ML-Peaks: CHIP-Seq peak detection pipeline using machine learning techniques

Sajad Amouei Sheshkal[1,2,4], Michael Alexander Riegler[1,2,3], Hugo Lewi Hammer[1,2]

[1] *Oslo Metropolitan University (OsloMet), Oslo, Norway*
[2] *Simula Metropolitan Center for Digital Engineering, Oslo, Norway*
[3] *UiT The Arctic University of Norway, Tromsø, Norway*
[4] *Ifocus eye clinic, Haugesund, Norway*

***Abstract*—CHIP-Seq data is critical for identifying the locations where proteins bind to DNA, offering valuable insights into disease molecular mechanisms and potential therapeutic targets. However, identifying regions of protein binding, or peaks, in CHIP-seq data can be challenging due to limitations in peak detection methods. Current computational tools often require manual human inspection using data visualization, making it challenging and resource demanding to detect all peaks, particularly in large datasets. CHIP-seq data poses difficulties in detecting peaks due to its high background noise, low signal-to-noise ratio, and variation in the size and shape of the peaks. To overcome these challenges, we propose a data preprocessing approach using sliding window and feature reduction techniques, and the resulting features can be further used in machine learning methods. Our machine learning methodology can accurately identify peaks using a small training set, which represents a distinct advantage over commonly used statistical approaches, as it has a greater capacity for learning from data.**

We tested our methodology on the H3K9me3_TDH_BP CHIP-Seq dataset exploring a range of different machine learning methods, sliding window settings, and feature reduction techniques to detect peak values without human intervention. Our pipeline efficiently detected the peaks, and achieved an F1-score of 0.9644 and a false positive rate of 0.1030.

***Index Terms*—Peak detection, CHIP-Seq dataset, Machine learning, Sliding window, Feature reduction.**

I. Introduction

Chromatin immunoprecipitation followed by high-throughput sequencing (CHIP-Seq) is a widely used technique for studying the binding of DNA-associated proteins, such as transcription factors, to the genome [1]. The method enables researchers to map the binding sites of DNA-associated proteins and their target genes, providing important insights into the regulation of gene expression. The CHIP-Seq data generated from this technique requires further analysis to identify the regions of the genome where DNA-associated proteins bind. To aid in this process, researchers use a variety of software tools, including both graphical tools and peak detection algorithms. This process is known as peak detection, and it is a key step in the CHIP-Seq data analysis pipeline [2].

Graphical tools, such as the UCSC genome browser [3], provide a visual interface for exploring genomic data, including the locations of functional elements. These tools allow scientists to view their data in the context of the genome, alongside other relevant information, such as gene annotations and conservation information. This visual representation can be particularly useful in the early stages of data exploration, as it helps to identify trends and patterns that may not be immediately apparent from the raw data.

However, graphical tools suffer from three main disadvantages. At a single base resolution, peak start and end locations are not obvious on visual inspection. Second, visual interpretation is inherently subjective, which makes it difficult for other researchers to reproduce it. Finally, it is not enough time for researchers to visually inspect and identify peaks across the entire genome. For these reasons, it is useful to use computational methods, and peak detection algorithms, to systematically and accurately identify CHIP-Seq peaks, in addition to using the UCSC Genome Browser [3] for data visualization and exploration.

Peak detection algorithms are used to identify the regions of the genome that are enriched with CHIP-Seq signals, representing the binding sites of DNA-associated proteins. There are several peak detection algorithms available, including MACS [2], SICER [4], and SPP [5], which are based on different statistical models and assumptions about the distribution of the CHIP-Seq signals. However, the accuracy and reliability of these algorithms can vary based on the specific requirements of the research question and the characteristics of the CHIP-Seq data. Statistical methods are relatively straightforward, but have limited capacity for acquiring knowledge from previously identified peaks (training data). As a result, it is a need for machine learning methods with better learning capacity, and to further carefully evaluate their performance to be able to select the appropriate algorithm for a specific study [6].

In recent years, machine learning techniques, such as random forest [7] and Support Vector Machine (SVM) [8], have been increasingly used for peak detection in CHIP-Seq data analysis [6], [9], [10]. These algorithms have shown promising results in terms of accuracy, sensitivity, and specificity in detecting binding sites compared to traditional peak detection algorithms [11].

In this paper, we present a peak detection pipeline that is illustrated in Figure 1, moving from left to right. The pipeline begins by utilizing a sliding window approach to explore the dataset and extract sub-sequences from the data.

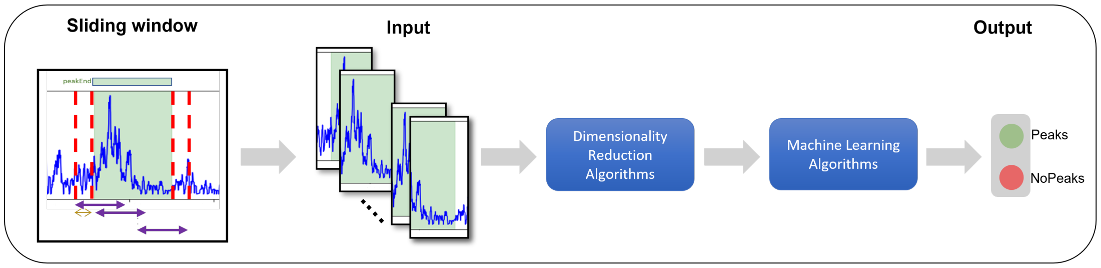

Sliding window
peakEnd
Input
Output
Dimensionality Reduction Algorithms
Machine Learning Algorithms
Peaks
NoPeaks

Fig. 1: ML-Peaks pipeline overview. The proposed sliding window technique extracts sub-sequences from labeled regions in the chip-seq dataset. Feature reduction followed by machine learning algorithms yield binary classification: Peaks or NoPeaks.

Dimensionality reduction algorithms are then applied to the extracted sub-sequences, which are subsequently used as input for machine learning models to conduct experiments for peak and noise detection.

Our main contributions are as follows:

1) We propose a flexible sliding window approach for CHIP-Seq data analysis, which yields high accuracy.
2) We demonstrate the effectiveness of feature reduction methods in enhancing machine learning algorithms for peak detection. By reducing the dimensionality of the input data, we optimize the performance of the machine learning algorithms and achieve more accurate peak detection results.
3) We compare the performance of different machine learning algorithms for peak detection.

Overall, our paper presents a flexible and effective pipeline for peak detection in CHIP-Seq datasets using machine learning and offers valuable contributions to the field of CHIP-Seq data analysis.

We conducted our experiments on Google Colaboratory, using version 1.0 of the platform. The hardware specifications of the system are as follows: Intel(R) Xeon(R) CPU @ 2.20GHz, 25 GB RAM, and Disk space 107 GB. We used Python version 3.8 and scikit-learn [12] library version 1.2.2 to implement machine learning algorithms and evaluation metrics. The code used to perform the experiments described in this article is open-source and available on GitHub [13]. Users can modify the code to suit their needs and reproduce the experimental results in the paper, promoting transparency and reproducibility in scientific research.

II. Data and Methods

A. Data description

The data used in this paper was obtained from the UCI Machine Learning Repository [14]. The data examine histone modifications such as histones H3K4me1, H3K9me3, H3K27ac, H3K27me3, and H3K36me3 and label files containing labeled regions. Some segments of the long genomic sequences had been labeled by human experts using a genome visualization browser. In this paper, we used H3K9me3_TDH_BP CHIP-Seq sequences. This dataset contains CHIP-Seq data for the histone modification H3K9me3 in the TDH-BP cell line. In the context of peak detection, the H3K9me3_TDH_BP dataset provides a rich resource for evaluating and developing algorithms for identifying regions of high signal intensity, commonly referred to as ”peaks”, in CHIP-Seq data. The peaks in the dataset correspond to regions of the genome where the H3K9me3 histone modification is enriched and may represent sites of regulation or epigenetic changes.

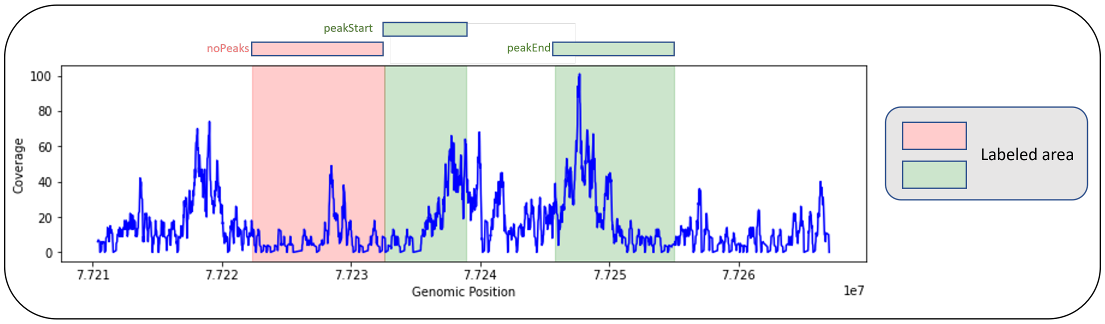
The dataset consists of 540 labeled sequences and contains both positive and negative control sequences, providing a robust evaluation resource for peak detection algorithms. Figure 2a shows a part of a sequence, including the labels peakStart, peakEnd, noPeaks and parts that were not annotated. The data is presented in a tab-delimited format with four columns: chromosome, start position, end position, and signal intensity, and saved using run-length encoding. Both peaks (peakStart and peakEnd) and noPeaks classes have almost the same number of sequences, 253 peak sequences and 287 non-peak sequences of varying lengths, but noPeaks sequences are on average much longer than peaks sequences. Table I lists the attributes of the H3K9me3_TDH_BP dataset.

TABLE I: H3K9me3_TDH_BP chip-seq data attributes.

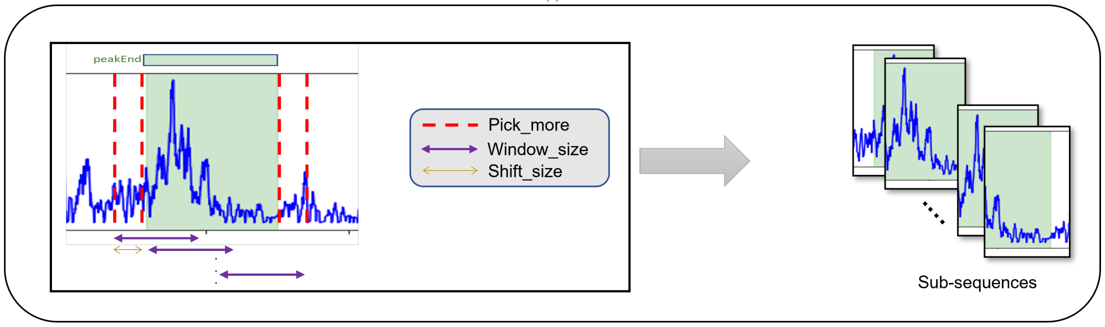
Dataset attribute	Value
Dataset name	H3K9me3_TDH_BP
Data type	Chip-Seq
Domain	Broad peaks
Number of files	240 files
File format	bedgraph
Data labels	peakStart, peakEnd and NoPeaks
Number of labeled regions	540 regions
Number of Peak Sequences	253
Number of NoPeak Sequences	287
The total length of all Peaks sequences	4,122,705
The total length of all NoPeaks sequences	11,590,675

B. Data preprocessing

The H3K9me3_TDH_BP CHIP-Seq dataset comprises labeled sequences that are represented as vectors of non-negative integer count data. This data is stored on disk space using the run-length encoding [14] method as tables, with each table containing information for a labeled sequence. These tables consist of four columns: the first column indicates the chromosome number, the second column indicates the starting point of the genomic position in the vector, the third column indicates the end point of the genomic position in the vector,



(a)



(b)

Fig. 2: The chip-seq data and the proposed sliding window. Figure 2a A partial sequence of the chip-seq dataset is shown, with colored backgrounds indicating annotated regions and unannotated regions displayed without coloration. Figure 2b To extract sub-sequences, the proposed sliding window approach is employed on the annotated regions, with the window_size, shift_size, and pick_more parameters utilized in the process.

and the fourth column specifies the value to be placed in the vector from the starting point to the end of the genomic position. In our preprocessing module, we first decoded the sequences and extracted the labeled regions based on the label files that indicated which parts of the sequences were labeled. We redefined the labels, by introducing the label peaks as every region has peakStart or peakEnd, resulting in the two labels peaks and noPeaks.

To divide the labeled regions into more sub-sequences, we introduced a sliding window approach, as shown in Figure 2b. The idea is that to predict the class in a given genomic position, the most important information is in some neighborhood, or window, around the position. The labeled regions varied in length, and using the sliding window approach on each region resulted in a different number of labeled sub-sequences. This approach includes adjustable parameters that influence the number of labeled sub-sequences generated, which can impact the performance of machine learning algorithms.

This approach enables us to evaluate three important factors that impact the extraction of sub-sequences, namely window_size, shift_size, and pick_more, see Figure 2b. The variable window_size specifies the size of the sub-sequences to be extracted from the labeled regions. The pick_more factor specifies the part of the unlabeled regions to use along with the labeled regions to create additional sub-sequences. The shift_size factor controls the amount of horizontal shift between each extracted feature. By extracting labeled sub-sequences, our method enables the identification of additional patterns from labeled regions and provides more data for pattern extraction for machine learning algorithms.

To evaluate the effectiveness of our pipeline for peaks and noise identification, we conducted several experiments using the proposed sliding window algorithm to generate sub-sequences of specific lengths.

C. *Feature reduction*

In the field of machine learning, feature reduction methods play a crucial role in reducing the dimensionality of the data while preserving the most informative features. Principal Component Analysis (PCA) [15], Linear Discriminant Analysis (LDA) [16], and Independent Component Analysis (ICA) [17] are among the most widely used feature reduction techniques.

PCA [15] is an unsupervised method that reduces the dimensionality of the data by projecting it onto a lower-dimensional subspace while retaining the maximum amount of variance. PCA [15] is particularly useful when dealing with high-dimensional data, such as epigenetic data and sequences, where the number of features exceeds the number of samples.

ICA [17] is another unsupervised feature reduction method that seeks to separate the observed data into statistically independent sources. By separating the data into independent sources, ICA [17] can reduce the complexity of the data and improve the interpretability of the results.

LDA [16], on the other hand, is a supervised feature reduction method that seeks to maximize the separation between classes while minimizing the within-class variance. LDA [16] is commonly used in classification tasks, where the goal is to identify the class membership of a given sample.

In addition to the aforementioned feature reduction techniques, we also employed hand-crafted methods to reduce the dimensionality of the data. Specifically, we calculated the maximum and average values for each sub-sequence, as well as the mean, range of values (maximum-minimum) along an axis called Peak-to-Peak (PTP) [18], and Standard Deviation (STD) of each sub-sequence. These methods can provide valuable insights into the underlying patterns in the data.

D. Machine learning models

In this study, several machine learning approaches from diverse families were chosen for experimentation. These methods were specifically selected due to their unique features and included: Decision tree [19], XGBoost [20], Adaboost [21], Bagging [22], Random forest [7], and SVM [8]. The algorithms were implemented using the scikit-learn [12] library in Python.

Adaboost [21] and Bagging are both iterative boosting algorithms that use a set of independent learners that are combined into strong learners. The combination of the learners reduces overfitting and improves the stability of the method. Bagging is an ensemble learning method that trains multiple models on different subsets of the training data and combines their predictions to obtain a better result. Random Forest [7] is an extension of the Bagging method that selects a random subset of features for each split in the decision tree [19]. This method is known for its ability to handle high-dimensional data effectively and reduce overfitting.

The Decision tree [19] and XGBoost [20] algorithms are both tree-based models that utilize a hierarchical structure to make predictions. The decision tree [19] is a simple and interpretable model that can handle both continuous and categorical variables. XGBoost [20], short for eXtreme Gradient Boosting, is an optimized version of the gradient boosting algorithm. It is widely used for its ability to handle missing values and high-dimensional data effectively. SVM [8] is a supervised learning algorithm that is known for its ability to handle non-linear problems and high-dimensional data. SVM [8] has been widely used in various fields, including bioinformatics and genomics, due to their performance in handling complex data structures.

Besides employing sophisticated machine learning algorithms, we also use a DummyClassifier algorithm, which utilizes elementary rules for prediction. This algorithm functions as a baseline in conjunction with other algorithms and consistently predicts the class with the most frequency in the training data. The DummyClassifier algorithm can be configured to adopt different strategies, and we have employed three of these strategies, namely ”most frequent”, ”stratified”, and ”uniform”.

E. Evaluation metrics

To evaluate the performance of the machine learning algorithms used in this study, several metrics were utilized. The performance of the algorithms was evaluated on the test dataset using accuracy, precision, recall, F1-score, balanced accuracy, and Matthews Correlation Coefficient (MCC).

Accuracy, precision, recall, and F1-score are commonly used metrics for evaluating the performance of machine learning algorithms. Accuracy measures the percentage of correct predictions made by the model, while precision measures the percentage of true positive predictions among all positive predictions. Recall, also known as sensitivity, measures the percentage of positive instances that were correctly detected by the model. The F1-score is the harmonic mean of precision and recall, and it provides a balance between the two metrics.

These metrics provide a comprehensive evaluation of the performance of the model, especially when dealing with imbalanced datasets. Balanced accuracy is a modification of accuracy that takes into account the imbalance in the class distribution of the data. This metric is particularly useful when dealing with imbalanced datasets, as it provides a more accurate evaluation of the performance of the model. MCC is a correlation coefficient that measures the quality of binary classifications. One of the primary challenges in peak detection algorithms is the identification of noise signals as true peaks. Consequently, the false positive rate in the confusion matrix can serve as a useful metric to compare and evaluate the performance of these algorithms. By examining the accuracy of peak detection and minimizing the false positive rate, researchers can develop more reliable and efficient peak detection methods.

III. Experiments and Results

We tested the performance of various machine learning methods on the extracted sub-sequences, which were presented to the models in eight different configurations of the extraction. The methods were tested on unmodified sub-sequences, as well as on subsequences that had been processed using the seven different dimension reduction methods, including PCA [15] with 2 and 3 components, ICA [17] with 2 and 3 components, LDA [16] with one component, calculation of average and a maximum of each sub-sequence, and calculation of mean, PTP [18], and STD for each sub-sequence.

The computed features used to train the popular machine learning methods such as Decision tree [19], XGBoost [20], AdaBoost [21], Bagging, Random Forest [7], and SVM [8] to classify between peaks and noPeaks. The default machine learning model settings and five-fold cross-validation were used for training. We used 70% of the data for training and five-fold cross-validation, and 30% for testing.

The noPeaks sequences were on average much longer than peaks sequences, and can therefore result in far more noPeaks than peaks sub-sequences. To address this imbalance problem, we used a higher shift_size factor for extracting sub-sequences from the noPeaks sequences compared to the peaks sequences.

To compare the impact of these changes on peak detection models, 138 experiments were conducted by altering window_size, shift_size, and pick_more parameters and selecting different feature reduction methods and machine learning models,and the main findings are summarized below.

TABLE II: The table displays the best result achieved by combinations of machine learning models, feature reduction techniques, and window sizes across all 138 experiments based on the evaluation metrics used.

ML Model	Feature reduction method	Window size	F1 score	Balanced accuracy	MCC	False positive rate
SVM [8]	LDA [16]	4001	**0.9644**	**0.9641**	**0.9290**	0.1260
XGBoost [20]	LDA [16]	4001	0.9638	0.9634	0.9281	**0.1030**
AdaBoost [21]	LDA [16]	4001	0.9626	0.9623	0.9258	**0.1030**
Random forest [7]	LDA [16]	4001	0.9489	0.9488	0.8979	0.2350
Bagging [22]	LDA [16]	4001	0.9489	0.9488	0.8979	0.2290
Baseline ”most_frequent”	LDA [16]	4001	0.3395	0.5	0.0	0.0
Baseline ”stratified”	LDA [16]	4001	0.4953	0.4954	-0.9100	0.2441
Baseline ”uniform”	LDA [16]	4001	0.5036	0.5037	0.7400	0.2550

TABLE III: This table shows comparing windows size used for making sub-sequences and effect on machine learning models. All sub-sequences fed to models without any feature reduction

Model	Window size	Shift size	F1 score
XGBoost [20]	2001	100	0.6993
	4001	100	0.7748
	10001	100	**0.8733**
	4001	500	0.7787

Table III, shows the performance of XGBoost [20] for different choices of window_size and shift_size. According to the results, using a window_size of $10,001$ and a shift_size of 100 led to the highest F1-Score of 0.8733. This indicates that peak detection accuracy can be improved by using a larger window size to scan the dataset. Nevertheless, the study also demonstrated that a combination of other settings and a smaller window size can result in a better outcome in some cases.

TABLE IV: The table compares the impact of feature reduction methods on labeled sequences within the H3K9me3_TDH_BP dataset on the Bagging machine learning model's ability to detect peaks and noises, measured by the F1-score criteria. The highest F1-score is highlighted in bold

Model	Feature reduction	F1 score
Bagging [22]	Raw sub-sequences	0.7540
	Min, Max	0.7039
	Mean, PTP [18], STD	0.7814
	PCA [15] with 2 components	0.7395
	PCA [15] with 3 components	0.7640
	ICA [17] with 2 components	0.7419
	ICA [17] with 3 components	0.7648
	LDA [16] with 1 component	**0.9489**

Abbreviations: Min = Minimum, Max = Maximum, STD = Standard deviation, PTP = Peak to peak function in numpy library, PCA = Principal component analysis, ICA = Independent component analysis, LDA= Linear discriminant analysis.

Table IV shows the performance of the Bagging machine learning model using different feature reduction methods. The results show that using LDA [16] for feature reduction resulted in the highest F1-Score of 0.9489. This indicates that LDA [16] is a highly effective method for reducing the dimensionality of the data and improving the performance of the machine learning models.

Table II displays the top five outcomes out of 138 experiments conducted. The SVM [8] algorithm, along with a window size of 4001 and LDA [16] feature reduction method, produced the highest performance in the current pipeline, achieving an F1-score of 0.9644. This indicates that using the SVM [8] algorithm, a window size of 4001, and applying LDA [16] for feature reduction can significantly enhance the accuracy of peak detection in this dataset.

Additionally, we observed that the XGBoost [20] and Adaboost [21] machine learning models had the lowest false positive rates, with values of 0.0103. The low false positive rates observed in XGBoost [20] and Adaboost [21] models suggest that these models have the potential to be used in other CHIP-Seq datasets for accurate peak detection. The results indicate that the machine learning algorithms have achieved considerably higher accuracy in contrast to the baseline models.

The outcome of our experiments demonstrates that machine learning methods can effectively address peak detection problems with high accuracy, without the need for domain experts. These techniques can distinguish between peak shapes and noises, which is a major challenge in this field, and achieve low false positive rates, thereby increasing the reliability of peak detection. Our findings indicate that the size of the window in the sliding window method plays a crucial role in extracting meaningful patterns from the data. A larger window size can capture more information, but it may also include more noise, whereas a smaller window size may miss important peaks. Therefore, choosing an appropriate window size is essential for accurate peak detection. Furthermore, our results reveal that feature reduction methods can significantly improve the performance of machine learning models in peak detection. These methods can remove irrelevant and redundant features, reducing the complexity of the model and improving its generalization ability. By selecting relevant features, feature reduction methods provide more suitable data for training machine learning models, resulting in higher accuracy and faster computation time.

IV. Conclusion

In this paper, we have introduced a pipeline for peak detection in CHIP-Seq genomic data, which combines sliding window, feature reduction techniques, and machine learning methods. To assess the performance of the pipeline, various evaluation metrics were utilized, demonstrating their effectiveness in accurately detecting peaks. Our findings indicate that the selection of window sizes can significantly impact feature extraction using the sliding window approach. Additionally, feature reduction methods can yield more relevant features, which can improve the performance of machine learning models trained on the dataset. Importantly, the use of feature reduction methods can reduce the dimensionality of the training data and decrease the training and testing time for machine learning algorithms. We evaluated several machine learning methods and found that SVM [8], XGBoost [20], and AdaBoost [21] outperformed other models in detecting peaks. Our study provides insight into the benefits of combining sliding window, feature reduction, and machine learning techniques in peak detection, and highlights the importance of optimizing these methods for specific CHIP-Seq genomic sequences.

In the future, the peak detection pipeline proposed in this study could be expanded and applied to a diverse range of CHIP-Seq datasets with different peak characteristics, such as sharp or broad peaks. Moreover, the sub-sequences located between peakStart and peakEnd could be leveraged to extract patterns that are associated with peaks. The combination of the sliding window, feature reduction and machine learning models used in this pipeline has shown promising results, but further optimization through fine-tuning could improve its accuracy and efficiency. Moreover, deep learning models and pre-trained models can be employed to extract peak patterns from extracted sub-sequences by the proposed sliding window, which could enhance the performance of the pipeline. Future research could explore the use of these models and evaluate their effectiveness compared to existing methods. In summary, our proposed peak detection pipeline has great potential for advancing CHIP-Seq data analysis and providing insights into the functional genomics of various biological systems. Further investigation and optimization of the pipeline could lead to more accurate and efficient detection of peaks in diverse datasets.

References

[1] P. J. Park, "Chip–seq: advantages and challenges of a maturing technology," *Nature reviews genetics*, vol. 10, no. 10, pp. 669–680, 2009.

[2] Y. Zhang, T. Liu, C. A. Meyer, J. Eeckhoute, D. S. Johnson, B. E. Bernstein, C. Nusbaum, R. M. Myers, M. Brown, W. Li *et al.*, "Model-based analysis of chip-seq (macs)," *Genome biology*, vol. 9, no. 9, pp. 1–9, 2008.

[3] C. M. Lee, G. P. Barber, J. Casper, H. Clawson, M. Diekhans, J. N. Gonzalez, A. S. Hinrichs, B. T. Lee, L. R. Nassar, C. C. Powell *et al.*, "Ucsc genome browser enters 20th year," *Nucleic acids research*, vol. 48, no. D1, pp. D756–D761, 2020.

[4] C. Zang, D. E. Schones, C. Zeng, K. Cui, K. Zhao, and W. Peng, "A clustering approach for identification of enriched domains from histone modification chip-seq data," *Bioinformatics*, vol. 25, no. 15, pp. 1952–1958, 2009.

[5] P. V. Kharchenko, M. Y. Tolstorukov, and P. J. Park, "Design and analysis of chip-seq experiments for dna-binding proteins," *Nature biotechnology*, vol. 26, no. 12, pp. 1351–1359, 2008.

[6] R. Nakato and T. Sakata, "Methods for chip-seq analysis: a practical workflow and advanced applications," *Methods*, vol. 187, pp. 44–53, 2021.

[7] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, pp. 197–227, 2016.

[8] S. Suthaharan and S. Suthaharan, "Support vector machine," *Machine learning models and algorithms for big data classification: thinking with examples for effective learning*, pp. 207–235, 2016.

[9] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[10] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows–wheeler transform," *bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[11] M. A. DePristo, E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl, A. A. Philippakis, G. Del Angel, M. A. Rivas, M. Hanna *et al.*, "A framework for variation discovery and genotyping using next-generation dna sequencing data," *Nature genetics*, vol. 43, no. 5, pp. 491–498, 2011.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] S. A. Sheshkal, "ML-Peaks: Chip-seq peak detection pipeline using machine learning techniques," https://github.com/sajadamouei/peak-detection-chip-seq, 2023.

[14] T. D. Hocking, "Chipseq Data Set," https://archive.ics.uci.edu/ml/datasets/chipseq, 2018, [Online; accessed 05-January-2023].

[15] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[16] S. Ioffe, "Probabilistic linear discriminant analysis," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part IV 9*. Springer Berlin Heidelberg, 2006, pp. 531–542.

[17] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[18] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[19] O. Z. Maimon and L. Rokach, *Data mining with decision trees: theory and applications*. World scientific, 2014, vol. 81.

[20] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[21] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[22] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine learning*, vol. 36, pp. 105–139, 1999.