"Ultra Chrome" Unique Deep CNN model for predicting gene expression from histone modifications

Davide Ghiotto [†] 1236660

***Abstract*—Future vehicular communication networks call for new solutions to support their capacity demands, by leveraging the potential of the millimeter-wave (mm-wave) spectrum. Mobility, in particular, poses severe challenges in their design, and as such shall be accounted for. A key question in mm-wave vehicular networks is how to optimize the trade-off between directive Data Transmission (DT) and directional Beam Training (BT), which enables it. In this paper, learning tools are investigated to optimize this trade-off. In the proposed scenario, a Base Station (BS) uses BT to establish a mm-wave directive link towards a Mobile User (MU) moving along a road. To control the BT/DT trade-off, a Partially Observable (PO) Markov Decision Process (MDP) is formulated, where the system state corresponds to the position of the MU within the road link. The goal is to maximize the number of bits delivered by the BS to the MU over the communication session, under a power constraint. The resulting optimal policies reveal that adaptive BT/DT procedures significantly outperform common-sense heuristic schemes, and that specific mobility features, such as user position estimates, can be effectively used to enhance the overall system performance and optimize the available system resources.**

***Index Terms*—Histone Modifications, Deep Learning, Convolutional Neural Networks, Recurrent Neural Networks.**

I. Introduction

The main contributions of *UltraChrome* can be summarized as follows:

- Is the first implementation of (Spatial) Convolutional Neural Network used to tackle gene expression prediction tasks
- Sightly improves previous state-of-the-art Deep Learning models if compared over all the available cell types
- Generalize over the cell type leading to one unique model for all the available cell types

II. Related Work

- ***DeepChrome*** : proposes the first deep convolutional neural network model to predict gene expression from histone modifications
- ***AttentiveChrome*** : advances the previous *DeepChrome* model by applying selective attention techniques to develop a better and more understandable model

[†]Department of Mathematics, University of Padova, email: d.ghiotto@studenti.unipd.it

III. Processing Pipeline

The input data consists of 56 cells types each of them containing different genes represented by 5 different histone modifications over 100 bins.
Starting from the raw input generated and formatted by *Singh et al. (2016)* [1] I transformed the 5×100 matrix in one *3D* tensor of shape $5 \times 100 \times 1$ in order to correctly feed it to the convolution layer at the beginning of the deep learning model. After the convolution I placed some regularization layers and then a multi-layer architecture. The last step is simply a softmax operation to reduce the computation into a binary classification task.

IV. Signals and Features

My setup is identical to *Singh et al. (2016)* [1] where they primarily focus on the regions around TSS, based on the observations from Cheng et al. (2011) showing that signals close to the TSS are the most informative.
They divided the 10 000 basepair (bp) DNA region (65000 bp) around the transcription start site (TSS) of each gene into bins of length 100 bp. Each bin includes 100 bp long adjacent positions flanking the TSS of a gene. In total, they consider five core histone modification marks from REMC database (Kundaje et al., 2015). These five histone modifications are selected as they are uniformly profiled across all cell-types considered in that study. This makes the input for each gene a 5×100 matrix, where columns represent different bins and rows represent histone modifications. For each bin, they report the value of all 5 histone signals as the input features for that bin Fig.1.
Here I formulate the gene expression prediction as a binary classification task. Specifically, the outputs of my models are labels 1 and -1, representing gene expression level as high or low, respectively.
Following what done by *Singh et al. (2016)* [1] I split the dataset into training (6601), validation (6601) and test (6600).
In addition to this procedure I standardized the dataset using mean and standard deviation of the training set. I applied this scaling to improve the performance of the neural networks learning algorithm.

V. Learning Framework

UltraChrome architecture is composed of different layers. The diagram in figure X (??) provides a visual reference of

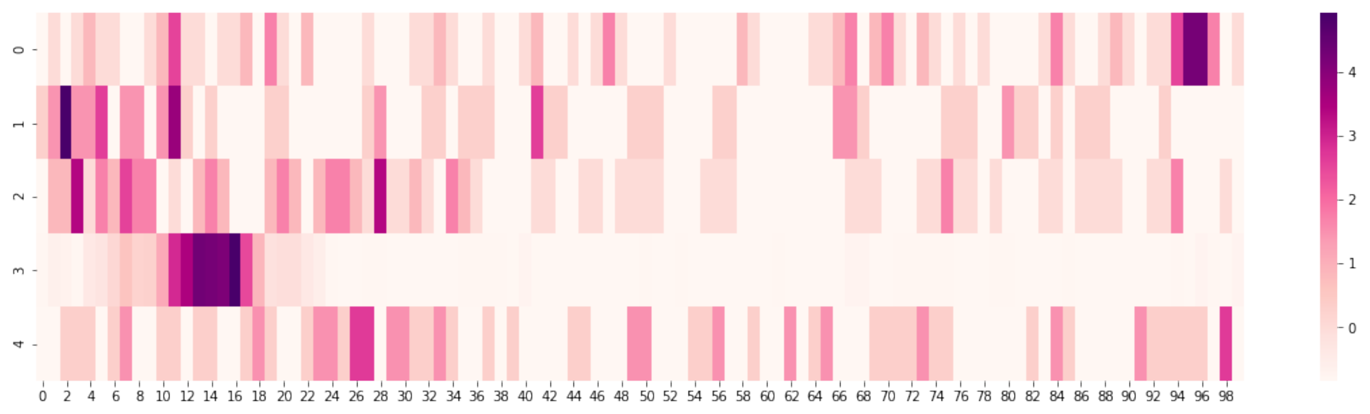

Fig. 1: Example of the matrix representation of the input

the model that can be summarized as follows:

- **Input layer**: the input is a tensor of shape $5 \times 100 \times 1$.
- **Convolution**: I applied a spatial (2D) convolution with F filters with a kernel size of 5. This operation performs a sliding window over all bins positions and across all 5 histone modifications. The convolution enables the model to extract visual, local and global features of the distribution of the modifications.
- **Pooling**: to preserve local features I inserted a *max pool* layer of shape (1×5): the particular choice of the shape aimed to capture the variability just over the bins and not also across the different modifications.
- **Dropout**: I inserted a *dropout* layer with a rate of 0.5 after the pooling to help the generalization ability of the model and reduce overfitting.
- **Flattening**: this flattening step is just useful to format the input for the following fully connected architecture.
- **Fully Connected layers**: the core of the learning is achieved by exploiting a multi layer, fully connected, architecture composed of 3 different layers with N_1, N_2, N_3 neurons each of them with *relu* activation function to produce non-linearities.
- **Batch normalization**: in order to reduce the gradient vanishing problem I inserted 2 *Batch Normalization* layers between the 3 fully connected layers.
- **Output layer**: the final layer is just a Dense layer with one neuron that maps to the desired output in the form of a probability score.

VI. Results

A. Validation of Hyper-parameters

In order to select the best model I chose the hyper parameters validating them on the validation set. I tested different options and values:

- Number of filters: $\{50|100\}$
- Number of neurons per layer:
  - Low : $[32, 128, 32]$ (3 layers)
  - $High$: $[64, 256, 64]$ (3 layers)
  - $Deep$: $[64, 256, 256, 64]$ (4 layers)
- Dropout: $\{Yes|No\}$
- Batch Normalization: $\{Yes|No\}$

In the Tab. 1 are reported a compact subset of the most significative results of the validation step. As we can see,

Hyper-parameters				**AUC validation set**		
Filters	Neurons	Dropout	BN	Min	Max	Mean
50	Low	No	No	0.653	0.935	0.764
100	Low	No	No	0.653	**0.939**	0.765
100	High	No	No	0.653	0.936	0.763
100	High	Yes	No	**0.654**	0.933	0.763
100	High	Yes	Yes	0.652	0.937	**0.766**
100	Deep	Yes	Yes	0.648	0.937	0.765

TABLE 1: Hyper parameter tuning for *UltraChrome* on validation set

the best combination is obtain with 100 filters, high number of neurons, dropout and batch normalization. However the differences are not large from one combination to another: small changes in the second part of the architecture (the fully connected multi-layers) are not so relevant on the overall results, suggesting that the main contribution maybe come from the convolution itself rather than the particular deep models. Moreover, the last combination reported, with a deeper architecture, do not lead to better results, but only increases the complexity of the network.

B. Other models

I tried some other models:

- RFC
- Base NN
- ChromeR (RNN)

C. Global and individual models

As mentioned in the introduction I, my contributions are also related to the type of training approach: I trained my models over the full dataset, unaware of the different cell types (the same principle applies to validation and test set).

However, in order to compare the original baselines of

DeepChrome and *AttentiveChrome* I also trained my NNs in an individual way: for each cell type I trained a model, validated on its set and tested only on his test set.
Implementing *DeepChrome* model myself allowed me to reproduce both configurations (global and individual).
Individual configurations led to better results in terms of AUC score but required to differentiate for the cell-type also at test time: this particular setup poses a strong hypothesis on the problem formulation and execution in the sense that we must obtain the cell-type separately before testing the unseen data with our model.

D. Test set AUC Comparison

In the Tab 2 are reported the AUC scores for the implemented methods and the two baselines from previous related work. As we can see from the results, *AttentiveChrome* is the best *individual* model in terms of mean AUC. However, reporting just the mean AUC averaged over all 56 cell types is not so interesting as we look at only one single score. This the reason why I also trained my global model individually for all the different cell types. This arrangement puts my algorithm on the same level as *DeepChrome* and *AttentiveChrome* , enabling me to discriminate the single improvement on the different cell types.
The plot Fig. 2 shows the performance of some *global* models. The plot in Fig. 3 shows the performance of some *individual* models.

Model	**Training type**	**AUC**
Random Forest Classifier	*Global*	0.7942
Base Neural Network	*Global*	0.7930
ChromeR	*Global*	0.7961
DeepChrome	*Global*	0.8032
UltraChrome	*Global*	0.8045
Base Neural Network	*Separated*	0.7986
ChromeR	*Separated*	0.8009
DeepChrome	*Separated*	0.8068
AttentiveChrome	*Separated*	**0.8115**
UltraChrome	*Separated*	0.8114

TABLE 2: AUC scores comparison on test set

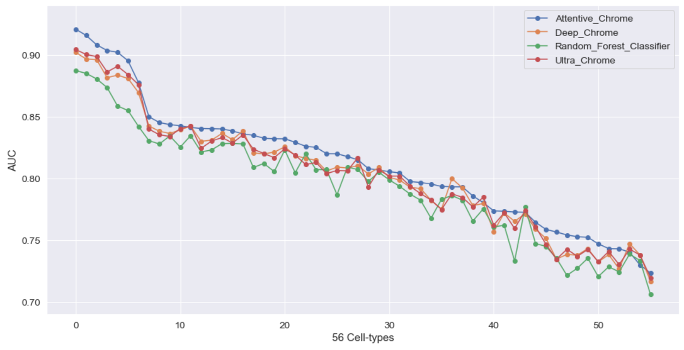

Fig. 2: AUC global models

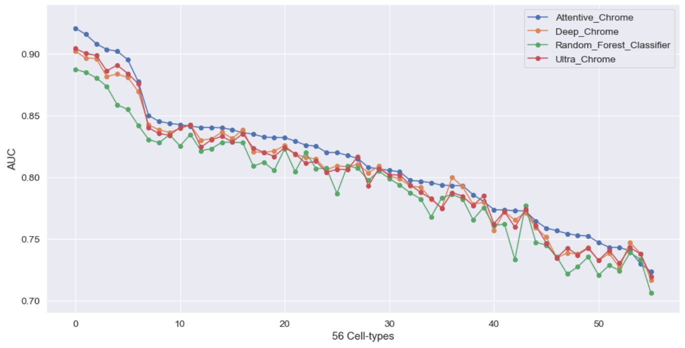

Fig. 3: AUC individual models

E. Improvements Matrix

To graphically visualize the number of improvements from one model to another I built an improvements matrix, defined as follows:

$$\boldsymbol{IM} = [x_{ij}] \tag{1}$$

where:

- $i, j \in \mathcal{M}$
- $\mathcal{M}$: models set
- x_{ij} : # improvements of model i over model j.

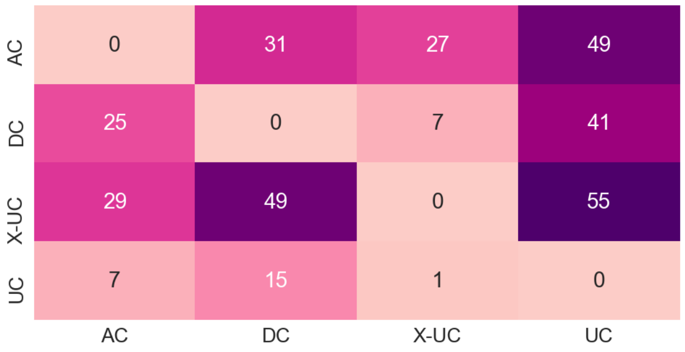

Fig. 4: Improvements Matrix

The matrix defined by Eq. (1) is reported in the Fig. (4) for the 4 best models. If we look at row level we have (going downwards) :

- *AttentiveChrome* (AC) : 31+27+49 $= 107$ improvements
- *DeepChrome* (DC) : $25 + 7 + 41 = 73$ improvements
- *UltraChrome* (individual) (X-UC) : $29 + 49 + 55 = 133$ improvements
- *UltraChrome* (global) (UC) : $7 + 15 + 1 = 23$ improvements

Observing the improvements over all the cell types, individual *UltraChrome* results as the best model even if compared to *AttentiveChrome* . However if we look at global *UltraChrome* the results are not as good, but the different problem setup generates this unfair comparison: the individual models are specifically trained, validated and tested on one cell type individually; instead *UltraChrome* reaches good AUC scores in a global scenario.
There are several implications to these findings:

- the model incorporates all the cell types characteristics and can describe the phenomenon as a whole
- lower probability of overfitting the model as we increase both the complexity of the task and the input space
- better generalization ability over new cell types

VII. Concluding Remarks

In this paper I have presented *UltraChrome* a Deep Convolutional Neural Network that can effectively predict the gene expression by simultaneously analyzing spatial distribution of histone signals between all 5 kinds of modifications. Moreover, the model in cell type independent and do not need to know the cell type at test time, improving the generalization ability of the model and extending the previous models with this global approach.
The spatial convolution of *UltraChrome* intercepts patterns and interactions between different modifications. This approach is vital to better model the gene expression mechanism: as reported in this paper (cite here) histone modifications can positively or negatively affect other modifications. This phenomenon is called *crosstalk*.
Moreover, in the future, as new cell types and their relative histone modifications are added to the database we may be interested in instantly predict the gene expression without training a specific model for the new cell type. In this scenario a global model like *UltraChrome* can be useful to assign an initial indication of the gene expression to be refined later on with an individual and cell-specific model.
Future developments could improve the interpretability of *UltraChrome* model: the deep convolution internally maps the input features and extracts meaningful relations, but still miss a way to describe the biology underpinning these processes. This is relevant from a epigenetic point of view. Another improvement could be the integration of new histone modifications in addition to the 5 of the current setup. The new modifications could contribute to shed a light in more complex relation and *crosstalk* effects.

A. Project Remarks

Personally I found the challenge of this project pretty tough mainly because the domain was very specific and required me some time and effort to properly understand the context of the problem.
On the other hand, the project was also instructive. First, I had the chance to develop from scratch deep learning models and test several architectures and hyper-parameters. Second, I was forced to build a complete learning framework, starting from data collection all the way to the communication of the results. In particular I reckon to have been very useful to think about new ways of communicate my results in an effective and elegant manner.

References

[1] G. R. Ritambhara Singh, Jack Lanchantin and Y. Qi, "DeepChrome: deep-learning for predicting gene expression from histone modifications," *Bioinformatics*, pp. i639–i648, Sept. 2016.