Active Regulatory Regions Prediction using Deep Learning Methods

Alice Schiavone

Project developed in May 2022 for the "Bioinformatics" course
at Università degli Studi Statale di Milano, held by Professor Giorgio Valentini

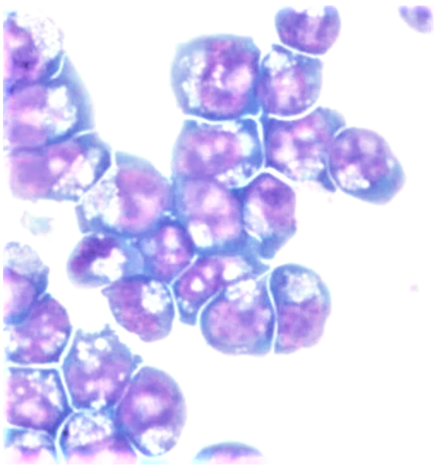

Figure 1: K562 cells. [13]

Abstract

To solve the problem of classification of active regulatory regions, we try to predict whether the enhancers and promoters of the cell line K562 are active or inactive. This binary classification problem has been approached in different ways: we explore the possibility of doing it by the implementation of deep learning techniques. First, a FFNN will be trained on the epigenomic data of K652, with levels of activation from FANTOM5. Secondly, a CNN is trained on the reference genomic sequence *hg38*. To fully grasp the complexity of the task, we concatenate the two networks into a MMNN, which uses both type of data to solve the task. Results are close to those of S.O.A., even if the training has been done on limited resources, and only on one cell line. This implies that more tests are needed to determine a statistical relevance.

Keywords: Active Regulatory Regions, Deep Learning, Supervised Learning

1 Introduction

Different pairs of nucleotides, connected sequentially in what we call DNA, are the fundamental letters of the words that form the instructions for ribosomes, macromolecules responsible for the production of proteins, one of the basic ingredients for life on Earth. Without proteins, organisms would have a hard time with catalysing metabolic reactions, having a strong structure for their cells, transporting molecules and even replicating DNA itself.

However, not every bit of DNA encodes a protein. If we focus our attention of the human DNA, which was recently entirely sequenced (determining base pairs and identifying genes), we can find 3.1 billions bases [11], of which only 22,300 genes encode proteins. [10]. In fact, a gene (a sequence of nucleotides) can either encode a protein, RNA or nothing at all. It is estimated that about 98% of the human DNA is non-coding DNA. These genes serve regulatory purposes, and are usually grouped in what we call regions.

Enhancers and promoters are usually small sequences of base pairs (50-1500 bases long the first, 100-1000 the second) that regulate the transcription of genes, usually localized close to them. That is why they are two of the regions that we call Cis-Regulatory Elements (CREs). When different types of RNA polymerase attach to a portion of DNA which includes an enhancer or a promoter, they are guided to the gene they need to express. While the function of a promoter is that of indicating where transcription should begin, an enhancer amplifies this initiations by interacting with the promoter.

If we know which promoter or enhancer is active, we should be able to tell which gene will be expressed. The task of this paper is to find a deep learning model model that yields statistically reliable results to the prediction of active regulatory regions. In particular, we focus on a single cell line, which is a population of cells from an organism which are immortal due to mutations which allow them to proliferate indefinitely, without degrading of the coding information.

Specifically, our cell line is that of K562 [Img 1], which comes from a 53-year-old woman with chronic myelogenous leukemia (a white blood cell cancer) in terminal blast crises. [7].

1.1 Previous work

Various approaches to the task of prediction of active regulatory regions have been studied. Among those which use similar models to the ones implemented by us, we have the CNN model by Min et al. [9], that achieved a AUROC of .874 and an AUPRC of .875 on multiple cell lines, including K562, with a p-value of $1.9 * 10^{-3}$ on a NVIDIA Tesla K80 CPU in less than 2h.

Also, Beer [3] has used a Gapped-Kmer Support Vector Machine to reach a AUROC score of .83 on K562, which shows that going deep is not strictly necessary. Li et al. [6] reached a auPRCs of 0.8565 on the three tasks on K652 (A-E, A-P, and BG) using the top ten features found by randomized deep feature selection. Their DECRES model reported a auPRC score beyond 0.89 on 6 cell lines (including K652), however, their sampling method balances the data by extracting 3000 samples from each class, which obviously solves the problem of unbalancing and improves classifiers performance accordingly.

2 Tasks

To predict the activation of a regulatory regions, we distinguish between the two tasks that we want to work on. By doing so, we can subdivide our goal in two problems:

- Active enhancers vs inactive enhancers (AEvsIE)
- Active promoters vs inactive promoters (APvsIP)

All the pre-processing and model training has been done exactly in the same way for both tasks.

2.1 Data

To expand the available data on Cis-Regulatory Regions, we match the features of K562 from ENCODE [1] to the active enhancers and promoters from FANTOM5 [2]. An epigenome is a record of changes which happens to the DNA of organisms in time. We can collect readings of different labs on the same cell line to gather more data about it. ENCODE offers a selection of experiment summaries, mostly ChIP-seq on different cell lines. Using the CAGE approach to identify the activity of transcription sites, FANTOM5 reports the transcription events on human and mouse genome on different tissues, which are "identified and annotated to provide precise location of known promoters as well as novel ones, and to quantify their activities". The unit of read activation is that of TPM (tag per million). We will feed this data to a Feed Forward Neural Network in a binary classification task to predict where that region is active or not.

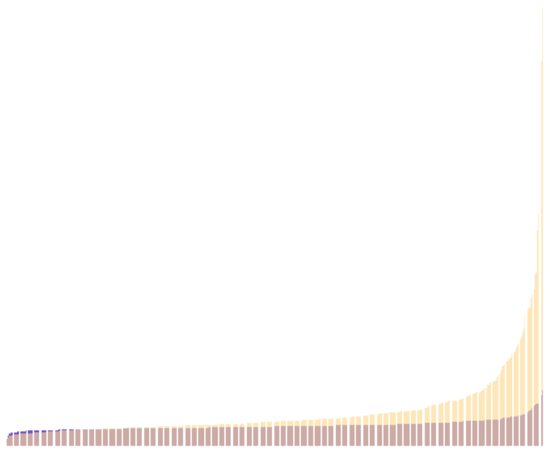
Figure 2: Distribution of values of the sorted averaged data. Enhancers in purple, promoters in yellow.

We will also use the *hg38* human genome assembly from the UCSC Genome Browser [8], bedded with respect to the FANTOM5 activation TPMs, for reference with respect to the K652 cell line. This data will be fed to a Convolutional Neural Network.

2.1.1 Data Preprocessing

	Enhancers	Promoters
# Samples	63285	99881
# Features	429	429
# NaN	102	496
Avg #NaN/sample	.002	.005
Avg #NaN/feature	.238	1.16

Table 1: Information about the datasets.

There could be many problems with this type of task, mostly related to the data. As we can see from Table 1, we have some missing values (NaN) spread throughout the features, with the promoters having an average of more than 1 NaN per feature. This can be solved with **data imputing**. We used *sklearn.impute.KNNImputer* to compute a value to replace the missing ones based on the 5 neighboring values.

Another problem comes with the distribution of the values. We can see [Figure 2] that while enhancers TPMs tend to be uniform throughout all features, the same doesn't happen for promoters, which follow a steep exponential trend. This can be solved by **scaling** the values with robust Z-scoring, which instead of subtracting the mean and dividing by the standard deviation (which are heavily influenced by outliers), substructs the median Q_2 and divides by the standard deviation between the interquantile range $IQR = Q_3 - Q_1$.

For some features it could happen that they display **constant values**. We look for these feature to drop them, because they don't add any relevant information to the task, but we don't find any.

If some feature displays some **correlation with the output** labels, it means that that feature is useful for learning. That is why we look for features that show some type of relationship with the output, so we can drop them from the dataset and reduce the dimensionality of the feature space. Because many types of relations exist, we look for 3 different types: linear correlation with Pearson's Coefficient, monotonic correlation with Spearman's Coefficient and for other interesting relationships using MINE (Maximal Information-based Nonparametric Exploration) [12]. We drop any feature that shows a correlation with a p-value more than a p-value threshold of 0.05.

A similar reasoning holds for **correlation between features**: if a feature is highly correlated with another, it means that we have a repetition of information within the dataset. We drop any feature that shows a correlation of more than 99%: we do not find any such feature.

Even after dropping some features, we still have a highly dimensional feature space. We want to perform **feature selection** in order to select them based on their importance. We do so using a Decision Tree classifier (See section 2.2.1).

	Inactive	Active
Enhancers	57761	5524
%	.91	.09
Promoters	66346	33535
%	.66	.34

Table 2: Label distribution with an activation threshold of 0.0 both for enhancers and promoters.

Choosing 0.0 as a threshold for the activation of both promoters and enhancers, as we can see from Table 2, our data is mostly unbalanced. We have a ratio of .505 Positive examples to Negatives for promoters, and .096 for enhancers. This of course will skew the model willingness to classify each instance towards the most numerous class. We do not perform any kind of sampling to avoid loosing information, but we will keep in mind about this fact while evaluating the results.

2.2 Models

For the task of binary classification, we implement the following models:

- Feed Forward Neural Network (FFNN) for the prediction of CERs (enchancers and promoters) on the K562 cell line
- Convolutional Neural Network, for the same tasks on the *hg38* human genome assembly
- Multi-Modal Neural Network, that training the two previous models on their respective data, aims at solving the task concatenating them

For reducing the number of features, a simple feature selection with a Random Forest Classifier has been computed.

To visualize the feature space we run T-SNE on a subsample of the data, and UMAP on the full dataset, which showed good results on genetic data [5].

2.2.1 Random Forest Classifier

To reduce the dimensionality and train models faster, we select features that achieve a minimum level of importance. The importance of a feature is computed with 3 runs on a Random Forest Classifier computed by *sklearn.ensemble.RandomForestClassifier* with criterion set to "gini" and random state set to 42. It is an ensemble learning method that constructs multiple decision trees, which outputs the class selected by most trees. However, for our purpose, we are not interested in the actual prediction power of the algorithm, instead we extract the features and their importance: the features on top of the tree are the most informative, so we get a hierarchy of attributes from which we can extract feature importance.

Based on the dimensionality that we wish to achieve, we can decide which threshold of importance makes us keep a feature or not. Once the 3 runs are averaged, and the median computed, we can decide to keep the features which reach an importance level above the median, or to select the n-top ones. Of course selecting the features with a positional measure like the median discards exactly half of the features, meaning a reduction of 50%. However we can see from Figure 3 that there isn't a clear distinction between most and less informative features. Given the power of deep learning methods, we are happy with the top 199 features for enhancers and top 205 for promoters.

More complex methods based on Decision Trees are available (like Boruta) but they have not been used due to the high computational cost with respect to the other

Layer	Shape	Size	Stride/Rate	Activation
Dense	32			ReLU
Dense	8			ReLU
Dropout			0.3	
Dense	1			sigmoid

Table 3: Architecture for the FFNN model. The middle part is repeated two times.

preprocessing and classification tasks: satisfactory results for this particular task can be achieved also with a simple model like Random Forests with multiple averaged runs.

2.3 UMAP and T-SNE

Given the high dimensionality of the feature space, we can try to visualize it by mapping it to a 2-d plane. We color in purple (dark) the instances which have been labelled as active, and yellow (light) the inactive ones.

In order to do that, we look into two different methods. The first one is UMAP, and the second is T-SNE. The latter is extremely more expensive to compute with respect to UMAP, so we have done it on a subsample of 10% of the original data. We can see that the T-SNE plot in Figure 4, with respect to Figure 5 made by UMAP, behaves similarly.

It is clear that both methods plot two defined clusters for the promoters reference genome, but it is in no way correlated to the actual promoter expression. A similar concept holds for the enhancers on the same dataset, but here T-SNE fails at making distinguishable clusters.

An interesting structure arises from the ENCODE data, which is more clearly defined by UMAP, however we can see that the positive labels get dispersed quickly for enhancers, due to the unbalancing.

3 Classification

Disclaimer: the work done has been strictly limited by the author hardware limitations. Hence, we advise the reader to repeat the experiments with more resources, if available.

As explained in Section 2.2, we train 3 different models on 3 types of datasets, of course for each task. The models are evaluated on a unique testing set which has been extracted before training, so that no training data would be in it. The testing set is 20% of the entire dataset.

3.1 Training

For each of the 3 models (and task), we train for 50 epochs, on 5 holdouts. This is due to some Google Colab limitation on GPUs when training different models for many epochs, hence the initial disclaimer. The models are trained locally on a Intel i5-6400 CPU @2.70GHz. To fully understand the power of deep learning methods, one would wish to train on adequate hardware (GPUs or TPUs) for many epochs, preventing overfitting by stopping the training when no further improvement is made on the training loss. Because this is not the case, the results of this work is highly influenced by resource limitations, and it would be recommended to train for at least 100 epochs on at least 10 holdouts to measure the statistical significance of the results.

Additionally, the training has been regulated with the EarlyStopping callback, which stopped training after no improvement over .005 on loss had been done in 2 epochs.

The holdout strategy is as follows: after removing the testing data from the dataset (with the same strategy) we extract stratified randomized folds, that take into consideration the classes distribution. This means that we draw random indices that will be used for selecting which sample would be used for training or testing. We do this for 5 splits.

The model training is optimized based on the *nadam* optimizer and the loss computed is the binary cross-entropy $L_{\log}(y,p) = -(y\log(p)+(1-y)\log(1-p))$ with a true label $y\{0,1\}$ and a probability estimate $p = Pr(y = 1)$, for all models. They are trained on batches of 32 samples.

Layer	Shape	Size	Stride/Rate	Activation
Conv1D	32	6	1	ReLU
Conv1D	8	4	1	ReLU
Dropout			0.3	
MaxPool1D		2		
GlobalAvgPool				
Dense	64			ReLU
Dense	1			sigmoid

Table 4: Architecture for the CNN model. The middle part is repeated two times.

3.2 Models Architecture

The models have been build and trained using Keras, with the parameters shown in Table 3 and Table 4. The Multi-Modal Neural Network concatenates the last hidden layer of the FFNN with the one from the CNN, and ends with a Dense layer with 64 neurons and ReLU activation, and a Dense layer with one neuron with sigmoid as activation.

Because our features are not that many, with respect to other tasks like image classification, we can achieve good results even with a low number of parameters, which is also advisable to avoid overfitting.

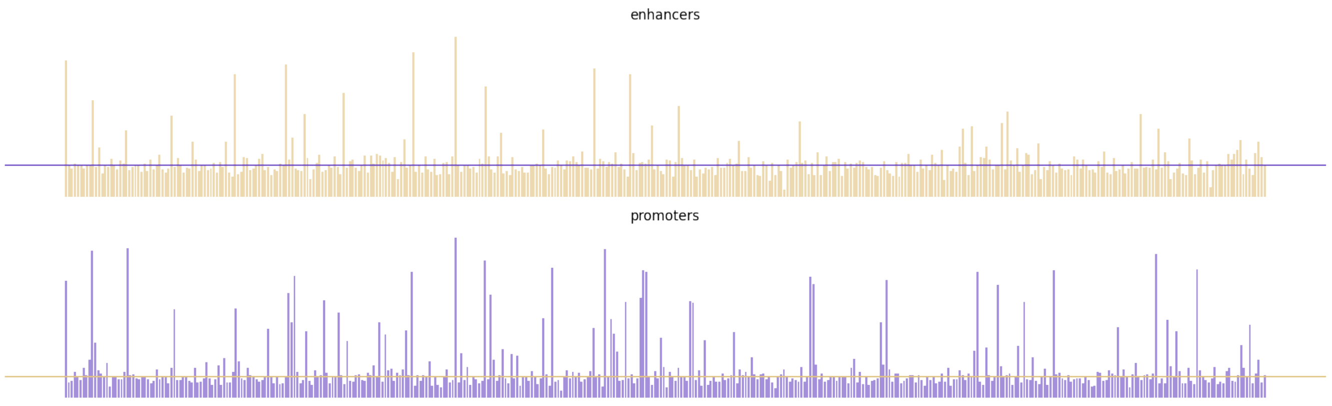

Figure 3: The results of 5 runs on the Decision Tree algorithm are plotted, with their median in opposite color. The final importance is computed as the mean of the different runs. The values are log-scaled for visualization purposes.

3.3 Metrics

We evaluate the models performance mostly on the following metrics:

- **AUROC**: We can model our binary classification task as a continuous random variable X. Given a threshold T, we classify as positive the instances that have a score above T, and negative otherwise. X follow a probability density function $f_1(x)$ if the instance is actually positive, and $f_0(x)$ if otherwise. We call $\text{TPR}(T) = \int_T^\infty f_1(x)\,dx$ the True Positive Rate and $\text{FPR}(T) = \int_T^\infty f_0(x)\,dx$ the False Positive Rate. The Area Under the Receiver Operating Characteristic is the area under the curve that plots parametrically $\text{TPR}(T)$ versus $\text{FPR}(T)$.
- **AUPRC**: it is the area under the curve of plotted recall ($\frac{TP}{TP+FN}$) with respect to precision ($\frac{TP}{TP+FP}$).
- **F1-score** or F-measure: harmonic mean of precision and recall
$$F_1 = 2 \times \frac{\text{PPV} \times \text{TPR}}{\text{PPV} + \text{TPR}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$
- **Balanced Accuracy**: computed as the sum of recall and specificity ($\frac{TN}{TN+FP}$) divided by 2
$$BA = \frac{TPR + TNR}{2}$$

All these metrics indicate a perfect performance when scoring 1.

4 Results

In order to have a stronger evaluation, we test our models on the same dataset. We trained the models multiple times, with the use of 5 holdouts. We can see from Figure 6 how they performed on both tasks.

Firstly, we plot the loss, which for binary cross-entropy shows how close the prediction probability is to the true value. We then look at the AUROC, which reaches a good result even with models so simple and trained for so little time. In particular, we can see that the FFNN scored an average of .891 for AEvsIE and of .897 for promoters! This is the best outcome, and it shows that even with a small network we can reach S.O.A. result, even if the previous models cited often trained on multiple cell lines, thought balancing the classes. It is also worth to mention that due to the limited resources it was not possible to perform a good tuning strategy, but rather we went with the standard ”try what looks best” approach. Possibly, applying methods like Bayesian Optimization [4], we could find the model that best fits the data.

A similar argument can be made for the other metrics, which show low variance throughout the 5 holdouts. The only model that performs poorly is the CNN on the reference genomic data. We are not impressed because we had a hint of that when looking at the decomposition of the data in 2 dimensions (Section 2.3).

5 Conclusions

We can get insights about this category of tasks by reviewing the methods and the results obtained, but we cannot jump to conclusions because we need to repeat the experiment on a more powerful machine. However, we can still say something about how we tackle this problem in the future.

The process of feature selection is important, and it could be something to focus on more. By removing features that do not add any information, we can help the

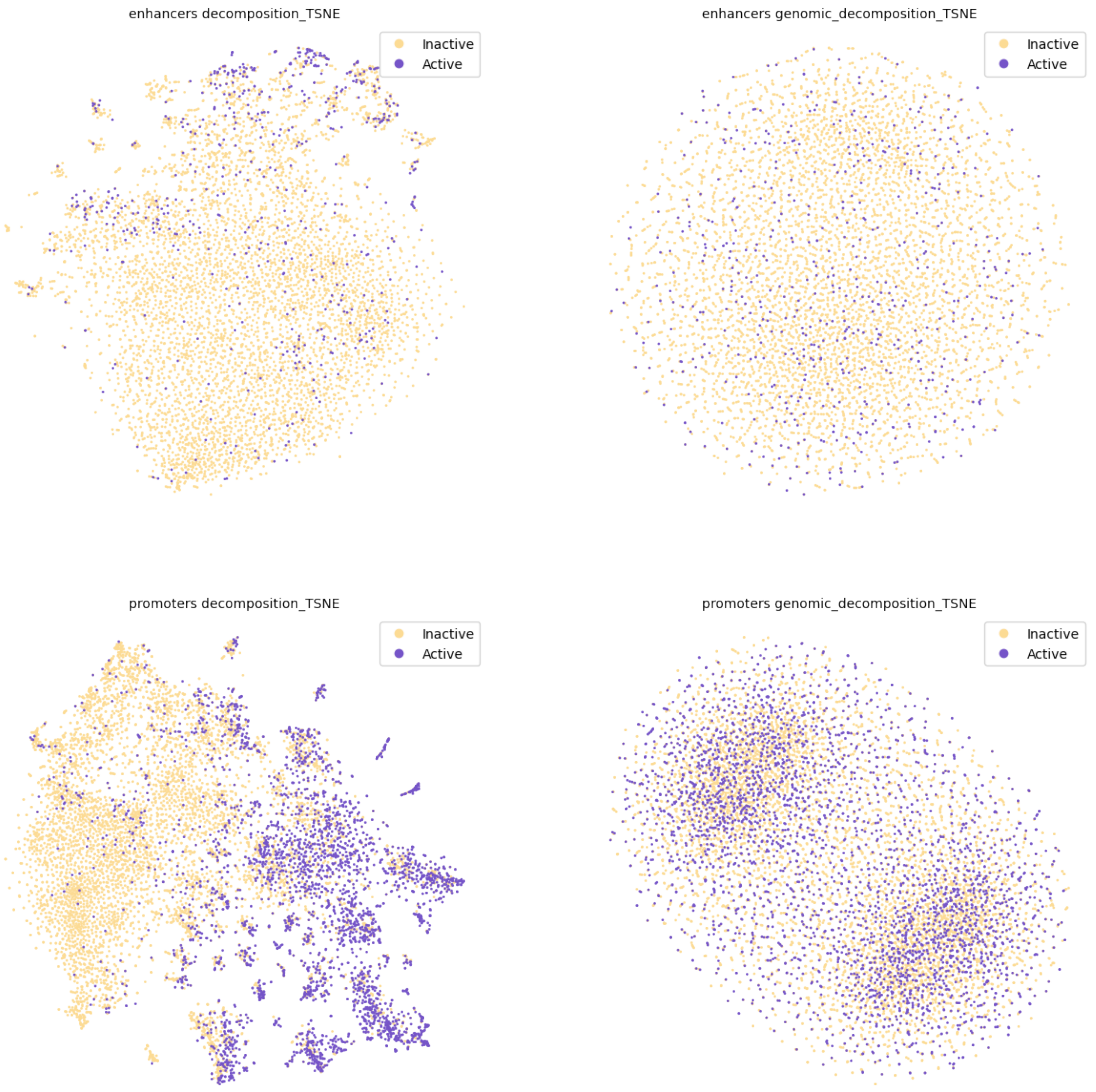

Figure 4: Feature decomposition using T-SNE on a subsample (10%). Activation threshold is set to 0 both for enhancers and promoters. On the left we have the data from ENCODE, on the right the UCSC Genome Browse hg38 genome. Labels from FANTOM5.

model to discriminate better. They also lower the computational times, and it would be interesting to compare the performance of a FFNN, trained on 200 features like we did, with an explainable model, like a SVM, trained only with the best features selected from algorithms like Random Forests or Boruta.

Visualizing our data has also prove to be beneficial, because the results that we obtained on the reference genome showed that it could have been difficult for a CNN to learn that type of data, has it is difficult to find clusters of active regions in the plotted decompositions.

It is also important to remember that the metrics used for unbalanced datasets like the ones we discussed have a lot of weight to the final model evaluation. If we judge

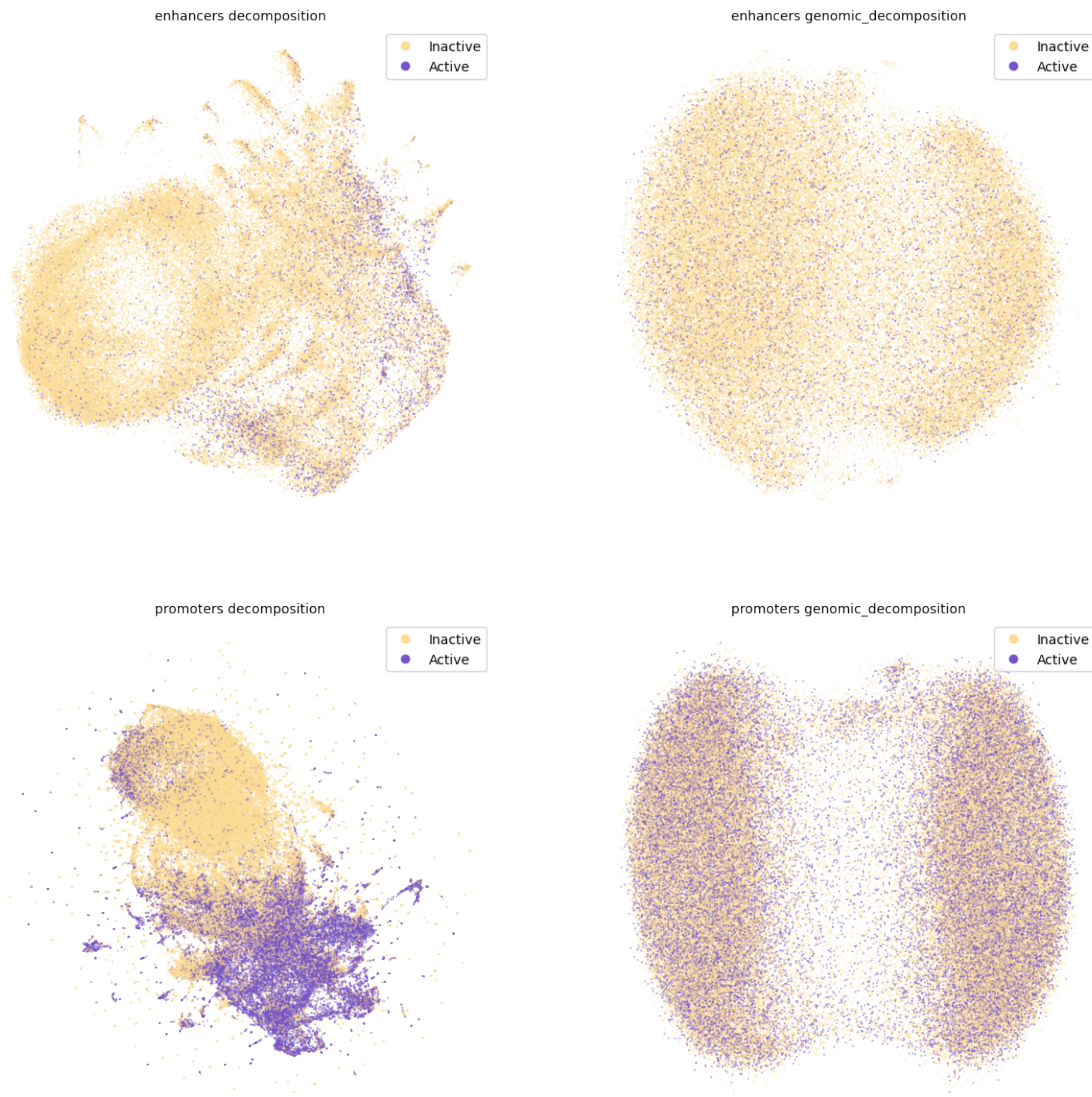

Figure 5: Feature decomposition using UMAP on the full dataset. Activation threshold is set to 0 both for enhancers and promoters. On the left we have the data from ENCODE, on the right the UCSC Genome Browse hg38 genome. Labels from FANTOM5.

our model based on accuracy only, we can expect the model to learn the distribution of the labels, rather than which feature distinguish one instance from another. In addition, it would be advisable to repeat the training with a greater number of holdouts, and to compute the statistical relevance of that result: is the model performing well only by chance, or can we say it is actually better than a naive classifier?

How we perform the selection of the instances for the holdouts is also relevant, mostly because we don't want the less present label to get lost in some sets and not appear in others.

Normally, one would like to discuss the threshold used for when we consider a region active or not, but for

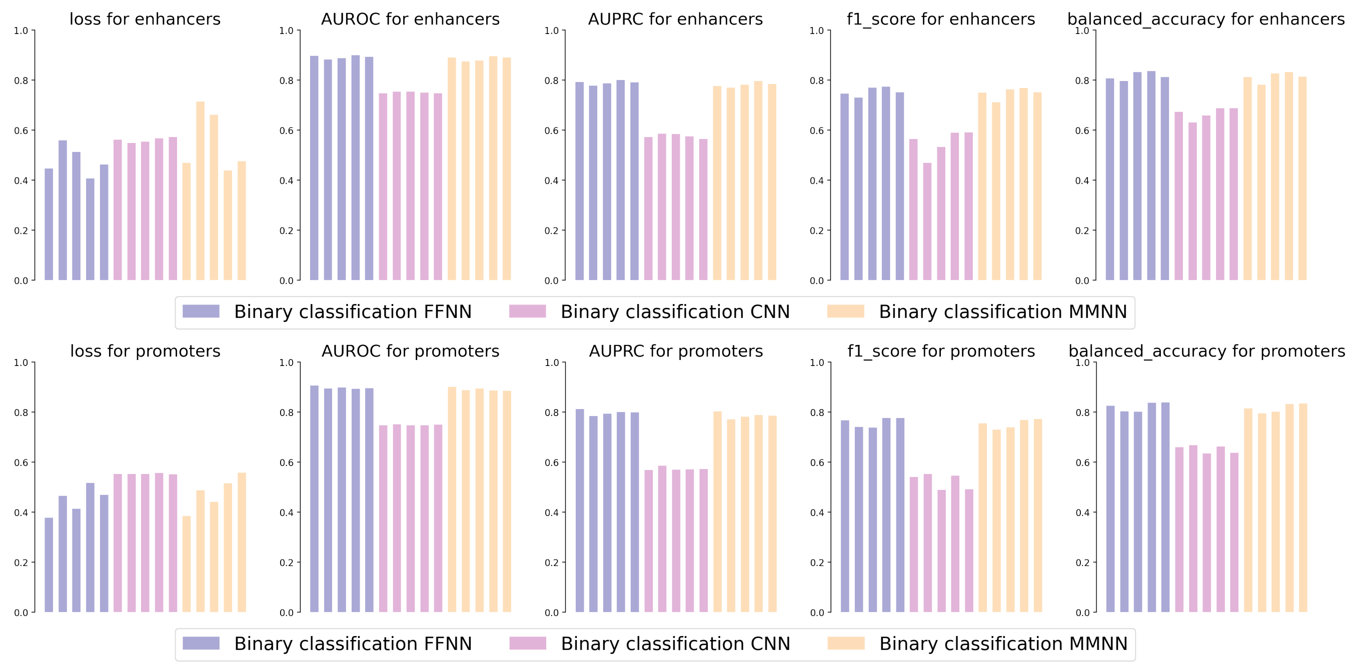

Figure 6: Evaluation results on unseen data for the task "Active Enhancers vs Inactive Enhancers" (top) and "Active Promoters vs Inactive Promoters" (down) for each model trained on 5 holdouts.

this cell line it was clear that no improvements could be made. Moreover, the FANTOM5 team recommends these thresholds, so cannot do anything more than accept the unbalance.

Lastly, we didn't notice improvements when using a Multi Modal approach for this particular task on this specific cell. This is not an indication of failure of the model, because it performed still very similarly to the DNN, however it would be interesting to compare this approach between different cell lines or adding new models that handle different data that could be useful to the task.

6 Acknowledgements

Much of this work is based on that of Luca Cappelletti, computer science researcher at UniMi, who also provided many libraries which were vital to the implementation of the tasks resolution. Among them, we have:

- *epigenomic_dataset*: a data wrapper which downloads enhancers and promoters
- *ucsc_genomes_downloader*: to download the reference genome
- *keras_bed_sequence*: to bed the reference genome with respect to the epigenomic data
- *keras_mixed_sequence* and *keras_bed_sequence*: to create Sequence objects to be fed to the models
- *extra_keras_metrics*: for additional Keras metrics

Model	Loss	Accuracy	AUROC
FFNN	**.477**	**.831**	**.891**
CNN	.560	.708	.749
MMNN	.551	.827	.885

Table 5: Average of evaluation results of 5 holdouts on enhancers.

Model	Loss	Accuracy	AUROC
FFNN	**.447**	**.835**	**.897**
CNN	.552	.709	.747
MMNN	.476	.831	.889

Table 6: Average of evaluation results of 5 holdouts on promoters.

References

[1] Dunham et al. “An integrated encyclopedia of DNA elements in the human genome”. In: *Nature* 489.7414 (2012), pp. 57–74. DOI: 10.1038/nature11247.

[2] Noguchi et al. “FANTOM5 CAGE profiles of human and mouse samples”. In: *Scientific Data* 4.1 (2017), p. 170112. DOI: 10.1038/sdata.2017.112.

[3] Michael A. Beer. “Predicting enhancer activity and variant impact using gkm-SVM”. In: *Human Mutation* 38.9 (2017), pp. 1251–1258. DOI: https://doi.org/10.1002/humu.23185. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.23185. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.23185.

[4] Luca Cappelletti et al. “Bayesian Optimization Improves Tissue-Specific Prediction of Active Regulatory Regions with Deep Neural Networks”. In: *Bioinformatics and Biomedical Engineering*. Springer International Publishing, 2020, pp. 600–612. DOI: 10.1007/978-3-030-45385-5_54. URL: https://doi.org/10.1007/978-3-030-45385-5_54.

[5] Michael W. Dorrity et al. “Dimensionality reduction by UMAP to visualize physical and genetic interactions”. In: *Nature Communications* 11.1 (Mar. 2020). DOI: 10.1038/s41467-020-15351-4. URL: https://doi.org/10.1038/s41467-020-15351-4.

[6] Wasserman Li Shi. “Genome-wide prediction of cis-regulatory regions using supervised deep learning methods”. In: *BMC Bioinformatics* 19.1 (May 2018), p. 202. ISSN: 1471-2105. DOI: 10.1186/s12859-018-2187-1.

[7] C B Lozzio and B B Lozzio. “Human chronic myelogenous leukemia cell-line with positive Philadelphia chromosome.” In: *PubMed* 45.3 (1975), pp. 321–34. URL: https://pubmed.ncbi.nlm.nih.gov/163658/.

[8] Karen H Miga et al. “Centromere reference models for human chromosomes X and Y satellite arrays”. en. In: *Genome Res.* 24.4 (Apr. 2014), pp. 697–707.

[9] Xu Min et al. “Predicting enhancers with deep convolutional neural networks”. In: *BMC Bioinformatics* 18.13 (Dec. 2017), p. 478. ISSN: 1471-2105. DOI: 10.1186/s12859-017-1878-3. URL: https://doi.org/10.1186/s12859-017-1878-3.

[10] Mihaela Pertea and Steven L Salzberg. “Between a chicken and a grape: estimating the number of human genes”. In: *Genome Biology* 11.5 (2010), p. 206. DOI: 10.1186/gb-2010-11-5-206. URL: https://doi.org/10.1186/gb-2010-11-5-206.

[11] Allison Piovesan et al. “On the length, weight and GC content of the human genome”. In: *BMC Research Notes* 12.1 (Feb. 2019). DOI: 10.1186/s13104-019-4137-z. URL: https://doi.org/10.1186/s13104-019-4137-z.

[12] David N. Reshef et al. “Detecting Novel Associations in Large Data Sets”. In: *Science* 334.6062 (2011), pp. 1518–1524. DOI: 10.1126/science.1205438. eprint: https://www.science.org/doi/pdf/10.1126/science.1205438. URL: https://www.science.org/doi/abs/10.1126/science.1205438.

[13] Fang Tan et al. “Essential role for ALCAM gene silencing in megakaryocytic differentiation of K562 cells”. In: *BMC Molecular Biology* 11.1 (Dec. 2010). DOI: 10.1186/1471-2199-11-91. URL: https://doi.org/10.1186/1471-2199-11-91.