DREAM Challenge 2022
Predicting gene expression using millions of random promoter sequences
by KircherLab - predicting-gene-expression DREAM Challenge

Abstract

Understanding the regulation of genes is an important step towards the interpretation of sequence alterations causing disease. The regulatory activity can be measured through functional assays but are limited in their throughput given the large universe of hundreds of millions potential variants. Therefore, predictive models are built that try to learn the regulatory property of a DNA sequence. Here we present a simple convolutional neural network that can be trained with expression data of high-throughput assay and predict the potential activity of a short sequence. Because such experimental data is biased towards GC content, we implemented a unique GC correction step on the training data so that the model can focus its decisions on motifs within the sequence rather than the general nucleotide composition. This makes the model more interpretable towards learned motifs and enables the ability to measure sequence alterations, a step forward understanding diseases.

1. Description of data usage

Because we used a classification approach using 18 classes (equal size of bins) we first selected the classified bin by rounding the measured sequence effect. Then we randomly split up the data into 70% training, 20% test, and 10% as validation set. Sequences that were not the size of 110 (including adapters) were removed. As for the model's output, the classified bins were also one-hot encoded among 18 classes using the function `pandas.get_dummies()`.

We recognized a high correlation (0.34 Pearson correlation) between GC content and measured sequence effect. We wanted the model to be unbiased in terms of GC content. This enables a better interpretation of models regarding the biology behind the activity of sequences. Otherwise, GC rich motives will be favorized for high activity and GC poor motives for low activity, respectively. When models are being used for in-silico variant predictions changes from G or C to A or T will have by design a higher effect than between G and C or A and T. This is not in line with the observation of mutational robustness to frequent transitional over transversional changes [1].

Therefore, we sampled the training set sequences in each bucket using the overall GC distribution (all input data). Another approach would be using the GC distribution from the organism of random sequences of the same length. For each bucket we sampled 80% without replacement of the training sequences with this approach. We note that the buckets are very imbalanced. High count buckets are in the center of the effect range (see "4. Other important features" for more discussion on that).

Finally, sequences are one-hot encoded using a custom written data loader (`kipoi_dataloader`) for the challenge from Kipoi [2] (See `training/sequences.py`).

2. Description of the model

We used a convolutional neural network that can be trained on one-hot encoded sequences and returns the 18 different classes/bins. It simple contains of two blocks connected in series with two convolutional layers a max pooling layer and a dropout layer. Then a dense layer followed, with another dropout and finally two dense layers. We keep the final activation function linear and do not use any softmax or similar function. The model has 2,194,768 total

parameters, 2,193,068 are trainable and 1,700 are non-trainable. A schematic figure of the model can be found in Figure 1 and layers are described in Table 1.

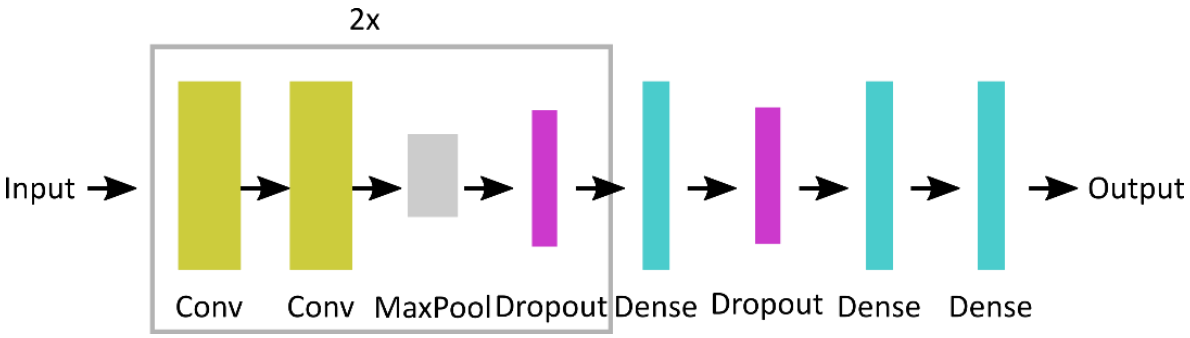

Figure 1 Schematic version of the model.

Layer type (activation)	Output Shape	Param #
InputLayer	[(None, 110, 4)]	0
Conv1D (softmax)	(None, 104, 250)	7250
BatchNormalization	(None, 104, 250)	1000
Conv1D (softmax)	(None, 97, 250)	500250
BatchNormalization	(None, 97, 250)	1000
MaxPooling1D	(None, 48, 250)	0
Dropout 0.1	(None, 48, 250)	0
Conv1D (softmax)	(None, 46, 250)	187750
BatchNormalization	(None, 46, 250)	1000
Conv1D (softmax)	(None, 45, 100)	50100
BatchNormalization	(None, 45, 100)	400
MaxPooling1D	(None, 45, 100)	0
Dropout 0.1	(None, 45, 100)	0
Flatten	(None, 4500)	0
Dense (sigmoid)	(None, 300)	1350300
Dropout 0.3	(None, 300)	0
Dense (sigmoid)	(None, 300)	90300
Dense (linear)	(None, 18)	5418

Table 1 Summary of model.

3. Training procedure

We are using mean squared error (MSE) as loss function during training. Training was done with early stopping using Tensorflow `EarlyStopping` method with patience of 10 and restoring the best weights. As metrics we report MSE, mean absolute error (MAE) and accuracy (acc). Finally, training was done in 20 epochs and metrics of training and validation set are reported in Table 2.

epoch	acc	mae	mse	val_acc	val_mae	val_mse
0	0.261	0.096	0.046	0.234	0.094	0.047
1	0.284	0.093	0.045	0.173	0.100	0.050
2	0.291	0.093	0.045	0.138	0.099	0.049
3	0.297	0.092	0.045	0.264	0.091	0.047

4	0.302	0.091	0.044	0.218	0.092	0.047
5	0.307	0.091	0.044	0.264	0.091	0.045
6	0.311	0.090	0.044	0.290	0.091	0.045
7	0.317	0.089	0.044	0.299	0.090	0.045
8	0.322	0.089	0.044	0.299	0.088	0.044
9	0.326	0.088	0.044	0.301	0.089	0.044
10	0.331	0.088	0.044	0.291	0.089	0.045
11	0.335	0.087	0.043	0.295	0.088	0.045
12	0.340	0.087	0.043	0.292	0.088	0.045
13	0.344	0.087	0.043	0.293	0.089	0.045
14	0.348	0.087	0.043	0.281	0.089	0.045
15	0.352	0.086	0.043	0.295	0.088	0.045
16	0.355	0.086	0.043	0.280	0.089	0.045
17	0.358	0.086	0.043	0.292	0.088	0.045
18	0.361	0.086	0.043	0.288	0.088	0.045
19	0.364	0.085	0.043	0.294	0.088	0.045

Table 2 Statistics for each epoch.

Our model returns for each of the 18 classes one score. Now we must transform them back to a sequence effect from 0 to 17. We compute the value using the weighted mean over all buckets. The weights are the predictions made by our classifier. Final performance on our training, validation and test set is 0.75, 0.73 and 0.73 using Pearson correlation. We note that the Pearson correlation of the model with all input training data (without GC sampling) has the same correlation as with GC sampling.

We fully build the whole sampling/training process in the workflow management system Snakemake [3] using conda as package managing system and a config file for the parameters. We did not use the TPU support and run it on our GPUs (Tesla V100-SXM2-32GB) with CUDA version 11.6. For retraining our models, please install conda (we recommend mamba), create an environment (`conda create env -n snakemake_env`), activate it (`conda activate snakemake_env`) and install snakemake (`conda install snakemake`). Then setup a config file (example in `config/example.conf.yml`) and run snakemake with `snakemake –use-conda –configfile <your_config_file>` in the cloned github repository. The generated results are in the result folder. For more information on the workflow please read the readme on our github repository.

After the model predicts the weights on other sequences, we recommend the numpy function average to transform the classification predictions into a final prediction:

```
numpy.average(range(0,len(predictions)),weights=predictions)
```

4. Other important features

We note that our classification approach is not in-line with the data generation of the MPRA. Our data model favors a clear assignment to one bucket. This is achieved by a majority vote by removing sequences that are ambiguous. E.g., with a min support per bucket and a majority assignment to it like 75%. With this approach we expect a lower number of data but with less noise than the actual approach using the weighted mean over all assignment to buckets. In addition, the majority vote approach might lead to a more even distribution of sequences across the effects/buckets.

5. Contributions and Acknowledgement

5.1. Contributions

Name	Affiliation	Email
Sebastian Röner	Berlin Institute of Health at Charité – Universitätsmedizin Berlin, Berlin, Germany	sebastian.roener@bih-charite.de
Pyaree Mohan Dash	Berlin Institute of Health at Charité – Universitätsmedizin Berlin, Berlin, Germany	pyaree-mohan.dash@bih-charite.de
Max Schubach	Berlin Institute of Health at Charité – Universitätsmedizin Berlin, Berlin, Germany	max.schubach@bih-charite.de

5.2. Acknowledgement

Computation was performed on the HPC for Research cluster of the Berlin Institute of Health at Charité – Universitätsmedizin Berlin.

6. References

[1] M. Kircher *et al.*, “Saturation mutagenesis of twenty disease-associated regulatory elements at single base-pair resolution,” *Nat. Commun.*, vol. 10, no. 1, p. 3583, 08 2019, doi: 10.1038/s41467-019-11526-w.

[2] Ž. Avsec *et al.*, “The Kipoi repository accelerates community exchange and reuse of predictive models for genomics,” *Nat. Biotechnol.*, vol. 37, no. 6, pp. 592–600, Jun. 2019, doi: 10.1038/s41587-019-0140-0.

[3] F. Mölder *et al.*, “Sustainable data analysis with Snakemake,” *F1000Research*, vol. 10, p. 33, Jan. 2021, doi: 10.12688/f1000research.29032.1.

7. Feedback

Thank you for hosting this challenge!