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

Allen Chieng Hoon Choong
Faculty of Cognitive Sciences and Human Development
Universiti Malaysia Sarawak
Kota Samarahan, Malaysia
allen.choong@gmail.com

*Nung Kion Lee
Faculty of Cognitive Sciences and Human Development
Universiti Malaysia Sarawak
Kota Samarahan, Malaysia
nklee@unimas.my

***Abstract*—Convolutionary neural network (CNN) is a popular choice for supervised DNA motif prediction due to its excellent performances. To employ CNN, the input DNA sequences are required to be encoded as numerical values and represented as either vectors or multi-dimensional matrices. This paper evaluated a simple and more compact ordinal encoding method versus the popular one-hot encoding for DNA sequences. We compared 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 method but with significant reduction in training time. In addition, the one-hot encoding performances were rather consistent across various datasets but would require suitable CNN configuration to perform well. The ordinal encoding with matrix representation performed best in some of the evaluated datasets. This study implied that the performances of CNN for DNA motif discovery depends on the suitable design of the sequence encoding and representation. The good performances of the ordinal encoding method demonstrates that there are still rooms for improvement for the one-hot encoding method.**

***Index Terms*—DNA sequence encoding, convolutionary neural networks, motif discovery**

I. Introduction

CNN (Convolutional Neural Network) [1], [2] is currently one of the most widely used deep learning methods in machine learning due to its powerful modelling capability on complex and large-scale datasets. Recently, CNN has been widely used for learning DNA sequence datasets related to regulatory regions and other functional landmarks [3]–[6]. The advantage of CNN is its learning can be performed without the need of engineered features. The intrinsic features in the raw dataset are learned through the many layers 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 are called filters. A filter convolves the inputs from the previous layer to produce a reduced sample. It only connects to a patch of the previous layer, which is named as "receptive field". Moreover, all neurons in the filters detect the same features of the previous layer but at different map locations. Different filters might detect different types of features [7]. In a DNA dataset, the features might represent different motifs enriched in the input DNA sequences. In addition, [7] stated that the exact locations and frequency of a feature 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 greatly reduced. After the last pooling layer, it has a fully connected multi-layers perceptron neural networks.

CNN is designed to effectively models multi-dimensional 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 on DNA datasets, existing works typically encode the nucleotides in DNA sequences by using the one-hot method [3], [5], [6]. That is, each nucleotide is encoded with a binary vector of four bits with one of them 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 encoding method draws similarity to the Position Frequency Matrix [8]. In which, the values in a vector are considered as the probability of finding the four bases at a certain position in a DNA sequence. Once encoded, 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 were employed 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. They preserved the original order the bases appeared in a biological sequence after the encoding. While the indirect methods engineered a fixed number of features (numerical values) 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.

This study is supported by the Minister of Education Malaysia, Fundamental Research Grant Scheme-FRGS/SG03(01)/1134/2014(01)

978-1-5386-0765-7/17/$31.00 2017 IEEE, *Corresponding author

In recent years, CNN and other deep learning techniques are gaining popularity in supervised DNA motif prediction because of their excellent performances. For example, in DeepBind, it achieved an average of 0.85 AUC for evaluation of 137 transcription factors, while MEME-Chip achieved only 0.82. Some of the recent works of using deep learning neural networks 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 proposed a simple method to transform DNA sequences into matrix representation. The matrices are fed as input to CNN for classification model construction. We evaluated our proposed encoding method to the most popular one-hot encoding method using the three sets of sequence datasets. The next section presents the method of this study including datasets used, evaluation metric, and CNN structure. The Results section presents the 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 input examples are assumed to be represented as vectors or matrices of numerical values. In this study, we evaluated the utility of using ordinal encoding method to encode each nucleotide letter and thus a DNA sequence as a 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 caused long training time and only short sequence was computationally feasible for large-scale dataset. For examples, in DeepBind the sequence length is limited to 14101bp long, while Basset uses 600bp input sequences. 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 this study, to reduce the dimensions of input sequence the nucleotides were 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 with heuristic. But our preliminary evaluation indicated that those were good choices (results not shown).

The advantages of ordinal encoding is it requires less computer memory resource and computations. 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 a 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

We have prepared three sets of datasets to compare the ability of ordinal and one-hot encoding method to learn the motif features enriched in the DNA sequences. The datasets from Pazar database [13] were chosen for the comparative evaluation. Pazar is an open-access and open-source database that stores the transcription factor and regulatory sequence annotation independently [13]. The datasets were experimentally validated TFBSs. A set of mouse TF datasets were gathered from Pazar. The details of the datasets are shown in Table I. Another set of datasets collected from Pazar is the transcription factors of human. The information of the datasets are shown in Table II

Besides the datasets from Pazar, seven ChIP-seq datasets used in ENSPART [14] were also been selected (NRSF,FOXA1, CREB, FOXA2, OCT4, CTCF, STAT1).

To avoid classes imbalance problem, same number of sequences from each dataset was sampled. Furthermore, the sequences were truncated by removing bases that are beyond 900 bp. In the preliminary study (results not shown), using longer sequence lengths only gives marginal improvement on the accuracy rates but at the price of higher computational cost. Sequences that were shorter than 900 bp were padded with the vector $[0.25, 0.25, 0.25, 0.25]$ for the one-hot encoding, while 0.00s were 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 compared three (3) sequence representation using the two encoding methods:(a) one-hot with matrix representation; (b) ordinal encoding with square matrix (Square) representation; and (c) ordinal with 1-dimensional vector representation (1D). The CNN architectures employed in the simulation were shown in Table IV. The CNN labeled "3Layer" used three layers CNN with ordinal encoding and square matrix representation.

TABLE IV
CNN'S LAYERS SETUP USED IN THE SIMULATIONS.

	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
Fully connected 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 used a 30×30 input layer and "1D" used 900×1 as a one-dimensional input layer. In addition, we also uses a three-layers CNN with the 30×30 square matrix

TABLE I
INFORMATION OF MOUSE DATASETS COLLECTED FROM PAZAR

Dataset	GEO	Number of sequences	Average length of sequences	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 sequences	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 TRAINING, VALIDATION, AND TESTING FOR EACH TF DATASET.

Group	ChIP-Seq	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 in 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

with ordinal encoding. All the CNNs employed the dropout rate of 0.5. Dropout was applied before the output layer to reduce overfitting of training [15]. All the layers used ReLU activation function as recommended by [16].

The TensorFlow [17] was chosen as implementation of the CNN. The softmax activation function was 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 was used in each epoch. Maximum epoch was set at 20000.

D. Evaluation Metric

The Area under curves (AUC) [18] was used as the performance metric for the evaluation. We compared the one-hot and ordinal encoding methods using the tensor-flow implementation of the CNN. In addition, we also compared with Basset and gkm-SVM. A five-fold cross-validation was used for all the datasets. Basset and gkm-SVM did output AUC values in their output. While for the CNN implemented by the TensorFlow, the AUC values were computed by our own implementation.

III. RESULTS

Table V shows the comparison of average AUC rates from 5-fold cross-validation using the ChIP-seq datasets. It was observed that the square encoding method performed better than one-hot encoding for all the datasets. That is a surprising result since the one-hot has been the state-of-the-art encoding method for DNA sequences in most of the recent CNN works. It was also noted that the CNN with the square matrix performed better than gkm-SVM for 4 out of the 7 datasets. Other than that, Basset 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 was not better than the square encoding with 2 layers. Another observation was that CNN-based methods performed better that SVM in most of the evaluated datasets. This shows that CNN was more powerful in feature learning since it was able to learn the different abstraction of the features through its convolutionary-pooling

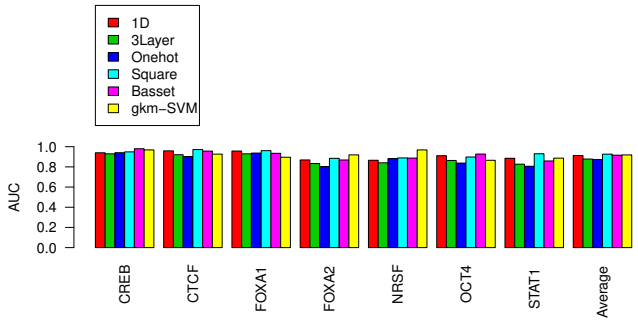

Fig. 1. Comparisons of the CNN average AUCs using the 1D, one-hot, and square encoding . The AUC values are obtained from average of 5-fold cross-validation using the ChIP-seq datasets.

layers. gkm-SVM employed k-mer feature for the modeling. To give a clearer comparison, Fig. 1 shows the graph of the average AUC values for all the datasets.

Table III shows the comparisons of AUC values of CNN, Basset, and gkm-SVM using ordinal or one-hot encoding for the mouse datasets. It was noted that Basset performed better than other methods (gkm-SVM, CNN with ordinal encoding) in 9 out of 10 of the datasets. It obtained an average AUC value of 0.949 for ten of the datasets. However, it was noted that the CNN with square encoding produced AUC values which are quite close to the Basset's performance (average AUC value on 10 datasets was 0.92) and significantly better than gkm-SVM for 6 out of the 10 datasets. We also noted that the architecture used by the CNN might not be the most optimal because the one-hot encoding results should be quite close to the Basset since they were using the same encoding and representation. This highlights the difficulty of benchmarking CNN using different representations since their performances were also affected by the architecture and parameter values choices. It is also surprising to see that the AUC values of CNN using the 1D sequence encoding were very close to Basset. Fig. 2 illustrates the average AUC values for the mouse datasets.

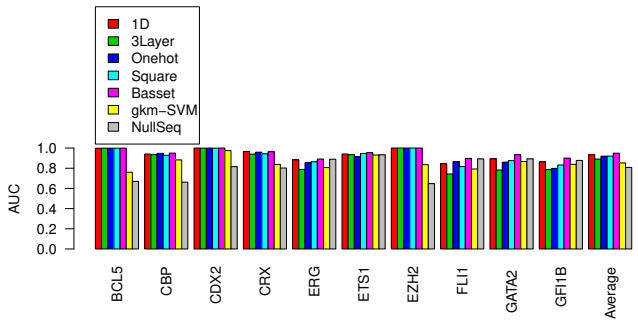

Fig. 2. Comparison of the CNN average AUCs using the 1D, one-hot, and square encoding. The AUC values are obtained from an average of 5-fold cross-validation using the mouse datasets.

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. However, Basset obtained best average of (0.91) of all the human datasets. Generally, the results are quite mixed for the human datasets since the best predictors for different datasets were distributed to all the methods used. However, the one-hot with CNN did not perform as good as other methods using the the mouse datasets. Fig. 3 shows the average AUC values for all the compared methods.

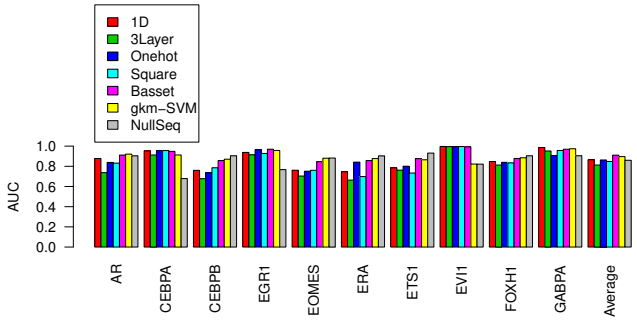

Fig. 3. Comparison of the CNN average AUCs using the 1D, one-hot, and square encoding. The AUC values are obtained from an average of 5-fold cross-validation using the human datasets.

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 sequences. Specifically, the encoded sequences were meant for CNN learning which expect the input examples were in vector or matrix format. What are the desired properties of an encoding method for CNN to model effectively the DNA sequences enriched with motifs? That question is challenging because it would require domain experts with the understanding of the relevant features for the prediction of various classes of motifs. For instance, it was discovered that the frequencies of the short sequences (k-mers) in ChIP sequences were useful for the construction of classification model [19], [20]; while for enhancers it was found that the short repeats were important for their location identification [21]. In addition, for predicting 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 their identification [21]. Other than that, for histone marks which were associated with active enhancers, previous studies found no clear sequence patterns that can be signals for their identification [22]. However, past studies have demonstrated that the k-mers were discriminating features for various histone marks [23], [24] and ChIP sequences [19]. The one-hot encoding method which assumed the features in DNA sequences can be detected by filters (i.e. PFMs) in the convolutionary layers may not be able to detect those features mentioned. Therefore, one-hot encoding may not detect distinctive features in different means of DNA sequences.

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 CHIP-SEQ DATASETS

ENSPART	1D	3Layer	One-hot	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	One-hot	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	One-hot	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

Our method was termed direct encoding as opposed to indirect encoding [9]. The ordinal encoding was a direct encoding method which preserved the original order and position of each nucleotide in a DNA sequence. Even after the 1D vector was reshaped, the nucleotides ordering was still preserved in the resulted matrix but in a different locations (e.g. rows). The CNN would still be able to detect those features because of the filters scanning.

Our evaluation results using various datasets showed that the ordinal encoding with square matrix representation has good performance on ChIP datasets. Nevertheless, Basset, which is using the one-hot encoding, has optimized architecture to achieve good performances across all the datasets. While that is the case, the results of the ordinal encoding with matrix representation are quite close to Basset (one-hot) for the Mouse and Human datasets. The advantage of the ordinal encoding is its shorter input dimension which reduced the computations of filters scanning in the convolutionary layer. Likewise, the kernels in the pooling layer would require less computations.

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 saved almost one third of the running time in comparison to one-hot and 1D encoding method.

The 1D and one-hot encoding methods have almost the same running time because the number of computations required at both pooling and convolutionary layers grows with the length of matrix (vector). Since the implementation is different in terms of programming language and data structure between Basset (using Torch7 and Lua) and TensorFlow (Python), it is hard to compare the running time for both tools. Basset took 17m20s for the ChIP-seq datasets, while it took 29m11s and 29m2s for the Human and Mouse datasets, respectively.

TABLE VIII
TOTAL CNN RUNNING TIME USED FOR CHIP-SEQ, MOUSE, AND HUMAN DATASETS USING CNN IMPLEMENTED BY TENSORFLOW.

	ChIP-Seq	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 [25], CNN architecture decision is task-specific and therefore, further tuning may necessary to find the suitable architecture configuration.

The main conclusions from this study are:

- The ordinal encoding method for DNA sequences performed comparably to the one-hot encoding. While the one-hot encoding has better interpretation, it has not much computational and performance advantages over the simple ordinal encoding. This study did not investigate the convergence behavior of both encoding methods.
- The reduced input dimensions using the ordinal encoding allowed CNN to learn the input examples with reduced time. However, the speedup gained is depended on the implementation of the CNN tool.
- Different encoding and representation methods may require customize tuning of CNN architecture and parameters used. Currently, there is still lack of guidelines on choosing those parameters for learning DNA sequences.

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, 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, 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, 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, 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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/

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

[19] 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, 2014.

[20] 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.

[21] 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, 2017.

[22] 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.

[23] 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.

[24] 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, 2012.

[25] 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, 2016.