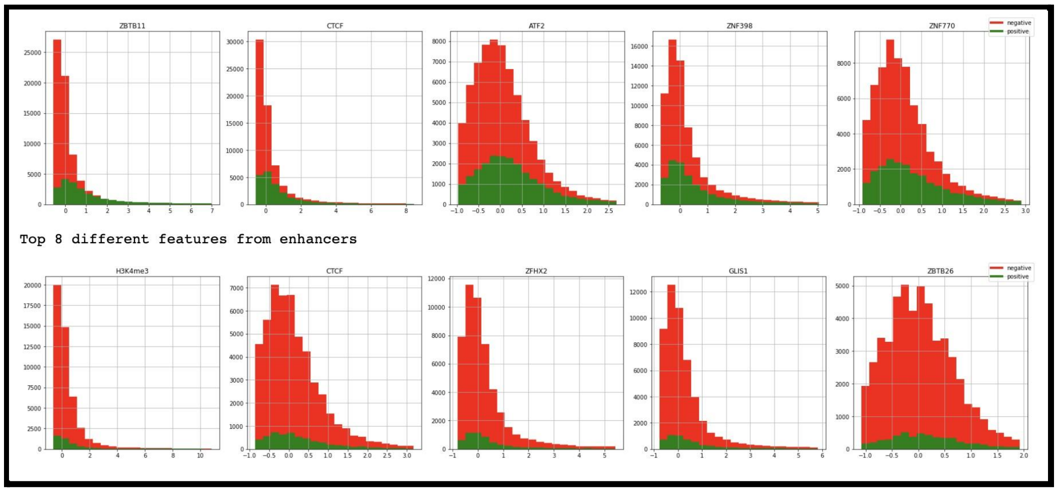


Figure 3: distributions of the top 5 different features in the epigenomic dataset

PROMOTERS				
ZNF549	ZNF302	ZNF223	AEBP2	ZNF677
ZNF34				
ENHANCERS				
ZNF19	ZNF677	ZBTB49	ZNF433	ZNF202
ZNF157	ZNF791	ZNF416	PRDM2	ZNF211
ZNF404	ZNF768	ZNF23	ZSCAN26	ZNF169
ZNF777	ZNF670	BCL6B	ZNF549	ZNF155
ZNF140	ZNF426	ZNF292	ZNF530	ZNF626

table 1: features correlated with the output

Correlation between features

If two features have a high correlation score (a value greater than 0.95), it means that they bring almost the same information and the presence of both in the dataset increases the complexity of the model. So, it is sufficient to keep only one and drop the other. In this case, only the Pearson test has been computed. If two features are highly correlated, the feature that should be removed is the one with lower entropy.
In the epigenomic dataset, there are not extremely correlated features.
In figure 4 and figure 5, it is possible to see the pairplots of the most correlated features for both of the regions.

In the models evaluation, we will find and drop the features highly correlated inside the training set for each holdout split.

Data visualization

Data visualization is the practice of translating information into a visual context, such as a map or graph, to simplify the identification of patterns, trends and outliers in large datasets[17].
For the data visualization, the selected algorithm is t-SNE.
T-distributed stochastic neighbor embedding (t-SNE) is a technique of non-linear dimensionality reduction which lends itself to embedding of high-dimensional datasets in a space of two or three dimensions.
The algorithm models the points so that the nearby objects in the original space turn out to be close in the dimensionality-reduced space and distant points in the original space result to be far-away in the dimensionality-reduced space, trying to preserve the local structure.
The figure 6 and 7 show t-SNE, performed on the datasets.

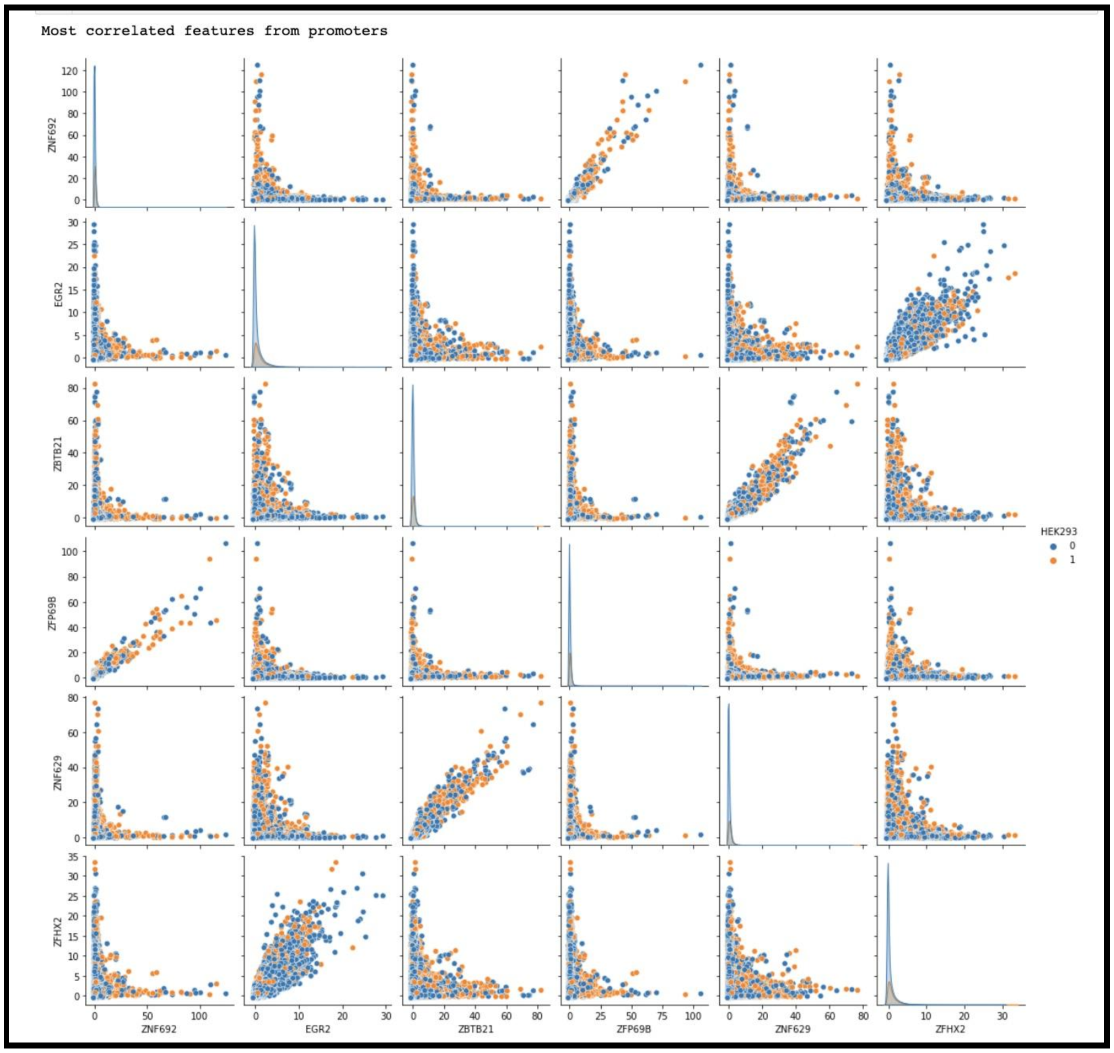


Figure 4: most correlated feature for promoters

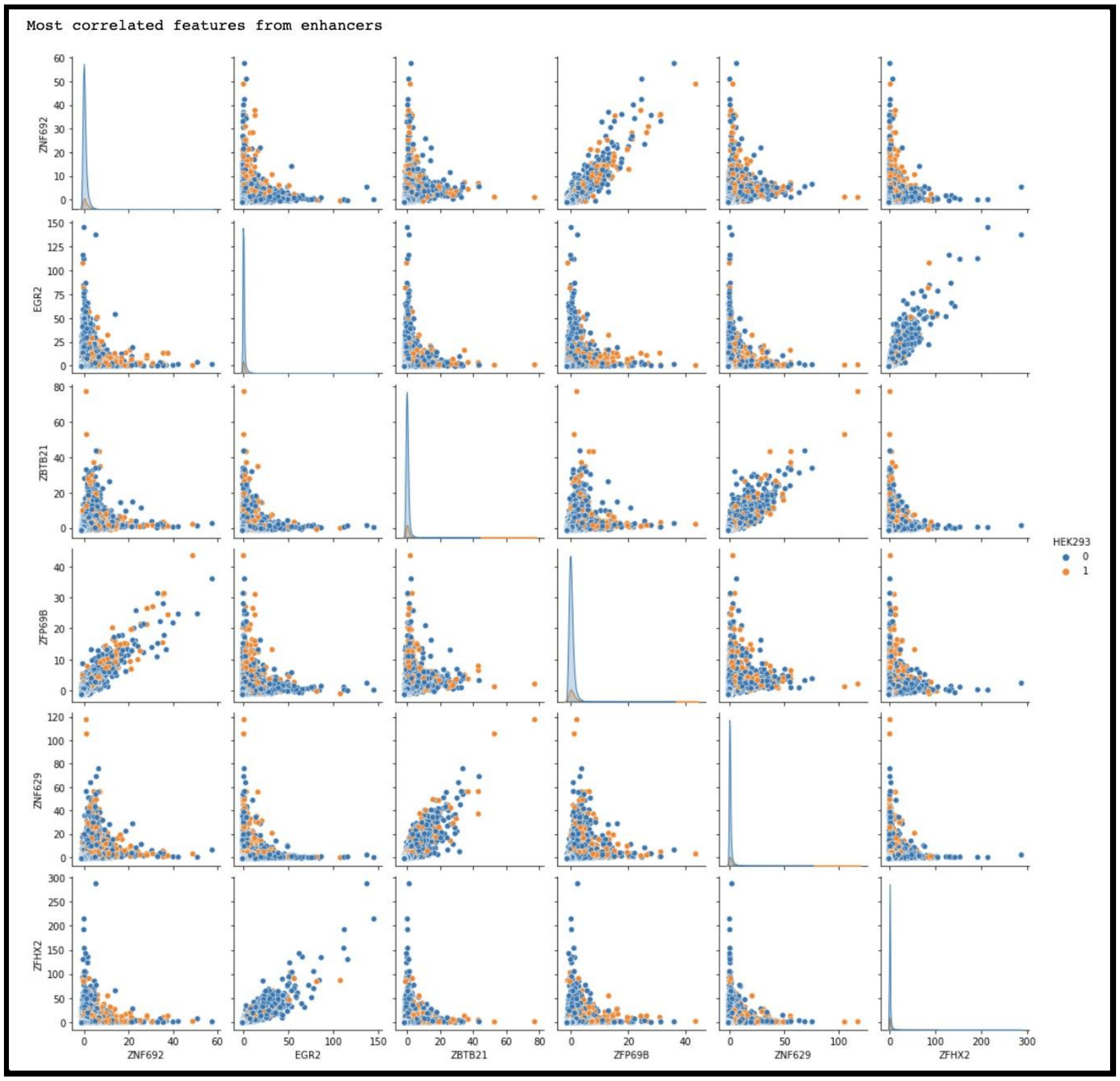


Figure 5: most correlated features for enhancers

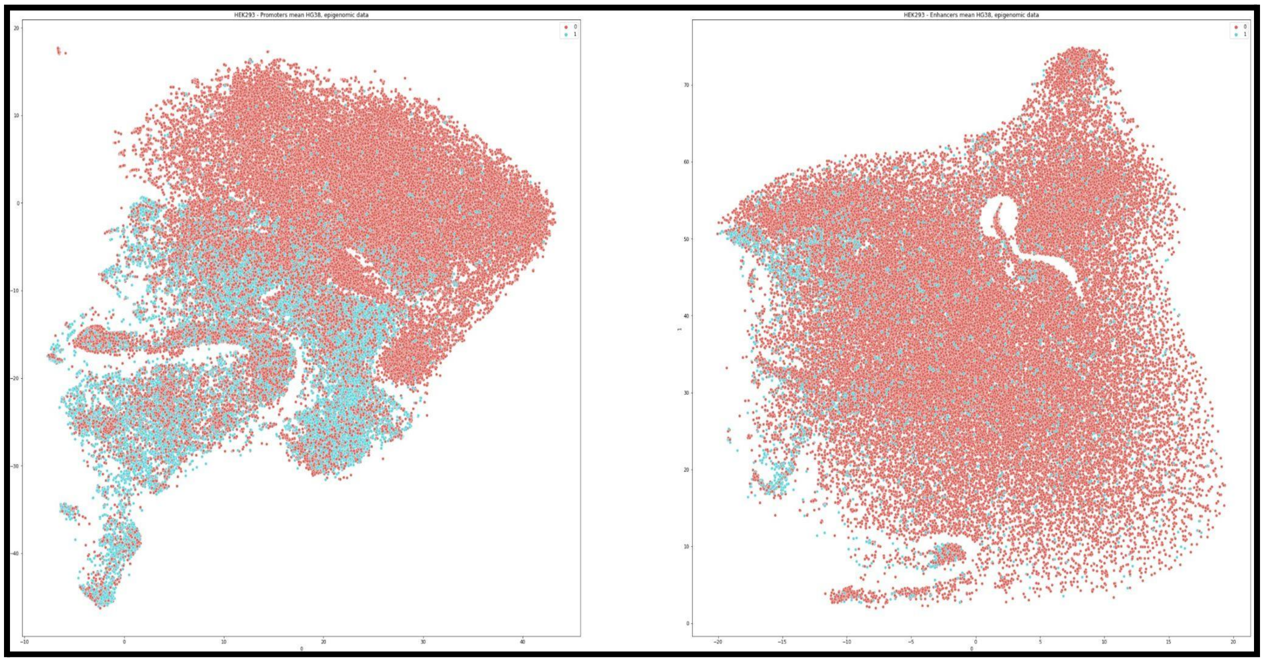


Figure 6 : t-SNE of epigenomic data : promoters (at left) and enhancers (at right)

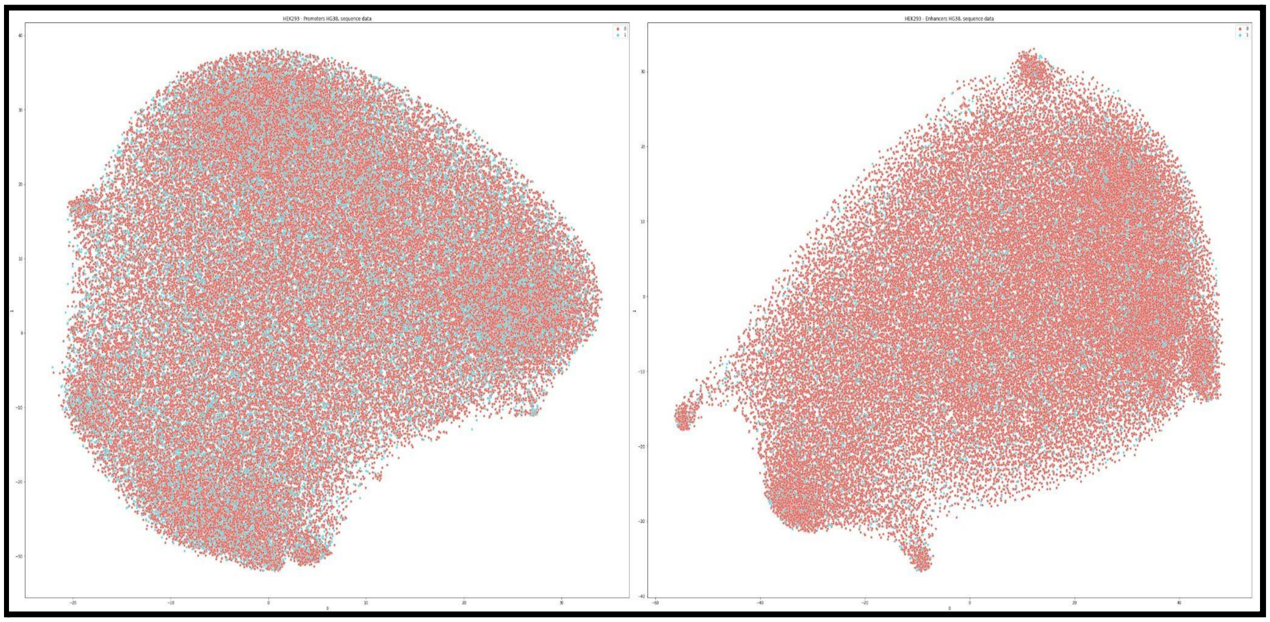


Figure 7: t-SNE of sequence data: promoters (at left) and enhancers (at right)

Holdouts

The hold-out method for model evaluation represents the mechanism of splitting the dataset into training and test datasets. The model is trained on the training set and then tested on the testing set to get the most optimal performance.
In the project, I used a variation of hold-out, the stratified holdout, where each class is represented by the same percentage of samples in the training set and in the test set. Using stratified holdouts, we can have a better average performance with a reduced number of iterations, compared to the simple holdout. This procedure can be done multiple times in order to obtain more reliable results.
To perform the stratified holdouts I adopted the method StratifiedShuffleSplit[18] from the machine learning library Scikit-learn.
It creates a generator of indexes that we can apply to the dataset in order to split it into the training and test sets. The selected percentage of the test set is 30% and the chosen number of iteration is 2 because performing many iterations (i.e. 10) could be very computationally heavy.

Models

In this project, I used the classification task to distinguish the active and inactive promoters and enhancers inside the given cell line HEK293. The neural networks that have been used are: feed forward neural network (FFNN), convolutional neural network(CNN) and multimodal neural network (MMNN).

FFNN

A Feed Forward Neural Network is an artificial neural network in which the connections between nodes does not form a cycle. The feed forward model is the simplest form of neural network as information is only processed in one direction and never backwards [19].
In particular, I used Multilayer perceptron (MLP),a kind of FFNN, made up of multiple perceptrons. In MLP, there are at least three layers: an input layer, one or more hidden layers and an output layer. The neurons of hidden layers use non linear functions as activation function in order to separate non linear separable data.
I applied the feed forward neural network to the epigenomic dataset.

The structure of the FFNN consists of the following layers:

- Input layer[20]: through this layer, we provide the model with the input data. Input shape corresponds to the shape of input data, that in our case is the number of features of the epigenomic dataset
- Dense layer[21]: it is the regular deeply connected neural network layer, made of the indicated number of units, on which it is applied the activation function.
- Dropout layer[22]: it applies the dropout regularization, in order to avoid overfitting. It sets input units to 0 with a frequency of rate at each step during training time.
- Batch Normalization layer[23]: it applies a transformation on the inputs that maintains the mean output close to 0 and the output standard deviation close to 1. During training the layer normalizes

its output using the mean and standard deviation of the current batch of inputs; instead, during evaluation the layer normalizes its output using a moving average of the mean and standard deviation of the batches it has seen during training. The aim of batch normalization is to make the training faster and more stable
- ReLU layer[24]: it performs the Relu activation function, returning element-wise max(x, 0).

I put the BatchNormalization layer before Relu to keep it from performing normalization also on the zeros; it may cause misbehaviour .

CNN

Convolutional neural networks are deep neural networks characterized by the usage of the convolution operation and pooling operation, applied to the corresponding layers . In the convolutional layer, the element involved in carrying out the convolution operation is called Kernel/Filter, K, which moves to the right with a certain stride value till it parses the complete width of the input space. Every time, it is performed a matrix multiplication operation between K and the portion P of the input over which the kernel is hovering. The results are called future maps, and so there is a future map for each executed computation. The Pooling layer is responsible for reducing the spatial size of the future maps. There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value over a spatial window of the feature maps; while, Average Pooling returns the average. The convolution and pooling can be executed multiple times inside a CNN. Subsequently, the future maps are flattened into a 1-dimensional array, which will be the input of a Fully Connected Neural Network that drives the final classification decision.
The convolutional neural network lends itself to be adopted to genome sequence because it is able to handle fixed-size sequence data and to capture locality, in which FFNN fails.

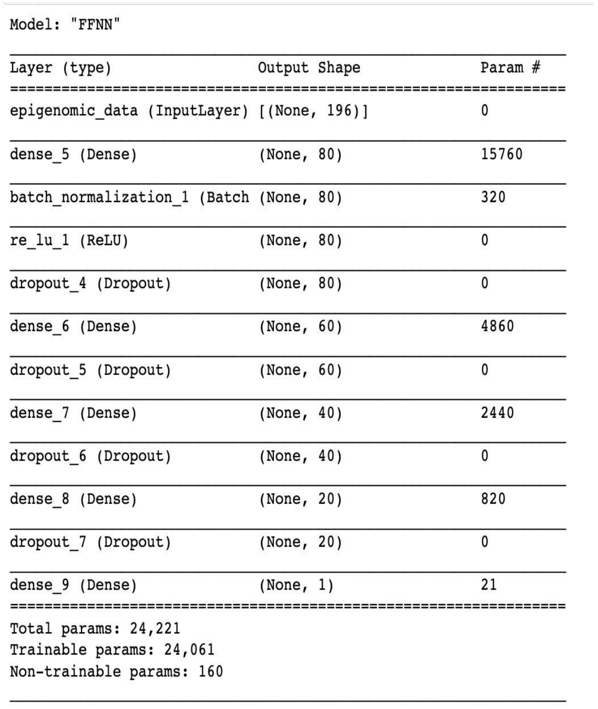
The CNN presents the layers, described below:
- Input layer: it is described above
- CONV1D[25]: it applies convolution between the input patch and the kernel with the selected size.
- MaxPooling1D[26]: it applies the max-pooling operation.
- Flatten layer[27]: it flattens the input
- Dropout layer
- BatchNormalization layer
- Dense layer: it is one of the layers that compose the fully connected network

MMNN

Multimodal neural networks (MMNN) combine two Deep Neural Networks, that, in our case, are FFNN and CNN.
The MMNN uses both epigenomic data , inputs of the FFNN, and sequence data, inputs of the CNN, trying to produce better results using information coming from both sources.
The chosen approach for the MMNN is to start the training from the already pre-trained last layers and input layers of the other two networks.

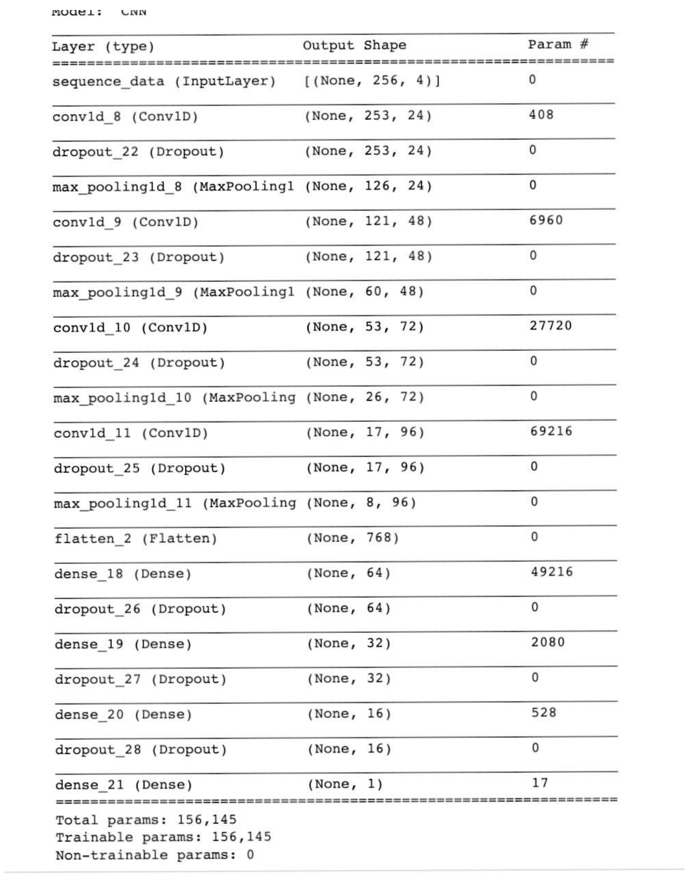
In order to tie together the networks, it is performed the concatenation of their last hidden layers. On top of the concatenation layer, as we can see in figure 13, more other layers are added.

In all the implemented networks, I used, as loss-function, *binary-cross entropy* because we are executing a binary classification task and the optimizer *nadam*. Regarding the training,I selected the Earlystopping callback[29], that stops the training when a monitored metric has stopped improving. In this specific case, we stop training when, after 2 epochs, the loss is no longer decreasing by a minimum value of 0.001.



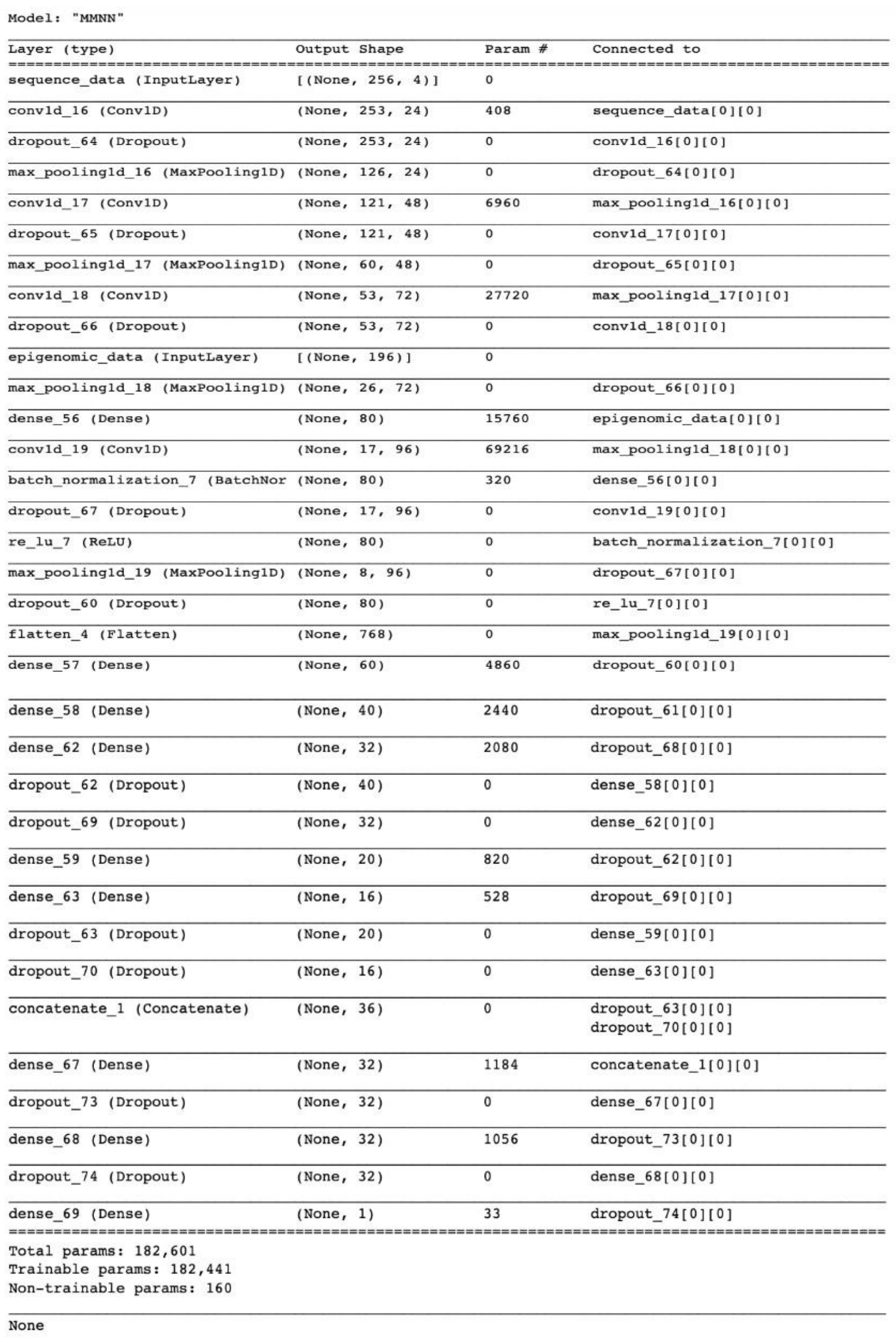
```
Model: "FFNN"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
epigenomic_data (InputLayer) [(None, 196)]             0
_________________________________________________________________
dense_5 (Dense)              (None, 80)                15760
_________________________________________________________________
batch_normalization_1 (Batch (None, 80)                320
_________________________________________________________________
re_lu_1 (ReLU)               (None, 80)                0
_________________________________________________________________
dropout_4 (Dropout)          (None, 80)                0
_________________________________________________________________
dense_6 (Dense)              (None, 60)                4860
_________________________________________________________________
dropout_5 (Dropout)          (None, 60)                0
_________________________________________________________________
dense_7 (Dense)              (None, 40)                2440
_________________________________________________________________
dropout_6 (Dropout)          (None, 40)                0
_________________________________________________________________
dense_8 (Dense)              (None, 20)                820
_________________________________________________________________
dropout_7 (Dropout)          (None, 20)                0
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 21
=================================================================
Total params: 24,221
Trainable params: 24,061
Non-trainable params: 160
_________________________________________________________________
```

Figure 8 : summary of the FFNN



```
Model: CNN
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequence_data (InputLayer)   [(None, 256, 4)]          0
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 253, 24)           408
_________________________________________________________________
dropout_22 (Dropout)         (None, 253, 24)           0
_________________________________________________________________
max_pooling1d_8 (MaxPooling1 (None, 126, 24)           0
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 121, 48)           6960
_________________________________________________________________
dropout_23 (Dropout)         (None, 121, 48)           0
_________________________________________________________________
max_pooling1d_9 (MaxPooling1 (None, 60, 48)            0
_________________________________________________________________
conv1d_10 (Conv1D)           (None, 53, 72)            27720
_________________________________________________________________
dropout_24 (Dropout)         (None, 53, 72)            0
_________________________________________________________________
max_pooling1d_10 (MaxPooling (None, 26, 72)            0
_________________________________________________________________
conv1d_11 (Conv1D)           (None, 17, 96)            69216
_________________________________________________________________
dropout_25 (Dropout)         (None, 17, 96)            0
_________________________________________________________________
max_pooling1d_11 (MaxPooling (None, 8, 96)             0
_________________________________________________________________
flatten_2 (Flatten)          (None, 768)               0
_________________________________________________________________
dense_18 (Dense)             (None, 64)                49216
_________________________________________________________________
dropout_26 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_19 (Dense)             (None, 32)                2080
_________________________________________________________________
dropout_27 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_20 (Dense)             (None, 16)                528
_________________________________________________________________
dropout_28 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_21 (Dense)             (None, 1)                 17
=================================================================
Total params: 156,145
Trainable params: 156,145
Non-trainable params: 0
```

Figure 9 : summary of the CNN



```
Model: "MMNN"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
sequence_data (InputLayer)      [(None, 256, 4)]     0
__________________________________________________________________________________________________
conv1d_16 (Conv1D)              (None, 253, 24)      408         sequence_data[0][0]
__________________________________________________________________________________________________
dropout_64 (Dropout)            (None, 253, 24)      0           conv1d_16[0][0]
__________________________________________________________________________________________________
max_pooling1d_16 (MaxPooling1D) (None, 126, 24)      0           dropout_64[0][0]
__________________________________________________________________________________________________
conv1d_17 (Conv1D)              (None, 121, 48)      6960        max_pooling1d_16[0][0]
__________________________________________________________________________________________________
dropout_65 (Dropout)            (None, 121, 48)      0           conv1d_17[0][0]
__________________________________________________________________________________________________
max_pooling1d_17 (MaxPooling1D) (None, 60, 48)       0           dropout_65[0][0]
__________________________________________________________________________________________________
conv1d_18 (Conv1D)              (None, 53, 72)       27720       max_pooling1d_17[0][0]
__________________________________________________________________________________________________
dropout_66 (Dropout)            (None, 53, 72)       0           conv1d_18[0][0]
__________________________________________________________________________________________________
epigenomic_data (InputLayer)    [(None, 196)]        0
__________________________________________________________________________________________________
max_pooling1d_18 (MaxPooling1D) (None, 26, 72)       0           dropout_66[0][0]
__________________________________________________________________________________________________
dense_56 (Dense)                (None, 80)           15760       epigenomic_data[0][0]
__________________________________________________________________________________________________
conv1d_19 (Conv1D)              (None, 17, 96)       69216       max_pooling1d_18[0][0]
__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 80)           320         dense_56[0][0]
__________________________________________________________________________________________________
dropout_67 (Dropout)            (None, 17, 96)       0           conv1d_19[0][0]
__________________________________________________________________________________________________
re_lu_7 (ReLU)                  (None, 80)           0           batch_normalization_7[0][0]
__________________________________________________________________________________________________
max_pooling1d_19 (MaxPooling1D) (None, 8, 96)        0           dropout_67[0][0]
__________________________________________________________________________________________________
dropout_60 (Dropout)            (None, 80)           0           re_lu_7[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 768)          0           max_pooling1d_19[0][0]
__________________________________________________________________________________________________
dense_57 (Dense)                (None, 60)           4860        dropout_60[0][0]

__________________________________________________________________________________________________
dense_58 (Dense)                (None, 40)           2440        dropout_61[0][0]
__________________________________________________________________________________________________
dense_62 (Dense)                (None, 32)           2080        dropout_68[0][0]
__________________________________________________________________________________________________
dropout_62 (Dropout)            (None, 40)           0           dense_58[0][0]
__________________________________________________________________________________________________
dropout_69 (Dropout)            (None, 32)           0           dense_62[0][0]
__________________________________________________________________________________________________
dense_59 (Dense)                (None, 20)           820         dropout_62[0][0]
__________________________________________________________________________________________________
dense_63 (Dense)                (None, 16)           528         dropout_69[0][0]
__________________________________________________________________________________________________
dropout_63 (Dropout)            (None, 20)           0           dense_59[0][0]
__________________________________________________________________________________________________
dropout_70 (Dropout)            (None, 16)           0           dense_63[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 36)           0           dropout_63[0][0]
                                                                 dropout_70[0][0]
__________________________________________________________________________________________________
dense_67 (Dense)                (None, 32)           1184        concatenate_1[0][0]
__________________________________________________________________________________________________
dropout_73 (Dropout)            (None, 32)           0           dense_67[0][0]
__________________________________________________________________________________________________
dense_68 (Dense)                (None, 32)           1056        dropout_73[0][0]
__________________________________________________________________________________________________
dropout_74 (Dropout)            (None, 32)           0           dense_68[0][0]
__________________________________________________________________________________________________
dense_69 (Dense)                (None, 1)            33          dropout_74[0][0]
==================================================================================================
Total params: 182,601
Trainable params: 182,441
Non-trainable params: 160
__________________________________________________________________________________________________
None
```

Figure 10: summary of the MMNN

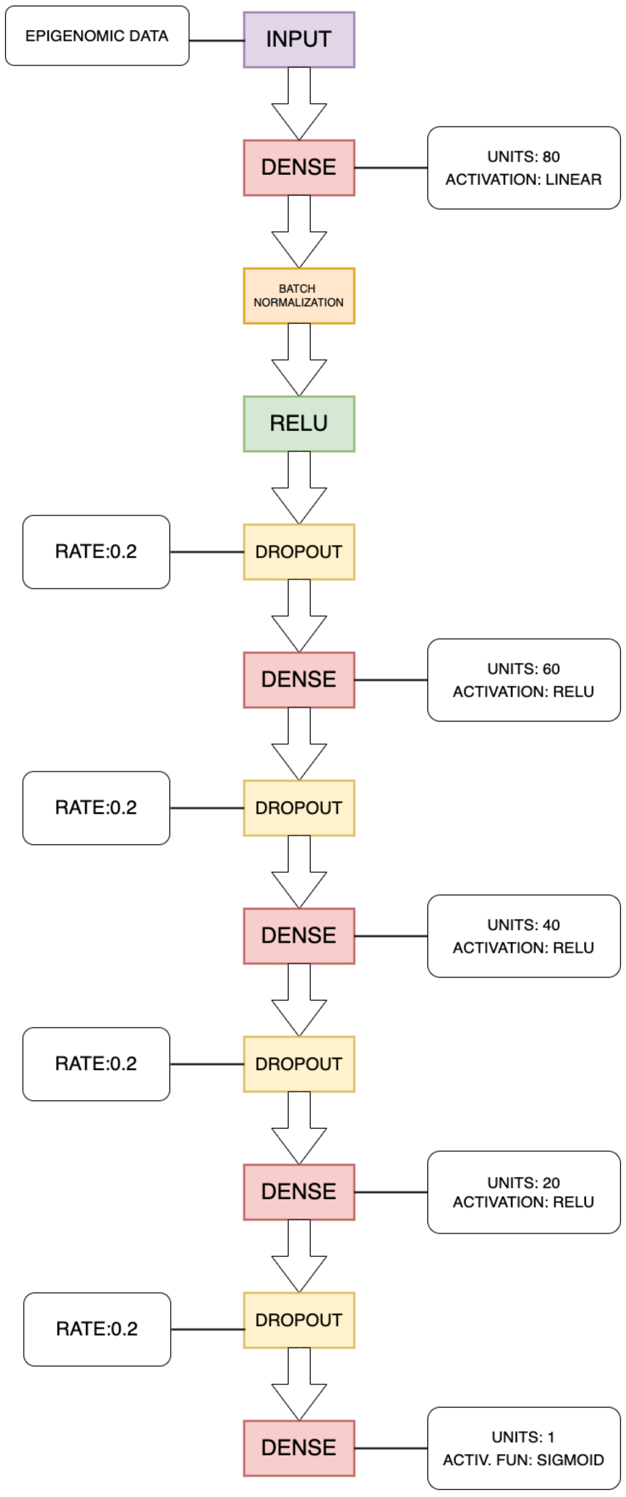


Figure 11: Feed Forward Neural Network structure

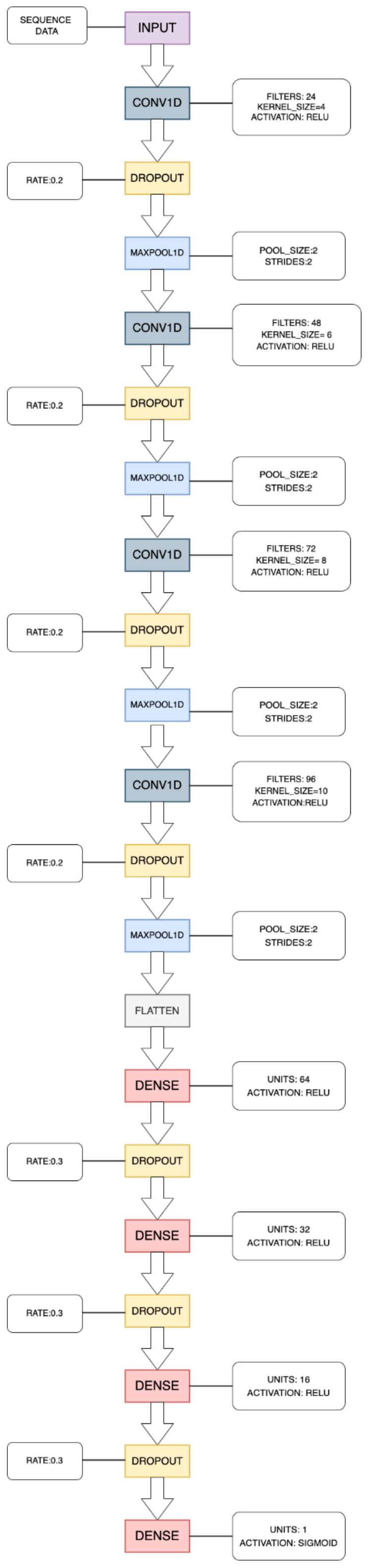


Figure 12 : Convolutional Neural Network structure

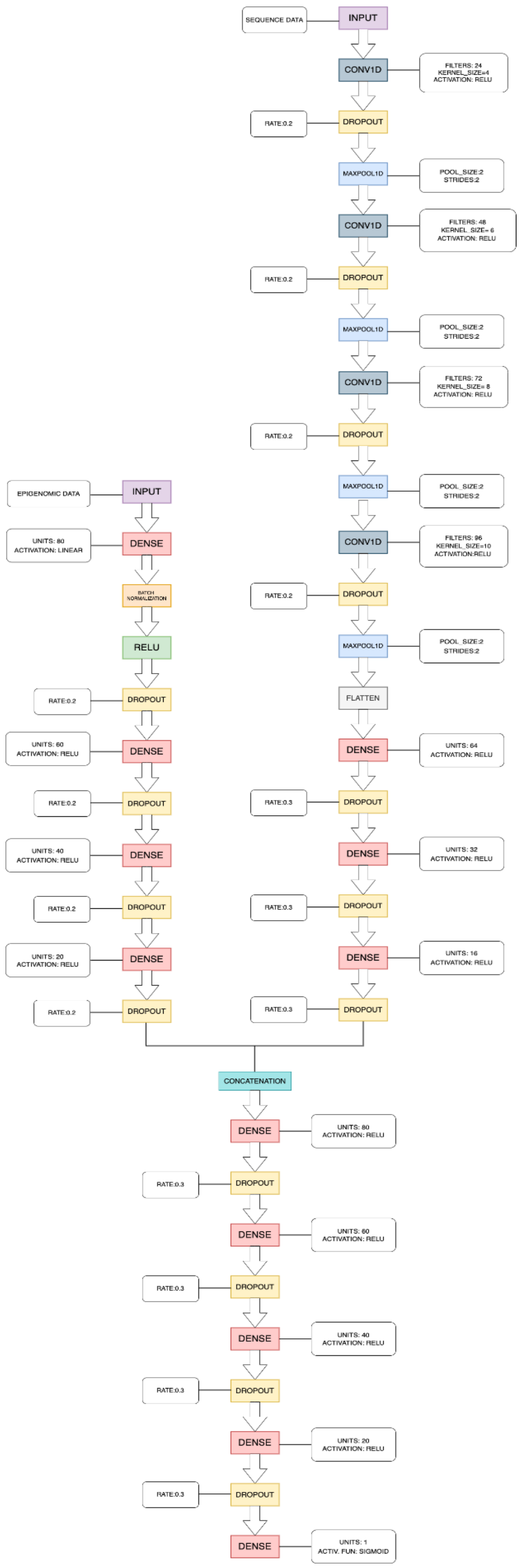


Figure 13 : Multimodal Neural Network structure

Results

Wilcoxon test

To compare the models performance, the Wilcoxon test has been adopted.
Wilcoxon test aims to verify if there exist statistical differences in performance of the models. If the resulting p value is greater than a given threshold (in our, case,0.05), it means there is no evidence of a statistically significant difference between the performances of the considered models; on the contrary, if the p_value is lower than the threshold, we can assert that one model performs better than the other. To select the best one, we have to compare the means of the values for each metric obtained during the evaluation. For the Wilcoxon test,I took into consideration only the most important metrics: AUROC,AUPRC and accuracy.

Comparison between aggregation metrics in the epigenomic dataset

To examine how the adopted aggregation metric in the epigenomic data influences the model's performance, I applied the FFNN to epigenomic dataset, considering both the metrics. From the barplots in figure 14 , it results that the model's performance is the same, if we take the max instead of the mean as aggregation metric.
Also, as you can see from the following table (table 2), the Wilcoxon test doesn't individuate a statistically significant difference between the two models.

TASK	METRIC	P VALUE
AE_vs_IE	AUROC	0.65
AE_vs_IE	AUPRC	0.179
AE_vs_IE	accuracy	0.65
AP_vs_IP	AUROC	0.179
AP_vs_IP	AUPRC	0.179
AP_vs_IP	accuracy	0.65

Table 2: Wilcoxon test

General models performance

As we can see in figure 15, considering the classification of the enhancers, the FFN and the MMNN strongly overfit.
In figure 17 which shows the training histories, it is possible to observe, for the two models mentioned above, how the train and test values of the loss change with the number of epochs: while the test curve rises with each epoch, the train curve decreases rapidly. It is a clear sign of overfitting because the model's performance improves (the train loss becomes lower and lower) on the training data, at each epoch, and gets worse with the test data (the test loss grows very quickly); therefore is not able to generalize.
Also, if we take into consideration AUROC and AUPRC, the gap between the test and train curves in the last epochs is significant; this is another indicator of overfitting.
Regarding the classification of the promoters, we can find a strong overfitting with CNN; indeed, if we observe the plots (figure 16) relative to its training history, we

can notice a similar behaviour with respect to the plots analyzed above .
From the accuracy standpoint, we have on average better results with "active_enhancers_vs_inactive_enhancers" task".
If we pay attention to the AUPRC, the classification of the enhancers overcomes the other task; this happens because, as we can see in figure 2, there is a lower balance between the two classes.

The wilcoxon test, performed on the models, shows the absence of a statistical difference between each pair of models (table 3).

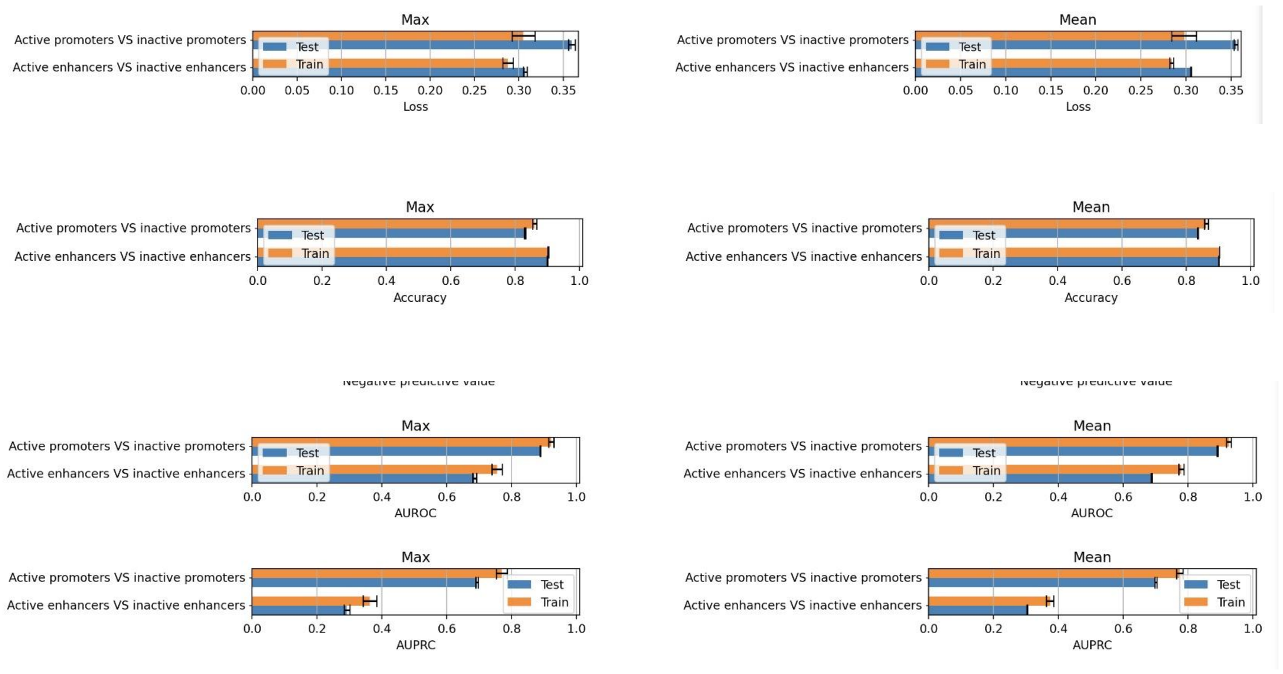


Figure 14 : barplots of loss, accuracy, AUROC and AUPRC with which we can compare the models performance using max or mean as aggregation metric for the epigenomic dataset

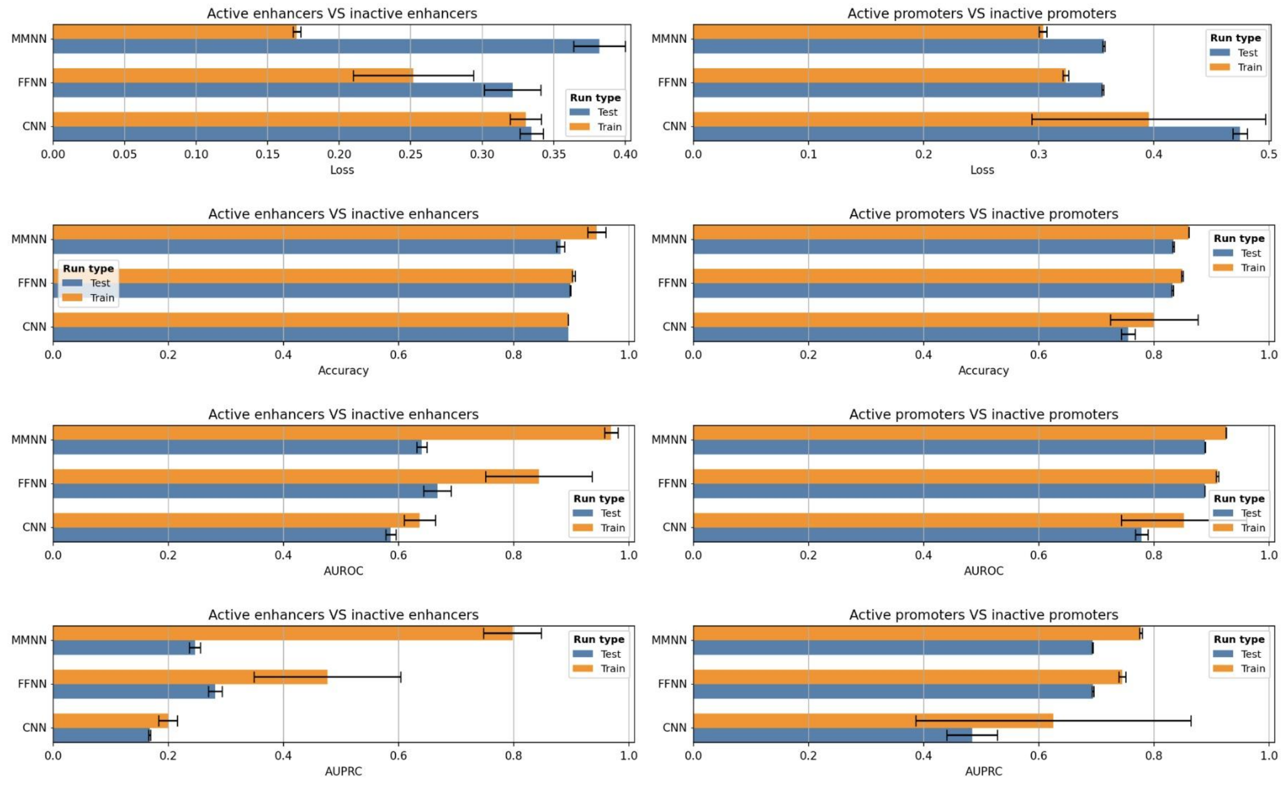
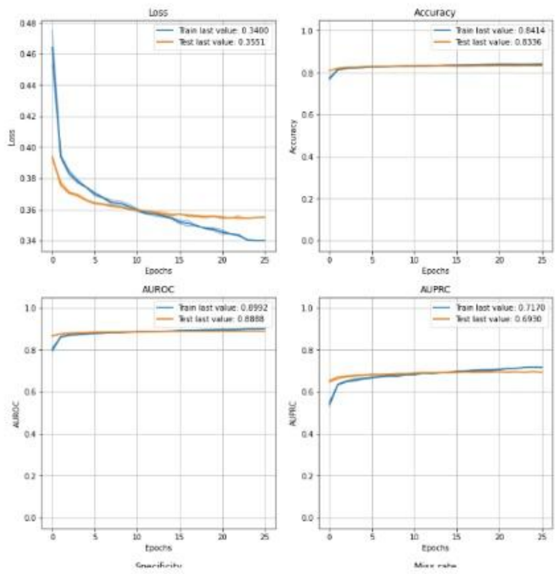


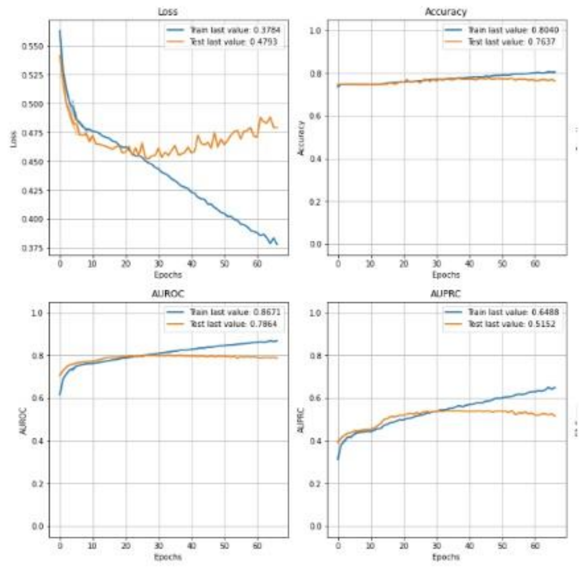
Figure 15: barplots representing the train and test performances grouped by task and model

TASK: ACTIVE_PROMOTERS_VS_INACTIVE_PROMOTERS

FFNN



CNN



MMNN

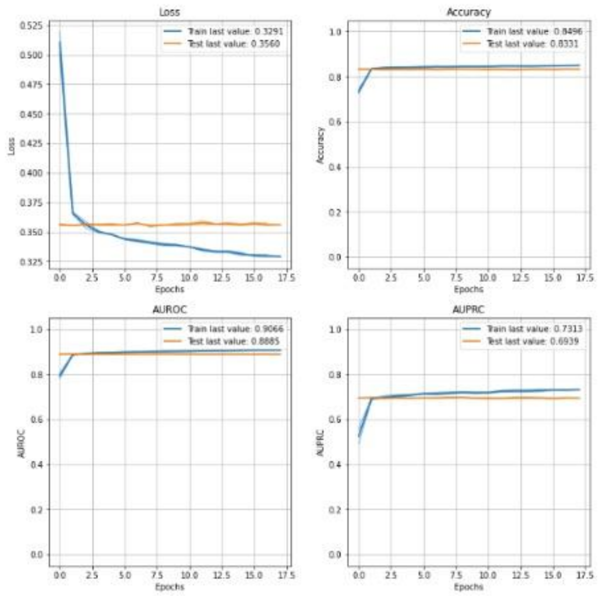
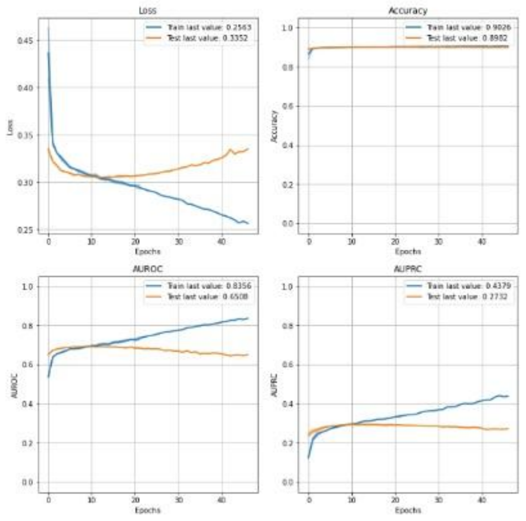


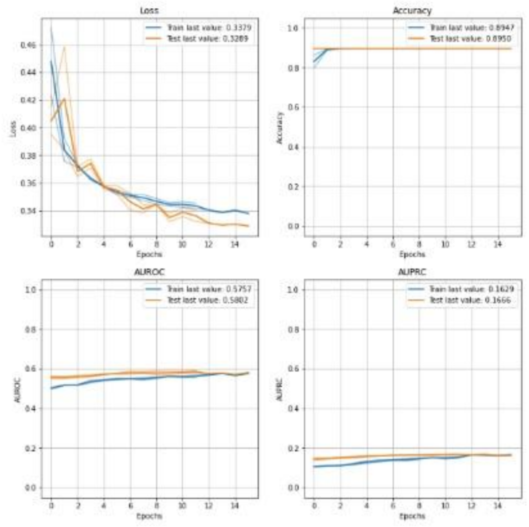
Figure 16:training history of “active_promoters_vs_inactive_promoters” task relative to loss, accuracy, AUROC and AUPRC

TASK: ACTIVE_ENHANCERS_VS_INACTIVE_ENHANCERS

FFNN



CNN



MMNN

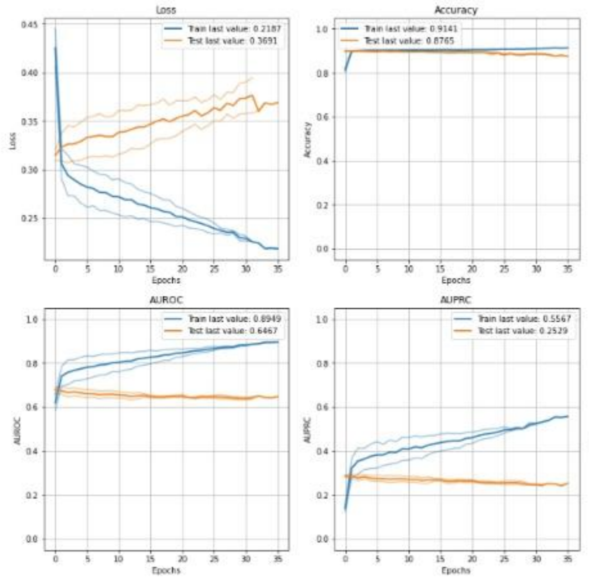


Figure 17 :training history of “active_enhancers_vs_inactive_enhancers task”, relative to loss, accuracy, AUROC and AUPRC

TASK	MODEL A	MODEL B	METRICS
AE_vs_IE	FFNN	CNN	ALL : 0.179
AE_vs_IE	FFNN	MMNN	ALL : 0.179
AE_vs_IE	CNN	MMNN	ALL : 0.179
AP_vs_IP	FFNN	CNN	ALL : 0.179
AP_vs_IP	FFNN	MMNN	AUROC:0.65 AUPRC,ACC:0.179
AP_vs_IP	CNN	MMNN	ALL : 0.179

Table 3: Wilcoxon test between the built models considering accuray,AUROC,AUPRC

References

[1]https://academic.oup.com/bib/article/7/1/86/264025?login=true

[2]https://www.sciencedirect.com/science/article/pii/S0303264715001604?casa_token=a_IFw_pI1QUAAAAA:9EIVHMCLnbo23DZsKxgwwV4ry2UOZtclBrxGePgnPtmswaaQunbQQI0i1QIu9kOMLxyWgSU

[3]https://www.addgene.org/mol-bio-reference/promoters/#:~:text=A%20promoter%20is%20a%20region,translated%20into%20a%20functional%20protein.

[4]https://en.wikipedia.org/wiki/HEK_293_cells

[5]https://www.sciencedirect.com/topics/neuroscience/immortalised-cell-line

[6]https://en.wikipedia.org/wiki/ENCODE

[7]https://en.wikipedia.org/wiki/UCSC_Genome_Browser

[8]https://www.nature.com/collections/jcxddjndxy/

[9]https://genome.ucsc.edu/cgi-bin/hgGateway

[10]https://github.com/AnacletoLAB/epigenomic_dataset

[11]https://github.com/LucaCappelletti94/ucsc_genomes_downloader

[12]https://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/

[13]https://pypi.org/project/keras-bed-sequence/

[14]https://developers.google.com/machine-learning/data-prep/transform/normalization

[15]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

[16]https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/how-to/correlation/methods-and-formulas/#:~:text=The%20p%2Dvalue%20for%20Pearson's,n%20%E2%80%93%202%20degrees%20of%20freedom.

[17]https://www.techtarget.com/searchbusinessanalytics/definition/data-visualization

[6]https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

[18]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

[19]https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

[20]https://keras.io/api/layers/core_layers/input/

[21]https://keras.io/api/layers/core_layers/dense/

[22]https://keras.io/api/layers/regularization_layers/dropout/

[23]https://keras.io/api/layers/normalization_layers/batch_normalization/

[24]https://keras.io/api/layers/activation_layers/relu/

[25]https://keras.io/api/layers/convolution_layers/convolution1d/

[26]https://keras.io/api/layers/pooling_layers/max_pooling1d/

[27]https://keras.io/api/layers/reshaping_layers/flatten/

[28]https://theaisummer.com/regularization/

[29]https://keras.io/api/callbacks/early_stopping/