Comparative analysis and prediction of nucleosome positioning using integrative feature representation and machine learning algorithms

Guo-Sheng Han¹,²*, Qi Li¹,²† and Ying Li¹,²

From 19th International Conference on Bioinformatics 2020 (InCoB2020)
Virtual. 25-29 November 2020

*Correspondence: hangs@xtu.edu.cn
†Guo-Sheng Han and Qi Li have contributed equally
¹ Department of Mathematics and Computational Science, Xiangtan University, Xiangtan 411105, Hunan, China
Full list of author information is available at the end of the article

Abstract

Background: Nucleosome plays an important role in the process of genome expression, DNA replication, DNA repair and transcription. Therefore, the research of nucleosome positioning has invariably received extensive attention. Considering the diversity of DNA sequence representation methods, we tried to integrate multiple features to analyze its effect in the process of nucleosome positioning analysis. This process can also deepen our understanding of the theoretical analysis of nucleosome positioning.

Results: Here, we not only used frequency chaos game representation (FCGR) to construct DNA sequence features, but also integrated it with other features and adopted the principal component analysis (PCA) algorithm. Simultaneously, support vector machine (SVM), extreme learning machine (ELM), extreme gradient boosting (XGBoost), multilayer perceptron (MLP) and convolutional neural networks (CNN) are used as predictors for nucleosome positioning prediction analysis, respectively. The integrated feature vector prediction quality is significantly superior to a single feature. After using principal component analysis (PCA) to reduce the feature dimension, the prediction quality of H. sapiens dataset has been significantly improved.

Conclusions: Comparative analysis and prediction on H. sapiens, C. elegans, D. melanogaster and S. cerevisiae datasets, demonstrate that the application of FCGR to nucleosome positioning is feasible, and we also found that integrative feature representation would be better.

Keywords: Nucleosome classification, Frequency chaos game representation, Support vector machine, Extreme learning machine, Extreme gradient boosting, Convolutional neural networks

© The Author(s) 2021. Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated in a credit line to the data.
Background

The nucleosome is the basic structural unit of eukaryotic chromatin. It is formed by the combination of histones and DNA. The core is an octamer formed by two copies of each histones H2A, H2B, H3 and H4, DNA is wound around it about 1.65 turns. Among them, the DNA wrapped around the octamer is called core DNA, which is 147 base pairs in length; the DNA sequence that connects two adjacent nucleosomes is called linker DNA, which ranges from 20 to 60 base pairs [1]. In eukaryotic cells, nucleosomes play a crucial role in the process of genome expression, DNA replication, DNA repair and transcription [2–6]. In addition, studies have demonstrated that abnormal histone modifications in the nucleosome structure are directly related to diseases such as tumors [7] and lupus erythematosus [8]. Therefore, the mechanism of nucleosome positioning in DNA sequence has an extremely important research value, which is also one of the hot spots in current epigenetics research.

The precise position of the nucleosome on the DNA sequence in the whole genome is called nucleosome positioning. Early experiments mainly used micrococcal nuclease to process chromatin to achieve nucleosome positioning [9]. In recent years, benefiting from the development and application of high-throughput experimental techniques, such as chromatin immunoprecipitation-chip (ChIP-chip), chromatin immunoprecipitation sequencing (ChIP-Seq), many breakthroughs have been made in nucleosome positioning experiments. The nucleosome positioning maps of different species such as Saccharomyces cerevisiae [10, 11], Homo sapiens [12], Caenorhabditis elegans [13], Drosophila melanogaster [14], etc. have been obtained, which provides a large amount of data basis for researchers to carry out theoretical research and prediction.

Much of the research in nucleosome positioning is based on DNA sequence analysis [15, 16]. The DNA sequence consists of four nucleotides: A, T, C and G. Studies have shown that the affinity between genomic DNA sequences and histones is clearly dependent on sequence order, which indicates that the DNA sequence order does affect the position of nucleosome formation. Although some provide the support that nucleosome positioning is affected by multiple factors such as DNA sequence, ATP-dependent nucleosome remodeling enzymes and transcription factors [17, 18]. Many researchers used sequence analysis methods to express nucleosome DNA sequence characteristics and then performed nucleosome positioning and recognition.

In the past decade, with the popularity of machine learning algorithms, a multitude of computational models based on DNA sequence information have been proposed. Chen et al. proposed the "iNuc-Physchem" nucleosome prediction model using 12 physico-chemical features of DNA, which identified the core DNA and linker DNA of the yeast genome nucleosome [19]. Later, the research group also established a biophysical model based on the deformation energy of DNA sequences to predict the sequence of nucleosomes [20]. Guo et al. used pseudo k-tuple nucleotide composition to successfully express the feature vector of the DNA sequence, and used the support vector machine (SVM) classifier to train H. sapiens, C. elegans and D. melanogaster [21]. 3LS model used similar methods and combined the distribution of different numbers of nucleotide combinations in the sequence to further improve the prediction accuracy [22]. ZCMM model based on the Z-curve (z-curve) theory and the position weight matrix (PWM), the prediction performance is excellent on D. melanogaster [23].
Deep learning is also applied to nucleosome positioning and achieved good prediction quality. These deep learning models all used one-hot encoding. Gangi et al. [24] constructed a deep learning model that integrates convolutional layers and long short-term memory networks. LeNup model added the Inception module and gated convolutional network to the convolutional neural network to improve the nucleosome positioning [25].

In this work, we firstly will use frequency chaos game representation to construct DNA sequence features. This feature representation method has not been used in nucleosome positioning before. Secondly, we also integrated FCGR with other feature vectors and adopted the principal component analysis (PCA) algorithm to achieve the feature dimensionality reduction. Finally, various machine learning algorithms such as support vector machine (SVM), extreme learning machine (ELM), extreme gradient boosting (XGBoost), multi-layer perceptron (MLP), and convolutional neural networks (CNN) will be used to perform comparative analysis and prediction of nucleosome positioning.

**Results**

**Rule of performance evaluation**

Cross validation is a statistical analysis method used to validate the model. The basic idea is to divide the original data into a training set and a test set. First, use the training set to train the model, and then use the test set to test the classification or prediction performance of the obtained model. In this work, we used K-fold cross-validation to evaluate the performance of the predictor through four parameters: sensitivity ($S_n$), specificity ($S_p$), accuracy (ACC), and Mathew’s correlation coefficient (MCC). The specific definition are as follows:

\[
\begin{align*}
S_n &= \frac{TP}{TP+FN} \\
S_p &= \frac{TN}{TN+FP} \\
ACC &= \frac{TP+TN}{TP+TN+FP+FN} \\
MCC &= \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FN) \times (TP+FP) \times (TN+FN) \times (TN+FP)}}
\end{align*}
\]

where TP, TN, FP and FN are the numbers of true positives, true negatives, false positives and false negatives, respectively [25]. $S_n$ is the true positive rate. When $S_n=1$, it means that all core DNA of nucleosomes have been correctly predicted. $S_p$ is true negative rate. When $S_p=1$, it means that all linker DNAs are correctly predicted. ACC reflects the ratio of the number of correctly predicted samples of each category to the total sample. MCC comprehensively evaluates the prediction results. $MCC \in [-1,1]$. $MCC = -1$ means that the correlation is completely opposite. $MCC = 1$ means that the prediction result is completely correlated with the true category. $MCC = 0$ means that the prediction is completely random.

Receiver operating characteristic curve (ROC curve) and area under curve (AUC) are often used to evaluate the pros and cons of a binary classifier. Area under curve (AUC) is the area under the Roc curve, usually between 0.5 and 1. As a value, AUC can be used to evaluate the quality of the classifier more intuitively. The larger the AUC value, the better. Taking into account the length of the paper, this paper only calculates the AUC value and does not draw the ROC curve one by one.
Performance of predictors

According to the characteristics of FCGR described above, the different values of K nucleotide will affect the feature expression of the DNA sequence [26]. A large K value means a high feature dimension. And generally, high-dimensional features are relatively sparse, and the fitting quality may not be outstanding. Obviously, choosing an appropriate K value will have a greater impact on the classification effect of each classifier. Some studies have combined DNA sequence features [22, 23, 27, 28]. Similarly, FCGR can also use different combinations of K nucleotide values as feature vectors.

Feasibility of FCGR

In this work, we flatten the FCGR matrix into a normalized vector (1-D) corresponding to the frequency of K nucleotides as the input of SVM and ELM [27]. The input of MLP and CNN models are not only single-channel FCGR images (2D) [26, 27], but also multiple K-value images, the image size is 64 × 64. For the input of multi-K-value images, we leveraged multiple channels to feed in the combination of K values when training the model, and used simple averaging to calculate the final prediction result. To find the appropriate value of K or combination, we use 10-fold cross-validation. Figure 1 shows the classification accuracy of each classifier with different K values and combinations.

For SVM, the accuracy of H. sapiens, C. elegans reaches its peak with K = 1, 2 and 4; the accuracy of D. melanogaster was the highest with K = 2 and 4. For ELM, the accuracy of D. melanogaster reaches an peak when K = 2; the accuracy of H. sapiens reaches its peak when K = 2 and 4; the classification accuracy of C. elegans is best with K = 1, 2 and 4 like using SVM.

For MLP, the accuracy of H. sapiens and D. melanogaster reaches its peak with K = 3, 4 and 5; the classification accuracy of C. elegans is best with K = 3 and 4. For CNN, H. sapiens have the best classification quality when using the FCGR image with K = 4; the accuracy of C. elegans reaches its peak with K = 4 and 5; the accuracy of D. melanogaster

![Fig. 1](image_url) The histogram (a-d) shows the accuracy of using SVM, ELM, MLP and CNN with K = 1, 2, 3, 4, 5 or combinations.
reaches its peak with $K = 3, 4$ and $5$. Table 1 clearly shows the best prediction results for four species via 10-fold cross-validation.

For S. cerevisiae dataset, we used SVM, ELM and MLP to achieve $S_n = S_p = ACC = MCC = AUC = 1$ via 10-fold cross-validation when $K = 3$ or $4$. There may be room for improvement in the predicted quality of the other three datasets.

**Comparison of the results with integrative features**

In addition, we also integrated FCGR with other feature representations [29–32], such as DAC, TAC, DACC, TACC, PC-PseDNC, PC-PseTNC, and input them into SVM and ELM. Besides, we added the extreme gradient boosting (XGBoost) algorithm. The comparative analysis results are shown in Tables 2, 3 and 4 respectively.

From the results in Tables 2, 3 and 4, the combination of FCGR and DAC as feature vectors have a greater prediction quality. XGBoost performance is relatively stable, and each prediction results have little difference, especially for inputting high-dimensional features. However, after some high-dimensional feature vectors are input into SVM and ELM, the prediction results are relatively poor. It shows that XGBoost is more suitable for processing high-dimensional features.

**Comparison of the results with dimensionality reduction**

Considering the high dimensionality of the integrative feature vector, it is possible that high-dimensional feature vectors would bring the curse of dimensionality, which leads to overfitting of the prediction result. Therefore, we also adopted the principal component analysis (PCA) algorithm [33] to achieve feature dimensionality reduction. Then, the feature vector after dimensionality reduction is input into SVM, ELM and XGBoost respectively. In the process of using PCA to dimensionality reduction, the cumulative contribution rate of the retained principal components will directly affect the

### Table 1

| Species       | Method  | K              | ACC   | $S_n$ | $S_p$ | MCC   | AUC   |
|---------------|---------|----------------|-------|-------|-------|-------|-------|
| H. sapiens    | FCGR-SVM| 1 + 2 + 4      | 0.8708| 0.8980| 0.8439| 0.7432| 0.9300|
|               | FCGR-ELM| 2 + 4          | 0.8332| 0.8773| 0.7896| 0.6695| 0.8969|
|               | FCGR-MLP| 3 + 4 + 5      | 0.8565| 0.8768| 0.8365| 0.7144| 0.9186|
|               | FCGR-CNN| 4              | 0.8585| 0.8746| 0.8426| 0.7185| 0.9214|
| C. elegans    | FCGR-SVM| 1 + 2 + 4      | 0.8603| 0.8948| 0.8263| 0.7229| 0.9295|
|               | FCGR-ELM| 1 + 2 + 4      | 0.8754| 0.8944| 0.8566| 0.7515| 0.9421|
|               | FCGR-MLP| 3 + 4          | 0.8537| 0.8613| 0.8462| 0.7092| 0.9225|
|               | FCGR-CNN| 4 + 5          | 0.8495| 0.8839| 0.8156| 0.702 | 0.9181|
| D. melanogaster| FCGR-SVM| 2 + 4         | 0.8113| 0.7831| 0.8400| 0.6241| 0.8791|
|               | FCGR-ELM| 2 + 4         | 0.7910| 0.7648| 0.8175| 0.5833| 0.8595|
|               | FCGR-MLP| 3 + 4 + 5      | 0.8117| 0.8000| 0.8235| 0.6238| 0.8848|
|               | FCGR-CNN| 3 + 4 + 5      | 0.8108| 0.8014| 0.8204| 0.6228| 0.8854|
| S. cerevisiae | FCGR-SVM| 4              | 1     | 1     | 1     | 1     | 1     |
|               | FCGR-ELM| 3 or 4        | 1     | 1     | 1     | 1     | 1     |
|               | FCGR-MLP| 4              | 1     | 1     | 1     | 1     | 1     |
|               | FCGR-CNN| 4              | 0.9997| 1     | 0.9994| 0.9995| 1     |

*Best values are in bold*
dimensionality reduction effect. Therefore, we calculated the accuracy of 95%, 93%, 90%, 88% and 85% of the contribution rate of the retained principal components respectively. Figures 2, 3 and 4 shows the classification accuracy of each classifier with different contributing rate of principal component. And the results of the optimal contribution rate of the principal components corresponding to each predictor are shown in Tables 5, 6 and 7 respectively.

From Tables 5, 6 and 7, We have noticed that the prediction quality has been improved after dimensionality reduction through PCA for H. sapiens. It is increased by 4.57%, 3.12%, 6.00%, 9.03%, 3.56% in ACC, Sn, Sp, MCC and AUC when we combined FCGR vectors and TAC for using with XGBoost. However, the prediction quality has been not improved significantly for C. elegans. Especially when ELM was used, its prediction quality decreased slightly. For D. melanogaster, similarly, there is no significant improvement.

**Comparison with other algorithms**

To verify the effectiveness of our method, we compared the prediction results of the optimal performing predictors in Tables 1, 2, 3 and 4 with other models using the same
datasets. DLNN-5 [24] is a deep learning model with a convolution kernel size of 5, and ZCMM [23] is based on SVM. Tables 8, 9, 10 and 11 shows that our methods perform prominently on H. sapiens and S. cerevisiae datasets. For S. cerevisiae dataset, we used SVM, ELM and MLP to achieve $Sn = Sp = ACC = MCC = AUC = 1$ via 10-fold cross-validation when $K = 3$ or 4. Compared with the model that based on DNA deformation energy in the original paper [20], the prediction performance has been obviously lifted. For H. sapiens, combined FCGR vectors and TAC for using with XGBoost is higher than ZCMM in ACC, $Sn$, $Sp$, MCC, AUC by 10.87%, 15.58%, 5.23%, 21.25%, 8.81%, respectively. The performance of CNN is slightly better than ZCMM and DLNN-5. For C. elegans, compared with ZCMM, we use ELM to increase the evaluation indicators by 2.20%, 10.64%, 1.56%, 13.15%, 3.01% when combined FCGR vectors with $K = 1, 2$ and 4.

### Table 3

| Method | Feature | parameter | ACC  | Sn   | Sp   | MCC  | AUC  |
|--------|---------|-----------|------|------|------|------|------|
| SVM    | FCGR + DAC | $K = 4, \text{lag} = 2$ | 0.8574 | 0.8863 | 0.8290 | 0.7164 | 0.9283 |
|        | FCGR + TAC | $K = 4, \text{lag} = 2$ | 0.8561 | 0.8824 | 0.8302 | 0.7137 | 0.9272 |
|        | FCGR + DACC | $K = 4, \text{lag} = 2$ | 0.8471 | 0.8777 | 0.8171 | 0.6961 | 0.9122 |
|        | FCGR + TACC | $K = 4, \text{lag} = 2$ | 0.8470 | 0.8641 | 0.8301 | 0.6949 | 0.9179 |
|        | FCGR + PCPseDNC | $K = 4, \lambda = 8, w = 0.5$ | 0.8576 | **0.8921** | 0.8236 | 0.7176 | 0.9275 |
|        | FCGR + PCPseTNC | $K = 4, \lambda = 8, w = 0.5$ | 0.8539 | 0.8839 | 0.8244 | 0.7096 | 0.9275 |
|        | All features | | 0.8431 | 0.8461 | 0.8401 | 0.6867 | 0.9139 |
| ELM    | FCGR + DAC | $K = 4, \text{lag} = 2$ | **0.8707** | 0.8863 | **0.8555** | **0.7421** | **0.9355** |
|        | FCGR + TAC | $K = 4, \text{lag} = 2$ | 0.8696 | 0.8890 | 0.8505 | 0.7400 | 0.9358 |
|        | FCGR + DACC | $K = 4, \text{lag} = 2$ | 0.8684 | 0.8831 | 0.8539 | 0.7376 | 0.9358 |
|        | FCGR + TACC | $K = 4, \text{lag} = 2$ | 0.8680 | 0.8917 | 0.8447 | 0.7371 | 0.9329 |
|        | FCGR + PCPseDNC | $K = 4, \lambda = 8, w = 0.5$ | 0.8624 | 0.8847 | 0.8405 | 0.7258 | 0.9318 |
|        | FCGR + PCPseTNC | $K = 4, \lambda = 8, w = 0.5$ | 0.8557 | 0.8847 | 0.8271 | 0.7132 | 0.9262 |
|        | All features | | 0.8597 | 0.8863 | 0.8336 | 0.7210 | 0.9271 |
| XGBoost | FCGR | $K = 1 + 2 + 4$ | 0.8487 | 0.8797 | 0.8182 | 0.6995 | 0.9202 |
|        | FCGR + DAC | $K = 4, \text{lag} = 2$ | 0.8416 | 0.8652 | 0.8182 | 0.6842 | 0.9165 |
|        | FCGR + TAC | $K = 4, \text{lag} = 2$ | 0.8433 | 0.8707 | 0.8163 | 0.6882 | 0.9169 |
|        | FCGR + DACC | $K = 4, \text{lag} = 2$ | 0.8462 | 0.8703 | 0.8225 | 0.6938 | 0.9170 |
|        | FCGR + TACC | $K = 4, \text{lag} = 2$ | 0.8417 | 0.8676 | 0.8163 | 0.6848 | 0.9162 |
|        | FCGR + PCPseDNC | $K = 4, \lambda = 8, w = 0.5$ | 0.8450 | 0.8749 | 0.8156 | 0.6917 | 0.9199 |
|        | FCGR + PCPseTNC | $K = 4, \lambda = 8, w = 0.5$ | 0.8493 | 0.8789 | 0.8202 | 0.7004 | 0.9178 |
|        | All features | | 0.8481 | 0.8695 | 0.8271 | 0.6973 | 0.9195 |

All features means the feature vector = FCGR + DACC + TACC + PC-PseDNC + PC-pseTAC, and the parameters are consistent with the parameters of the corresponding feature. Parameter K indicates the values of K nucleotide in FCGR; lag indicates the distance of lag along the sequence; λ represents the highest counted rank (or tier) of the correlation along a DNA sequence; w is the weight factor ranged from 0 to 1.

Best values are in bold.
## Table 4: The prediction results of integrative feature representation for D. melanogaster via 10-fold cross-validation by SVM, ELM and XGBoost

| Method | Feature | parameter | ACC  | Sn   | Sp   | MCC  | AUC  |
|--------|---------|-----------|------|------|------|------|------|
| SVM    | FCGR + DAC | K = 4, lag = 2 | 0.8047 | 0.7862 | 0.8235 | 0.6103 | 0.8762 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.8089 | 0.7835 | 0.8347 | **0.6190** | 0.8747 |
|        | FCGR + DACC | K = 4, lag = 2 | 0.7753 | 0.7772 | 0.7733 | 0.5509 | 0.8295 |
|        | FCGR + TACC | K = 4, lag = 2 | 0.7560 | 0.6772 | **0.8361** | 0.5199 | 0.8247 |
|        | FCGR + PC-pseDNC | K = 4, λ = 8, w = 0.5 | 0.8073 | 0.7797 | 0.8354 | 0.6162 | **0.8803** |
|        | FCGR + PC-pseTNC | K = 4, λ = 8, w = 0.5 | 0.8057 | 0.7835 | 0.8284 | 0.6129 | 0.8769 |
|        | All features | | 0.7510 | 0.6628 | 0.8204 | 0.5078 | 0.7987 |
| ELM    | FCGR + DAC | K = 2, lag = 2 | 0.7920 | 0.7779 | 0.8063 | 0.5847 | 0.8644 |
|        | FCGR + TAC | K = 2, lag = 2 | 0.7917 | 0.7807 | 0.8028 | 0.5839 | 0.8651 |
|        | FCGR + DACC | K = 2, lag = 2 | 0.7769 | 0.7617 | 0.7923 | 0.5544 | 0.8503 |
|        | FCGR + TACC | K = 2, lag = 2 | 0.7694 | 0.7735 | 0.7653 | 0.5391 | 0.8460 |
|        | FCGR + PC-pseDNC | K = 2, λ = 8, w = 0.5 | 0.7896 | 0.7631 | 0.8165 | 0.5806 | 0.8651 |
|        | FCGR + PC-pseTNC | K = 2, λ = 8, w = 0.5 | 0.7595 | 0.7341 | 0.7853 | 0.5206 | 0.8400 |
|        | All features | | 0.7847 | 0.7810 | 0.7884 | 0.5700 | 0.8576 |
| XGBoost | FCGR | K = 1 + 2 + 4 | 0.7976 | 0.7797 | 0.8158 | 0.5959 | 0.8725 |
|        | FCGR + DAC | K = 4, lag = 2 | 0.7873 | 0.7717 | 0.8032 | 0.5751 | 0.8613 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.7877 | 0.7624 | 0.8133 | 0.5768 | 0.8647 |
|        | FCGR + DACC | K = 4, lag = 2 | 0.7724 | 0.7814 | 0.7632 | 0.5450 | 0.8532 |
|        | FCGR + TACC | K = 4, lag = 2 | 0.7824 | 0.7693 | 0.7958 | 0.5638 | 0.8542 |
|        | FCGR + PC-pseDNC | K = 4, λ = 8, w = 0.5 | 0.7997 | 0.7824 | 0.8172 | 0.6001 | 0.8725 |
|        | FCGR + PC-pseTNC | K = 4, λ = 8, w = 0.5 | 0.7988 | 0.7790 | 0.8190 | 0.5989 | 0.8775 |
|        | All features | | 0.7951 | 0.7793 | 0.8112 | 0.5909 | 0.8718 |

All features means the feature vector = FCGR + DACC + TACC + PC-pseDNC + PC-pseTNC, and the parameters are consistent with the parameters of the corresponding feature. Parameter K indicates the values of K nucleotide in FCGR; lag indicates the distance of lag along the sequence; λ represents the highest counted rank (or tier) of the correlation along a DNA sequence; w is the weight factor ranged from 0 to 1.

Best values are in bold.

---

**Fig. 2** The histogram (a-c) shows the accuracy of using SVM, ELM and XGBoost with contributing rate of principal component = 0.95, 0.93, 0.9, 0.88, 0.85 for H. sapiens.
Comparison with other advanced methods

In addition to DLNN-5 and ZCMM models, there are some other advanced methods for nucleosome prediction in the same dataset. LeNup model utilizes improved convolutional neural networks, which adds inception modules and gated convolutional networks [25]. 3LS is based on the linear regression model [22]. LeNup used...
the 20-fold cross-validation and provided comparison data with 3LS for H. sapiens, C. elegans and D. melanogaster. Therefore, we utilized the results provided by LeNup for comparative analysis in Tables 12, 13 and 14.

LeNup has the best overall prediction effect. The accuracy of C. elegans is 0.9188, and the average accuracy of other species are also over 0.88. The prediction result of our method is relatively close to it on the H. sapiens dataset. For C. elegans, ELM with FCGR performs slightly worse than 3LS, ACC, Sn, Sp, MCC, AUC decreased by 0.29%, 3.44%, 0.51%, 1.86% respectively.

**Discussion**

Firstly, the results in Table 1 and Fig. 1 clearly showed that the FCGR feature of the combined K value is better than the single K value, and the SVM output better prediction results. When training CNN and MLP models, we utilized multi-channel multiple K-value input images, and the prediction accuracy had been improved. All these indicated that FCGR feature combinations with different K values can better express sequence features, thereby improving models’ prediction accuracy.

Secondly, we further integrated FCGR with other feature representations, and combined three types of machine learning algorithms to compare prediction results (Tables 2, 3, 4). Besides, we performed PCA dimensionality reduction processing on

| Method | Feature | Parameters | PCA% | ACC | Sn | Sp | MCC | AUC |
|--------|---------|------------|------|-----|----|----|-----|-----|
| SVM    | FCGR    | K = 1 + 2 + 4 | 0.85 | 0.8758 | 0.8966 | 0.8552 | 0.7528 | 0.9288 |
|        | FCGR + DAC | K = 4, lag = 2 | 0.9 | 0.8749 | 0.8856 | 0.8644 | 0.7507 | 0.9314 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.85 | 0.8752 | 0.8878 | 0.8626 | 0.7513 | 0.9306 |
|        | FCGR + DAC | K = 4, lag = 2 | 0.95 | 0.8410 | 0.8236 | 0.8583 | 0.6825 | 0.9138 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.95 | 0.8369 | 0.8170 | 0.8565 | 0.6749 | 0.9099 |
|        | FCGR + PCPseDNC | K = 4, λ = 8, w = 0.5 | 0.88 | 0.8727 | 0.8896 | 0.8561 | 0.7463 | 0.9284 |
|        | FCGR + PCPseTNC | K = 4, λ = 8, w = 0.5 | 0.9 | 0.8719 | 0.8878 | 0.8561 | 0.7444 | 0.9281 |
|        | All features | | 0.95 | 0.7746 | 0.6388 | 0.9087 | 0.5698 | 0.8906 |
| ELM    | FCGR    | K = 1 + 2 + 4 | 0.88 | 0.8428 | 0.8636 | 0.8222 | 0.6866 | 0.9075 |
|        | FCGR + DAC | K = 4, lag = 2 | 0.88 | 0.8461 | 0.8724 | 0.8200 | **0.6936** | 0.9128 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.85 | **0.8469** | 0.8698 | 0.8244 | 0.6952 | **0.9129** |
|        | FCGR + DAC | K = 4, lag = 2 | 0.9 | 0.8458 | 0.8763 | 0.8157 | 0.6936 | 0.9095 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.88 | 0.8439 | 0.8808 | 0.8074 | 0.6902 | 0.9072 |
|        | FCGR + PCPseDNC | K = 4, λ = 8, w = 0.5 | 0.85 | 0.8454 | 0.8645 | **0.8265** | 0.6918 | 0.9107 |
|        | FCGR + PCPseTNC | K = 4, λ = 8, w = 0.5 | 0.88 | 0.8437 | 0.8627 | 0.8248 | 0.6882 | 0.9069 |
|        | All features | | 0.88 | 0.8447 | **0.8843** | 0.8057 | 0.6923 | 0.9118 |
| XGBoost | FCGR    | K = 1 + 2 + 4 | 0.95 | 0.8513 | 0.8733 | 0.8296 | 0.7037 | 0.9175 |
|        | FCGR + DAC | K = 4, lag = 2 | 0.85 | 0.8537 | 0.8667 | 0.8409 | 0.7080 | 0.9172 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.95 | **0.8859** | **0.9045** | **0.8674** | **0.7725** | **0.9491** |
|        | FCGR + DAC | K = 4, lag = 2 | 0.93 | 0.8364 | 0.8601 | 0.8130 | 0.6741 | 0.9014 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.95 | 0.8395 | 0.8667 | 0.8126 | 0.6805 | 0.9050 |
|        | FCGR + PCPseDNC | K = 4, λ = 8, w = 0.5 | 0.95 | 0.8463 | 0.8711 | 0.8217 | 0.6937 | 0.9147 |
|        | FCGR + PCPseTNC | K = 4, λ = 8, w = 0.5 | 0.93 | 0.8498 | 0.8645 | 0.8352 | 0.7003 | 0.9155 |
|        | All features | | 0.95 | 0.8423 | 0.8729 | 0.8122 | 0.6864 | 0.9051 |

*PCA%* means contributing rate of principal component
Best values are in bold

Table 5. PCA dimensionality reduction results via 10-fold cross-validation for H. sapiens.
feature vectors to prevent high-dimensional features from causing overfitting (Tables 5, 6, 7). Although the overall prediction quality has improved after the PCA dimensionality reduction processing with the integrated feature, superior results are obtained for using FCGR feature representation. These also further illustrated the advantages of FCGR features representation.

Here we compared the results of the proposed method with other advanced algorithms. Slightly superior results are achieved with our algorithm on H. sapiens and S. cerevisiae datasets, but there are gaps in the other two datasets. On the one hand, it explains the feasibility of our method; on the other, our work has room for improvement.

**Conclusions**

In this work, we used FCGR to represent the features of the DNA sequence and applied it to the nucleosome positioning. Our experiments have achieved positive results. Especially when multiple features are used in combination, the prediction quality can be improved. The advantage of this representation is that the time consumed in the process of constructing features is shortened, and the features are clear and intuitive. The quality of integrating features representation is also acceptable. Particularly after we use
PCA for dimensionality reduction, the prediction quality of H. sapiens dataset has been improved. This demonstrates the feasibility of the method.

In this paper, we also tried a simple CNN model with FCGR image and got mediocre results. Since deep learning is now increasingly used in bioinformatics. In the further research of nucleosome positioning, we will try to build a more efficient deep learning prediction model to achieve prediction of DNA represented in the form of images, such as FCGR image.

Table 7  PCA dimensionality reduction results via 10-fold cross-validation for D. melanogaster

| Method | Feature | Parameters | PCA% | ACC  | Sn   | Sp   | MCC  | AUC  |
|--------|---------|------------|------|------|------|------|------|------|
| SVM    | FCGR    | K = 1 + 2 + 4 | 0.88 | 0.8108 | 0.7786 | 0.8435 | 0.6235 | 0.8785 |
|        | FCGR + DAC | K = 4, lag = 2 | 0.95 | 0.8070 | 0.7855 | 0.8288 | 0.6152 | 0.8768 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.93 | 0.8115 | 0.7831 | 0.8404 | 0.6245 | 0.8766 |
|        | FCGR + DAC | K = 4, lag = 2 | 0.95 | 0.7809 | **0.7931** | 0.7684 | 0.5621 | 0.8343 |
|        | FCGR + TACC | K = 4, lag = 2 | 0.95 | 0.7678 | 0.6879 | **0.8491** | 0.5440 | 0.8363 |
|        | FCGR + PCPseDNC | K = 4, λ = 8, w = 0.5 | 0.9 | 0.8085 | 0.7752 | 0.8425 | 0.6190 | 0.8773 |
|        | FCGR + PCPseTNC | K = 4, λ = 8, w = 0.5 | 0.93 | 0.8097 | 0.7772 | 0.8428 | 0.6215 | 0.8761 |
|        | All features | | 0.95 | 0.7593 | 0.70414 | 0.81544 | 0.52275 | 0.80283 |
| ELM    | FCGR + DAC | K = 2, lag = 2 | 0.95 | 0.7817 | 0.7690 | 0.7947 | 0.5642 | 0.8544 |
|        | FCGR + TAC | K = 2, lag = 2 | 0.95 | 0.7859 | **0.7735** | 0.7986 | 0.5723 | 0.8552 |
|        | FCGR + DAC | K = 2, lag = 2 | 0.9 | 0.7530 | 0.7524 | 0.7537 | 0.5064 | 0.8262 |
|        | FCGR + TAC | K = 2, lag = 2 | 0.95 | 0.7365 | 0.7472 | 0.7256 | 0.4733 | 0.8018 |
|        | FCGR + PCPseDNC | K = 2, λ = 8, w = 0.5 | 0.93 | 0.7837 | 0.7597 | **0.8081** | 0.5685 | 0.8587 |
|        | FCGR + PCPseTNC | K = 2, λ = 8, w = 0.5 | 0.88 | 0.7678 | 0.7283 | **0.8081** | 0.5379 | 0.8448 |
|        | All features | K = 2 | 0.95 | 0.7727 | 0.7714 | 0.7740 | 0.5455 | 0.8437 |
| XGBoost | FCGR | K = 1 + 2 + 4 | 0.9 | 0.8037 | **0.7824** | **0.8253** | **0.6085** | **0.8772** |
|        | FCGR + DAC | K = 4, lag = 2 | 0.9 | 0.7877 | 0.7683 | 0.8074 | 0.5763 | 0.8630 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.88 | 0.7930 | 0.7635 | 0.8252 | 0.5879 | 0.8671 |
|        | FCGR + DAC | K = 4, lag = 2 | 0.93 | 0.7741 | 0.7690 | 0.7793 | 0.5486 | 0.8506 |
|        | FCGR + TAC | K = 4, lag = 2 | 0.88 | 0.7654 | 0.7576 | 0.7733 | 0.5313 | 0.8461 |
|        | FCGR + PCPseDNC | K = 4, λ = 8, w = 0.5 | 0.88 | 0.7988 | 0.7769 | 0.8211 | 0.5987 | 0.8753 |
|        | FCGR + PCPseTNC | K = 4, λ = 8, w = 0.5 | 0.85 | 0.7974 | 0.7772 | 0.8179 | 0.5960 | 0.8727 |
|        | All features | K = 2 | 0.93 | 0.7647 | 0.7590 | 0.7705 | 0.5295 | 0.8406 |

*PCA%* means contributing rate of principal component
Best values are in bold

Table 8  Comparison of our predictors with other models via 10-fold cross-validation for S. cerevisiae

| Method             | ACC   | Sn   | Sp   | MCC  | AUC  |
|--------------------|-------|------|------|------|------|
| Deformation energy [20] | 0.981 | 0.982 | 0.980 | 0.963 | ~    |
| FCGR-SVM           | 1     | 1    | 1    | 1    | 1    |
| FCGR-ELM           | 1     | 1    | 1    | 1    | 1    |
| FCGR-MLP           | 1     | 1    | 1    | 1    | 1    |
| FCGR-CNN           | 0.9997| 1    | 0.9994 | 0.9995 | 1    |
Methods

Dataset descriptions
To compare the results of the predictors, the datasets of this work downloaded from two published papers [20, 21]. The first group of datasets involved H. sapiens, C. elegans and D. melanogaster from the paper by Guo et al. [21]. The length of each DNA sequence is 147 bp. The second dataset involved S. cerevisiae genome from the paper by Chen et al. [20]. The length of each DNA sequence is 150 bp. Both of the datasets contain two types of samples: nucleosome-forming sequences (positive data) and nucleosome-inhibiting sequences (negative data). And none of the sequences included has ≥ 80% pairwise sequence identity with any other. The details of the datasets are shown in Table 15.

DNA sequence feature representation
Except for the above mentioned, common DNA sequence representation methods include basic kmer (Kmer) [34], reverse complementary kmer (RevKmer) [35], etc. based...
on deoxyribonucleic acid composition, and some are based on the correlation between nucleotide physical and chemical indicators, such as dinucleotide-based autocovariance (DAC), trinucleotide-based autocovariance (TAC) [29], etc. and pseudo k-tuple nucleotide composition (PseKNC) [21] based on pseudo deoxyribonucleic acid composition. These feature representation methods have specific calculation formulas and iterative functions, and some calculations are more complex and require a long time. This paper will mainly use a simple and intuitive feature representation.

Chaos game representation (CGR) is a graphical representation method of gene sequence based on chaos theory proposed by Jeffrey in 1990 [36]. The method is as follows: The four nucleotides \{A, T, G, C\} are located at the four vertices of the plane coordinate system, and the position of each nucleotide in the DNA sequence in the plane is \(P_i\). According to formula (2) draw the coordinate point of each nucleotide:

\[
P_i = 0.5 \cdot (P_{i-1} + N_i), \quad i = 1, \ldots, L \text{ and } P_0 = (0.5, 0.5)
\]
Among them, $P_0$ is the given starting point, $L$ is the length of the DNA sequence, and $N_i$ represents the corresponding coordinate of the $i$-th nucleotide, where $A = (0,0)$, $T = (1,0)$, $G = (1,1)$, $C = (0,1)$. This method draws a corresponding image of a DNA sequence through the iterative function and makes the nucleotides in the sequence correspond to the points on the image one by one [36–40]. From Fig. 5, we can see the CGR graphical representation of the two types of sample sequences in the *H. sapiens* dataset.

Divide the CGR image into $2^K \times 2^K$ sub-blocks and calculate the number of points appearing on each sub-block, we can determine the frequency of $K$ nucleotide combinations, and then convert the CGR image into a $2^K \times 2^K$ matrix, which is called frequency chaos game representation (FCGR) [39]. For example, we divided the CGR graph of Fig. 5a into a $2^3 \times 2^3$ matrix and calculated the number of occurrences of the midpoint of each sub-block, and obtain the frequency matrix shown in Table 16.

FCGR can be used not only as a numerical matrix, but also as a grayscale image. The original CGR image is divided into $4^K$ sub-blocks. The darker the sub-block, the more

---

**Table 11** Comparison of our predictors with other models via 10-fold cross-validation for D. melanogaster

| Method     | Feature | ACC   | Sn    | Sp    | MCC   | AUC   |
|------------|---------|-------|-------|-------|-------|-------|
| DLNN-5 [24]|         | 0.8560| 0.8781| 0.8333| ~     | ~     |
| ZCMM [23]  |         | 0.9362| 0.9226| 0.7964| 0.7000| 0.9110|
| SVM        | FCGR    | 0.8113| 0.7831| 0.84   | 0.6241| 0.8791|
|            | FCGR + DAC | 0.8089| 0.7835| 0.8347| 0.619 | 0.8747|
|            | FCGR + TAC | 0.8115| 0.7831| 0.8404| 0.6245| 0.8766|
|            | FCGR + DACC | 0.7809| 0.7931| 0.7684| 0.5621| 0.8343|
|            | FCGR + TACC | 0.7678| 0.6879| 0.8491| 0.544 | 0.8363|
|            | FCGR + PseDNC | 0.8085| 0.7752| 0.8425| 0.619 | 0.8773|
|            | FCGR + PseTNC | 0.8097| 0.7772| 0.8428| 0.6215| 0.8761|
|            | All features | 0.7593| 0.70414| 0.81544| 0.52275| 0.80283|
| ELM        | FCGR    | 0.791 | 0.7648| 0.8175| 0.5833| 0.8595|
|            | FCGR + DAC | 0.792 | 0.7779| 0.8063| 0.5847| 0.8644|
|            | FCGR + TAC | 0.7971| 0.7807| 0.8028| 0.5839| 0.8651|
|            | FCGR + DACC | 0.7769| 0.7617| 0.7923| 0.5544| 0.8503|
|            | FCGR + TACC | 0.7694| 0.7735| 0.7653| 0.5391| 0.846 |
|            | FCGR + PseDNC | 0.7806| 0.7631| 0.8165| 0.5806| 0.8651|
|            | FCGR + PseTNC | 0.7678| 0.7283| 0.8081| 0.5379| 0.8448|
|            | All features | 0.7847| 0.781 | 0.7884| 0.57  | 0.8576|
| XGBoost    | FCGR    | 0.8037| 0.7824| 0.8253| 0.6085| 0.8772|
|            | FCGR + DAC | 0.7877| 0.7683| 0.8074| 0.5763| 0.863 |
|            | FCGR + TAC | 0.793 | 0.7635| 0.8232| 0.5879| 0.8671|
|            | FCGR + DACC | 0.7741| 0.769 | 0.7793| 0.5486| 0.8506|
|            | FCGR + TACC | 0.7824| 0.7603| 0.7958| 0.5658| 0.8542|
|            | FCGR + PseDNC | 0.7997| 0.7824| 0.8172| 0.6001| 0.8725|
|            | FCGR + PseTNC | 0.7988| 0.779 | 0.819 | 0.5989| 0.8775|
|            | All features | 0.7951| 0.7793| 0.8112| 0.5909| 0.8718|
| MLP        | FCGR    | 0.8117| 0.8000| 0.8235| 0.6238| 0.8848|
| CNN        | FCGR    | 0.8108| 0.8014| 0.8204| 0.6228| 0.8854|

The table shows the optimal results of each classifier, and the specific parameters are shown in the previous tables.

Best values are in bold.
dots appear in the sub-blocks; the lighter sub-blocks, indicates that the number of dots in the color block is small, and the pixel value of the image is between 0 and 255 [39]. From Fig. 6, we can see the FCGR image of the sample sequence with K = 3, 4 and 5,

Support vector machine
Support vector machine (SVM) is a commonly used two-class classification model. Compared with other classification algorithms, it has a good classification effect and strong generalization ability on small data sets. It can also handle nonlinear classification problems through nuclear techniques. Thus, support vector machines have also been widely used in the field of bioinformatics [19, 21, 23]. Its basic idea is to map the sample from the original low-dimensional space to a high-dimensional space, so that the sample can find a partitioning hyperplane with the largest interval in the feature space, and separate samples of different categories.
In this paper, we will use the python package (Scikit-learn 0.23), which can be downloaded from https://scikit-learn.org/stable/index.html. This package contains the SVM module, and the implementation is based on libsvm. We will train the SVM with the radial basis function (RBF) kernel, meanwhile two parameters will be considered: penalty parameter C and kernel coefficient Gamma. In the training process, we used the grid optimization method to determine the best values of the two parameters.

**Extreme learning machine**

Extreme learning machine (ELM) was proposed by Guang-Bin Huang. The algorithm is a new machine learning algorithm based on single hidden layer feedforward neural networks (SLFNs). Compared with traditional algorithms, ELM has a faster learning speed while maintaining learning accuracy. The core idea is to randomly select the input layer weight and hidden layer bias of the network, and get the corresponding hidden node output [41]. The network structure of ELM model is shown in Fig. 7.
Table 14  Comparison of our predictors with other advanced models via 20-fold cross-validation for D. melanogaster

| Method | Feature | ACC  | Sn   | Sp   | MCC  | AUC  |
|--------|---------|------|------|------|------|------|
| LeNup [25] |         | 0.8847 | 0.8974 | 0.8713 | 0.7828 | 0.9401 |
| 3LS [22] |         | 0.8341 | 0.8407 | 0.8274 | 0.6682 | 0.9147 |
| SVM | FCGR | 0.8117 | 0.7841 | 0.8396 | 0.6251 | 0.8782 |
| | FCGR + DAC | 0.8094 | 0.7876 | 0.8316 | 0.6201 | 0.8783 |
| | FCGR + TAC | 0.8118 | 0.8073 | 0.8163 | 0.6252 | 0.8863 |
| | FCGR + DACC | 0.7866 | 0.7997 | 0.7733 | 0.5738 | 0.8384 |
| | FCGR + TACC | 0.7767 | 0.7128 | 0.8418 | 0.5593 | 0.8412 |
| | FCGR + PseDNC | 0.8095 | 0.7992 | 0.8199 | 0.6209 | 0.8843 |
| | FCGR + PseTNC | 0.8108 | 0.8034 | 0.8183 | 0.6234 | 0.8848 |
| | All features | 0.7602 | 0.7014 | 0.8200 | 0.5255 | 0.8059 |
| ELM | FCGR | 0.7912 | 0.7651 | 0.8204 | 0.5842 | 0.8601 |
| | FCGR + DAC | 0.7924 | 0.7752 | 0.8098 | 0.5862 | 0.8689 |
| | FCGR + TAC | 0.7932 | 0.7776 | 0.8091 | 0.5877 | 0.8619 |
| | FCGR + DACC | 0.7793 | 0.7710 | 0.7877 | 0.5599 | 0.8537 |
| | FCGR + TACC | 0.7697 | 0.7686 | 0.7709 | 0.5403 | 0.8456 |
| | FCGR + PseDNC | 0.7910 | 0.7659 | 0.8165 | 0.5837 | 0.8648 |
| | FCGR + PseTNC | 0.7691 | 0.7455 | 0.7930 | 0.5395 | 0.8433 |
| | All features | 0.7878 | 0.7859 | 0.7899 | 0.5763 | 0.8637 |
| XGBoost | FCGR | 0.8037 | 0.7821 | 0.8257 | 0.6088 | 0.8771 |
| | FCGR + DAC | 0.7891 | 0.7741 | 0.8042 | 0.5791 | 0.8648 |
| | FCGR + TAC | 0.7948 | 0.7762 | 0.8137 | 0.5910 | 0.8690 |
| | FCGR + DACC | 0.7814 | 0.7790 | 0.7839 | 0.5634 | 0.8540 |
| | FCGR + TACC | 0.7706 | 0.7648 | 0.7765 | 0.5417 | 0.8508 |
| | FCGR + PseDNC | 0.8010 | 0.7786 | 0.8239 | 0.6036 | 0.8728 |
| | FCGR + PseTNC | 0.8074 | 0.7958 | 0.8193 | 0.6165 | 0.8831 |
| | All features | 0.7979 | 0.7745 | 0.8218 | 0.5972 | 0.8739 |
| MLP | FCGR | 0.8127 | 0.8003 | 0.8253 | 0.6272 | 0.8893 |
| CNN | FCGR | 0.8116 | 0.8036 | 0.8198 | 0.6252 | 0.8854 |

Best values are in bold

Fig. 5  CGR of DNA sequences. a H. sapiens nucleosome-inhibiting sample and b H. sapiens nucleosome-forming sample
The experiment reference used David Lambert’s Python version of ELM resources, which can be downloaded from the ELM web portal (https://www.ntu.edu.sg/home/egbhuang/). The code can be found on https://github.com/dclambert/Python-ELM.

**Extreme gradient boosting**

Extreme gradient boosting (XGBoost) is an open source machine learning project developed by Tianqi Chen et al. [42]. It is one of the boosting algorithms, which has the characteristics of high efficiency, flexibility, high accuracy, and strong portability. It is applied in the field of biomedicine [43].

The idea of XGBoost algorithm is to continuously add trees and perform feature splitting to complete the construction of a tree. In the whole process, each addition of a tree is learning a new function to fit the residual of the previous prediction. When the training is completed, K trees will be obtained. If we want to predict the score of a sample, according to the features of this sample, each tree will fall to a corresponding leaf node,

### Table 15 The quantity composition of the four species datasets

| Species       | N-f | N-i | Total |
|---------------|-----|-----|-------|
| H. sapiens    | 2273| 2300| 4573  |
| C. elegans    | 2567| 2608| 5175  |
| D. melanogaster| 2900| 2850| 5750  |
| S. cerevisiae | 1880| 1740| 3620  |

N-f indicates nucleosome-forming sequences (positive data) and N-i indicates nucleosome-inhibiting sequences (negative data)

### Table 16 The frequency matrix of CGR image on H. sapiens nucleosome-inhibiting sample

|   | 1   | 3   | 2   | 4   | 5   | 7   | 4   | 2   | 0   | 0   | 0   | 0   |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 | 3   | 1   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 3 | 1   | 4   | 1   | 1   | 0   | 2   | 3   | 3   | 3   | 3   | 3   | 3   |
| 2 | 0   | 0   | 0   | 2   | 2   | 2   | 2   | 2   | 2   | 2   | 2   | 2   |
| 4 | 4   | 3   | 4   | 7   | 0   | 2   | 2   | 2   | 2   | 2   | 2   | 2   |
| 5 | 1   | 0   | 4   | 2   | 3   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 3 | 7   | 2   | 2   | 5   | 1   | 2   | 0   | 0   | 0   | 0   | 0   | 0   |
| 8 | 4   | 5   | 0   | 4   | 0   | 4   | 1   | 1   | 1   | 1   | 1   | 1   |
| 12| 2   | 2   | 1   | 3   | 2   | 2   | 2   | 2   | 2   | 2   | 2   | 2   |

**Fig. 6** FCGR image of H. sapiens nucleosome-inhibiting sample with different K. a K = 3, b K = 4 and c K = 5
and each leaf node corresponds to a score. Finally, we only need to add up the scores corresponding to each tree to get the predicted value of the sample.

In this experiment, we used the python package (xgboost 1.2.0), which can be downloaded from https://github.com/dmlc/xgboost.

**Multilayer perceptron**

Multilayer perceptron (MLP) is also called deep neural networks (DNNs) [44]. MLP is based on the extension of perception. Multiple hidden layers are introduced between the input layer and the output layer, and the neurons between the layers are fully connected. So, both the hidden layer and the output layer in MLP are fully connected layers.

For the MLP, we used the AI Studio (https://aistudio.baidu.com/aistudio/index) experimental platform and PaddlePaddle (https://www.paddlepaddle.org.cn/) deep learning framework provided by Baidu (https://www.baidu.com/) to implement the experimental model with python (https://www.python.org/). MLP has three hidden layers with ReLU activation function [45], each layer contains 50 neurons, the output layer uses a softmax activation function. Besides, MLP is trained by 5 epochs, with Adamax optimizer a learning rate of 0.001. Adamax algorithm is a variant of Adam algorithm based on infinite norm, which makes the algorithm of learning rate update more stable and simple [46]. We use cross entropy as our loss function.

**Convolutional neural network**

Convolutional Neural Network (CNN) is a representative algorithm of deep learning. It has demonstrated extraordinary advantages in the field of computer vision and has also been widely used in bioinformatics [47, 48]. Convolutional neural networks can automatically extract features from input data. Compared with fully connected neural networks, it can simplify model complexity and effectively reduce model parameters [49]. Convolutional neural networks are applied to the general framework of image...
mode, mainly composed of convolutional layers, activation function, pooling layers and fully connected layers [49, 50].

Owing to the limitation of the sample data volume, during the training process, we need to prevent the over-fitting problem faced by CNN, so we add a batch normalization (BN) layer [51] after the convolutional layer and add a dropout layer [52] after the fully connected layer. In our network, the convolutional layer uses a $3 \times 3$ convolution kernel, the number of filters in the first layer is 64, and the second is 32. The pooling layer use the maximum pooling of $2 \times 2$, with stride $= 2$. The first fully connected layer neurons’ number is 100, and the second is 50. Then, the dropout probability of the subsequent dropout layer is 0.5. Except the softmax activation function used in the output layer, the activation function in the other layers is Relu. CNN is training by 20 epochs, with Adamax optimizer a learning rate of 0.001. The loss function is cross entropy. Like MLP, we also used the AI Studio experimental platform and PaddlePaddle deep learning framework provided by Baidu to implement the experimental model in python. The specific network structure is shown in Fig. 8.

**Fig. 8** The architecture of our CNN model

---

**Abbreviations**

ACC: Accuracy; AUC: Area under curve; CGR: Chaos game representation; CNN: Convolutional neural networks; DAC: Dinucleotide-based autocovariance; DACC: Dinucleotide-based auto-cross covariance; ELM: Extreme learning machine; FCGR: Frequency chaos game representation; MCC: Mathew’s correlation coefficient; MLP: Multilayer perceptron; PCA: Principal component analysis; PC-PseDNC: Parallel correlation pseudo dinucleotide composition; PC-PseTNC: Parallel correlation pseudo trinucleotide composition; $S_n$: Sensitivity; $S_p$: Specificity; SVM: Support vector machine; TAC: Trinucleotide-based autocovariance; TACC: Trinucleotide-based auto-cross covariance; XGBoost: Extreme gradient boosting.

**Acknowledgements**

The authors would like to thank anonymous reviewers for their constructive comments, which helped us to greatly improve the manuscript.

**About this supplement**

This article has been published as part of BMC Bioinformatics Volume 22 Supplement 6, 2021: 19th International Conference on Bioinformatics 2020 (InCoB2020). The full contents of the supplement are available online at https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-22-supplement-6.

**Authors’ contributions**

GSH and YL directed the research. GSH and QL designed the experiments. QL ran all the experiments and wrote the paper. All authors read and approved the final manuscript.

**Funding**

This work was supported in part by Natural Science Foundation of Hunan Province of China (Grant No. 2016JJ3116), Key Foundation of Hunan Educational Committee (Grant No. 19A497). The funding bodies had no role in the design of the study and collection, analysis, and interpretation of data and in writing the manuscript. Publication costs are funded by Key Foundation of Hunan Educational Committee (Grant No. 19A497).

**Availability of data and materials**

The datasets of this work can be downloaded from two published papers [20, 21]. The python source code used in this work are freely available at https://github.com/lliqi-echo/Comparative-analysis-and-prediction-of-nucleosome-positioning.
Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
1 Department of Mathematics and Computational Science, Xiangtan University, Xiangtan 411105, Hunan, China. 2 Key Laboratory of Intelligent Computing and Information Processing of Ministry of Education and Hunan Key Laboratory for Computation and Simulation in Science and Engineering, Xiangtan University, Xiangtan 411105, Hunan, China.

Received: 4 February 2021 Accepted: 8 February 2021
Published: 2 June 2021

References
1. Luger K, Mäder AW, Richmond RK, et al. Crystal structure of the nucleosome core particle at 2.8 A resolution. Nature. 1997;389:251–60.
2. Nocetti N, Whitehouse I. Nucleosome repositioning underlies dynamic gene expression. Genes Dev. 2016;30(6):660–72.
3. Bai L, Morozov AV. Gene regulation by nucleosome positioning. Trends Genet. 2010;26(11):476–83.
4. Schalch T, Duda S, Sargent DF, et al. X-ray structure of a tetranucleosome and its implications for the chromatin fibre. Nature. 2005;436:138–41.
5. Eaton ML, Kyrkni G, Sukhyan K, et al. Conserved nucleosome positioning defines replication origins. Genes Dev. 2010;24(8):748–53.
6. Rodriguez J, Lee L, Lynch B, et al. Nucleosome occupancy as a novel chromatin parameter for replication origin functions. Genome Res. 2017;27(2):269–77.
7. Desvouges B, Sanchez MP, Ramirez-Parra E, et al. Impact of nucleosome dynamics and histone modifications on cell proliferation during Arabidopsis development. Heredity. 2010;105(1):80–91.
8. Bruns A, Bliss S, Hausdorf G, et al. Nucleosomes are major T and B cell autoantigens in systemic lupus erythematosus. Arthritis Rheum Off J Am Coll Rheumatol. 2000;43(10):2307–15.
9. Kornberg RD. Structure of chromatin. Annu Rev Biochem. 1977;46:931–54.
10. Yuan GC, Liu YJ, Dion MF, et al. Genome-scale identification of nucleosome positions in S.cerevisiae. Science. 2005;309(5734):626–30.
11. Lee W, Tillo D, Bray N, et al. A high-resolution atlas of nucleosome occupancy in yeast. Nat Genet. 2007;39(10):1235–44.
12. Ozsolak F, Song JS, Liu XS, et al. High-throughput mapping of the chromatin structure of human promoters. Nat Biotechnol. 2007;25(2):244–8.
13. Valouev A, Ichikawa J, Tontonatto T, et al. A high-resolution, nucleosome position map of C. elegans reveals lack of universal sequence-dictated positioning. Genome Res. 2008;18(7):1051–63.
14. Mavrich TN, Jiang C, Ioshikhes IP, et al. Nucleosome organization in the Drosophila genome. Nature. 2008;453(7193):358–62.
15. Xi L, Fondufe-Mittendorf Y, Xia L, et al. Predicting nucleosome positioning using a duration Hidden Markov Model. BMC Bioinformatics. 2010;11:346.
16. van der Heijden T, van Vuig JT, Logie C, et al. Sequence-based prediction of single nucleosome positioning and genome-wide nucleosome occupancy. PLoS One. 2012;7:109(38):E2514–22.
17. Segal E, Fondufe-Mittendorf Y, Chen L, et al. A genomic code for nucleosome positioning. Nature. 2006;442:772–8.
18. Struhl K, Segal E. Determinants of nucleosome positioning. Nat Struct Mol Biol. 2013;20(3):267–73.
19. Chen W, Lin H, Feng PM, et al. iNuc-PhysChem: a sequence-based predictor for identifying nucleosomes via physicochemical properties. PLoS ONE. 2012;7(10):e47843.
20. Chen W, Feng P-M, Ding H, et al. Using nucleosome positioning energy to analyze nucleosome positioning in genomes. Genomics. 2016;107(2–3):69–75.
21. Guo SH, Deng EZ, Xu LQ, et al. iNuc-PseKNC: a sequence-based predictor for predicting nucleosome positioning in genomes with pseudo k-tuple nucleotide composition. Bioinformatics. 2014;30(11):1522–9.
22. Awazu A. Prediction of nucleosome positioning by the incorporation of frequencies and distributions of three different nucleotide segment lengths into a general pseudo k-tuple nucleotide composition. Bioinformatics. 2017;33(12):42–8.
23. Cui Y, Xu Z, Li J. ZCMM: a novel method using Z-curve theory based and position weight matrix for predicting nucleosome positioning. Genes (Basel). 2019;10(10):765.
24. Di Gangi M, Lo Bosco G, Rizzo R. Deep learning architectures for prediction of nucleosome positioning from sequences data. BMC Bioinformatics. 2018;19(Suppl 14):418.
25. Zhang J, Peng W, Wang L. LeNuP: learning nucleosome positioning from DNA sequences with improved convolutional neural networks. Bioinformatics. 2018;34(10):1705–12.
26. Rizzo R, Fiannaca A, La Rosa M, et al. Classification experiments of DNA sequences by using a deep neural network and chaos game representation. In: Proceedings of the 17th international conference on computer systems and technologies 2016. 2016. p. 222–8.
27. Touati R, Messaoudi I, Oueslati AE, et al. New intraclasshelitrons classification using DNA-image sequences and machine learning approaches. IRBM. 2020. https://doi.org/10.1016/j.irbm.2019.12.004.
28. Lu M, Liu S, Sangahia AK, et al. Nucleosome positioning with fractal entropy increment of diversity in telemmedicine. IEEE Access. 2018;6:33451–9.
29. Dong Q, Zhou S, Guan J. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. Bioinformatics. 2009;25:2655–62.
30. Chen W, Zhang X, Brooker J, et al. PseKNC-General: a cross-platform package for generating various modes of pseudo nucleotide compositions. Bioinformatics. 2015;31(1):119–20.
31. Telf VB. Nucleosome positioning: resources and tools online. Brief Bioinform. 2016;17(5):745–57.
32. Liu B, Zhang D, Xu R, et al. Combining evolutionary information extracted from frequency profiles with sequence-based kernels for protein remote homology detection. Bioinformatics. 2014;30(4):472–9.
33. Lever J, Krzywinski M, Altman N. Principal component analysis. Nat Methods. 2017;14:641–2.
34. Liu B, Fang L, Wang S, et al. Identification of microRNA precursor with the degenerate K-tuple or Kmer strategy. J Theor Biol. 2015;385:153–9.
35. Noble WS, Kuehn S, Thurman R, et al. Predicting the in vivo signature of human gene regulatory sequences. Bioinformatics. 2005;21(suppl_1):i338–343.
36. Jeffrey HJ. Chaos game representation of gene structure. Nucleic Acids Res. 1990;18(8):2163–70.
37. Hoang T, Yin C, Yau SS. Splice sites detection using chaos game representation and neural network. Genomics. 2020;12(2):1847–52.
38. Joseph J, Saksikumar R. Chaos game representation for comparison of whole genomes. BMC Bioinformatics. 2006;7:243.
39. Almeda JS, Carriço JA, Maretezk A, et al. Analysis of genomic sequences by chaos game representation. Bioinformatics. 2001;17(5):429–37.
40. Poi S, Dong W, Chen X, et al. Fast and accurate genome comparison using genome images: The Extended Natural Vector Method. Mol Phylogenet Evol. 2019;141:106633.
41. Huang GB, Zhu QY, Siew CK. Extreme learning machine: theory and applications. Neurocomputing. 2006;70(1–3):489–501.
42. Chen T, Guéstrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM. 2016. p. 785–94.
43. Li W, Yin Y, Quan X, et al. Gene expression value prediction based on XGBoost algorithm. Front Genet. 2019;10:1077.
44. Löcher HE, Eger D, Sperlea T, et al. Deep learning on chaos game representation for proteins. Bioinformatics. 2020;36(1):272–9.
45. Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on machine learning. Haifa: Omnipress; 2010. p. 807–14.
46. Kingma DP, Ba J. Adam: a method for stochastic optimization. Comp Sci. 2014.
47. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. 2012. p. 1097–105.
48. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.

Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.