Evaluation of Convolutionary Neural Networks Modeling of DNA Sequences using Ordinal versus one-hot Encoding Method

Allen Chieng Hoon Choong* and Nung Kion Lee†
*†Faculty of Cognitive Sciences and Human Development
Universiti Malaysia Sarawak,
94300, Kota Samarahan, Sarawak, Malaysia
*allen.choong@gmail.com †nklee@unimas.my

***Abstract*—Convolutionary neural networks (CNN) has been widely used for DNA motif discovery due to its high accuracy. To employ CNN for DNA motif discovery task, the input DNA sequences are required to be encoded as numerical values and represented as either vector or multi-dimensional matrix. This paper evaluates the simple and more compact ordinal encoding method versus the popular one-hot encoding for DNA sequences. We compare the performances of both encoding methods using three sets of datasets enriched with DNA motifs. We found that the ordinal encoding performs comparable to the one-hot encoding method but with significant reduction in training time. In addition, the one-hot encoding performance is quite consistent across various datasets but would require suitable CNN configuration to perform well. The ordinal encoding with matrix representation performs best in some of the evaluated datasets. This study implies that the performance of CNN for DNA motif discovery depends on the suitable design of the sequence encoding and representation. In addition, the CNN architecture and configuration require some tuning to suit different encoding methods.**

I. Introduction

CNN (Convolutional Neural Network) [1], [2] is currently one of the most widely used deep learning method in machine learning due to its powerful modelling capability on complex and large-scale datasets. Recently, CNN has been widely used for supervised learning of DNA sequences for the prediction of regulatory regions and other functional landmarks [3]–[6]. The advantage of CNN is the training can be performed without the need of engineered features. It can learn and discover the intrinsic features in the raw dataset through the many layered structure which represents the different abstraction of features. The layers in a CNN consist of convolutionary and pooling layers. A convolutionary layer consists of multiple maps of neurons which is called feature map or filter. A feature map convolves the input from the previous layer to produce a reduced sample. It is different from the fully connected network in the sense that it is only connected to a patch of the previous layer, which is named as “receptive field”. Moreover, all neurons in the feature maps scan the same feature of the previous layer but different locations. Different feature maps might detect different types of feature such as edges of the image, or the sequence motifs in DNA data sequences [7].

In addition, [7] stated that the exact location and frequency of the features are unimportant to the learning purpose because the final output of the deep learning is recognition of the input data. On the other hand, the pooling layer summarizes the adjacent neurons by computing their activity. As a result, the model parameters are reduced. After the last pooling layer in the CNN, it has a fully connected multi-layers perceptron neural networks.

CNN is designed to effectively models multidimensional input data. Thus, it is powerful in solving problems related to computer vision and image recognition [2] where the data consists of images. To employ the CNN for solving DNA motif discover problem, existing works typically encode nucleotides in DNA sequences using the one-hot method [3], [5], [6]. That is, each base pair in a sequence is encoded with a binary vector of four bits with one of its is hot (i.e. 1) while others are 0. For instance $A = (1, 0, 0, 0)$, $G = (0, 1, 0, 0)$, $C = (0, 0, 1, 0)$, and $T = (0, 0, 0, 1)$. This sequence representation method draws similarity to the Position Frequency Matrix [8]. In which, a vector indicates the probability of occurrences of the four bases at a certain position in a DNA sequence. Therefore, an input DNA sequence of length l is represented as $4 \times l$ matrix. Or in another word, a ”2D image” with one channel.

Methods for converting biological sequences into numerical values have been existed in numerious past studies [9]. Those encoding methods can be categorized into direct and indirect encoding [9]. Direct methods represent each nucleotides/amino acids with a numerical value or vector of numerical values. It preserved the original order the bases appear in a biological sequence after the encoding. While the indirect methods engineered a set of features (numerical forms) from the biological sequences. The features can be based on frequency counts of various k-mers (short sequence segments of length *k* bp), biological or biochemical properties.

In recent years, CNN and other deep learning techniques have gaining popularity in solving supervised DNA motif prediction because of their good performances in comparison with the support-vector-machine. Some of the recent works of using deep learning for DNA motif discovery are DeepBind

[3], DeepSEA [6], Basset [5], TFImpute [10], and FIDDLE [11]. All of those works are using one-hot encoding to encode the input DNA sequences.

In this study, we propose a simple method to transform DNA sequences into matrix representation. The matrices (i.e images) are fed as input to CNN for classification model construction. We evaluate our propose encoding method to the most popular one-hot encoding method using the ChIP datasets. This paper is organized as follows. The next section presents our method of this study including datasets used, evaluation metric, and CNN structure. The Results section presents our evaluation results with discussion. The last section discusses the main findings and some issues for future studies.

II. Method

A. Ordinal encoding

In CNN, the raw input dataset have to be represented as vectors or matrices of numerical values. In this study, we evaluate the utility of using ordinal encoding method to encode each nucleotide letter and thus a DNA sequence as vector of numerical values. One of the problems with the one-hot encoding is the curse-of-dimensionality [12], in which for each input sequence of length l, the number of input values would be $4 \times l$. It causes long training time and only short sequence is computationally feasible for large-scale dataset. While the one-hot encoding has been claimed to represent the PFM [3] and has a clear meaning of interpretation, for machine learning, that is unnecessary useful since our aim is to learn the features associated with different class labels. In our method, to reduce the dimensions of every input sequence the nucleotides are encoded with numerical values. That is, A is represented by 0.25, C by 0.50, G by 0.75, and T by 1.00, respectively. For the unknown nucleotide N, its value is 0.00. It is difficulty to justify how those numbers are decided rather than heuristic. But our preliminary evaluation indicated that those are good choices (results not shown).

The advantage of ordinal encoding is its ability to reduce the memory consumption and thus speedup the CNN learning process. Moreover, a one-dimensional vector can be arbitrarily reshaped to a two-dimensional matrix. Suppose we have a row vector v of size $1 \times l$, where l is an even integer, that represents an input sequence. To transform the vector to a matrix of size $m \times n$, we simply reshape the vector v to $m \times n$ matrix. If l is not multiple of n, zero values are padded at the last row.

B. Datasets

The datasets from Pazar database [13] are chosen for our comparative evaluation. Pazar is an open-access and open-source database that stores the transcription factor and regulatory sequence annotation independently [14]. The datasets are experimentally validated TFBSs. Besides the datasets from Pazar, seven ChIP-seq datasets used in ENSPART [15] are also been selected (NRSF,FOXA1, CREB, FOXA2, OCT4, CTCF, STAT1).

The second group of datasets gathered is the transcription factors of mouse from Pazar. The details of the datasets are shown in Table I

The third group of datasets collected is the transcription factors of human from Pazar. The information of the datasets are shown in Table II

To avoid classes imbalance problem, same number of sequences from each dataset is sampled. Furthermore, the sequences are truncated by removing bases that are beyond 900 bp. In our preliminary study (results not shown), using longer sequence length gives marginal better accuracy rates but at the price of higher computational cost. Sequences that are shorter than 900 bp are padded with the vector $[0.25, 0.25, 0.25, 0.25]$ for the one-hot encoding, while 0.00s are added for the ordinal encoding.

Table III shows the number of samples from each dataset, number of sequences as training sets, validation sets, and testing sets.

C. CNN Architecture

The evaluation study compares three (3) sequence representation methods:(a) one-hot; (b) ordinal encoding with square matrix (Square); and (c) 1D vector. The CNN architectures used for the evaluation are shown in Table IV. The CNN labeled "3Layer" is using three layers CNN with square encoding method.

TABLE IV
CNN LAYERS SETUP USE IN THE SIMULATION.

	Onehot	Square	1D	3layer
Input layer	900×4	30×30	900×1	30×30
1st conv layer	12×4	6×6	36×1	6×6
1st activation function	ReLU	ReLU	ReLU	ReLU
1st max pooling	2×2	2×2	2×2	2×2
2nd conv layer	8×4	6×6	36×1	6×6
2nd activation function	ReLU	ReLU	ReLU	ReLU
2nd max pooling	2×2	2×2	2×2	2×2
3rd conv layer				4×4
3rd activation function				ReLU
3rd max pooling				2×2
Full connection layer	1024	1024	1024	1024
Dropout	0.5	0.5	0.5	0.5
Output layer	one-hot	one-hot	one-hot	one-hot

"Square" representation uses a 30×30 input layer and "1D" uses 900×1 as a one-dimensional input layer. In addition, we also uses a three-layers CNN with the 30×30 ordinal encoding. All the CNNs employed the dropout rate of 0.5. Dropout is applied before the output layer to reduce overfitting of training [16]. All the layers use ReLU activation function as recommended by [17].

The TensorFlow [18] is chosen as implementation of the CNN. The softmax activation function is used at the output layer. ADAM optimizer with the learning rate 10^{-4} is chosen as the adaptive learning method. The mini-batch of 200 is used in each epoch. Maximum epoch is set at 20000.

D. Evaluation Metric

Area under curves (AUC) [19] is used as the performance metric for the evaluation. Each CNN configuration, Basset,

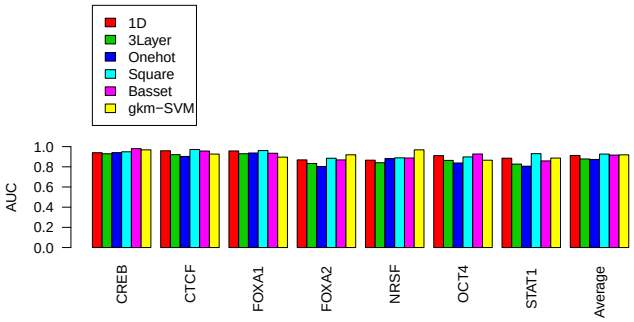

Fig. 1. Comparison of the CNN average AUCs using the 1D, one-hot, and square encoding . The AUC values are obtained from average of 5-fold cross-validation using the ENSPART datasets.

and gkm-SVM are tested with five testing sets and the average of the AUCs are collected for comparison. Basset and TensorFlow do produce AUCs as the final output. While for the gkm-SVM the AUC of its prediction has to be computed separately.

III. Results

Table V shows the average AUC rates from 5-fold cross-validation using the three sequence representations. It can be observed that the square encoding method performed better than one-hot encoding for all the datasets. This is a surprising result for the ChIP datasets since the one-hot has been the state-of-the-art encoding method for DNA sequences in most of the recent CNN works. It is also noted that the CNN with the square encoding performed better than gkm-SVM for 5 of the datasets. Basset, which uses the one-hot encoding method, performed only better than CNN with square encoding in 2 of the 7 datasets. The results indicate that the two layers CNN might not be the most optimal architecture for the one-hot encoding use. However, the 3 convolutionary-pooling layers used by Basset is not better than the square encoding with 2 layers. Another observation is that CNN-based methods performed better that SVM-based method in most of the datasets. This shows that CNN is more powerful in feature learning since it is able to learn the different abstraction of the features through its convolutionary-pooling layers. gkm-SVM employed k-mer feature for modeling. To give a clearer comparison, Figure 1 shows the graph of the average AUC values for all the datasets.

Table III shows the comparisons of AUC values between different encoding methods using CNN, Basset, and gkm-SVM. It is noted that for the mouse datasets, Basset performed better than other methods in 9 out of 10 of the datasets. However, it is noted that the square encoding AUC values are quite close to the Basset performance and significantly better than gkm-SVM for 6 out of the 10 datasets. We also noted that the architecture use by CNN might not be the most optimal because the one-hot encoding results should be quite close to the Basset since they are using the same representation. This highlights the difficulty of benchmarking CNN using different representations since their performances are also affected by the choices of architecture design and parameters being used. It is also surprising to see that the AUC values of CNN using the 1D sequence encoding are very close to Basset. Figure 2 illustrates the average AUC values for the mouse datasets.

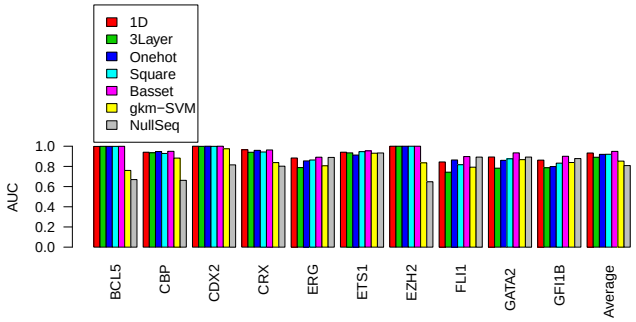

Fig. 2. Comparison of the CNN average AUCs using the 1D, one-hot, and square encoding. The AUC values are obtained from an average of 5-fold cross-validation using the mouse datasets.

On the human datasets (i.e. Table VII), the gkm-SVM performed the best in terms of average AUC values for 6 out of 10 of the datasets. Nevertheless, Basset obtained best average of (0.9105) of all the human datasets. Generally, the results are quite mixed for the human datasets since the best predictors for different datasets are distributed to all the methods used. However, clearly, the one-hot with CNN does not perform as good as other methods for these datasets. Figure 3 shows the average AUC values for all the compared methods.

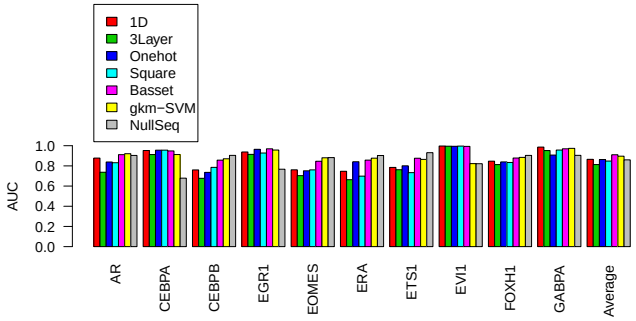

Fig. 3. Comparison of the CNN average AUCs using the 1D, one-hot, and square encoding. The AUC values are obtained from an average of 5-fold cross-validation using the human datasets.

IV. Discussion and Conclusion

This study investigated how a simple ordinal encoding method would perform in comparison with the state-of-the-art one-hot encoding method for DNA sequence representation. Specifically, the encoded sequences are meant for CNN learning. What are the desired properties of an encoding method for CNN to model effectively the DNA sequences enriched with motifs? This question is challenging because it would require domain experts with the understanding of the relevant

features that are useful for the prediction of various classes of motifs. For instance, for the ChIP sequences, it is found the frequencies of the short sequences (k-mers) are useful for the construction of classification model [20], [21]; while for enhancers it was identified that the short repeat are important for identification [22]. In addition, for the cis-regulatory modules (CRM), the existent of a set of motif signals clustered within a pre-defined region length in DNA sequences are useful signature for identification [22]. Other than that, for histone marks which are associated with active enhancers, our previous study found no clear sequence patterns that can be signals for their identification [23]. However, past studies have demonstrated that the k-mers are discriminating features for various histone marks [24], [25] and ChIP sequences [20]. The one-hot encoding and matrix representation method is not able to capture the frequency domain of the features mentioned.

Our method is termed direct encoding as opposed to indirect encoding [9]. The ordinal encoding is a direct encoding method which preserved the original order and position of each nucleotide in a DNA sequence. Even after the 1D vector is reshaped, the nucleotides ordering is still intact.

Our evaluation results using various datasets showed that the ordinal encoding with square matrix representation has good performance on ChIP datasets. Neverthelss, Basset, which is using the one-hot encoding has optimized architecture to achieve good performances. While that is the case, the results of the ordinal encoding with matrix representation are quite closed to Basset (one-hot) for the Mouse and Human datasets. The advantage of the ordinal encoding is its shorter input dimension and that can save training time significantly versus the one-hot encoding.

Table VIII shows the total running time for the three sets of datasets for training the CNN classifiers. The simulation ran on a Linux PC, with 8GB RAM, iCore7 (2.5GHz) processor and 4GB NVidia GeForce 930M graphic card. It is clearly seen that the Square encoding has saving of almost one third of the running time in comparison to one-hot and 1D encoding method. The 1D and one-hot encoding methods have the same length, therefore their total running time are quite similar. Because the implementation is different between Basset (using Torch7 and Lua) and TensorFlow (Python), it is hard to compare the running time for both tools. Our results show that Basset took 17m20s for the ENSPART datasets, while is 29m11s and 29m2s for the Human and Mouse datasets, respectively.

TABLE VIII
TOTAL CNN TRAINING TIME USED FOR ENSPART, MOUSE, AND HUMAN DATASETS USING TENSORFLOW.

	ENSPART	Mouse	Human
One-hot	89m38.2s	159m30.5s	123m31.6s
Square	24m19.2s	56m28.0s	58m45.0s
1D	87m46.9s	119m49.7s	121m7.6s
3Layer	23m13.3s	52m22.9s	52m20.8s

Our evaluation results also demonstrated that the 3Layer CNN did not perform well for the square representation. According to [26], CNN architecture decision is task-specific and therefore, possibly further tuning is necessary to find suitable architecture. This is the possible explanation that the 3Layer structure does not enhance the prediction performance.

The main conclusions from this study are summarized below:

- The nucleotides can be encoded as numerical values directly for CNN learning and its performance is not much different from the one-hot encoding.
- The reduced input dimension using the ordinal encoding allows CNN to learn faster in terms of computation time. However, the speed gained depends on the implementation of the CNN tool.
- Different encoding and representation methods may require customized tuning of CNN architecture and parameters used. Currently, there are still lacked of guidelines on choosing those configurations.

ACKNOWLEDGMENT

This work contributes as part of the Minister of Education Malaysia, Fundamental Research Grant Scheme-FRGS/SG03(01)/1134/2014(01).

REFERENCES

[1] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[3] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning," *Nature Biotechnology*, vol. 33, no. 8, pp. 831–838, jul 2015.

[4] C. Angermueller, H. Lee, W. Reik, and O. Stegle, "Accurate prediction of single-cell DNA methylation states using deep learning," *bioRxiv*, 2016.

[5] D. R. Kelley, J. Snoek, and J. Rinn, "Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks," *Genome Research*, 2016.

[6] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning-based sequence model." *Nature Methods*, vol. 12, no. 10, pp. 931–4, oct 2015.

[7] C. Angermueller, T. Pärnamaa, L. Parts, O. Stegle, F. Albert, S. Treusch, A. Shockley *et al.*, "Deep learning for computational biology." *Molecular systems biology*, vol. 12, no. 7, p. 878, jul 2016.

[8] G. D. Stormo, "DNA binding sites: representation and discovery," *Bioinformatics*, vol. 16, no. 1, pp. 16–23, jan 2000.

[9] N. K. Lee, D. Wang, and K. W. Tan, *Neural Networks Applications in Information Technology and Web Engineering*. Kuching, Sarawak: Borneo Publishing Co, 2005, ch. Protein classification using neural networks : A review, pp. 2–14.

[10] Q. Qin and J. Feng, "Imputation for transcription factor binding predictions based on deep learning," *PLOS Computational Biology*, vol. 13, no. 2, p. e1005403, feb 2017.

[11] U. Eser and L. S. Churchman, "FIDDLE: An integrative deep learning framework for functional genomic data inference," *bioRxiv*, 2016.

[12] P. Ng, "dna2vec: Consistent vector representations of variable-length k-mers," *arXiv:1701.06279 [q-bio.QM]*, 2017.

[13] E. Portales-Casamar, S. Kirov, and J. Lim, "PAZAR: a framework for collection and dissemination of cis-regulatory sequence annotation," *Genome Biology*, vol. 8, no. 10, p. R207, 2007.

[14] E. Portales-Casamar, D. Arenillas, J. Lim, M. I. Swanson, S. Jiang, A. McCallum, S. Kirov, and W. W. Wasserman, "The PAZAR database of gene regulatory information coupled to the ORCA toolkit for the study of regulatory sequences," *Nucleic Acids Research*, vol. 37, no. Database, pp. D54–D60, 2009.

[15] N. K. Lee, A. C. H. Choong, and N. Omar, "ENSPART: An Ensemble Framework Based on Data Partitioning for DNA Motif Analysis," in *2016 IEEE 16th International Conference on Bioinformatics and Bioengineering (BIBE)*. IEEE, oct 2016, pp. 87–94.

[16] N. Srivastava, G. Hinton, and A. Krizhevsky, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[18] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[19] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[20] M. Ghandi, D. Lee, M. Mohammad-Noori, M. A. Beer, T. Manolio, M. Maurano, R. Humbert *et al.*, "Enhanced Regulatory Sequence Prediction Using Gapped k-mer Features," *PLoS Computational Biology*, vol. 10, no. 7, p. e1003711, jul 2014.

[21] D. Lee, R. Karchin, and M. A. Beer, "Discriminative prediction of mammalian enhancers from dna sequence," *Genome Research*, vol. 21, no. 12, pp. 2167–2180, 2011.

[22] L. L. Colbran, L. Chen, and J. A. Capra, "Short dna sequence patterns accurately identify broadly active human enhancers," *BMC Genomics*, vol. 18, p. 536, Jul. 2017.

[23] N. K. Lee, P. K. Fong, and M. T. Abdullah, "Modelling complex features from histone modification signatures using genetic algorithm for the prediction of enhancer region." *Bio-medical materials and engineering*, vol. 24, no. 6, pp. 3807–3814, 2014.

[24] S. Nazeri, N. K. Lee, and M. Norwati, "Comparisons of enhancers associated marks prediction using k-mer feature," in *International Conference of IT in Asia (CITA15)*, Kuching, Sarawak, May 2015.

[25] D. U. Gorkin, D. Lee, X. Reed, C. Fletez-Brant, S. L. Bessling, S. K. Loftus, M. A. Beer, W. J. Pavan, and A. S. McCallion, "Integration of chip-seq and machine learning reveals enhancers and a predictive regulatory sequence vocabulary in melanocytes," *Genome Research*, vol. 22, no. 11, pp. 2290–2301, Jul. 2012.

[26] H. Zeng, M. D. Edwards, G. Liu, and D. K. Gifford, "Convolutional neural network architectures for predicting DNAprotein binding," *Bioinformatics*, vol. 32, no. 12, pp. i121–i127, jun 2016.

TABLE I
INFORMATION OF MOUSE DATASETS COLLECTED FROM PAZAR

Dataset	GEO	Number of sequences	Average length of sequence	File size
BCL6	GSE16723	1906	200.00	430.6k
CBP	GSE29362	6372	89.96	738.5k
CDX2	GSE14586	59097	500.00	31.1M
CRX	GSE20012	9661	182.47	2.0M
ERG	GSE22178	36167	398.00	15.3M
ETS1	GSE29362	2359	127.22	361.5k
EZH2	GSE18776	4502	1449.55	6.6M
FLI1	GSE22178	19601	398.00	8.3M
GATA2	GSE22178	9234	398.00	3.9M
GFI1B	GSE22178	8853	398.00	3.8M

TABLE II
INFORMATION OF HUMAN DATASETS COLLECTED FROM PAZAR

Dataset	GEO	Number of sequences	Average length of sequence	File size
AR	GSE28126	10363	520.62	5.7M
CEBPA	GSE29195	1644	107.71	219.7k
CEBPB	GSE31939	17949	463.88	8.8M
EGR1	GSE21665	35354	1114.39	40.3M
EOMES	GSE26097	61234	623.51	39.8M
ERA	GSE25021	48007	324.12	16.8M
ETS1	GSE17954	19420	562.58	11.4M
EVI1	GSE25210	38636	200.00	8.7M
FOXH1	GSE29422	29292	250.88	8.1M
GABPA	GSE24933	15740	98.40	2.0M

TABLE III
NUMBER OF SAMPLES FOR EACH DATASET OF THE THREE CNN LEARNING GROUPS

Group	Human (from ENSPART)	Mouse (Pazar)	Human (Pazar)
Number of datasets	7	10	10
Datasets	CREB, CTCF, FOXA1, FOXA2, NRSF, OCT4, STAT1	BCL6, CBP, CDX2, CRX, ERG, ETS1, EZH2, FLI1, GATA2, GFI1B	AR, CEBPA, CEBPB, EGR1, EOMES, ERA, ETS1, EVI1, FOXH1, GABPA
Number of samples each dataset	1657	1906	1644
Number of training sets	8119	13342	11508
Number of validation sets	1160	1906	1644
Number of testing sets	2320	3812	3288
Total number of sequences	11599	19060	16440

TABLE V
COMPARISON OF THE AUCS OF 1D, 3LAYER, ONEHOT, AND SQUARE OF THE AVERAGE OF 5-FOLD AUCS WITH BASSET AND GKM-SVM ON ENSPART DATASETS

ENSPART	1D	3Layer	Onehot	Square	Basset	gkm-SVM
CREB	0.9400	0.9295	0.9408	0.9488	**0.9801**	0.9681
CTCF	0.9591	0.9206	0.9028	**0.9717**	0.9563	0.9262
FOXA1	0.9572	0.9296	0.9369	**0.9609**	0.9344	0.8958
FOXA2	0.8688	0.8329	0.8030	0.8848	0.8684	**0.9190**
NRSF	0.8657	0.8406	0.8820	**0.8889**	0.8871	**0.9681**
OCT4	0.9110	0.8636	0.8378	0.8979	**0.9270**	0.8654
STAT1	0.8854	0.8268	0.8061	**0.9302**	0.8585	0.8866
Average	0.9124	0.8777	0.8728	**0.9262**	0.9160	0.9185

TABLE VI

COMPARISON OF THE AVERAGE AUCS USING DIFFERENT SEQUENCE ENCODING METHODS WITH CNN VERSUS BASSET AND GKM-SVM ON THE MOUSE DATASETS

						gkm-SVM*	
Mouse	1D	3Layer	Onehot	Square	Basset	Complement	NullSeq
BCL5	0.9980	**0.9990**	0.9988	0.9985	**0.9991**	0.7602	0.6695
CBP	0.9406	0.9351	**0.9473**	0.9284	**0.9496**	0.8824	0.6621
CDX2	**1.0000**	0.9989	**1.0000**	0.9989	**1.0000**	0.9747	0.8152
CRX	**0.9651**	0.9403	0.9592	0.9429	0.9630	0.8377	0.8022
ERG	0.8834	0.7883	0.8546	0.8635	**0.8913**	0.8062	0.8892
ETS1	0.9408	0.9343	0.9126	**0.9461**	**0.9542**	0.9308	0.9332
EZH2	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.8354	0.6478
FLI1	0.8442	0.7430	0.8637	0.8168	**0.8974**	0.7923	0.8922
GATA2	0.8932	0.7826	0.8603	0.8755	**0.9342**	0.8670	0.8924
GFI1B	0.8623	0.7866	0.7986	0.8323	**0.9004**	0.8387	0.8774
Average	0.9328	0.8908	0.9195	0.9203	**0.9489**	0.8525	0.8081

* The "complement" uses the complement datasets as the negative datasets; while "NulSeq" uses gkm-SVM null sequence function to generate the negative sequences.

TABLE VII

COMPARISON OF THE AVERAGE AUCS USING DIFFERENT SEQUENCE ENCODING METHODS WITH CNN VERSUS BASSET AND GKM-SVM ON HUMAN DATASETS

						gkm-SVM	
Mouse	1D	3Layer	Onehot	Square	Basset	Complement	NullSeq
AR	0.8775	0.7372	0.8385	0.8314	0.9114	**0.9207**	0.9035
CEBPA	0.9525	0.9123	0.9558	**0.9560**	0.9482	0.9124	0.6785
CEBPB	0.7601	0.6771	0.7357	0.7853	0.8569	0.8705	**0.9041**
EGR1	0.9378	0.9142	0.9642	0.9275	**0.9693**	0.9570	0.7670
EOMES	0.7613	0.7035	0.7515	0.7605	0.8459	0.8803	**0.8821**
ERA	0.7466	0.6633	0.8413	0.6984	0.8572	0.8765	**0.9028**
ETS1	0.7849	0.7622	0.8009	0.7332	0.8758	0.8644	**0.9309**
EVI1	**0.9968**	0.9951	0.9943	0.9962	0.9933	0.8224	0.8217
FOXH1	0.8468	0.8127	0.8394	0.8339	0.8779	0.8843	**0.9039**
GABPA	**0.9863**	0.9519	0.9073	0.9569	0.9694	0.9737	0.9045
Average	0.8651	0.8130	0.8629	0.8479	**0.9105**	0.8962	0.8599