Extracting Biological Insights From Genomics Data Using Machine Learning Approaches

Lance D. Hentges
University College



A thesis presented for the degree of
Doctor of Philosophy

University of Oxford
Michaelmas 2021

ACKNOWLEDGEMENTS

There have been many people who have helped me to complete the work presented in the following pages. The most central being my supervisors Jim Hughes and Steve Taylor, whose ideas and insights continue to amaze me even years after knowing them. It's impossible to even imagine this project without them, and their support of me during my time as a student has undoubtedly changed my life.

I also need to thank the Hughes Group who were incredibly helpful, both in and out of the pub. They always had my back and were usually my first beta testers - this was especially true of Damien Downes in both regards. Working alongside Ron Schweissenger meant that I could bounce ideas off of a fellow deep learning doctoral student, and was another person I was very lucky to share a group with.

The AVI Group was also pivotal to this work. Simon McGowan was incredibly helpful in polishing our web tool implementation, offering user experience insight and even proofing drafts of our manuscript for publication. Both Ewan MacMahon and Duncan Tooke also deserve recognition - their knowledge and patience during painstaking testing allowed our software to be hosted at the Centre for Computational Biology's cluster, granting easy access to our tool worldwide.

Spanning both the Hughes and AVI groups is Martin Sergeant, the co-creator of our analysis tool. Martin is a brilliant programmer, and his web interface does more than just displaying the model outputs online, it brings the results to life. Besides offering quality control to users, it made it easier for me to visualise and improve my own work.

It has been an incredible honour to be hosted by the prestigious Weatherall Institute of Molecular Medicine. I owe a debt of gratitude to Doug Higgs for his encouragement throughout my time working here - while my focus has always leaned slightly out of his wheelhouse, that never stopped him from asking thoughtful questions that got to the heart of the science at hand. Furthermore the Higgs Group, alongside the Milne Group, put in countless hours using our software for real analysis, for which I am wholly appreciative.

I am also grateful to the VISION Project Consortium and University College, University of Oxford, whose financial contributions helped make my work possible. The VISION project also provided me with an outlet to some of the most prominent researchers in genome biology outside of Oxford. Complementing that, my time at Univ exposed me to world class researchers across all disciplines and was the setting for some of the most memorable moments during my time in Oxford.

Kicking this whole thing off was my time in the Systems Biology Programme

in the Doctoral Training Centre. It was there I was encouraged to explore ideas that led to this project, making some lifelong friends in the process.

Lastly, I am very fortunate to have some very special people in my life. My family, especially my brother Grant who I could always count on for some good cheer and a laugh, and of course my parents who were always ready to help, even from an ocean away. Finally my partner Megan, who really means the world to me. Her unwavering support helped me push myself when things felt tough, and has been so central to this as with so many other things in my life.

CONTENTS

Abstract **1**

List of Figures **2**

List of Tables **7**

1 Introduction **8**
1.1 Regulation in the genome 9
1.1.1 Gene expression 10
1.1.2 Features of regulatory elements 13
1.2 Locating regulatory elements 14
1.2.1 War and peaks 17
1.3 Machine learning in biology 20
1.3.1 Deep learning in genomics 23
1.4 Thesis aims 26

2 Data and Candidate Peaks **28**
2.1 Introduction 28
2.2 Results 32
2.2.1 Data acquisition 32
2.2.2 Hand-labelling peaks 34
2.2.3 Candidate peak comparison 39
2.2.4 P-value comparison 44
2.3 Discussion 50

3 Deep Learning **57**
3.1 Introduction 57
3.2 Results 65
3.2.1 CNN model 65
3.2.2 Chromatin capture conformation interactions 69
3.2.3 Autoencoder 74
3.2.4 Wide and deep model 75
3.2.5 Enhancing ChIPmentation signal 81
3.3 Discussion 84

4 LanceOtron **89**
4.1 Introduction . . . 89
4.2 Results . . . 93
4.2.1 Reanalysis of existing peak call . . . 94
4.2.2 Transcription factor ChIP-seq . . . 95
4.2.3 Histone ChIP-seq . . . 99
4.2.4 ATAC-seq and DNase-seq . . . 102
4.3 Discussion . . . 103

5 Methods **108**
5.1 Data and Candidate Peaks - Chapter 2 methods . . . 108
5.1.1 Data acquisition . . . 108
5.1.2 Hand-labelling peaks . . . 109
5.1.3 Candidate peak comparison . . . 112
5.1.4 P-value comparison . . . 113
5.2 Deep Learning - Chapter 3 methods . . . 115
5.2.1 CNN model . . . 115
5.2.2 Chromatin capture conformation interactions . . . 116
5.2.3 Autoencoder . . . 117
5.2.4 Wide and deep model . . . 117
5.2.5 Enhancing ChIPmentation signal . . . 119
5.3 LanceOtron - Chapter 4 methods . . . 120
5.3.1 Test data and model performance . . . 121
5.3.2 Motif analysis . . . 122

6 Discussion **123**
6.1 Summary . . . 124
6.1.1 LanceOtron . . . 124
6.1.2 Labelling data . . . 124
6.1.3 Candidate peak calling . . . 125
6.1.4 Measuring enrichment . . . 127
6.1.5 CNN design . . . 128
6.1.6 ChIPmentation . . . 128
6.1.7 Wide and deep model . . . 129
6.2 Contribution to the field . . . 129
6.3 Future . . . 131

A Appendix **133**
A.1 ENCODE training and testing data . . . 133
A.2 Additional LanceOtron benchmarking . . . 135

References **137**

Applying machine learning algorithms to analyse signal from genome sequencing assays

Lance D. Hentges

Abstract

ATAC-seq, ChIP-seq, and DNase-seq have revolutionised molecular biology by allowing researchers to identify important DNA-encoded elements genome-wide. Regions where these elements are found appear as peaks in the analogue signal of an assay's coverage track, and despite the ease with which humans can visually categorize these regions, meaningful peak calls from whole genome datasets require complex analytical techniques. Current methods focus on statistical tests to classify peaks, reducing the information-dense peak shapes to simply maximum height, and discounting that background signals do not completely follow any known probability distribution for significance testing. Deep learning has been shown to be highly accurate for image recognition, on par or exceeding human ability, providing an opportunity to reimagine and improve peak calling. Here, a large, labelled dataset is built by classifying peak and noise regions by hand. These data are used to explore supervised and unsupervised machine learning techniques, as well as to assess the reliability of p-values in chromatin binding assays. The culmination of this work was the development of a peak calling framework, LanceOtron, which combines multifaceted enrichment measurements with deep learning image recognition techniques for assessing peak shape. In benchmarking transcription factor binding, chromatin modification, and open chromatin datasets, LanceOtron outperforms the long-standing, gold-standard peak caller MACS2 through its improved selectivity and near perfect sensitivity. In addition to command line accessibility, a graphical web application was designed to give any researcher the ability to generate optimal peak calls and interactive visualizations in a single step. Furthermore, the general applicability of this technique is demonstrated in chromatin conformation capture experiments. This establishes that the signals extracted and analysed may be found across a range of genomic data, and extending this technique to new datatypes is possible requiring only minor adjustments and model retraining.

LIST OF FIGURES

1.1 Coding and noncoding regions of the genome discovered since 1980. Figure adapted from Gates et al., 2021. 10
1.2 Traditional model of promoter-enhancer interaction, adapted from Haberle and Stark, 2018 11
1.3 Overview of ChIP-seq assay, adapted from Park, 2009 15
1.4 Sequenced fragments from ChIP-seq are aligned to a reference genome, forming characteristic peaks at areas of increased coverage density. Adapted from Brookhaven National Laboratory 16
1.5 Visualisation showing what particular convolutional filters are optimised for. As the layers progress, the filters detect increasingly intricate and complex patterns. Figure adapted from Chollet, 2016. 25

2.1 UCSC browser session showing signal from H3K4me1, H3K4me3, H3K27ac histone ChIP-seq as well as DNase-seq from ENCODE data. H3K4me1 is considered a broad histone mark, while H3K4me3, and H3K27ac are considered narrow histone marks. Compared to broad marks, the signal from narrow histone ChIP-seq is much more similar in shape to transcription factor ChIP-seq and the open chromatin tracks ATAC-seq and DNase-seq. 31
2.2 Selecting and labelling ENCODE datasets. 34
2.3 Candidate peak parameters used for labelling training data. Multiple parameters were varied to create candidate peak calls, shown here with simulated data. A rolling mean reduces the amount of random noise without distorting the signal. This makes the coverage track appear as a smoothed line; the height of the coverage signal is affected by the size of smoothing window used. The threshold affects the coordinates of the enriched region - or if the region is even considered enriched. Averages (λ values) affect how enrichment is defined by varying what the signal in the peak is compared against. 36
2.4 Candidate peaks marked as black bars above the coverage track, identified using the Savitzky-Golay filter for DNase-seq in K562 cells. Despite the high quality of these coverage data, numerous regions, including some with near zero enrichment, were identified by the algorithm. 41

2.5 Raw coverage of fragments (orange) of ChIP-seq experiment with quantity of enrichment definitions passed overlaid (blue). In the 10 kb shown on the left, only the highlighted region qualified as a candidate peak using the dynamic mean algorithm. The detected start and end coordinates of the peak are automatically detected, marked by the vertical dashed lines on the right. 44
2.6 Comparing candidate peaks found with the concurring enrichments algorithm vs called peaks using MACS2 on 10 ENCODE transcription factor ChIP-seq datasets. As expected, there are a large number of unique candidate peaks and relatively few unique MACS2 peaks. 45
2.7 Receiver operator characteristic (ROC) curves of p-values calculated from hand labelled data. P-values were calculated using a Poisson distribution comparing the maximum height of a region to a range of backgrounds. The full view of the ROC as well as the calculated area under the curve show p-values can be used accurately and are robust across a range of different backgrounds (left), however the plateauing end behaviour suggests that some peaks cannot be identified this way and minimising false negatives to zero will drastically increase false positives. 47
2.8 Precision, recall, and F1-score for p-values calculated from hand labelled data. P-values were calculated using a Poisson distribution and using a range of backgrounds as well as the chromosome mean. Consistent with the ROC analysis, the values calculated for these metrics were robust across p-value calculations with the exception of chromosome mean. 48
2.9 Dimensionality reduction using PCA, t-SNA, and UMAP of labelled peaks and noise. The values included for each region were 10 p-values, calculated based on its maximum height compared against a background of 10 kb to 100 kb. PCA, a linear technique, showed the most obvious separation between classes. 48
2.10 Accuracy using a logistic regression model to predict peak and noise classifications from the labelled dataset. Models were built using different measurements for the regions signal (maximum height, average height, and maximum height after 400 bp smoothing), range of background (50 kb, 100 kb, 200 kb, and 400 kb), and number of p-values calculated within the range (10, 20, 40). 49

3.1 Error rate of the top performing models used at the ImageNet competition 2011 to 2016. 58

3.2 Principles of convolutional filters. Top: Applying a convolutional filter which looks for a diagonal pattern in the simplified image. The dot product between the subsection and the filter is calculated, returning a value scaled to how well the pattern matched. N.B. for CNNs, the weights of the filter are randomised, and trained such that the most useful patterns are detected. Bottom: The convolutional filter is applied across all coordinates, creating a new representation of the image; convolutional filters can be applied to this as well to find patterns within the patterns. 59
3.3 Structure of a convolutional autoencoder. Mirror images of a conventional CNN are centred around a layer whereby all output matrices are flattened and built up again through upsampling and convolution; the target output is the same as the input, making this an unsupervised learning technique of machine learning. 63
3.4 Three inputs signals used for the CNN model. For each regions of interest, a 3x2000 matrix of signal, in the form of normalised coverage, is delivered to the model: 2000 base pairs (bp) of surrounding the region (blue), signal only from the region of interest expanded to 2000 bins (red), and 20,000 bp compressed into 2000 bins (green). 67
3.5 Accuracy and loss collected across training epochs for the CNN-only peak caller. Training was stopped before accuracy increased in training data but decreased in validation data which would indicate over fitting. 68
3.6 Receiver operator characteristic (ROC) curve with calculated area under the curve (AUC) for each of the different classifications of the CNN-only peak caller. The full plot (left) and the zoomed in plot (right) both show near perfect accuracy for this dataset. 69
3.7 Confusion matrix showing class predictions from the CNN-only peak caller. These testing data were composed of unambiguous peak and noise regions from five experiment types and were unseen during model training. Noise regions were combined from all data sources, and also indicate the models performance for detecting peaks versus noise (without predicting the experiment type). Near perfect scores were achieved for each classification type. 70
3.8 Comparison showing the HBA locus using several common chromatin conformation capture techniques, highlighting the high resolution capabilities of MCC. Figure adapted from Hua et al., 2021. . 71
3.9 When consensus peak calls were used from the different methods, they overwhelmingly were found with the same TADs as the loci being captured. Figure adapted from Hua et al., 2021. 73
3.10 Venn diagram showing the number of different regions identified by three peak calling methods: a convolutional neural network (CNN), a Poisson model, and MACS2. 74
3.11 t-SNE of the central latent layer of an autoencoder trained with candidate peaks. Unseen test data include labelled peaks and noise as well as unlabelled regions (left). The latent layer outputs were also input into a Bayesian Gaussian mixture model to predict clusters, shown in blue to yellow, present in the data (right). 75

3.12 Wide and deep model overview. A candidate peak (highlighted blue) is compared to its surroundings from multiple viewpoints; enrichment measurements are input into a logistic regression model. Signal from the candidate region's central 2 kilobases (kb) are input into a convolutional neural network (CNN). Outputs from the logistic regression model and CNN are combined with the multiple enrichment measurements in a multilayer perceptron which outputs a final overall peak score. 77
3.13 Model accuracy compared across pooling type (top left two panels), convolutional blocks (bottom left two panels), convolutional layers per block (top right two panels), and total convolutional filters (bottom right two panels). 5,000 models were created with the top models, defined as those with accuracy of 98.5% or greater when classifying unseen data (left panels of the sets of two), compared to all models (right panels of the sets of two). 79
3.14 Model accuracy compared against filter (kernel) size (top) and number (bottom). 5,000 models were created with the top models, defined as those with accuracy of 98.5% or greater when classifying unseen data (left), compared to all models (right). 80
3.15 Five fold cross validation strategy for training and testing (left), and the accuracy of the top 10 models found from hyperparameter testing. The structure and hyperparameters from the top performing model (99.89% average accuracy) were used for subsequent work using the wide and deep model design. 80
3.16 Autoencoder pretraining strategy. The structure of the CNN (left) was used to design the autoencoder (middle), perfectly replicating it on the input side and of the autoencoder while using a mirrored structure for the output side. Once the autoencoder is trained, the weights it learns can be transferred to the CNN (right). This strategy can be employed to include additional data not used to train the CNN. 82
3.17 Examples of predicted ChIP-seq signal using a generative adversarial network to transform ChIPmentation signal, as well as the original ChIPmentation and ChIP-seq coverage tracks for the corresponding areas. While the correlation improved overall using the GAN, many regions were only very slightly modified (top) to improve signal, while others were grossly transformed (bottom). . . . 83

4.1 Histogram showing the total number of candidate peaks called by LanceOtron and their associated Peak Score over all in-house datasets . LanceOtron's deep learning model tends to score enriched regions being either peak or noise at high probabilities. 95
4.2 Reanalysis of ENCODE peak call using Score Peaks module. Signal from each peak is reduced to 2D using t-SNE and coloured using LanceOtron's Peak Score, with thumbnail images selected from areas with a high Peak Score and a low Peak Score. 96
4.3 Comparison of peak calling for CTCF ChIP-seq 97
4.4 Comparison of CTCF motifs found for peak calls 98

4.5 Venn diagram of CTCF peak calls from LanceOtron and MACS2. Regions which did not intersect were assessed for overlap with promoters or enhancers. 98
4.6 A, Thumbnail images from the most highly enriched regions called exclusively by either LanceOtron (top) or MACS2 (bottom). B, Average coverage of the regions called exclusively by either LanceOtron (top) or MACS2 (bottom) for CTCF experimental track, control track, and DNase-seq open chromatin track. 99
4.7 Benchmarking LanceOtron against MACS2 for peak calling histone ChIP-seq. A, Model performance metrics using 10 Mb of labelled genomic regions of ENCODE ChIP-seq datasets for H3K27ac in HAP-1 cell line, and B, H3K4me3 in MG63 cell line. C, ChIP-seq dataset labelled by Oh et al. for H3K27ac in GM12878 cell line and D, H3K4me3 in K562 cell line. 100
4.8 Benchmarking LanceOtron against MACS2 for calling open chromatin. A, Model performance metrics using labelled genomic regions of an ENCODE ATAC-seq dataset in MCF-7 cell line and B, DNase-seq in A549 cell line. 102
4.9 Screenshots taken from the LanceOtron webtool. Three module options are given depending on analysis needed (left). Required data fields and coverage track upload (tutorial page shown, right). 103
4.10 Screenshots taken from the LanceOtron webtool. Completed analysis showing interactive charts, built in genome browser, and quick navigate bed file with thumbnail previews. 105

A.1 Performance benchmarks of LanceOtron compared to MACS2 with various parameters. MACS2 performance scores are obtained using a default q-value of 0.05, while MACS2-strict uses a q-value of 0.01 (the same value as Chapter 4); MACS2-broad uses the broad parameter. LanceOtron-MACS2 uses LanceOtron's deep learning model to score MACS2 peak calls, using them as candidate peaks. 136

LIST OF TABLES

1.1 List of published peak callers. 18

2.1 Comparison of candidate peak attributes from the traditional window-threshold method (window=400; threshold=4*chromosome mean) and Savitzky-Golay filter (algorithmic window size and derivative-determined region boundaries) to MACS2 final peak call. 41

4.1 LanceOtron and MACS2 peak call comparison for transcription start sites (TSSs), and for active regions in open chromatin. Percentages and counts of peaks intersecting TSSs or active areas of the genome are given for 5,000 regions of LanceOtron and MACS2 peak calls, selected for being most enriched (highest Peak Score or q-value for LanceOtron and MACS2 respectively). 101

4.2 Performance benchmarking comparing LanceOtron with MACS2. . 104

A.1 ENCODE training and testing data. 135

CHAPTER

ONE

INTRODUCTION

The foundation of the modern field of genomics is closely tied to the works of the Human Genome Project (HGP) (Gates et al., 2021). Completed in 2001, the HGP published the human genome's sequence of nucleotides and detailed the locations of protein coding regions (Craig Venter et al., 2001; Lander et al., 2001). Alongside this came many optimistic predictions about the future of medicine and disease, notably including those from famed geneticist and director of the Human Genome Project Francis Collins. Undoubtedly, the progress in understanding Mendelian disease has been substantial and largely lives up to the lofty ideals purported at the turn of the millennium (Alkuraya, 2021). Mendelian diseases are caused by mutations occurring at a singular region of the genome, and produce a Mendelian pattern of inheritance. Often these disorders are caused by mutations in protein coding DNA for a single gene, though exceptions occur (French & Edwards, 2020), notably including some long studied thalassemias (Higgs et al., 1989). Beyond Mendelian disease, expectations for progress were also high for genetically complex diseases such as cancer and diabetes, however as these disorders continued to be pervasive some of these predictions have attracted criticism (Joyner & Paneth, 2019). Not all of Collins' visions for the future came to pass, especially in reference to some complex diseases or personalised

medicine. Progress, however, has been undeniable and alongside his predictions he presciently included the Bartha Knoppers quote, "As the radius of knowledge gets longer, the circumference of the unknown increases even more" (Collins & McKusick, 2001).

1.1 Regulation in the genome

At the turn of the millennium, viewing genetic disease through a lens focused on protein coding DNA was warranted given the relatively small number of non-coding elements discovered at this time (Fig. 1.1). Indeed, the turn towards noncoding DNA's role in regulation was made possible by the HGP's creation of a comprehensive list of protein coding regions. It has become increasingly clear that many causes of common diseases are not associated with aberrations to protein coding DNA, nor are they Mendelian in nature (which tend to exhibit more binary characteristics), but rather are connected to precise gene expression levels at a given time and place. Supporting this, the first published catalogue of genome wide association studies (GWAS), which connect single nucleotide polymorphisms (SNPs) to known traits or diseases, showed the overwhelming majority of SNPs were found in noncoding regions (Hindorff et al., 2009). Critically it was discovered that these SNPs were potential genomic regulatory elements (Maurano et al., 2012), highlighting the importance of techniques used to investigate these crucial elements. Central to the results of this work is the structure of chromatin, which is composed of DNA wrapped around histone proteins called nucleosomes, and linker DNA connecting adjacent nucleosomes. Regions which are not tightly bound by nucleosomes, called open chromatin regions, are considered active because they allow interaction directly with the DNA sequence and are frequently associated with regulatory elements. In the Maurano et al. work the authors showed these SNPs were not typically found in active regions of the genome for all tissue types examined, however they were found, or were in per-

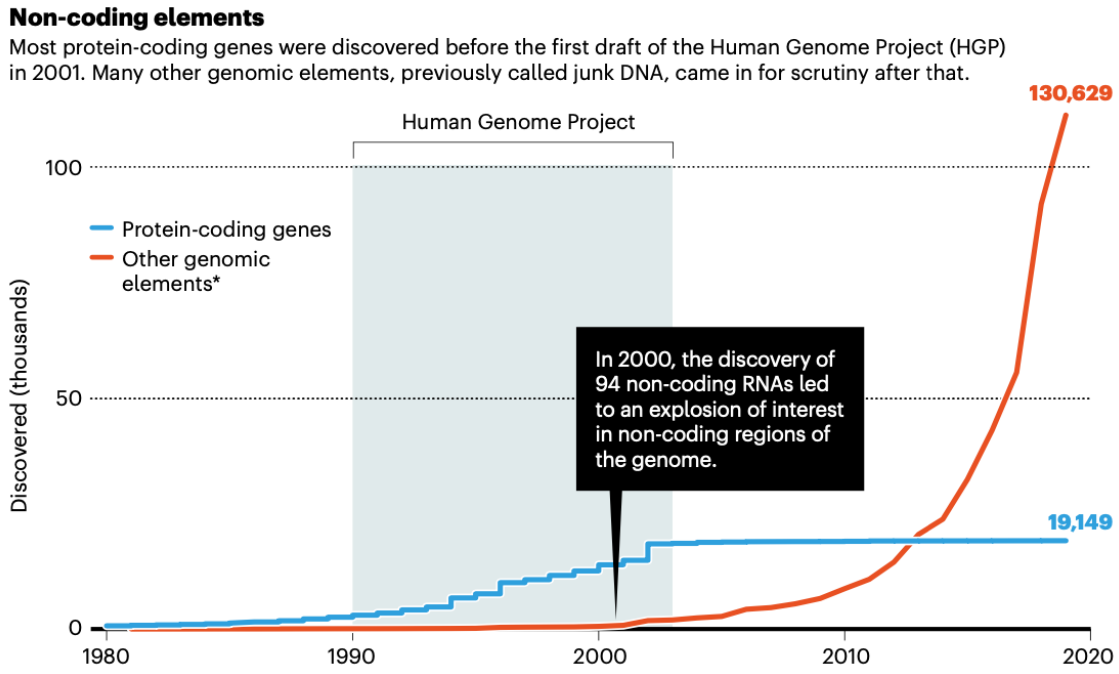


Figure 1.1: Coding and noncoding regions of the genome discovered since 1980. Figure adapted from Gates et al., 2021.

fect linkage disequilibrium with, active regions of the genome for at least one type of cell 76.6% of the time. Importantly some of these SNPs were found in regions that were selectively active in cell types which were pathogenically relevant - for instance finding SNPs associated with a cardiovascular disorder at active regions in heart tissue, or neurological SNPs found at active regions in brain tissue - providing strong evidence these DNA variants affected genome regulation. Because genome regulation has the capacity to drastically alter gene expression it plays an important role in complex disease, but this same mechanism also acts as the main driver of tissue specificity.

1.1.1 Gene expression

Despite the incredible diversity of specialised cell types in human body, cells all rely on the same genome for instructions. This feat is achieved through regulatory elements which work to affect cellular gene expression. Two types of elements are particularly important in this process: transcription factors (TFs) and *cis*-regulatory elements (CREs). Cell-type-specific TFs bind to specific regions in

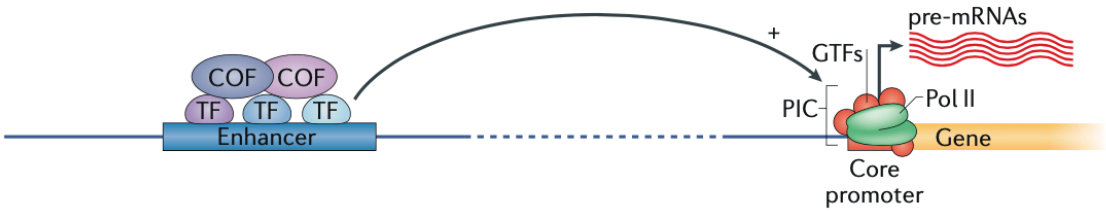


Figure 1.2: Traditional model of promoter-enhancer interaction, adapted from Haberle and Stark, 2018

the genome, which are determined by the complex interplay of DNA sequence-based TF binding motifs. These then recruit non-cell-type-specific general transcription factor machinery to the required regions in the genome and facilitate cell-specific gene expression; CREs are the noncoding sequences of DNA containing these binding sites and include promoters and enhancers (Wittkopp & Kalay, 2012). Genes are actually expressed through the process of transcription, whereby RNA polymerase II (PolII) converts the sequence of DNA to RNA (initially pre-messenger RNA or pre-mRNA), which is then followed by processing and eventual translation to protein. The core promoter contains the transcription start site (TSS), the genomic location where transcription begins, and extends out roughly 50 base pairs (bp) upstream and downstream. PolII and general transcription factors (GTFs) bind at the core promoter and make up the multi-protein unit called the pre-initiation complex (PIC). A promoter-bound PIC is sufficient to drive low level basal transcription, but increased expression is achieved by cell-specific TFs binding to CREs - either directly upstream of the promoter, or further away at enhancers. (Haberle & Stark, 2018). Enhancers are located distally from the promoter - sometimes even millions of base pairs away - but are in close physical proximity (i.e. in the 3D space of the nucleus) during gene expression (Fig. 1.2). Hundreds of thousands of enhancers are located across the genome, but only a selection are active in a given cell type.

Enhancers can be categorised as existing in one of four states: active, primed, poised, and inactive. Active enhancers exhibit several characteristic features: they are found in areas of open chromatin, bound by TFs and PolII, and pro-

duce enhancer RNA (eRNA). Primed enhancers are distinguished from active enhancers by the lack of production of eRNA and no binding of active chromatin marks, indicating that other cofactors are needed to trigger the enhancer's function. No eRNA is produced with poised RNA either, but this type carries an additional repressive signal on the enhancer associated nucleosome. Finally, inactive enhancers are not bound by transcription factors, are not associated with any histone modifications, and are found in regions of closed chromatin (Heinz et al., 2015).

The combination of TFs and CREs are able to affect gene expression by altering the chromatin environment locally and at a distance. While the precise mechanisms are an active area of research, there are measurable changes that occur which are used inform current models. Transcription factor binding sites (TFBSs) often have a specific sequence motif where the TFs bind, and although common in the human genome, only a small proportion of these motifs are stably bound by the TF (Lambert et al., 2018). Unbound TFBSs are associated with closed chromatin, and transcription factor binding opens up chromatin through nucleosome remodelling, making DNA further accessible to other TFs (Zaret, 2020). These changes are often accompanied by modifications to the local nucleosomes, specifically histone tail acetylations and methylations, which are laid down by the subsequent recruitment of general transcription factors. The patterns of histone modifications and binding of general TFs vary depending on the type of regulatory elements and can be useful for the annotation of regulatory elements into enhancer or promoter elements (Heinz et al., 2015).

Further complicating our understanding of regulatory processes, enhancers control genes from a distance and so are often referred to as distal regulatory elements. These distances can be highly variable, with some enhancers controlling genes only a few Kb away while others lie millions of base pairs away (Shlyueva et al., 2014). Another complication is that multiple enhancers may regulate a single gene, or a single enhancer may control more than one gene. It has now been

shown that such distal elements control their target genes by physically interacting in the three dimensional nuclear space of the nucleus (Long et al., 2016). This introduces another type of regulatory mechanism: boundary elements. Enhancers affect the transcription from gene promoters by physically interacting with them in the nucleus. Boundary elements act to prevent the interactions of certain enhancers and promoters by blocking their physical interactions in the nucleus, and so are part of the mechanisms that provide specificity in enhancer/promoter interactions.

While it is not known how boundary elements precisely work, the mechanism thought to best explain this process is the loop extrusion model (Fudenberg et al., 2016). This is an extremely active area of research and an ever developing body of both experimental and modelling work has supported and refined this model. Central to this is the TF CCCTC-binding factor (CTCF) and the protein complex cohesin. In this model cohesin is loaded on to the DNA and the strands are progressively extruded until DNA-bound boundary elements are reached. CTCF sites are proposed to block loop extrusion in an orientation dependent manner, reflected in the directionality of the CTCF TFBS. It has also been proposed that promoter-enhancer interactions may additionally block the DNA extrusion through cohesin.

1.1.2 Features of regulatory elements

Cis-regulatory elements are associated with characteristic chromatin features. While all CREs are typically found in areas of open chromatin, promoters and enhancers actually share other important similarities: they are both TF bound, and are associated with PolII which produces RNA (pre-mRNA at promoters and eRNA at enhancers). Furthermore, at the local nucleosome, histone 3, lysine 27 is acetylated (H3K27ac) in promoters of actively transcribed genes as well as active enhancers. However promoters can be distinguished from enhancers using the ratio of Histone 3 lysine 4 methylations, (specifically di- and trimethylation,

abbreviated H3K4me2 and H3K4me3 respectively) compared with monomethylation (H3K4me1). While promoters are frequently associated with high levels of H3K4me2/H3K4me3 compared to H3K4me1, enhancers show the reverse trend. Poised enhancers can also be detected from histone modifications, as this type is also associated with a lack of H3K27ac and has been reported to be enriched for histone H3 lysine 27 trimethylation (H3K27me3) (Heinz et al., 2015).

1.2 Locating regulatory elements

Determining if a genomic location is in a region of open chromatin, transcription factor bound, or if the local nucleosomes have any histone modifications present is central to understanding how a site may exhibit regulatory control over gene expression. Locating and annotating these regulatory regions is therefore critical to understanding both cell specificity and complex disease. This can be accomplished through molecular biology assays that selectively return the DNA sequences of regions which exhibit certain properties, including chromatin immunoprecipitation followed by sequencing (ChIP–seq) for TF-binding and histone modifications, and ATAC-seq and DNase-seq for open chromatin.

ChIP-seq was one of the early techniques utilising DNA sequencing in an assay, with the first known studies taking advantage of this technology in 2007 (Barski et al., 2007; Johnson et al., 2007; Mikkelsen et al., 2007; Robertson et al., 2007). It works *in vivo*, by first crosslinking DNA-binding proteins to DNA using formaldehyde. The chromatin is then sonicated into small fragments, and an antibody specific to the protein of interest is used to pull down the DNA-protein complexes via immunoprecipitation. The crosslinking process is reversed, and the protein associated DNA is sequenced (Park, 2009)(Fig. 1.3).

Unlike the DNA tightly wrapped in nucleosomes in closed chromatin, regions of open chromatin are hypersensitive to DNase-I cleavage. In DNase-seq, the chromatin in isolated nuclei is exposed to the action of the endonuclease DNase-

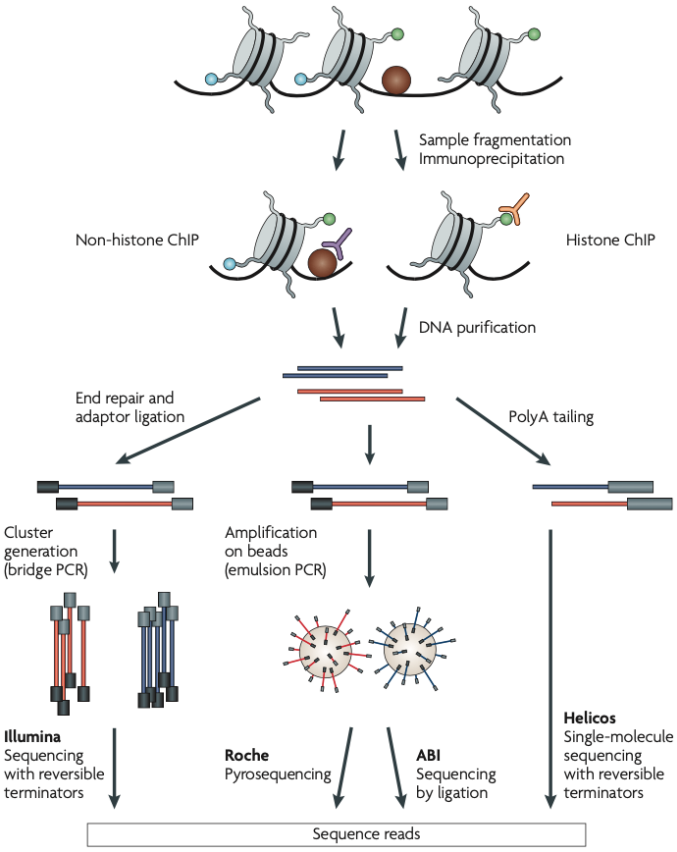


Figure 1.3: Overview of ChIP-seq assay, adapted from Park, 2009

I, which preferentially cuts the DNA at regions of low nucleosome density - these regions are therefore referred to as “DNase-I hypersensitivity sites” (DHSs). Fragments are size selected, as the regions cut by the DNase-I are considerably smaller on average, and then sequenced to create a map of DHSs which effectively marks areas of open chromatin (Boyle, Davis, et al., 2008).

Conceptually, locating open chromatin with ATAC-seq is similar to DNase-seq, but replacing DNase-I with a hyperactive Tn5 transposase. This has the added benefit of adding the necessary adapters required for sequencing the fragments in the same step which drastically decreased the number of cells required and experimental complexity at the bench (Buenrostro et al., 2013).

Data from ATAC-seq, ChIP-seq, and DNase-seq are processed in a similar fashion: enriched DNA fragments are sequenced, aligned to the genome, and areas enriched for these fragments are recorded. These data are arranged as coverage tracks across genomic coordinates, and increases in fragment density

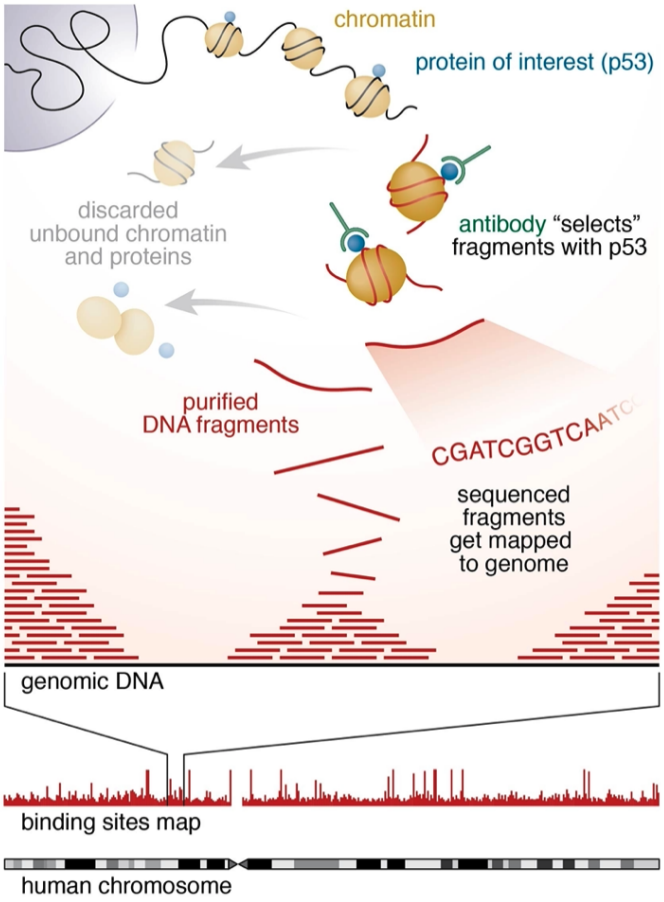


Figure 1.4: Sequenced fragments from ChIP-seq are aligned to a reference genome, forming characteristic peaks at areas of increased coverage density. Adapted from Brookhaven National Laboratory

at true-positive biological events are called “peaks" because of the characteristic pattern of fragments produced in these areas (Fig. 1.4). Besides these regions, enrichment also occurs due to biases and noise in the experimental procedures (Park, 2009) or systematic mapping errors common to areas of low complexity (Amemiya et al., 2019). Creating algorithms that can distinguish peaks from such experimental and computational noise, and are robust across methodologies, sequencing depth, diverse tissue types, and chromosomal structure has remained a challenge.

1.2.1 War and peaks

Identifying truly enriched regions in such experiments is at the heart of modern genomics. It is critical for both understanding general functional mechanisms in the genome as well as for annotating cell type specific functionally important regions to understand processes such as cellular differentiation and to interpret non-coding mutations. However, the extraction of information from such data by "peak calling" is not a solved problem, and perhaps the strongest evidence of this is the sheer number of tools that have been developed to address what is conceptually a simple task (Table 1.1). In nearly all the peak callers listed, and in the most popular peak caller used currently, MACS2, real peaks are distinguished from noise using statistical tests that compare enrichment from the region to background, which is assumed to consist of signal generated randomly. While the Poisson distribution models this better than other distributions (Thomas et al., 2016), background is in fact nonrandom (Wilbanks & Facciotti, 2010), appearing at increased levels in areas of open chromatin (Auerbach et al., 2009), at sites with inherent sequence bias and over regions of varying copy number (Vega et al., 2009). This must be considered when reviewing significance from statistical peak callers, as misclassification will occur at a higher rate than the p-value suggests. Relying solely on these significance scores may lead to high false positive rates, but also leaves room for potential false negatives, with the ratios of these errors depending on the parameters selected. Exacerbating this, default settings are routinely used, reducing accuracy nearly 10% on average from tuned parameters when using statistical peak callers (Hocking et al., 2016). With these tools, errors may be reduced by using matched negative controls (also known as input tracks) to calculate the level of background noise, though this increases the time and costs of the experiment. While peak callers such as MACS2 (Y. Zhang et al., 2008) do not strictly require negative control tracks, forgoing them may sacrifice performance (Stanton et al., 2017). Input tracks do control for some experimental bias but are still sensitive to chromatin activity, making statistical tests more prone

Peak caller	Citation
AREM	Newkirk et al., 2011
BayesPeak	Spyrou et al., 2009
BroadPeak	J. Wang et al., 2013
ChIPDiff	Gogol-Döring and Chen, 2012
CisGenome	Ji et al., 2008
CNN-Peaks	Oh et al., 2020
CSAR	Muiño et al., 2011
Dfilter	Kumar et al., 2013
dPeak	Chung et al., 2013
DROMPA	Nakato et al., 2013
EDD	Lund et al., 2014
F-Seq	Boyle, Guinney, et al., 2008
FindPeaks	Fejes et al., 2008
GEM	Guo et al., 2012
GeneProf	Halbritter et al., 2012
HATSEQ	Taskesen et al., 2013
HMCan	Ashoor et al., 2013
HOMER	Heinz et al., 2010
Hpeak	Qin et al., 2010
JAMM	Ibrahim et al., 2015
LONUT	R. Wang et al., 2013
MACS2	Y. Zhang et al., 2008
MICSA	Boeva et al., 2010
MOSAiCS	Kuan et al., 2011
MSPC	Jalili et al., 2015
NEXT-peak	Kim et al., 2013
PeakRanger	Feng et al., 2011
peakROTS	Elo et al., 2012
PeakSeq	Rozowsky et al., 2009
peakzilla	Bardet et al., 2013
PICS	X. Zhang et al., 2011
PolyaPeak	Wu and Ji, 2014
Q	Hansen et al., 2015
Qeseq	Micsinai et al., 2012
QuEST	Valouev et al., 2008
Ritornello	Stanton et al., 2017
RSEG	Song and Smith, 2011
SICER	Zang et al., 2009
SIPeS	C. Wang et al., 2010
SISSRs	Narlikar and Jothi, 2012
T-KDE	Y. Li et al., 2014
T-PIC	Hower et al., 2011
ZINBA	Rashid et al., 2011

Table 1.1: List of published peak callers.

to false negatives (Auerbach et al., 2009).

To address the well-known problems of peak callers, analysis pipelines employing quality control steps are common. The Encyclopedia of DNA Elements (ENCODE) consortium hosts numerous chromatin profiling assay datasets (Davis et al., 2018; Dunham et al., 2012) and has developed a robust set of guidelines including recommendations for input controls, sequencing depth, library complexity, and exclusion list regions where mapping errors are more prone to occur (Amemiya et al., 2019). Multiple replicates are encouraged, and procedures exist for combining peak calls for the most efficient reduction in error (Q. Li et al., 2011). Although these extensive measures greatly improve the reproducibility of peak calls, high-throughput visual inspection showed numerous erroneous peak calls remain (Sergeant et al., 2021).

The inability to reproduce published results is a prevalent concern amongst researchers and is due in part to the unintentional misuse of statistics (Baker, 2016). These issues include overstating the meaningfulness of the statistical test results and conflating significance with effect (Nuzzo, 2014) - traps commonly used peak callers fall prey to. Peak callers relying on statistical models simplify the complex analogue signal of a region into a single value (maximum height) and use it to calculate a p-value. Enrichment is calculated against a background signal incorrectly assumed to follow a known distribution, and this sole measurement is then falsely equated with peak quality and used to filter results. Quality control of the final peak call is typically limited to uploading the significant regions and coverage track to a genome browser such as UCSC (Kent et al., 2002) or IGV (Robinson et al., 2011), where sections of the genome can be manually scanned. Using only these tools makes anything beyond a cursory inspection tedious and impractical, but because of the incomplete link between statistical test results and peak quality, thoroughly exploring and refining peak calls is of particular importance.

Though extremely time consuming when done at scale, researchers have

been shown to effectively judge the quality of peaks using a genome browser. Rye et al. measured peak caller performance by creating a dataset of visually verified peak calls using the UCSC genome browser, and inadvertently measured the performance of the humans in the process (Rye et al., 2011). They found that transcription factor motifs, known to be associated with true biological signals, were recovered more often from the manually labelled peaks than from the peak callers. Amazingly they also found that 80% of the software's false positives could be detected even without an input track, because the human peak callers could identify that these regions "lacked the expected visual appearance of a typical ChIP-seq peak". Furthermore, while classifying regions by eye is seemingly dependent on an individual, Hocking et al. demonstrated a high consistency across labellers when judging peaks (Hocking et al., 2016). Visual inspection can be a credible method for peak calling, and software such as MIG (McGowan et al., 2013) and MLV (Sergeant et al., 2021) greatly improve the visual inspection process. However even with these tools, to comprehensively scan an entire human genome would be nearly impossible.

1.3 Machine learning in biology

In biology the relationships between experimental variables and outcomes are often extremely complicated. Even in model systems there are too many variables to be perfectly controlled, and making invariant, hard and fast rules can result in oversimplification beyond usefulness or just being wrong. In these situations, Machine learning (ML) can be used to find patterns in the data undetectable by humans, especially true when using large datasets with multiple outcome measurements. These are often the exact type of data available in genomics, and therefore applying machine learning algorithms here is a worthwhile venture capable of yielding important results.

Simply put, ML is programming computers to learn from data (Géron, 2019).

The "learning" aspect of machine learning means that rather than an algorithm carrying out the same operations regardless of the input, architecture is built to find and utilise the best parameters for the algorithm given the data, such that these parameters optimise some performance metric like accuracy or standard error. Most ML applications in biology can be categorised as either supervised or unsupervised learning. Within a ML dataset composed of many samples, each sample has a number of properties that are measured or known called features. In supervised learning, there is an additional supplied piece of information, called the target. Using the features as input, a supervised learning algorithm attempts to output a sample's associated target - these are typically used for regression, e.g. given a set of protein structures (features) of a molecule, what is the expected bioavailability in the bloodstream (target), or classification, e.g. given a set of gene expression levels (features), what is the cell's type (target). In unsupervised learning the target is not another external measurement or classification, but rather a value that can be calculated or inferred from the data itself. These algorithms are typically used for dimensionality reduction, e.g. if 1000 features exist for each sample, which 10 features exhibit the most variation between samples, and clustering, e.g. if there are multiple cell types in a population, how can the cell types be assigned to samples such that the variation of features within the same cell type is minimised.

Parameters play an important role in the structure of ML algorithms, as they are the values (called weights in many contexts) that the features are transformed by. For instance in a classification task, the ML algorithm would multiply each of the features of the input by some positive or negative weights, and output a score for each class present in the targets. If the performance metric being optimised was accuracy (typical for classification tasks), then the weights would be adjusted such that it maximised the correct class prediction for the dataset and minimised the incorrect classes. Data would be presented to the algorithm over multiple iterations until the accuracy plateaued or a fixed number of iterations

through the data, called epochs, was reached (N.B parameters that affect how the parameters/weights are updated are called hyperparameters). The major steps of a ML process can be summarised the following way:

1. **Data processing and algorithm selection** - Having representative data from the problem being addressed is critical, and often this requires construction and curation of a large dataset. The actual size of the dataset and artefact tolerance depends on the algorithm selected, heterogeneity of data, and performance requirements. Generally this process works in reverse, in that the data available and the information sought will largely direct the best type of algorithm to use. Capacity is a measurement based around the number of parameters a model has, and is an important consideration. A higher capacity generally means more complex pattern or subtle differences can be discerned from the data. Overfitting is a supervised learning problem where the model learns the target output not from general feature properties, but from essentially "memorising" specific samples. This has a greater potential to occur with high capacity models where the number of parameters exceeds the number of observations, and can be alleviated with very large datasets for training the model.

2. **Training the ML model** - The number of iterations that the model cycles through the data optimising parameters on can also cause overfitting. Generally datasets are divided into the training set, validation set, and testing set. The training set usually composed of around 80% of the total data, typically selected randomly, and is the only data the model uses to learn parameters from. The validation set is used as an additional measure of model performance, and to detect overfitting - if a model performs well on the training data, but poorly on the validation data, this is a sign the model simply "memorised" the training set and will not perform well on unseen data. Lastly, the testing set is used after all model adjustments are made to take the final performance measurements. Other hyperparameters, such as

learning rate or batch size, can also have a profound impact on algorithm behaviour, and must be properly explored to optimise model performance.

3. **Application** - Ultimately, a trained ML algorithm is used to make predictions or reveal information on unseen data.

1.3.1 Deep learning in genomics

With the introduction of next generation sequencing technology, the field of genomics was flooded with large and complicated datasets. This simultaneously coincided with a precipitous increase in computational power. These two factors laid the groundwork for the introduction of deep learning, a subfield of ML, in genomics: higher capacity models could be built and trained in weeks rather than years, and the increased depth of sequencing data allowed for their utilisation without fated overfitting. There are now numerous deep learning applications employed in genomics now, including predicting 3D genome folding (Schwessinger et al., 2020) or estimating the effects of sequence alterations on chromosomes (Zhou & Troyanskaya, 2015).

DL is often seen as cutting edge technology, but actually the framework for these models has existed for many decades, first coalescing out of work done on artificial neural networks (ANNs) the 1940s. However it was the 1950s brought about a monumental leap forward in the field when Frank Rosenblatt created a type of classification model capable of distinguishing between two categories by "learning from experience", using a model he called the perceptron. A perceptron, and ANNs in general, are composed of artificial neurons, which are actually based on biological neurons of the nervous system. Their function is straightforward: compute the weighted sum from multiple input features plus a bias value, and this total is passed to an activation function. In a classical perceptron design, if the weighted sum total of the input is positive then the neuron's output would be 1, otherwise its output would be 0 - this is analogous to a neuron receiving enough stimulus to generate an action potential as an all or nothing response. What the

ML algorithm learns is where to set the weights that are multiplied by the features, such that the neuron's activation function fires only when the sample belongs to the correct class, thus maximising accuracy.

Modern deep learning algorithms are remarkably similar to Rosenblatt's design, but with added levels of organisation. Artificial neurons remain central to the structure, but are composed as feed forward layers of neurons. A layer consists of multiple neurons, and the outputs from their activation functions are passed as inputs for the next layer of neurons. When a neural network has more than a single layer of neurons, it is considered a deep neural network (DNN), and when a DNN has three or more layer, any layer between the input and output layer is termed a hidden layer. With other ML algorithms careful attention must by directed towards selecting, and sometimes transforming or combining the most informative features - this process is called feature engineering and is often extremely difficult and time consuming. However with DNNs raw data can be used, and the layers will determine which features in which combinations are most informative by reducing the weights of uninformative paths. For this reason, much of the focus from feature engineering has shifted to model design and hyperparameter tuning in DL.

One of the most important types of DNNs today are convolutional neural networks (CNNs). CNNs are largely responsible for driving up the popularity of DL, a feat achieved because of their impressive image recognition performance. These work by using convolutional filters, essentially matrices of weights, optimised during model training to detect certain patterns. The pattern detection at early layers is simple, but as the feed forward network progresses, the patterns learned from the patterns detected in the early layers become increasingly complex (Fig. 1.5). Generally CNNs use a large number of convolutional filters and are capable of achieving extremely accurate classifications.

Training such powerful models comes with its own challenges; CNN models are very prone to overfitting because of their high capacity. Creating very large

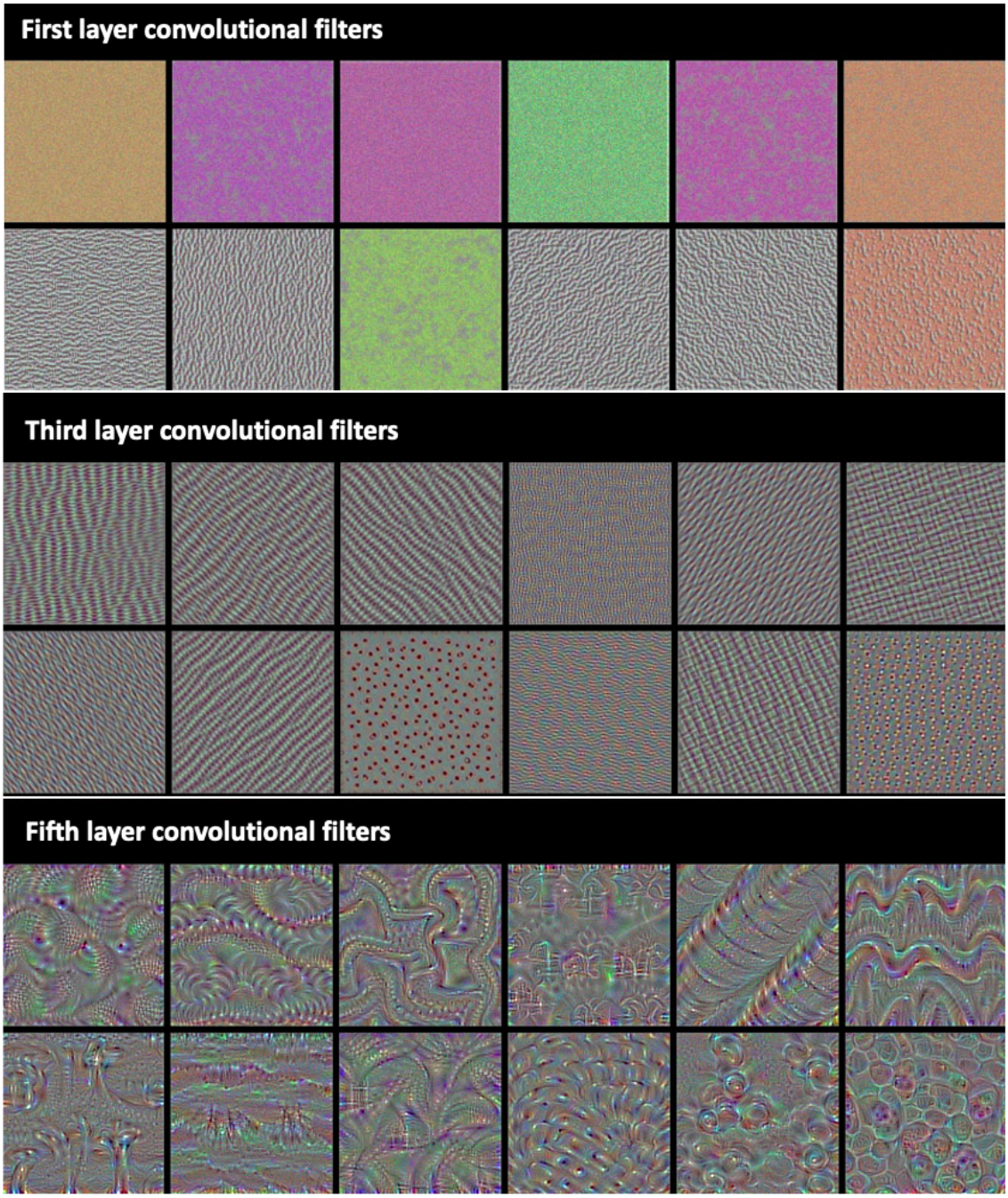


Figure 1.5: Visualisation showing what particular convolutional filters are optimised for. As the layers progress, the filters detect increasingly intricate and complex patterns. Figure adapted from Chollet, 2016.

datasets is an effective remedy to this problem, but this often requires the time consuming process of sample labelling. One strategy used to extend the usefulness of labelled data, and reduce the necessary number of samples is data augmentation. This technique is commonly used with image classification tasks, and involves making slight adjustments to the size or position of an image in the training data. This increases the number of training samples available and makes the model more invariant to slight changes to an image that have no meaningful impact on its classification. This does however need to be done in balance with the variation seen in the samples within the different classes. For instance imagine an image classification model designed to distinguish between cats and dogs, and uses thousands of images of different dogs, but only one image of a cat (named Chester) with thousands of modifications to the image (shifted up one pixel, rotated one degree clockwise, etc.). If this model were to classify a random image of a cat, it may strongly identify the image as "not Chester" and may lead to poor performance. For classification tasks, representative datasets which are equally balanced between class types are important for effective model training. Small imbalances may be accounted for by offsetting how the weights are updated during the training process, however depending on the data used some samples may need to be removed by downsampling.

1.4 Thesis aims

The special structure of CNNs, implemented using principles of machine learning, allow them to achieve extremely high performance for pattern detection. Though recognised because of their capability in image detection, they have since been widely adapted to solving problems in genomics. Peak calling has long been a problem in the field because it has been primarily addressed with low capacity statistical models. Finding the locations of regulatory elements is of central importance to understanding biology, as it informs cell specificity and adaptability.

Additionally it is critical to human health, as disregulation is the primary driver of some of the most prevalent diseases in the world. Therefore the aims of this thesis are as follows:

1. **Construct a labelled dataset and find algorithmic methods of detecting regions in a genomic coverage track for classification**
2. **Explore different CNN structures and their abilities to interpret coverage data**
3. **Build fully featured peak caller and benchmark against current tools**

CHAPTER

TWO

DATA AND CANDIDATE PEAKS

2.1 Introduction

Detecting the locations of biological events for ATAC-seq, ChIP-seq, or DNase-seq requires the use of peak callers for large genome wide searches. Peak calling can be broken into two separate problems: finding the coordinates of enriched regions, named candidate peak calling, and determining the regions significance. There are various ways to accomplish this, but generally the first step is to apply a smoothing filter to the genome coverage track with the intention of removing noise without distorting the pattern of coverage. Implemented as a rolling mean, the smoothing process works by setting a coordinate's value to the average signal from the surrounding region at a set window size. Afterwards enrichment is determined and candidate peaks called, merging adjacent coordinates above some threshold into a single region. The term "enrichment" is frustratingly vague and so its detection is implemented in various ways when peak calling. Here it may indicate signal above the average of a chromosome, some number of surrounding base pairs, or a comparable region in an input control track - it may also mean that the signal is measured at a certain p-value. The value used for the signal in the coverage track is also unclear and variably implemented, as it could be

the raw count of overlapping reads at the coordinate, smoothed signal, or averaged signal over the entire region of interest. Multiple definitions for enrichment therefore exist, without any true consensus. Further complicating matters, optimal smoothing window size is equal to the width of a true peak, an unknown value capable of varying by orders of magnitude, and choosing a specific window size will preferentially identify regions of that size (Thomas et al., 2016).

The most common peak callers, including the most widely cited tool MACS2, use statistical models to call peaks. While the strategy of using p-values as the definition of enrichment is itself debatable, the way in which they are applied is also problematic. In these models, the expected level of background is determined using a distribution (generally Poisson or less frequently binomial), however signal outside of binding events is actually non-random and does not follow any archetypal distribution. This, in combination with the multiple testing associated with candidate peak calling, means that p-values provided by these tools "can be overly significant by many orders of magnitude" (Chitpin et al., 2019). Even if the p-values were accurate, enrichment is not equivalent to a biological event. High enrichment is often associated with binding in these experiments, however this may also occur in the case of sequencing errors, inappropriate read alignment, or - most commonly in statistical peak callers - when the background signal is low or zero. False negatives are also possible, and are more frequently seen when ChIP-seq experiments are called with p-value-based enrichment models that use an input track to calculate expected background signal. Here, increased fragment density appears in the experimental tracks, with a small concomitant increase in the input control track. This increased signal in the control precludes the region from reaching the p-value threshold, but importantly the regulatory elements detected when using ChIP-seq nearly always occur in open chromatin, and therefore are associated with increased signal in control tracks (Auerbach et al., 2009); significance thresholds are often set high to help ward off false positives, further exacerbating this issue. P-values are not

consistent amongst true peaks, and thresholding with this metric means striking a balance between false positives and false negatives - the only common element preserved in these experiments is the "peak" shape of the enriched signal. This has been shown in several works throughout the years whereby humans accurately perform peak calling by visually inspection, even outperforming statistical based algorithms (Hocking et al., 2016; Oh et al., 2020; Rye et al., 2011).

Given the incredible size of the human genome, systematically scanning coordinates and marking locations by hand is impractical, bordering on impossible. Deep learning models excel at identifying and classifying complex patterns, for instance when CNNs are used for image recognition. Traditional peak callers reduce a peak to simply a maximum height, but the relationship between the different coverage levels across the coordinates of a peak, i.e. the peak's shape, is actually rich with data which allows humans, or CNNs, to judge its quality. In order to utilise this functionality, peak calling needs to be framed as an image recognition task, and candidate peak calling naturally fits in with this process. Implementing a CNN, however, brings with it signal size constraints that need to be accounted for. Critically, the same size region must be delivered to the model. Regions smaller than the fixed size can be accommodated by including surrounding signal or adding flanking zeros, but larger regions have to be cut off or scaled - either option increasing the complexity of the task, and in some cases making it impossible. Making the region considered by the CNN too large may also cause potential issues, as more than one enriched region could appear in this window. For instance, if two regions appear in a single window and one is a true peak and the other is enriched noise, the classification becomes ambiguous. Furthermore from a practical standpoint, the larger the size of the region window the larger the demand for computational resources.

The restriction encountered with CNN models is mirrored in peak calling tools, as models designed to identify peaks with a narrow binding pattern rarely work well at identifying those with a broad pattern. Narrow or punctate peaks are sharp

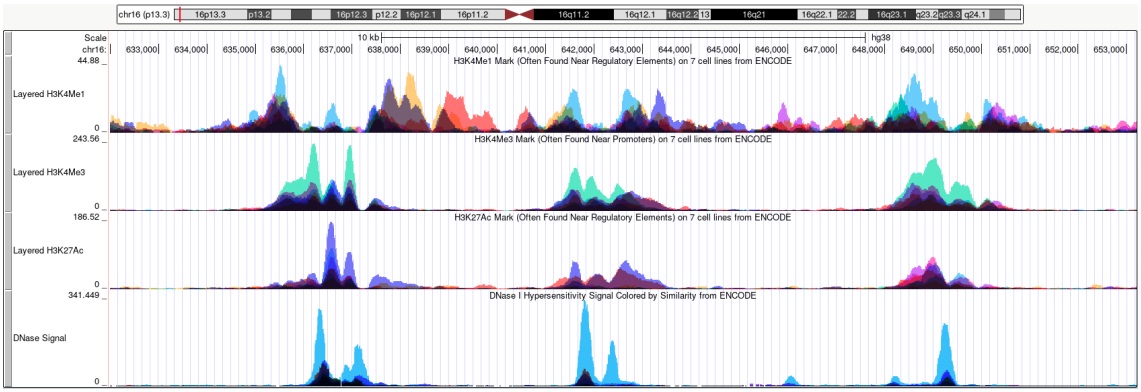


Figure 2.1: UCSC browser session showing signal from H3K4me1, H3K4me3, H3K27ac histone ChIP-seq as well as DNase-seq from ENCODE data. H3K4me1 is considered a broad histone mark, while H3K4me3, and H3K27ac are considered narrow histone marks. Compared to broad marks, the signal from narrow histone ChIP-seq is much more similar in shape to transcription factor ChIP-seq and the open chromatin tracks ATAC-seq and DNase-seq.

peaks generally occupying a few hundred base pairs each, and found commonly in ATAC-seq, DNase-seq, and transcription factor ChIP-seq tracks. Broad peaks are more general enriched areas, sometimes lacking a clear summit and extending for large portions of chromosomes, even occupying up to millions of base pairs (Song & Smith, 2011). Histone ChIP-seq may produce either type, with the most common narrow histone mark targets being H3K4me3 and H3K27ac, and the most common broad targets H3K4me1 and H3K36me3 (Landt et al., 2012) (Fig. 2.1). While the process of peak calling narrow and broad peaks is similar, the goal of this task is often reversed in broad peak calls: rather than seeking to discover regions of binding, generally the region of interest is already known and being inspected for its association with certain histone marks.

In this chapter, I build the collection of data for training CNN peak calling models. I chose to limit the data types to narrow peaks, which includes both ATAC-seq and DNase-seq open chromatin tracks, as well as transcription factor ChIP-seq, and two types of histone ChIP-seq with narrow binding profiles (H3K27ac and H3K4me3). In deep learning, the proficiency of a model is dependent on the quantity and quality of data used. For this reason, I used 50 datasets from the ENCODE consortium. ENCODE works across laboratories around the world, and

defines the standard for data collecting and processing for these experiments; importantly these datasets contain a variation in quality expected from the vast majority of scientists working with these data. Given the numerous ways in which enrichment is defined, I used multiple approaches to identifying candidate peaks; in subsequent work this allows DL models utilising these training data the ability to recognise a range of peak start and end coordinates, and is analogous to data augmentation strategies used in traditional image recognition tasks. I tested using a parameter-free approach as a means of candidate peak calling, and while it has great potential to improve the precise identification of a peak's start and end coordinates, it was an overly inclusive method of identifying regions of interest when used with these data. Furthermore, I designed a candidate peak calling algorithm that offers specific advantages to a CNN peak caller, and like the parameter-free method accurately approximated start and end coordinates. Finally, analysis was performed to find the best method for using p-values to classify regions as peaks or noise. For this a logistic regression model combining multiple p-values was built to improve accuracy over single measurements alone.

2.2 Results

2.2.1 Data acquisition

To create an accurately labelled set of candidate peaks I inspected and categorised regions by hand. This first required constructing a list of well-suited experiments to examine. The ENCODE consortium hosts thousands of tracks for ATAC-seq, ChIP-seq, and DNase-seq - far too many to investigate comprehensively, however a representative sampling was necessary for best model performance. Therefore my strategy was to use their data portal to build the list of potential human datasets in an inclusive manner, but select from it randomly. Only a handful of filters were used to limit the experiments, and centred on removing low quality tracks or ones with missing information. There are three different sta-

tuses that an experiment can be classified as in ENCODE: released, revoked, or archived. Revoked or archived status suggests the datasets were substandard or followed outdated protocols, so I limited the search to only "released" tracks. Control tracks were another necessary requirement for transcription factor and histone ChIP-seq experiments, and experiments lacking these were also removed. Only experiments that included files for aligned reads (BAM files) mapping to the most recent human reference genome at the time (GRCh38, often called hg38) were considered. In theory these filtered experiments were not necessarily low quality, however they were more likely to be older and potentially less relevant; furthermore aligning sequencing data to a reference genome is time consuming and computationally expensive. The metadata supplied by ENCODE is extensive and complete for nearly all experiments publicly available, however in several cases information was missing regarding how tracks were sequenced (single-end or paired-end) - these datasets were also excluded. Ten datasets were selected for each type of experiment: ATAC-seq, TF ChIP-seq, H3K27ac ChIP-seq, H3K4me3 ChIP-seq, and DNase-seq. Paired-end sequenced datasets experiments were also preferentially selected, since this process dramatically improves data quality and is commonly used in many molecular biology laboratories. Because paired-end was favoured in the selection process, single-end sequenced datasets were only included for H3K4me3 ChIP-seq, where only six paired-end experiments were available. The total number of experiments meeting all criteria was 3,902, composed of 74 ATAC-seq, 911 DNase-seq, 305 H2K27ac ChIP-seq, 463 H3K4me3 ChIP-seq, and 2,149 TF ChIP-seq tracks. I randomly sampled 10 datasets for each experiment type, and in the H3K4me3 experiments all six paired-end experiments were used and four single-end experiments were randomly selected. The final set of experiments resulted in 38 unique biosample types, 9 unique transcription factor ChIP-seq targets plus 2 histone ChIP-seq targets (Fig. 2.2).

Further details and technical considerations can be found in Methods, and

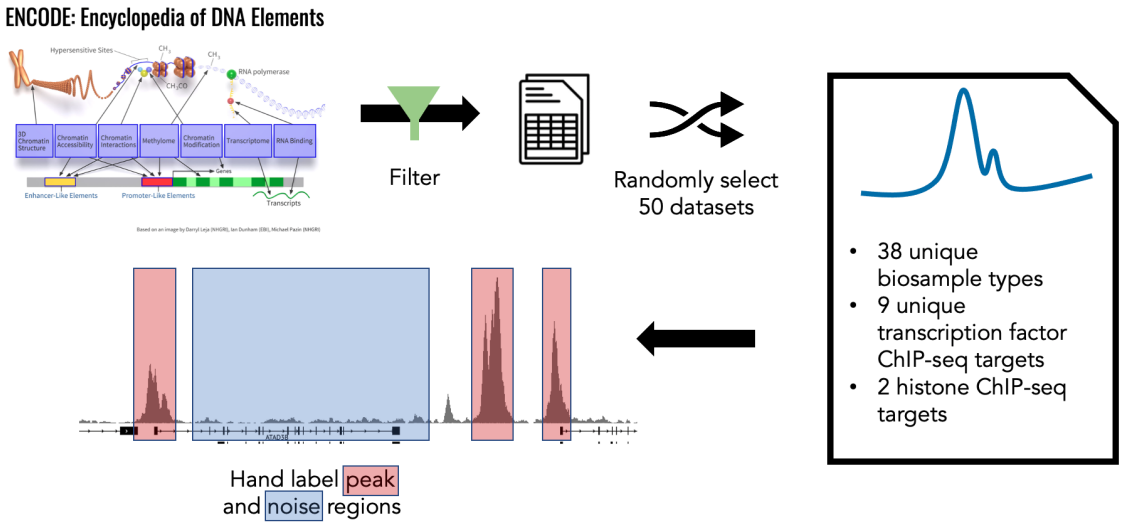


Figure 2.2: Selecting and labelling ENCODE datasets.

scripts used as well as their outputs can be found at https://github.com/LHentges/LanceOtron/tree/master/model_training.

2.2.2 Hand-labelling peaks

Accurately labelling regions as peaks or enriched noise required first subjecting the filtered list of datasets to candidate peak calling, then visually inspecting and classifying the individual regions. Because an area of a coverage track may be considered enriched under countless definitions, a multifaceted approached was used for identifying putative peaks. The level of enrichment compared to surroundings, and the size of a smoothing window are two important points of variation when determining candidate peaks.

Generating regions to label

I chose 25 enrichment thresholds and smoothing windows combinations, using a geometric series with a ratio of two in order to cover a large range of values. Enrichment was measured using the value of the smoothed signal in the experimental track compared to the average signal of a "large surrounding area". If the smoothed signal from the coverage track was above a threshold (scaled to the

average signal of the large surrounding area) it was considered enriched. The thresholds tested for, using the geometric series strategy, were enrichment above the mean of the large surrounding area, 2-fold above the mean, 4-fold, 8-fold, and 16-fold above the mean - maintained for 50 contiguous base pairs. This minimum size eliminated the plethora of regions composed of large enrichments only occurring over a small number of bases, even when the signal was smoothed. The signal of the "large surrounding area" was either the average signal over the entire chromosome, or the local area maximum. Local area maximum here means either the mean signal from the surrounding 5 kb, 10 kb, or chromosome mean - with the greatest signal average (most stringent) of the three large surrounding areas selected for use (these averages of background areas are called λ values here and when referencing the Poisson distribution). Smoothing windows followed a similar geometric pattern, these affecting the height of the signal from the coverage track. When deciding on an appropriate range of windows the default value MACS2 was considered, which has a window of 150 bp (Y. Zhang et al., 2008), however peak caller evaluation showed it had better performance at 200 bp for narrow peak datasets (Thomas et al., 2016). It was decided to use smoothing windows of 100 bp, 200 bp, 400 bp, 800 bp, and 1600 bp for this process (Fig. 2.3). In addition to all of these generic candidate peaks, MACS2 was run on default settings for all experiment types, and using the control input tracks for histone and TF ChIP-seq. This combination of window-threshold candidate peak calls and MACS2 peak calls (after applying an additional data augmentation strategy) made up the regions to be hand labelled.

Addressing regions labelled at multiple coordinates

Data augmentation is a well-used technique in deep learning image recognition data, and involves making slight adjustments to the size or position of an image in the training data. Some data augmentation was utilised, i.e. there were instances where the same region could be identified but with different start and end

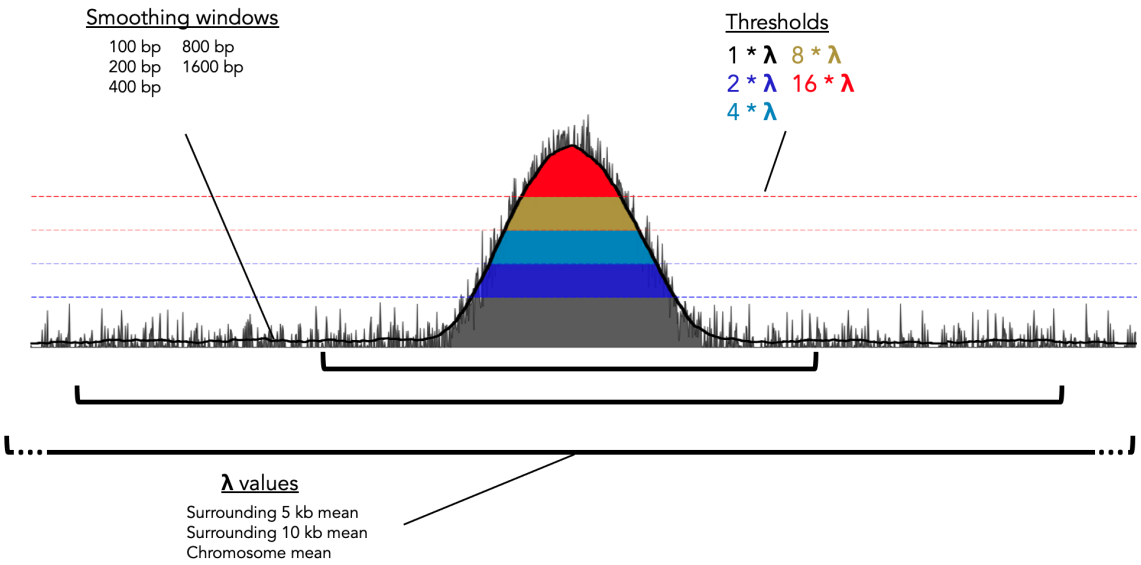


Figure 2.3: Candidate peak parameters used for labelling training data. Multiple parameters were varied to create candidate peak calls, shown here with simulated data. A rolling mean reduces the amount of random noise without distorting the signal. This makes the coverage track appear as a smoothed line; the height of the coverage signal is affected by the size of smoothing window used. The threshold affects the coordinates of the enriched region - or if the region is even considered enriched. Averages (λ values) affect how enrichment is defined by varying what the signal in the peak is compared against.

coordinates, but to avoid characterising the classes in an overly narrow way a randomisation strategy was implemented. A limited number of hand labelled start and end coordinates could be used for the same enriched region: one set from signal compared to chromosome mean, another from signal compared to local area maximum, and the final set from MACS2 (one additional set of coordinate, generated algorithmically, was also included and is described further below). The MACS2 calls always had exactly one set of coordinates with no overlapping regions, however for the other candidate peak calls the multiple definitions of enrichment often overlapped - especially true at locations of real binding events. For these all locations of overlapping candidate peaks were found, and one was selected at random to create the final list of regions available for labelling.

Selecting chromosomal blocks and setting datasets limits

Once the datasets had been selected from ENCODE, and the appropriate data augmentation limitations were applied, regions were selected randomly for la-

belling. From each dataset a 1 Mb continuous region was selected at random for each chromosome for autosomes and sex chromosomes only. If the start of the randomly selected region was near the end of the chromosome, the area considered was from that point to the chromosome end, then from the chromosome start extending out until a full 1 Mb was covered. By working with continuous regions, I hoped to replicate the same ratio of enriched regions a model would be likely to adjudicate. This was to ensure that rarely occurring peak shapes in enriched regions would be represented in the dataset, however the number of hand labelled regions was ultimately large enough that a simple random sampling of comparable size would likely have yielded similar results. Only candidate regions which were obviously peaks or noise were labelled as such. To ensure that the dataset was not biased towards particular experiments, a soft limit of 100 peaks was implemented; noise regions were far more common, so an additional randomisation strategy was carried out for enriched noise. After assessing a Mb region, if fewer than 100 verified peaks were identified, another Mb region from a different chromosome was assessed; this process continued until 100 or greater peaks were found or all of the chromosomes were inspected. A total of 736,753 regions were labelled this way (5,016 peaks and 731,737 noise regions) covering 499 Mb.

Algorithmic labelling

In testing users data, the threshold-smoothing window combination that generally worked best (all true peaks were candidate regions with the fewest number of enriched noise regions) was 4-fold enrichment above the mean at a smoothing window of 400 bp. Given this I additionally included regions at this enrichment definition, modified slightly by a custom multiple peak detection algorithm. This was done by labelling these regions as peaks if their coordinates intersected with the hand labelled peaks.

The multiple peak detection algorithm relies on the fact that the data being

analysed are narrow peaks, and are extremely unlikely to be broader than 2 kb. It first smooths the signal using a window (400 bp in this case), then finds all regions at or above an initial threshold (4-fold chromosome mean here). Next it examines the size of the region - regions below 50 bp are discarded as before, between 50 bp and 2 kb are left unchanged, and greater than 2 kb are reassessed at a higher threshold. If a region is greater than 2 kb it is first reassessed to find if there are any internal areas within the region greater than 5-fold above the chromosome mean. Once again these regions are assessed for size, following the same format as above - any regions still larger than 2 kb are subjected again to a reassessment at 6-fold enrichment. This recursive process continues until a region is between 50 bp and 2 kb, or the region is greater than 20-fold enriched.

Downsampling

The number of enriched noise regions were far more common in every chromosome segment examined, outnumbering the peaks by over 145 to one. Strategies exist to address unbalanced training data, the two most common being weighting the importance of the examples based on which class they belong to and downsampling. Because I hand labelled all samples personally, I was able to recognise there were a very large number of regions belonging to the noise class that were a single read in a region without any other signal. Given the minimal amount of diversity observed in the low level enriched noise regions, it was decided that downsampling would be used and a strategy was devised to preferentially reduce these regions. All noise regions with a maximum height in the 25th percentile or greater were included (3,658), and equal numbers below the 25th percentile were randomly sampled. This retained the regions that are harder to distinguish from peaks, at least when using enrichment level as the sole quality metric, while reducing the abundant low complexity regions over represented in the labelled data.

Model aided curation and correction of the labelled regions

Additional changes to the training data were made after observing the CNN's performance (Chapter 3) in order to identify any incorrect labels which may have occurred due to the addition of the algorithmic derived coordinates, or incorrectly identified hand labelled regions. After the CNN had reached its best performance accuracy and additional training was unlikely to improve results, all of the training data were scored with the model to identify misclassifications. This strategy is based on the notion that a trained CNN is unlikely to misclassify the very samples it is trained on, and so any such samples are worthy of reevaluation. Here the misclassified training samples were visually inspected for a second time to see if its original label was correct. This process revealed that a portion of the supplemental algorithmically labelled regions were in fact incorrectly labelled. Many of the mislabelled regions shared a common element, whereby the hand labelled peak regions' coordinates extended slightly beyond the peaks' increased coverage density, and algorithmically labelled regions were enriched noise immediately adjacent to these real peaks, overlapping by only a small number of bases. From this training data reconsideration process, 24 peaks and 1,187 noise regions were added to the dataset. Ultimately the final dataset was composed of 16,990 regions: 8,503 noise regions plus 8,463 peaks.

2.2.3 Candidate peak comparison

Working through the datasets, generally a smoothing window of 400 bp and a threshold of four fold chromosome mean was a passive enough filter to allow all true peaks through while ignoring many of the enriched noise regions. This is perhaps a good rule of thumb, however it obviously will not work with every dataset, and ultimately the track and peak call will need to be inspected for sensibility. The natural solution to this problem is to simple allow users to select parameters that yield the best results for their data. This does require trial and error, and even still

what constitutes optimal may be difficult to define, hindering reproducibility. For this reason I pursued an automated approach to limit the number of user selected parameters required.

Outside of genomics, peak calling is done using other established methods. First published in 1964 for the use in analytical chemistry, Savitzky and Golay designed a smoothing filter which can be applied to nearly any kind of signal to improve signal-to-noise ratio (Savitzky & Golay, 1964). Because of the way in which the algorithm is implemented, the Savitzky-Golay (SG) filter has the added benefit of calculating the derivatives of the signal in the same step as the smoothing without additional computational cost. The threshold parameter for the traditional candidate peak calling algorithm is used to find the start and end coordinates of the enriched region. Importantly, derivatives can show when the slope changes from increasing to decreasing and vice versa. This means the SG filter can be employed to find a candidate peak's start and end by recording the coordinates where the slope changes signs, reducing the number of parameters the user selects by one. The number of base pairs used to average across for smoothing still remains a parameter, however a process of automatically selecting the optimal smoothing window has been published specifically for the SG filter (Vivó-Truyols & Schoenmakers, 2006), potentially eliminating the need for any parameter selection for candidate peak calling.

I built an implementation of the SG filter for use with sequencing data, and used it to identify candidate peaks. When inspecting the candidate peak calls made using the SG filter, there were large stretches of chromosomes with densely clustered identified regions but lacking any real enrichment; this could be seen even in high quality, deeply sequenced tracks (Fig. 2.4). To quantify this, candidate peak calls from the SG filter and a window-threshold method were compared to final peak calls from MACS2. Data were obtained from ENCODE, and consisted of tracks obtained from primary liver tissue for CTCF TF ChIP-seq, H3K27ac and H3K4me3 histone ChIP-seq, as well as ATAC-seq and DNase-seq

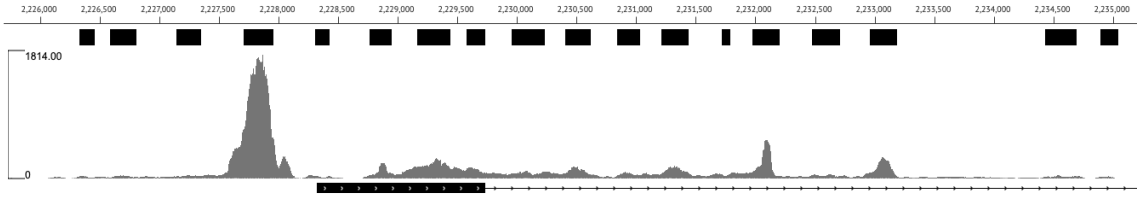


Figure 2.4: Candidate peaks marked as black bars above the coverage track, identified using the Savitzky-Golay filter for DNase-seq in K562 cells. Despite the high quality of these coverage data, numerous regions, including some with near zero enrichment, were identified by the algorithm.

Track	**Window-threshold**	**Savitzky-Golay**	**MACS2**
CTCF	1349	1738	1306
CTCF-control	75	326	
H3K27ac	1531	4295	2136
H3K27ac-control	69	294	
H3K4me3	1049	3392	1148
H3K4me3-control	43	508	
ATAC-seq	989	3085	532
DNase-seq	2519	7341	4019

Table 2.1: Comparison of candidate peak attributes from the traditional window-threshold method (window=400; threshold=4*chromosome mean) and Savitzky-Golay filter (algorithmic window size and derivative-determined region boundaries) to MACS2 final peak call.

open chromatin experiments. Comparison was done using chromosome 16 for each track, measuring the number of regions found (a single chromosome was used to reduce computation time, any reasonably sized chromosome could have been substituted). For the window-threshold method, 400 bp smoothing and 4-fold enrichment above the chromosome mean were used. MACS2 was used on default settings, using control input tracks for the ChIP-seq experiments. As expected from the preliminary examinations, in every instance the SG filter found more regions than either MACS2 or the window-threshold method (Table 2.1). Because these are candidate peaks, yet to be assessed for classification as either peaks or noise, this does not necessarily mean a decrease in accuracy. However candidate peak calling does determine the number of regions that are passed to the deep learning algorithm, a rate-limiting step in the process that scales to region count, meaning some speed is lost when using the SG filter.

In building the list for candidate peak to label, I noticed a trend that candidate regions that were true peaks were recognised under many of the definitions of enrichment, while noise was much more likely to be recognised by only a single definition. This is perhaps not surprising since the higher the enrichment, the less dependent a region is on the proper parameters in order to exceed a threshold. There are other important considerations however that make it more common for true peaks to be considered enriched under multiple definitions. If a true peak exists, and the coverage track is smoothed over some bp window, then the signal of the peak is preferentially improved over noise when the window size matches the number of bps of the peak (Thomas et al., 2016). In this scenario with the matched window size, the peak is more likely to be 2, 4, 8, etc. -fold enriched against background. From the perspective of enriched noise, if a single base pair is inappropriately enriched a smoothed signal will temper that enrichment; if multiple contiguous bases are required (such as the 50 bp minimum required for the candidate peaks algorithm above) this further excludes random noise. Random enrichment at contiguous bases of a region becomes increasingly rare as the size of the region grows. Unfortunately contiguous bases in regions are sometimes enriched (e.g. genome mapping errors), though separating out these events using smoothing windows and minimum enrichment is not possible as a general strategy.

Using these principles, I designed a candidate peak calling algorithm that uses the multiple enrichment definitions as the basis for finding regions. Each combination of an enrichment threshold at a particular smoothing window was considered a definition of enrichment, and the algorithm used 25 different definitions. The enrichment levels were 2, 4, 8, 16, or 32 fold greater than or equal to the mean-chromosome-signal after applying smoothing windows of 100 bp, 200 bp, 400 bp, 800 bp, or 1600 bp. Compared to the candidate peak selection process used for creating the labelled dataset above, the thresholds are slightly higher, removing mean chromosome signal (or 1 fold) enrichment and adding 32 fold enrichment.

This decision was based on the observation that a very large percentage of the genome was enriched above the mean for the various datasets thus making 1 fold enrichment too low of a bar to pass over to be useful. In addition, on tracks with high signal to noise ratio the mean chromosome signal was often very low and another definition at the high end of the range could be utilised. For each of these definitions, the enriched coordinates were recorded and tallied. This created a genome wide map of all of the regions considered enriched under any of the 25 different definitions. If a contiguous region of 50 bp or more had at least 5 concurring definitions it was marked as a candidate peak. For regions larger than 2 kb an adaptation of the reassessment process from the algorithmic labelling was applied, however rather than testing at an increased fold enrichment, it was reassessed at an additional enrichment definition. Therefore if a region had 5 or more concurring enrichment definitions and was greater than 2 kb, an assessment was done to see if any internal areas of the region were enriched under 6 or more definitions. These regions were again assessed for size and any regions still larger than 2 kb repeated the reassessment process, requiring an additional definition, and so on until the region was between 50 bp and 2 kb or the region was enriched under all 25 definitions. Rarely, cases would occur where at a certain enrichment definition it would be greater than 2 kb, but at one higher enrichment definition it would be less than 50 bp. In these cases the initial larger than 2 kb coordinates would be retained and the entire region would be considered as a candidate peak. Similar to the SG filter, this candidate peak calling algorithm did a good job of automatically detecting appropriate start and end coordinates for regions (Fig. 2.5). The numbers of smoothing windows, thresholds, and their values, along with the requisite number of definitions required can all be considered parameters, however in practice these default values performed exceedingly well across a range of datasets and experiment types (Chapter 4).

MACS2 is the standard peak caller used in the field and so it was important to compare the concurring enrichments candidate peak calling algorithm against

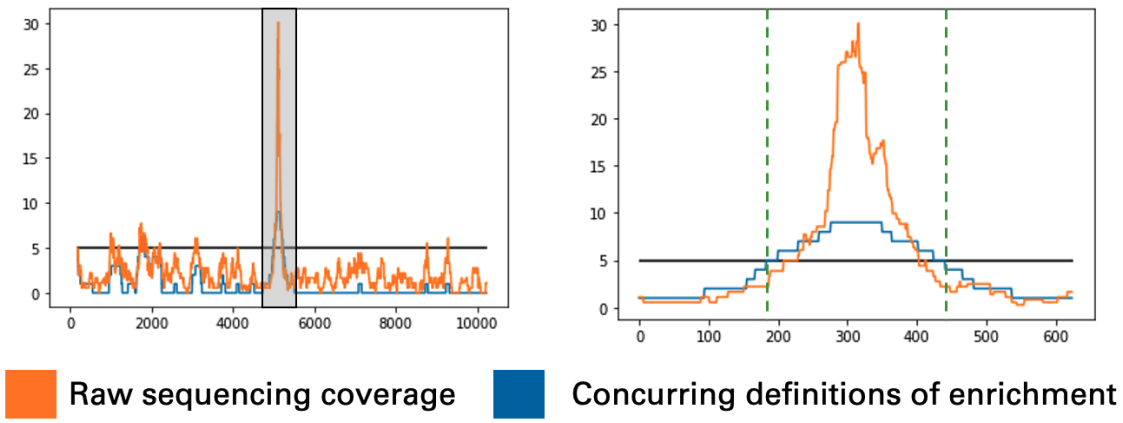


Figure 2.5: Raw coverage of fragments (orange) of ChIP-seq experiment with quantity of enrichment definitions passed overlaid (blue). In the 10 kb shown on the left, only the highlighted region qualified as a candidate peak using the dynamic mean algorithm. The detected start and end coordinates of the peak are automatically detected, marked by the vertical dashed lines on the right.

it. While commonly used for numerous sequence coverage tracks, MACS2 was originally designed for transcription factor ChIP-seq, and so the comparison was done using 10 different TF ChIP-seq tracks found from ENCODE. Comparing a final peak call (MACS2) to a candidate peak call (concurring enrichments algorithm) in this manner brings with it some important considerations. Candidate peaks may ultimately be dismissed as enriched noise. This means there should be a number of uniquely identified regions found in the candidate peak list not found in the final list. In a complementary fashion, the regions marked by MACS2 are areas its model has declared peaks, and should therefore coexist on the candidate peak list with relatively few unique MACS2 regions, though any false positives it misclassifies could either intersect with the candidate peak call or be uniquely identified by MACS2. Overall findings were very much as expected, with 2,260,296 unique candidate peaks identified, 5,830 unique MACS2 peaks called, and 119,565 regions shared between both methods (Fig. 2.6).

2.2.4 P-value comparison

P-values are commonly used to understand enrichment across every field of science, but the choices made as to how they are calculated affects their outcomes

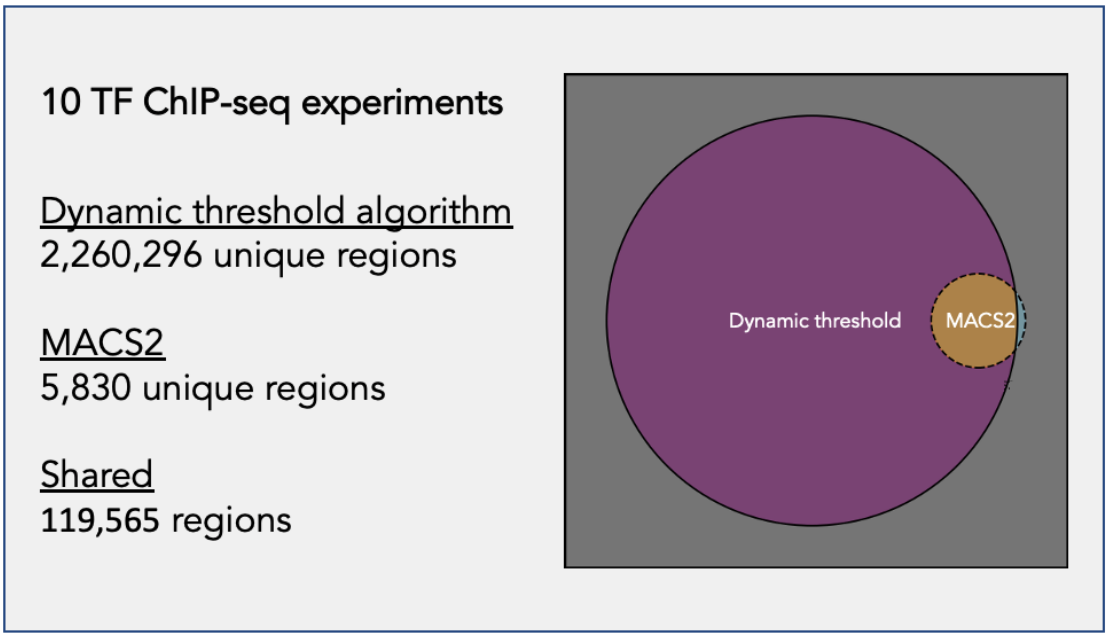


Figure 2.6: Comparing candidate peaks found with the concurring enrichments algorithm vs called peaks using MACS2 on 10 ENCODE transcription factor ChIP-seq datasets. As expected, there are a large number of unique candidate peaks and relatively few unique MACS2 peaks.

and interpretability. I wanted to test the most useful way to calculate p-values for ATAC-seq, ChIP-seq, and DNase-seq using the extensive labelled dataset I created to measure accuracy. Therefore different p-value calculations were performed and assessed. The p-values were calculated based on the Poisson distribution, varying the size of the region used to calculate background enrichment level. Signal was measured at maximum region height, as is standard procedure for typical peak calling algorithms. The background regions the signal was compared against, the λ value of the Poisson distribution, was the mean coverage for the surrounding 10 kb to 400 kb in 10 kb intervals, plus mean chromosome signal. Once a p-value is calculated, a decision also also must be made to determine the significance level at which a region is called a peak (the generally accepted norm of 0.05 is often considered too lenient). To garner an understanding of the performance irrespective of threshold, I plotted the receiver operating curve (ROC) which compares true positive rates to false positive rates. From this it was clear that the vast majority of regions can be classified accurately using

p-values at some threshold (though not necessarily the same threshold depending on the track). This is even consistent amongst various p-value calculations, though a 10 kb background is slightly worse than the other intervals and the mean chromosome signal is very inaccurate. Comparisons between the different measurements can be made by using the ROC and calculating the area under the curve (AUC). A completely random classification process would yield a AUC of 0.5, while a perfect classifier would be at 1.0. While the scores are generally in the high 90s - extremely good from an accuracy standpoint - there are several factors complicating this result. Firstly, the experiments being modelled are used for discovery, which means that a high value is placed on avoiding false negatives. This dataset is also balanced between peaks and enriched noise, however realistically there are orders of magnitude more noise regions occurring in candidate peaks called for an experiment - maintaining these same rates will result in an increased number of false positives amongst called peaks when evaluating a standard track. The end behaviour of the ROC curve is also interesting, as all of the measurements plateau. This occurred because even at the lowest thresholds these regions would not be enriched enough (according to any of the p-value calculation tested here), implying some true peaks simply cannot be identified with p-value thresholds alone (Fig. 2.7).

The threshold used for peak calling is a parameter that can generally be selected by the user depending on the tool being used, though the default value is often relied on. MACS2's default p-value is 0.01, and by selecting a specific threshold we can further analyse performance. The same p-value calculations were used to calculate precision, recall, and a combined F1-score which combines these metrics for an overall performance score (Fig. 2.8). Once again the values were extremely robust across the different backgrounds measured, excluding the chromosome mean. At this threshold, the recall is very high across the intervals - above 99% - but the precision is extremely low thus contributing to a low overall F1-score. The outlier in the analysis was the chromosome mean, where

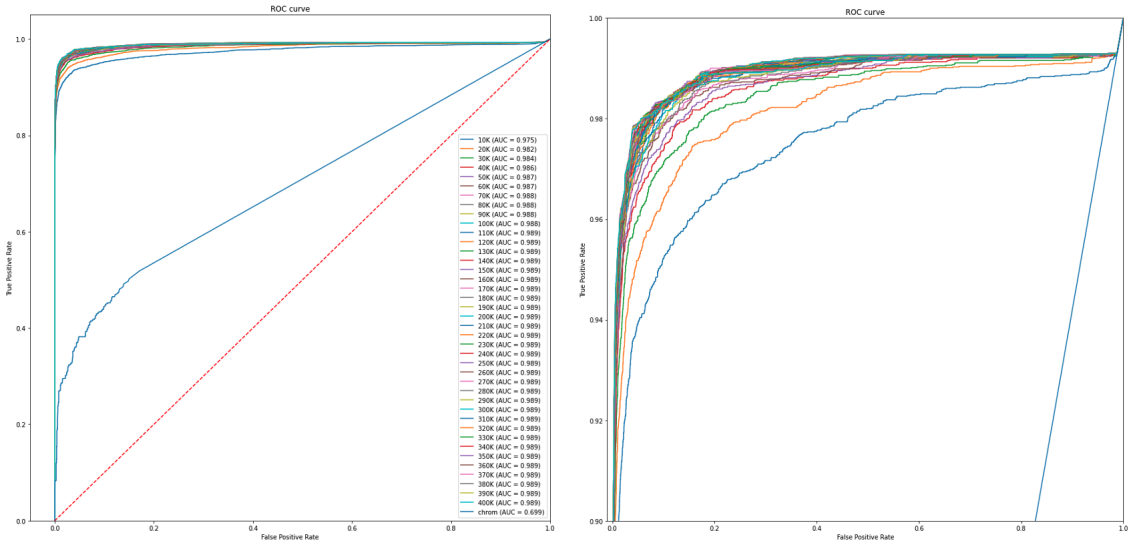


Figure 2.7: Receiver operator characteristic (ROC) curves of p-values calculated from hand labelled data. P-values were calculated using a Poisson distribution comparing the maximum height of a region to a range of backgrounds. The full view of the ROC as well as the calculated area under the curve show p-values can be used accurately and are robust across a range of different backgrounds (left), however the plateauing end behaviour suggests that some peaks cannot be identified this way and minimising false negatives to zero will drastically increase false positives.

the precision was high but the recall was low. The average chromosome signal was larger than the average surrounding signal for the intervals, which made a 0.01 p-value more stringent (on the ROC this threshold is nearer to the left). This improved precision at the cost of recall did not benefit the overall F1-score.

Given the countless ways in which enrichment can be defined, any single p-value is unlikely to give a universally accepted definition of enrichment, but several p-values in concert would likely improve results. In order to understand if a combination of values could be used to better classify regions, I used unsupervised machine learning techniques for dimensionality reduction. For this I chose 10 p-values for each region, calculated based on its maximum height compared against a background of 10 kb to 100 kb, and used PCA, t-SNA, and UMAP to create two-dimensional maps of each region. Interestingly, the linear technique PCA showed the most obvious separation between classes (Fig. 2.9).

Given this, I also wanted to see if results could be improved using a linear combination of all of the p-values. For this I used a binary logistic regression

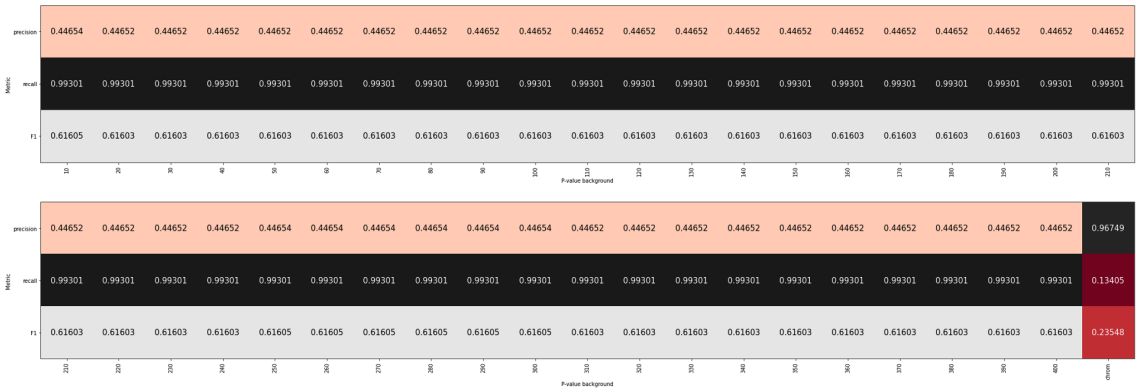


Figure 2.8: Precision, recall, and F1-score for p-values calculated from hand labelled data. P-values were calculated using a Poisson distribution and using a range of backgrounds as well as the chromosome mean. Consistent with the ROC analysis, the values calculated for these metrics were robust across p-value calculations with the exception of chromosome mean.

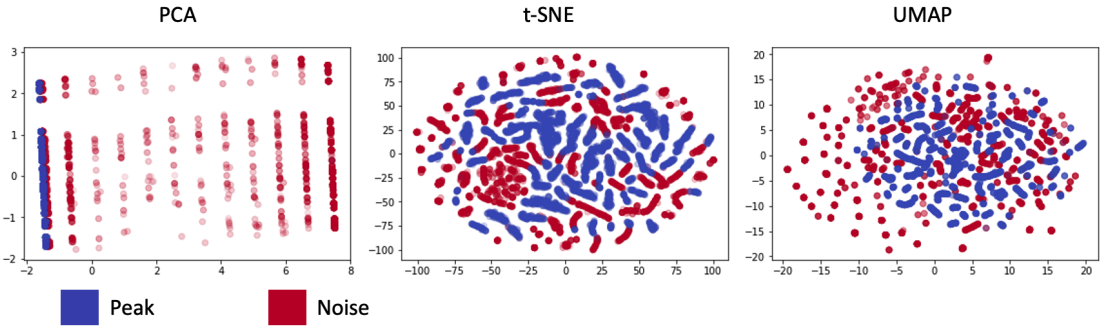


Figure 2.9: Dimensionality reduction using PCA, t-SNA, and UMAP of labelled peaks and noise. The values included for each region were 10 p-values, calculated based on its maximum height compared against a background of 10 kb to 100 kb. PCA, a linear technique, showed the most obvious separation between classes.

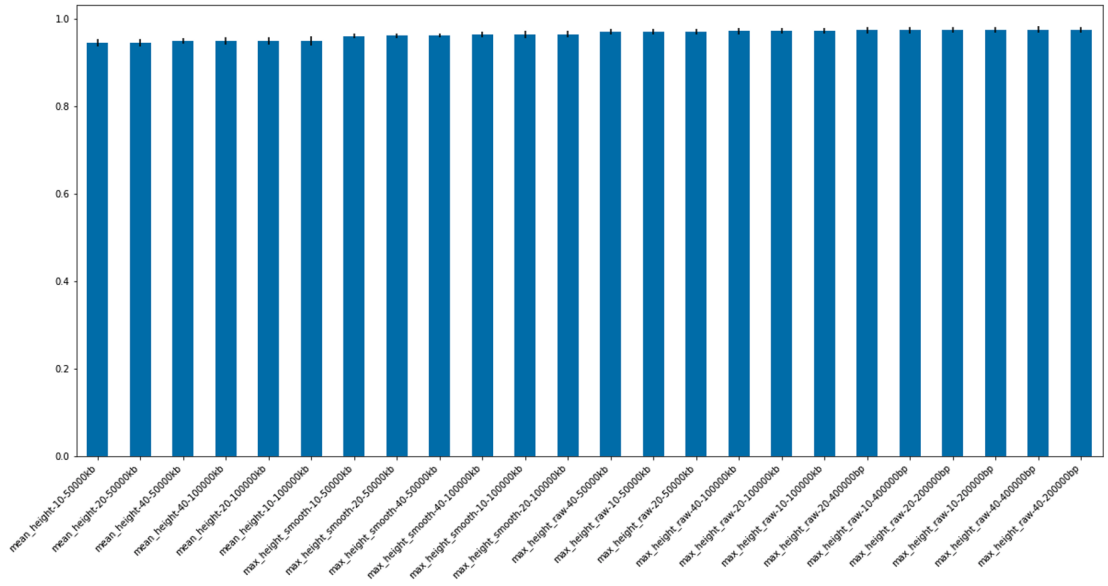


Figure 2.10: Accuracy using a logistic regression model to predict peak and noise classifications from the labelled dataset. Models were built using different measurements for the regions signal (maximum height, average height, and maximum height after 400 bp smoothing), range of background (50 kb, 100 kb, 200 kb, and 400 kb), and number of p-values calculated within the range (10, 20, 40).

model to predict whether a region was labelled peak or noise, testing the signal measurement types, range of base pairs covered, and number of p-values used in the model to make the prediction. The different types of signal measurements were maximum height, mean height, and maximum height after a 400 bp smoothing window was applied. Initially, only a range of 50 kb, 100 kb, and 200 kb were tested, however all of the 200 kb outperformed the smaller ranges, and so an additional range of 400 kb was added. The number of p-values tested in the range, divided into equally spaced regions, were either 10, 20, or 40. Model accuracy was measured, employing 5-fold cross validation. The most accurate model used maximum height, a range of 200 kb, and 40 p-value measurements, though performance was good generally (Fig. 2.10).

2.3 Discussion

Labelling datasets was an extremely time intensive task, though worthwhile because of the utility of the data. Here it was used to help understand the limits of p-value testing for classifying regions in chromatin profiling assays, but critically it also serves as the basis for the deep learning model (Chapter 3). For this reason high quality data were central to this work. Arguably simulated data could be generated to exact specifications, however it was equally important that the data was not only high quality but also representative of that used by molecular biology community; ENCODE perfectly fit this description. The combination of ENCODE's REST API and the data acquisition scripts here proved to work well together, and can be used for future works for other experiment types ENCODE hosts. The final dataset was large enough to allow for an extremely large capacity deep learning model. The goal of this work was to generate a tool that could easily be used regularly by laboratories for analysis, and as such considerations for processing time were necessary. Working with state of the art hardware, the model built pushes the limits of capacity while maintaining a reasonable processing time, however as technology progresses no doubt higher capacity models will become feasible. The script designed for accessing ENCODE was intentionally generic, and any subsequent data labelling could be completed without the additional work of engineering filters to access select subsections of ENCODE's database.

Ideally the labelling process would be as fast and painless as possible, and using the MLV software afforded me numerous benefits. This tool was designed to use the keyboard to rapidly cycle through regions in a bed file, display these regions in a genome browser, and tag them as peak or noise (assuming the region was not ambiguous) with a keystroke; this method allowed for the labelling of a mammoth 736,753 regions. While both quick and accurate, the vast majority of the time was spent considering obvious noise regions - many of which were ultimately disregarded from the final dataset. Very long stretches of "peak

desserts" exist in the tracks I inspected, and a better strategy would be to select a labelling region, and mark the areas of peaks and noise, then intersect candidate peak calls with those labels to find the coordinate. Rather than a simple inclusion intersection, as that introduced a number of falsely labelled peaks into the preliminary dataset, a rule that required the majority of the region to lie within the labelled area would eliminate many of the problematic labels I encountered with the algorithmic process.

The data curation process needs to account for the downstream use of data, and the labelling process resulted in an extremely high noise to peak ratio. Because this dataset was for use in deep learning models, this ratio of peaks to noise would have potential for negatively affecting performance. Given the imbalance often seen in these tracks, a model that always predicts noise would be extremely accurate: assuming the training data was representative of the track being tested, a model that predicted every enrichment was noise would achieve 99.3% accuracy. This model's accuracy comes at the cost of being entirely useless - clearly the ability to predict peaks is critical, and given that the goal of a peak caller is discovery, reducing false negatives is arguably more important than reducing false positives. Strategies exist to address unbalanced training data, the two most common being weighting the importance of the examples based on which class they belong to and downsampling. Because I hand labelled all samples personally, I was able to recognise there were a very large number of regions belonging to the noise class that were a single read in a region without any other signal. These low diversity samples are of little value for model training and so it was decided that downsampling would be used.

Before the careful construction of dataset described above, two proof of principle datasets were created and used to train and test deep learning models. The first dataset was constructed from datasets previously produced within our lab. Data was added ad hoc if the model made a large number of misclassification on a dataset being tested, and this process was carried out over a few

dozen datasets until misclassifications became rare events. Our group produced a number of H3K4me1 histone ChIP-seq tracks, a target that produces a broad peak, and these were also included in the training data. Unfortunately the CNN's peak calls for these experiments were inaccurate, though it did not seem to harm performance for the narrow peak tracks. This initial unstructured process yielded reasonably good results overall, and was the impetus to formalise the work and to build a full-scale tool.

Another dataset was also created which unfortunately had poor performance across the board. Given that CNNs can be capable of detecting patterns not readily seen by humans, optimistically I spent a great deal of time labelling ChIP-seq peaks which were ambiguous from the coverage track alone but could be determined from other information such as open chromatin and gene information. It was unclear if this information could possibly be encoded in the peak shape, undetectable by humans but recognisable by the algorithm. The CNN was not given access to the additional (and necessary) information I had when reviewing the regions, since this would not be available during a standard peak call. The subtle differences in peak shapes seen across a track could not recapitulate the information I accessed when labelling the ambiguous regions. Worse still, all regions, even those which were clearly peaks or noise, were also misclassified at a high rate.

When selecting values from parameters, I had complete freedom in choosing the numbers and the ranges tested. While I gained experience with what I perceived as sensible values for smoothing windows and thresholds, ultimately I relied heavily on geometric series of parameter values. This allowed for the accommodation of vastly different coverage tracks to achieve sensible candidate peak calls, however this does come at the expense of refinement. While there was a vast range in the quality and type of tracks successfully utilised for the labelled dataset, this approach could be improved by adjusting the range of values to the properties of the coverage track. Further analysis needs to be done ex-

amining which candidate peak calls most closely match the ideal start and end coordinates, and find any correlation to measurable properties of the datasets' statistics.

Over the course of building large annotated datasets, I personally inspected hundreds of chromatin profiling experiments. During this time, it was extremely rare that a reasonable - if not optimal - candidate peak call could be created using a smoothing window of 400 bp and a enrichment threshold of 4-fold the chromosome mean. Originally, it was thought that the resulting peak calling tool would use these values as sensible defaults for parameters, and users could simply adjust these if their results were unsuitable. However because candidate peaks here are passed to a deep learning algorithm where false positives could be identified and removed, more flexibility could be given to the candidate peak calls - i.e. as long as the true positives are included in the dataset, false positives can be tolerated to a degree. Any universal, parameter-free solution is unlikely to be perfectly accurate, but because the candidate peak call in this workflow is simply the first course filter in the process, the added flexibility makes room for this strategy. Initially an existing algorithm was found, the Savitzky-Golay filter, which had incredible promise given its successful utilisation in many fields of research. However when used on the coverage tracks used from the labelled dataset numerous noise regions were included, many of which were not included when using the window-threshold candidate peak calling algorithm. This precipitous increase in candidate peaks did not seem to identify any new true binding events in the tracks and ultimately made this method computationally expensive. While not used after testing here, the ability to use calculus to resolve the precise location of a peak's start and end coordinates, regardless of local signal biases, does offer an advantage to current methods. There may be solutions for reducing the number of candidate peaks called heuristically (e.g. limit a candidate peak call to the highest enriched 100,000 regions), or this technique may be used in conjunction with other candidate peak calling algorithm, possibly combining benefits of each

technique.

The vague definition of enrichment in coverage tracks from ATAC-seq, ChIP-seq, and DNase-seq encouraged exploration of different techniques for finding candidate peaks. At its core, the concurring enrichments method is simply adding a myriad of different definitions together. Combined with the centering and thresholding method from the labelling process, this algorithm functions essentially as a parameter-free candidate peak calling algorithm that works across a large range of tracks. Its efficacy was demonstrated when comparing the included regions to MACS2. Testing candidate peaks against MACS2 was limited to TF ChIP-seq. While MACS2 is very commonly used for the other experiment types focused on in this project, the decision to limit the track types came because of the exceedingly large number of false positives frequently seen in these other dataset. The expected results were observed comparing the final peak calls from MACS2 with the candidate peak calls from the algorithm.

The P-value comparisons across all thresholds, as displayed in the ROC curves, showed a marginal improvement using a background greater than 100 kb, but this has an upper limit as the chromosome mean showed the worst performance. Computationally, the chromosome mean is the fastest to calculate, as only a single value is used for all coordinates on said chromosome. Only a negligible difference exists in computation time amongst the different range calculations, but the larger sized background regions have larger regions displaying edge behaviour, e.g. an accurate rolling mean for the 400 kb background exists from coordinate 200,001 until the coordinate 200 kb away from the end as surrounding 400 kb does not precisely exist near the starts and ends of chromosomes. Practically speaking, tracks are not assessed at each and every p-value. When calculating performance metrics at the commonly used significance score of 0.01, varying the size of the background had a limited affect on the classification precision, recall, and F1 score. At this threshold when using the bp range values for background, over 99% of the true positives were identified, however at a serious

detriment to precision - in these cases the majority of significant regions were false positives. When using chromosome mean the average background signal was often higher than the measured background with the bp range strategy, effectively increasing the stringency of significance test. This improved precision but greatly reduced recall, and ultimately earned a far lower overall F1-score.

The different p-values calculated from a multitude of sizes from which background signal was measured yielded highly redundant results. Still, it could be valuable to combine different measurements to offer potential improvements. For instance if a true peak is adjacent to an area of high noise, perhaps the 10 kb and 20 kb background signals would be high enough that the peak was not significant, but the higher value ranges would still show significance. Complementary to this, if a true peak is found in a valley surrounded by high noise the larger ranges of background sizes may not be significant but the smaller ones may be. Unsupervised learning suggested that the peak and noise classes were largely linearly separable, and so a linear model was chosen. Logistic regression has the added benefit of taking the raw individual p-values without thresholding. Instead a probability of a region being peak or noise is generated, optimised for accuracy; regions above 50% probability were automatically classified as peaks. This resulted in high performing models, with accuracy ranges between 94.5% and 97.5%.

The work and results of this chapter laid the groundwork for the main findings of my project overall. Designing and building a deep learning model for peak calling requires an expansive dataset of precisely labelled samples to train from. Once trained, a mechanism for transforming the coverage tracks into signals the deep learning model can use is essential, otherwise the work would be limited to a proof of principle project rather than a tool for researchers to use. Finally the strategy of using multiple enrichment measurements, combined using a logistic regression model, is crucial for leveraging the best elements of significance testing while combining them with the sophisticated convolutional neural networks

designed in the next chapter; ultimately this combination results in the most accurate peak caller for these data types currently available.

CHAPTER

THREE

DEEP LEARNING

3.1 Introduction

Conceptually deep learning has existed for a long time, however in practice deep learning (DL) is a relatively new field. The most recent interest in deep learning after the so called "AI winter" was likely thrust into motion thanks to the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC). In the 2012 contest, a deep learning model (the AlexNet CNN) dramatically outperformed other models. This trend continued, resulting in the error rate falling from roughly 25% to 5% in just 5 years (Fig. 3.1).

Key to this advancement in the field was the use of numerous, or deep, layers of sequentially applied transformations, and in this case the primary transformations used were applied convolutional filters. In image recognition these are simply two-dimensional matrices of weights (also called tensors), which slide across the input calculating the dot product from the features (Fig. 3.2). In a deep learning (DL) model these outputs have convolutional filters directly applied to them in subsequent layers to uncover increasingly abstracted features. For instance, the first convolutional layer detects local patterns in the image, while each subsequent convolutional layer detects patterns within the patterns.

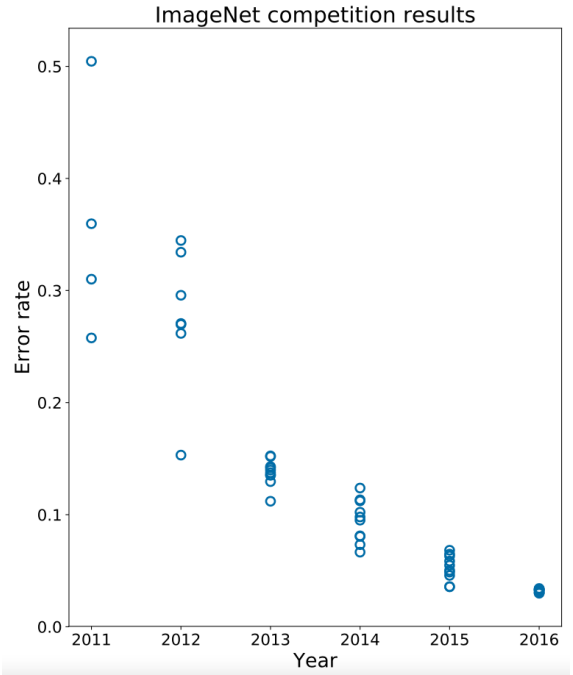


Figure 3.1: Error rate of the top performing models used at the ImageNet competition 2011 to 2016.

DL models learn weights, also called parameters, and the variables affecting how those weights are learned are called hyperparameters. Like in any new field, the exact model structure and variables (hyperparamters in this case) are not well established for all data types and a great variety of structures have been presented. The Visual Geometry Group (VGG) at the University of Oxford created a powerful model - VGGNet - which achieved the second highest accuracy at ILSVRC 2014. Other models surpassed its accuracy, however VGGNet remains popular even today as it performs well across a variety of data types due to its hyperparameter usage and structure. The group used a small receptive field for their convolutional filters - 3x3 pixels, which they describe as "the smallest size to capture the notion of left/right, up/down, center". The stride length, or the number of pixels (or inputs) a filter shifts over when moving for the subsequent calculation, was set to one. They also used pooling layers after some of the convolutional layers, forming convolutional blocks. Pooling reduces data dimensionality by reducing inputs in a specified way to a single output. Pooling size refers to the number of inputs reduced at a time which are commonly transformed by either

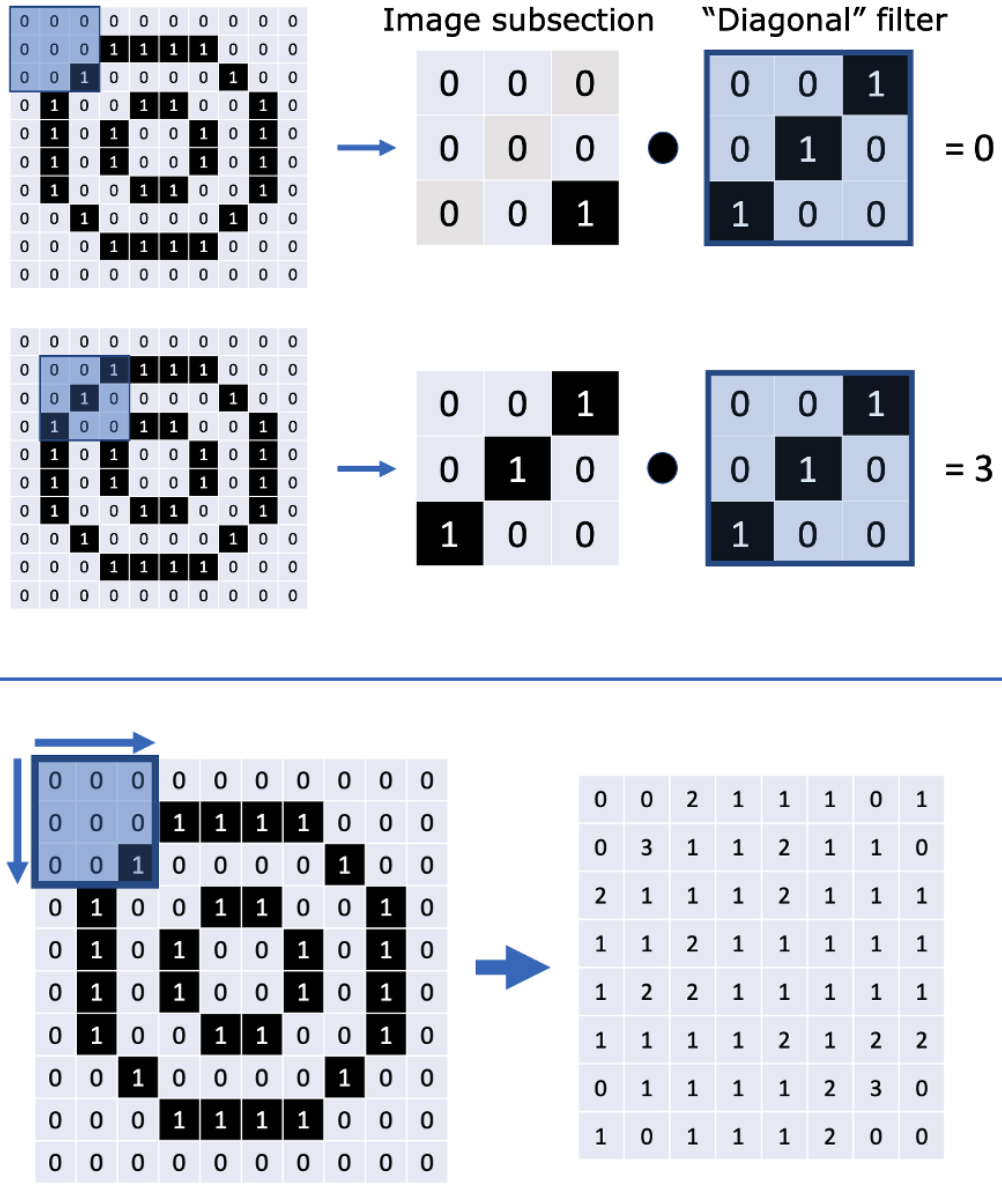


Figure 3.2: Principles of convolutional filters. Top: Applying a convolutional filter which looks for a diagonal pattern in the simplified image. The dot product between the subsection and the filter is calculated, returning a value scaled to how well the pattern matched. N.B. for CNNs, the weights of the filter are randomised, and trained such that the most useful patterns are detected. Bottom: The convolutional filter is applied across all coordinates, creating a new representation of the image; convolutional filters can be applied to this as well to find patterns within the patterns.

taking the average or using the maximum value of a pool (called average pooling or max pooling respectively). VGGNet used the max pooling type with a size of two. While finding the average would seem to promote accuracy as less data is lost, using the maximum value cultivates positional invariance. This is important in classification tasks, where the correct object label is agnostic of its location in the input data; e.g. a picture of a cat on the left side of an image versus the right - in both instances the classification should be "cat". The model was built as several different versions, each consisting of between one and four convolutional layers, with minimal filter size followed by a max pooling layer - variants ultimately had between eight and 16 convolutional layers and five pooling layers. These outputs were always combined using three fully connected layers, meaning each output of the previous layer directly connected to an input of the following layer, forming what is called a multilayer perceptron. CNNs can detect non linear patterns in data, and VGGNet achieved this through the use of the ReLU activation function. The values output by a model's layer are passed to this function - if its value is positive it passes to the next layer unchanged, otherwise the value of zero is passed. This is true for all layers with the exception of the output layer, which passes through a softmax activation function. Softmax is designed such that all of the outputs sum to one and is used to give the probability that a sample belongs to a particular class (Simonyan & Zisserman, 2014).

The activation function, such as ReLU, softmax, or sigmoid play an important role in neural networks. In early ANNs a step function was used because of its similarity to action potentials in biological neurons, but the fact that the function is nonlinear is actually critical to DNNs: if the layers were all linear, the result would simply be a linear output function. By having layers of nonlinear function, DNNs can learn far more complex relationships between the features and their targets. While the perceptron used the step function to determine activation, this was eventually replaced by the sigmoid function. It is differentiable across its domain, which has the important benefit of being compatible with gradient descent,

an extremely effective optimisation algorithm that can be used with DNNs. Gradient descent determines how weights should be altered to reduce error, and can be combined with the backpropagation algorithm to handle the multiple layers of DNNs (Rumelhart et al., 1986). In this algorithm, first input data passes through all layers of neurons with their weights randomly initialised (called the forward pass). The output of this is compared to the target, and the error is calculated. In the output layer, the amount each connection contributed to the error is calculated. This process is repeated, calculating error from lower layer connections working next through the hidden layers and finishing with the input layer. Finally gradient descent is used to calculate the optimal corrections to all weights.

The structure VGGNet highlights several important principles of structure design and hyperparameter selection. However early deep learning models were often difficult to train because of vanishing gradients. When using gradient descent the individual gradients, used to change the weights, become smaller and smaller as the layers get lower. In some cases the gradients are so small the lower layers are left virtually unchanged, preventing the model from converging on a good solution. However it has been shown that normalising the inputs of each layer substantially reduces the vanishing gradient problem, which can be accomplished through the addition of batch normalisation layers (Ioffe & Szegedy, 2015).

With these tools and the increased computing power, extremely high capacity models can be built, however there are several pitfalls to avoid. Firstly, a model's capacity has to be well suited to the data so that the samples are not "memorised", i.e. parameters becoming dedicated to correctly classifying the specific examples from the training data. This is called overfitting a model, and results in poor generalisation when the model is introduced to unseen data. Another related consideration is balancing the dataset and is closely connected to how training typically works: after seeing a mini batch of a set number of examples of the training data, the weights are updated to minimise the error (full batch, as

opposed to mini batch, means error is calculated from all training data at once and then updated; this strategy is rarely used because mini batch is much faster and creates near identical models). In classification tasks, if the different classes are distinctly different with representative variation present in the data, accurate results can be obtained regardless of the imbalance - this is because deep learning algorithms do not directly learn the frequency of the classes. However, as the complexity of the task increases and the distributions of signal being classified overlap between classes, models favour predicting classes present at higher frequency in the training data (Japkowicz, 2000). By balancing datasets through sampling, including batch normalisation layers, careful hyperparameter selection, and strategically designing model structure, extremely powerful CNNs can be built and trained.

CNNs are very good at classifying data composed of positionally ordered signals, however often in molecular biology there is often additional useful information that is known or collected. The primary benefit of the convolutional filters is the consideration of this positional signal, but other methods can be used when data are of a different type. The Wide and Deep neural network design utilises a full scale CNN, but additionally complements this with inputs that bypass the convolutional filters (Cheng et al., 2016). This is useful in cases where additional information outside of the raw signal is useful to the task performed by the model. For instance a peak's enrichment level and its shape provide different information. Enrichment simply compares signal at a point to its surroundings, and in the case of ATAC-seq, ChIP-seq, and DNase-seq which are bulk assays, suggests the proportion of the cell population where a sequenced fragment has been pulled down. However these are wet lab assays, relying on multistage processes capable of introducing noise at every step. The shape of a peak suggests if this enrichment is due to noise or from a biological event, as biological peaks have an easily identifiable pattern in their coverage tracks.

CNNs are an important class of supervised learning algorithms, and their de-

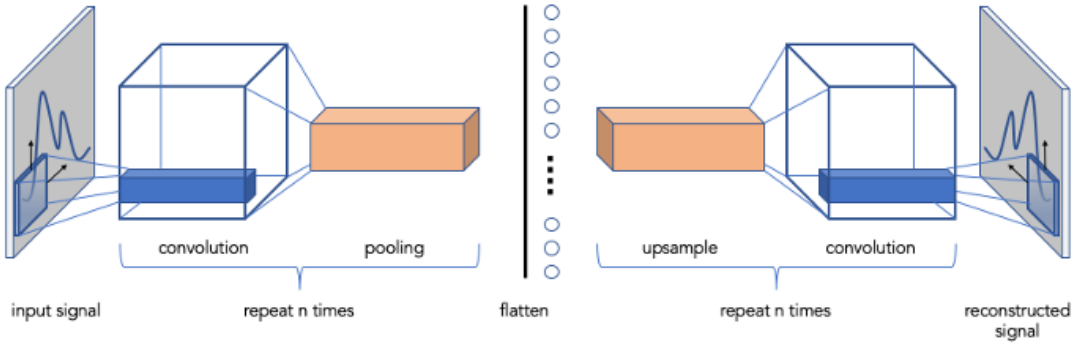


Figure 3.3: Structure of a convolutional autoencoder. Mirror images of a conventional CNN are centred around a layer whereby all output matrices are flattened and built up again through upsampling and convolution; the target output is the same as the input, making this an unsupervised learning technique of machine learning.

sign structure is utilised in a useful unsupervised learning technique: the convolutional autoencoder. A convolutional autoencoder is a type of neural network used to learn dense representations of input data (Géron, 2019). These work by training the model to predict the input after it has been condensed. The first half of the structure actually *is* a CNN used to condense and summarise the input data, and the second half is essentially a mirror of this just building up (generating a larger number of outputs than inputs) rather than breaking down (Fig. 3.3). The target outputs of the model are the exact values represented in the inputs, and because the targets are inherent in the data it is considered an unsupervised learning technique. In a typical CNN model, as the layers get deeper the information they output condenses. For an autoencoder this is true as well, except structurally this layer is actually the central layer, as these relatively few data points are then used to reconstruct the whole input data for a sample. Interestingly, the final output is not often the useful information provided by the model, as it is traditionally just an imperfect representation of the input. Instead the condensed information in the central layer, composed of far fewer dimensions than the input data, is ex-

tracted and used for clustering or further dimensionality reduction. Convolutional autoencoders are useful because they are capable of leveraging the impressive information compressing abilities of CNNs in an unsupervised environment.

Related to autoencoders are encoder-decoder model structures. In this system, the encoding process mirrors that of the first half of an autoencoder, compressing the input into a reduced number of dimensions. From here however, instead of building the latent space representation back into the input, it is transformed into another type of output. When translating language for instance, converting between languages one word at a time may result in nonsensical translated sentences. An encoder-decoder model might work to encode the entire sentence into a universal latent space representation that could then be reconstructed with a decoder specific to a particular language.

Training an encoder-decoder can be as simple as training two autoencoders with the same model structure, and swapping the second half of the autoencoders. In order to ensure that the translations were indeed accurate, after exchanging the halves of the trained models, additional training using the original input and the translated target is required; this also makes this a supervised learning task. Encoder-decoder models solve problems around translations, and related to this are tasks to enhancing signal, for instance "translating" standard resolution images into super resolution images. While standard encoder-decoders can be employed for this task, more commonly this is undertaken using generative adversarial networks (GANs) because of their potential for high performance. Introduced in 2014, GANs are complicated structures which are actually composed of two models. The first model is called the generator and creates fake target data, while the other model, the discriminator, attempts to distinguish generated data from real data (Goodfellow et al., 2014). Wrapping this in a machine learning framework mean that both models improve: the generator makes increasingly convincing fakes while the discriminator is able to detect "counterfeit" target outputs with greater precision.

Often deep learning models can produce quality data analysis from a single replicate. In biology, however, the complicated systems being interrogated and assessed present heterogeneous samples even in well controlled experiments. Well established methods exist for combining statistical tests from multiple replicates such as Fisher's approach. In classification models which utilise the softmax activation function, simple probabilities are output, allowing for the use of different methods. The way in which many CNNs work is to provide many examples of a classes of input data, and based on attributes it has automatically optimised for, it is able to make a prediction on data at a certain probability. The accuracy of these predictions is also measurable. This makes implementing a Bayesian updating approach for combining observations particularly well suited: the prior in the first sample is simply the chance of the class being observed in the dataset, then the posterior of this is used as the prior for the next sample. This can continue on in this fashion through the replicates, using the calculated posterior from the previous sample as the prior for the next sample. Bayesian updating provides a convenient way of combining information from multiple biological replicates analysed by CNNs.

3.2 Results

3.2.1 CNN model

A CNN was designed for the purpose of distinguishing enriched regions arising from biological events from noise, using the characteristic shape of these classes as described in Chapter 2. Here the signal being input into the model is the number of overlapping sequenced reads in a coverage track (after normalisation). Beginning with a structure similar to VGGNet, modification were applied after testing various numbers of convolutional layers, filters, sizes, and pooling layers. A CNN was ultimately designed through careful consideration of the area covered by the convolutional filters. The first convolutional block was composed

of five convolutional layers, with 32 filters of size two. This means a filter in the first convolutional layer outputs a single value summary based on the coverage from two base pairs. In binary terms, this two bp combination can be thought to have four states, the first base pair and second base pair low (0, 0), then low, high (0, 1), then high, low (1, 0), then high, high (1, 1). Each state could be recognised by the convolutional filter and passed to the next deeper layer. However, by having 32 filters even more subtle patterns could be recognised than the binary ones described above, for instance a (0.2, 0.85) pattern may be commonly seen with double peaks. Using a stride length of one, the next convolutional transformation is applied to the second base pair from the first summary, plus one new bp (the next in the sequence). The output is also placed in sequential order, whereby the next convolutional layer applies another size two filter to these outputs. Because of the stride length of one, this this second convolutional layer is effectively taking the values of two summaries, covering 3 unique bp, and making another single value summary. The process continues through the convolutional layers, with the output of the final convolutional layer in this block covering 6 bp using 160 filters. Each convolutional layer was followed by a batch normalisation layer to protect from vanishing/exploding gradients, and the convolution block finished with a pooling layer (size 2) to improve candidate peak coordinate invariance, plus a 25% dropout to prevent overfitting. This fed into another convolutional layer, this time with a large filter size of 32 (meaning that the area of the raw track considered was 38 bp). This larger filter size combining 32 precise convolutional summaries required a large number of potential patterns to be optimised for, and as such this layer trained 128 filters. This was the only convolutional filter of this block, which similarly concluded with size 2 pooling but 50% dropout. Finally, two fully connected layers were used, first with inputs of 233 (coming off of the convolutional block) with 1000 outputs and then feeding directly into another layer with 6 outputs, one for each class, returning the final output probabilities for each region.

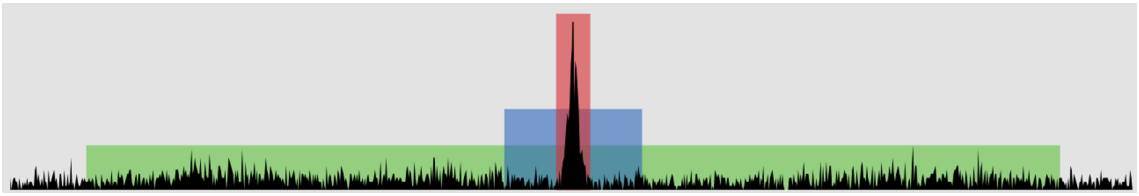


Figure 3.4: Three inputs signals used for the CNN model. For each regions of interest, a 3x2000 matrix of signal, in the form of normalised coverage, is delivered to the model: 2000 base pairs (bp) of surrounding the region (blue), signal only from the region of interest expanded to 2000 bins (red), and 20,000 bp compressed into 2000 bins (green).

Deep learning in general requires less feature engineering, as raw data is frequently used allowing the model to select the most relevant information to the task at hand. Initially this model followed this principle closely, only normalising the signal to the chromosome, finding the midpoint of the region and extracting the surrounding 2 kb signal (+-1000 bp from the midpoint). However better performance was achieved when additional attention was directed towards viewing a region from multiple perspectives using "zoom levels". Ultimately the signal extraction method implemented was coordinate based, forming a 3 x 2000 matrix for each of the candidate peak regions. In addition to the raw 2 kb, the signal of exclusively the candidate region was extracted, and expanded or contracted to 2000 points. Next the surrounding 20 kb were extracted, using the midpoint once again and finding the average signal over 10 bp (Fig. 3.4). The signal extraction process started by first normalising values across the chromosome such that it had a mean of zero and a standard deviation of one. Testing was done to see if the normalisation could be taken further, removing all enrichment information and preserving only the shape of the regions. Rather than applying the normalisation across a chromosome, the normalisation was applied per sample, such that every individual region had a mean of zero and standard deviation of one. This strategy resulted in fairly good general accuracy in small tester datasets, however the mistakes it made were somewhat egregious. By normalising out the context of the region, noise that would be filtered out by even a passive enrichment threshold were classified as peaks with high probability. For this reason this normalisation strategy was not used for further analysis.

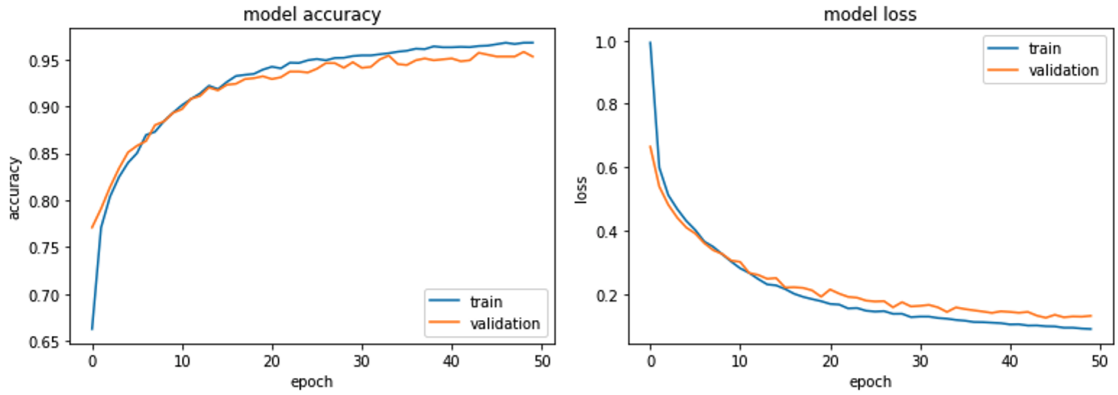


Figure 3.5: Accuracy and loss collected across training epochs for the CNN-only peak caller. Training was stopped before accuracy increased in training data but decreased in validation data which would indicate over fitting.

The data used for this model were low level enriched noise and very strong peaks, and the model performed extremely well differentiating them. Model training proceeded as expected, with a relatively smooth increase in accuracy and decrease in loss for both training and validation data (Fig. 3.5).

Included in the full scale training dataset were labels describing the source of a sample. Rather than classifying a region as a general peak, the model was designed to predict the probability of a region being a peak from a specific experiment type (all noise regions, regardless of source, were simply labelled "noise"). The model was trained with a softmax activation function for its output, so that the probability of the region being a peak from any source was simply the sum of the different peak probabilities (or mathematically equivalent, 1 - noise score). Hold out testing data (not used in training nor validation) was used to assess model performance and totalled 1000 regions. Because this is a multiclass classification task, the ROC curve has one line plotted for each class. This measures the model as if it were six separate binary classifiers (e.g. detecting if a region is an ATAC peak or not an ATAC peak). Despite the peaks often having indistinguishable differences to the naked eye, the model accurately predicted the source of the signal, reflected in AUC values approaching perfect scores for the different classifications. Noise classification achieved the highest AUC at over 0.99978. Because noise is combined from all data sources, this also gives the AUC for us-

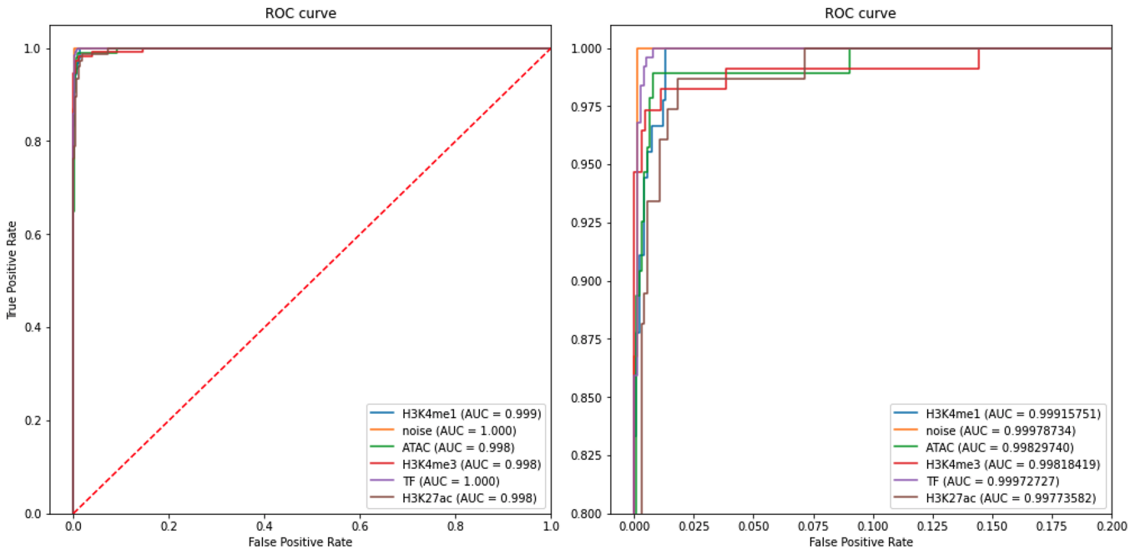


Figure 3.6: Receiver operator characteristic (ROC) curve with calculated area under the curve (AUC) for each of the different classifications of the CNN-only peak caller. The full plot (left) and the zoomed in plot (right) both show near perfect accuracy for this dataset.

ing the model to detect peaks versus noise regions without the additional source prediction (Fig. 3.6).

A confusion matrix was generated to further investigate the multiclass predictions. When the given label was noise, the model accurately predicted that the region was noise in every case; only twice did the model predict the region was noise when it was actually a peak. There were no particular instances of classes being commonly confused, accurately predicting the source of signal in 976 out of 1000 regions tested (Fig. 3.7).

3.2.2 Chromatin capture conformation interactions

ATAC-seq, ChIP-seq, DNase-seq, and ChIPmentation are used to find locations of DNA-protein binding activity. Far different from this, however, is Chromatin Capture Conformation (3C), which is used to detect physical contacts between *cis*-acting sequences (Dekker et al., 2002). This is done to better understand genomic regulation by investigating the three dimensional arrangement of DNA within cells using interaction frequencies. The experimental technique is carried out by first crosslinking DNA, making physically close DNA (though not necessar-

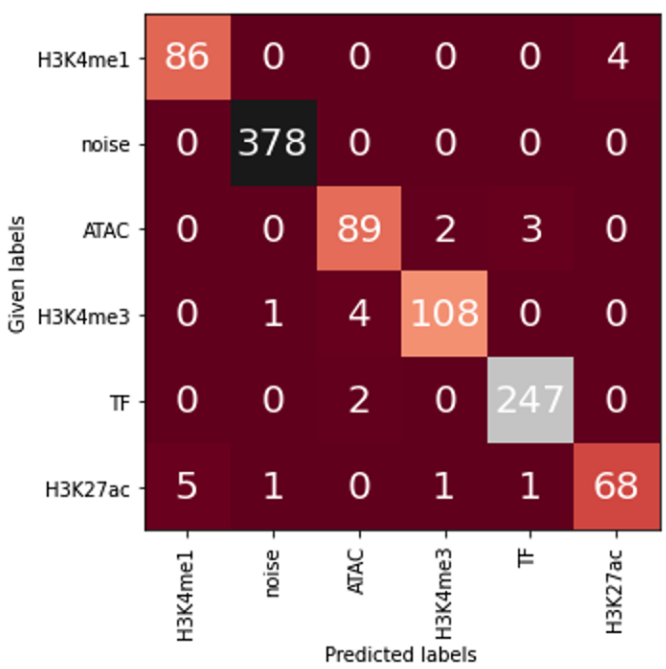


Figure 3.7: Confusion matrix showing class predictions from the CNN-only peak caller. These testing data were composed of unambiguous peak and noise regions from five experiment types and were unseen during model training. Noise regions were combined from all data sources, and also indicate the models performance for detecting peaks versus noise (without predicting the experiment type). Near perfect scores were achieved for each classification type.

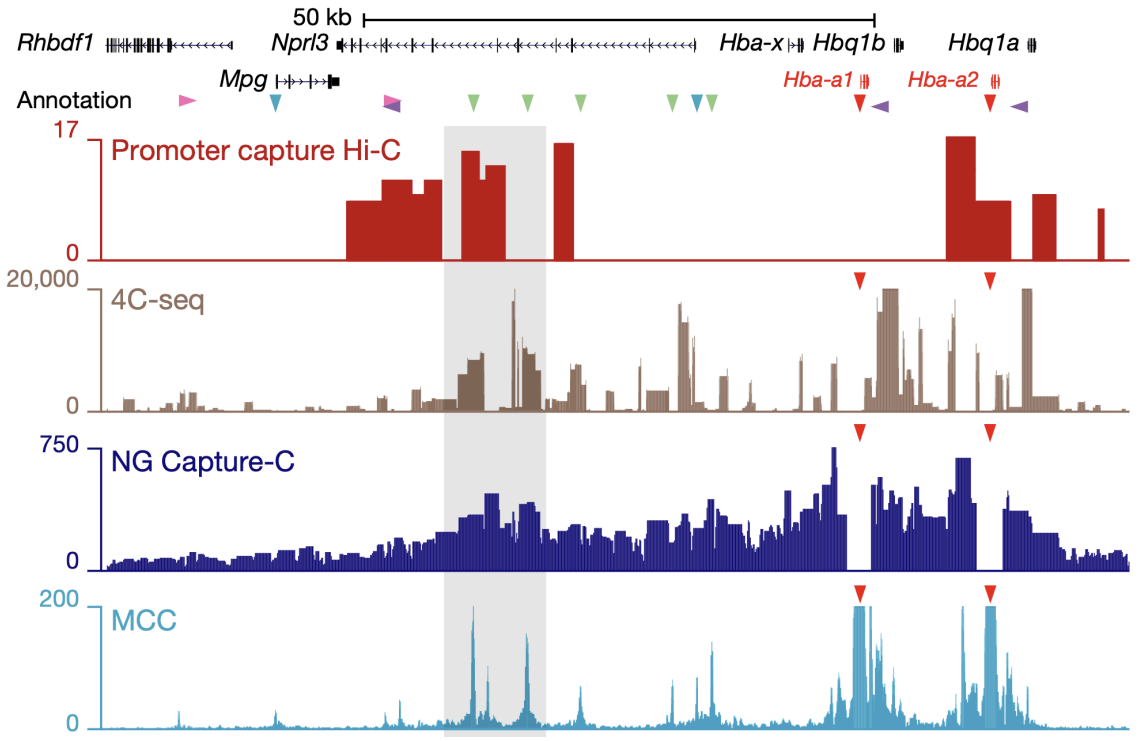


Figure 3.8: Comparison showing the HBA locus using several common chromatin conformation capture techniques, highlighting the high resolution capabilities of MCC. Figure adapted from Hua et al., 2021.

ily close in terms of linear DNA) locked together. Next the DNA is fragmented (typically enzymically), and the crosslinked DNA strands are ligated together. Sequence capturing techniques such as Capture-C use oligonucleotides to selectively pull down fragments from loci of interest, which are then mapped to a reference genome (Davies et al., 2015; Hughes et al., 2014). This approach has continued to be developed, with the latest iteration Micro-Capture-C (MCC) capable of base pair resolution (Hua et al., 2021). The higher resolution coverage tracks allow for an increased understanding of *cis*-acting sequences when compared with other methods (Fig. 3.8).

The fragmenting step and the depth of sampling largely limits the resolution of 3C techniques, with the frequency of the cut sites determining the size of the "bins" fragments are placed in. MCC uses micrococcal nuclease (MNase) for fragmentation, which does not require a specific cut site, and fragments DNA largely without bias (Hsieh et al., 2016; Kornberg, 1974). In addition the protocol is optimised to minimally disrupt nuclear architecture, which when deeply enriched us-

ing the Capture-C process generates coverage maps at a resolution very similar to chromatin profiling assays (e.g. ATAC-seq), representing a profound increase in resolution. Importantly, the shape of the peaks is also highly concordant with the shape of peaks generated by ATAC-seq assays, and suggests the possibility of adapting my work identifying chromatin peaks to identifying interaction peaks in MCC data. In order to show that the regions found by the MCC technique were both enriched, and possessed the shape of a region arising from a biological event, the CNN model was used to assess the coverage tracks produced for these analyses.

Evaluation was carried out on six replicates across two cell types (mouse erythroid and ES cells) at 330 viewpoints using the MCC technique. The CNN model was compared against the peak caller MACS2, as well as a standard Poisson model. The Poisson model simply used the max height of the region and background from the coordinate matched region of the input track. Combining biological replicates can affect results, and so the coverage tracks were assessed both by simply summing the reads from all replicates, as well as assessing the replicates individually and combining significance scores. When combining p-values Fisher's method was used for MACS2 and the Poisson model. However the output from the CNN model (and many deep learning classification models in general) is a probability of an observed class. In order to combine these probabilities a Bayesian updating method was used. For each replicate, a posterior probability was calculated using the model's output (peak or noise classification) along with the model's accuracy metrics for class predictions. This posterior was used as the new prior for the next replicate, with this pattern of calculations continuing through all of the replicates. This yielded probability estimates that used information from all of the observations in combination with knowledge of the model's performance. Though not customary in many fields, biological data often utilises multiple replicates and deep learning classification applications could benefit from this approach.

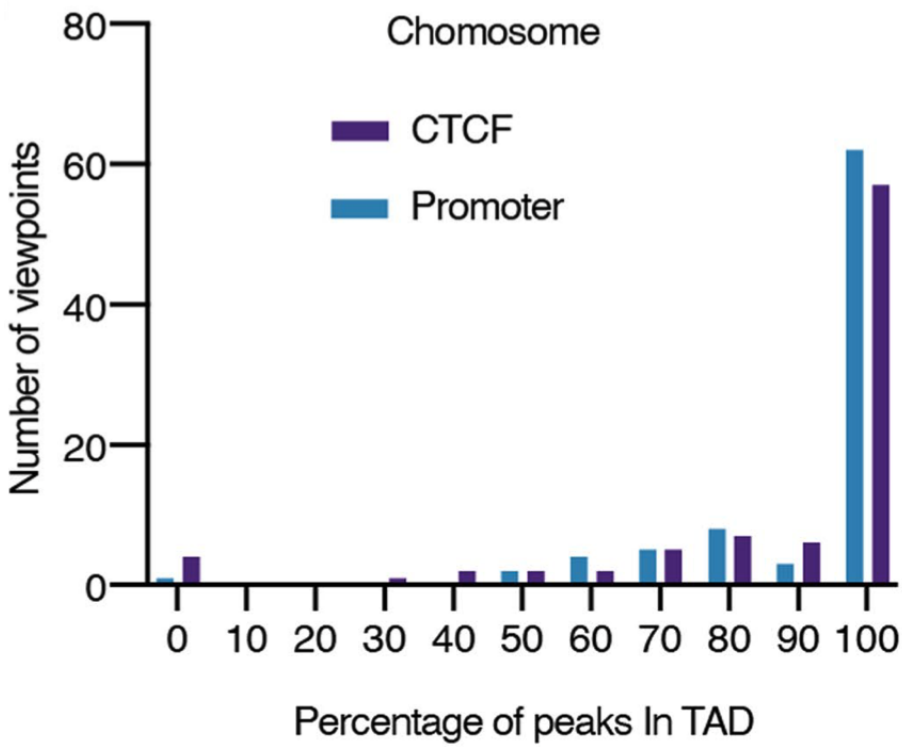


Figure 3.9: When consensus peak calls were used from the different methods, they overwhelmingly were found with the same TADs as the loci being captured. Figure adapted from Hua et al., 2021.

Peak calls were performed with all three methods. Across the experimental conditions a consensus region was found using all three peak callers in 1,319 out of 1,320 cases (330 capture sites * two cell types * two data combining strategies [individual replications and merged tracks]). Consensus regions were carried forward for further analysis, and as expected, these regions were preferentially found in the same topologically associated domains (TADs) as the capture sites (Fig. 3.9).

Outside of these primary consensus regions, the peak callers also identified other potential interactions, and sometimes these regions were only identified by a single method. This was particularly true using MACS2, which identified far more uniquely enriched regions (6,307) compared to the Poisson method (2,985) or the CNN model (986) (Fig. 3.10). This may be due to a high false positive rate, often associated with this method of peak calling when used for chromatin binding.

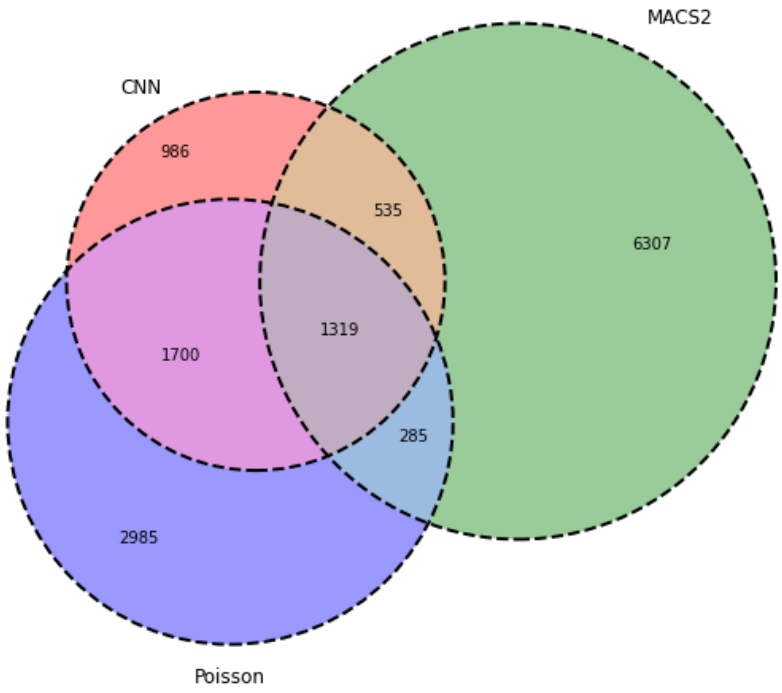


Figure 3.10: Venn diagram showing the number of different regions identified by three peak calling methods: a convolutional neural network (CNN), a Poisson model, and MACS2.

3.2.3 Autoencoder

Convolutional autoencoders are unsupervised models which are useful for condensing complex input samples into smaller representations with minimal loss. Supervised learning requires labelled datasets which are not necessary for autoencoders, however labels can give insight into how the algorithm behaves. If the condensed output of the central layers, called latent layer encoding, could be used to separate classes in an unsupervised fashion this would provide a method of automated data labelling replacing the tedious visual inspection process. To test this an autoencoder, using structure resembling the CNN described above, was trained using candidate peaks, and afterwards both labelled and unlabelled data were input into the model with the compressed central layer extracted for each region. These data were then run through an unsupervised learning classifier, a Bayesian Gaussian mixture model, to predict clusters. Unfortunately using the algorithm's groupings, no sets cleanly separated the peak regions from noise.

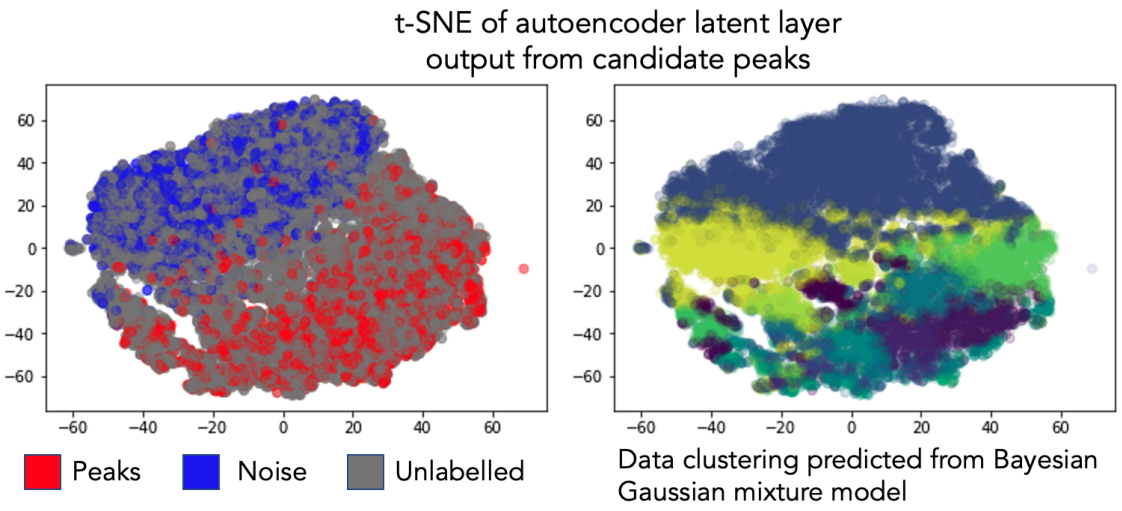


Figure 3.11: t-SNE of the central latent layer of an autoencoder trained with candidate peaks. Unseen test data include labelled peaks and noise as well as unlabelled regions (left). The latent layer outputs were also input into a Bayesian Gaussian mixture model to predict clusters, shown in blue to yellow, present in the data (right).

This is easily visualised using t-SNE to reduce the central layers' dimensions down to two for plotting, then overlaying the predicted clusters. The t-SNE does a reasonably good job separating the peaks from noise when applied to the unadulterated input data, and with the algorithms output overlaid it is clear a large cluster spreads over both peak and noise regions (Fig. 3.11).

While the latent layer encoding was unable to distinguish classes in this setup, further testing was done to see if an autoencoder could be used as a means of incorporating additional data in model training.

3.2.4 Wide and deep model

While the previous CNN showed good accuracy through feature engineering, additional benefits exist for deliberate separation of enrichment from region shape. Firstly, there are specialised models which excel in both areas. Previous work showed that a linear combination of enrichment values could accurately categorise peaks and noise in these experiments, though an upper limit existed (Chapter 2). CNNs are very good at classification, however peak height may be overly influential to the scoring - even if it is only a relative height scaled to

chromosome signal. Normalising the regions completely preserves the shape only, however testing with the CNN-only model showed this decreased accuracy. However in the case of a separated shape and enrichment model, the CNN could score a region on shape alone, the logistic regression model could score on enrichment alone, and these scores could be combined to form an overall peak score. This was the motivation behind building a wide and deep model, which was carried out using the labelled dataset described in the previous chapter.

The wide part of the model was composed of enrichment measurements plus output from a logistic regression model. The logistic regression model was based on testing of the different combinations of signal measurements, background size ranges, and number of enrichments calculated. In Chapter 2 it was shown that top performance was achieved using 200 kb range to measure background signal over 40 intervals, however the structure used was based on 10 background measurement values with 100 kb of coverage plus chromosome mean. While this model's results (sans chromosome mean) were slightly lower, the difference between it and the top performer were not statistically significant. This model was selected because the high performance came at a far lower computational cost with reduced edge behaviour effects. Edge cases occur when a region is too near the edge of a chromosome to fully calculate background, and so this change cut that area in half. Additionally these enrichment measurements used for the logistic regression were input directly as part of the wide portion of the model.

The deep part of the model was composed of a CNN with optimised structure and tuned hyperparameters. Input of 2000 bp of coverage signal for each region was used, each region being normalised such that its mean signal was one and standard deviation was zero. The output of the CNN, was combined with the output of the logistic regression (the enrichment score), as well as the p-values from the 11 enrichment measurements. The complete model structure included these values input into a multilayer perceptron, which output an overall peak score (Fig. 3.12).

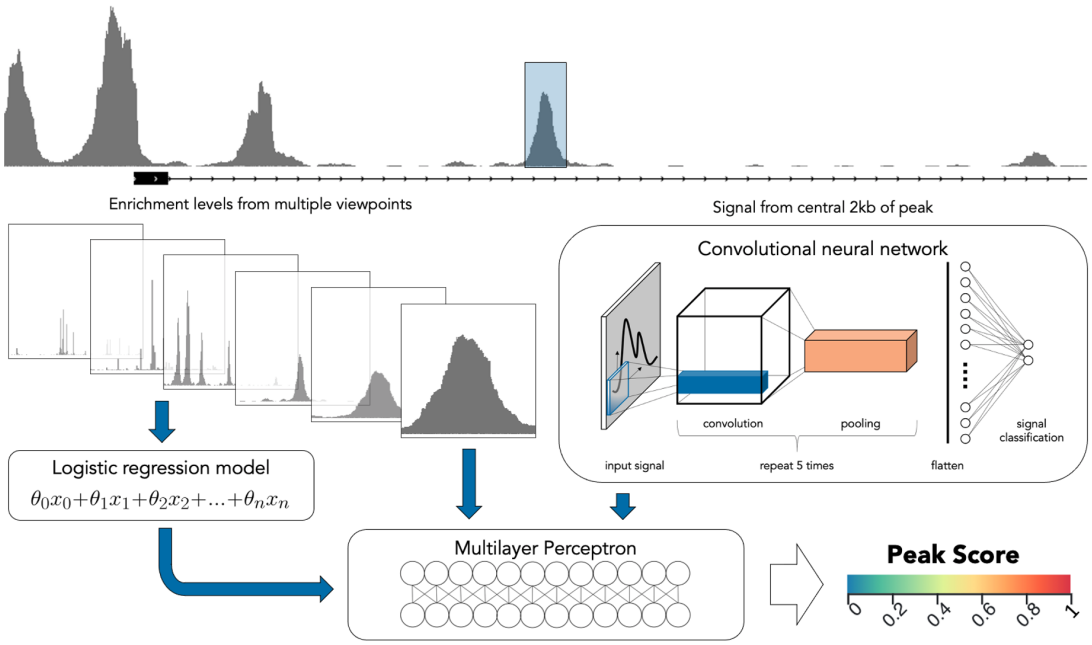


Figure 3.12: Wide and deep model overview. A candidate peak (highlighted blue) is compared to its surroundings from multiple viewpoints; enrichment measurements are input into a logistic regression model. Signal from the candidate region's central 2 kilobases (kb) are input into a convolutional neural network (CNN). Outputs from the logistic regression model and CNN are combined with the multiple enrichment measurements in a multilayer perceptron which outputs a final overall peak score.

Special considerations were made when structuring the model training process. This was to create not only an accurate final score produced by the model, but also to gather useful metrics used to generate this score. The logistic regression model was trained separately and weights locked for all subsequent training. As demonstrated in Chapter 2 the logistic regression model can be highly accurate without any other information, and is a useful metric in its own right. The cost function was constructed such that the model was penalised for both incorrectly assigning an overall peak score as well as shape score. This was divided such that 70% of the cost was derived from the overall peak score and 30% was from the shape score. Because many regions can be accurately predicted from enrichment, a separate penalty was incurred when the output of the shape score did not match the labelled data - this ensured that the model was not over-relying on enrichment and that the shape score remained accurate throughout training. Ultimately for each candidate peak, the 11 p-values collected for the model are

returned alongside the enrichment score, shape score, and overall peak score, thus giving a comprehensive assessment of each region.

During the construction of the CNN, several different model structures and hyperparameter values were considered for testing. Small scale test models, trained with subsets of data over only a few epochs showed promise using a range of setups. This included the number and composition of convolutional blocks. Good results were seen with anywhere from one to five convolutional blocks, composed of one to five convolutional layers, plus a possible pooling layer (either max pooling, average pooling, or no pooling). Important hyperparameter variations were connected to convolutional filters, with both their number (between 20 and 100) and their size (between two and 10 base pairs) showing solid performance using many different values without converging around an obvious optimum. These variables, along with several other important factors (dense layer construction, dropout rate, and learning rate) could be altered to form millions of unique models. To address this a total of 5,500 full-scale models were built, selecting structure/hyperparameter combinations at random. They were trained for up to 50 epochs, with early stopping when performance dropped for 4 consecutive epochs. While this sampling may not have yielded the top model of all possible combinations, it was used to investigate trends in the CNNs generally as well as the top performing models.

This brute force method of model optimisation yielded several interesting results. Firstly, while trends could be observed when varying these hyperparameters, there was no single factor able to guarantee high nor low performance; any particular hyperparameter was found as a top model in some ensemble of these options. Top performing models were defined as those which achieved greater than 98.5% accuracy on unseen testing data. Pooling showed perhaps the strongest influence, as many more top performing models used max pooling in their convolutional blocks. Analysing the number of convolutional blocks, also connected to pooling, showed that the most number of top performing models

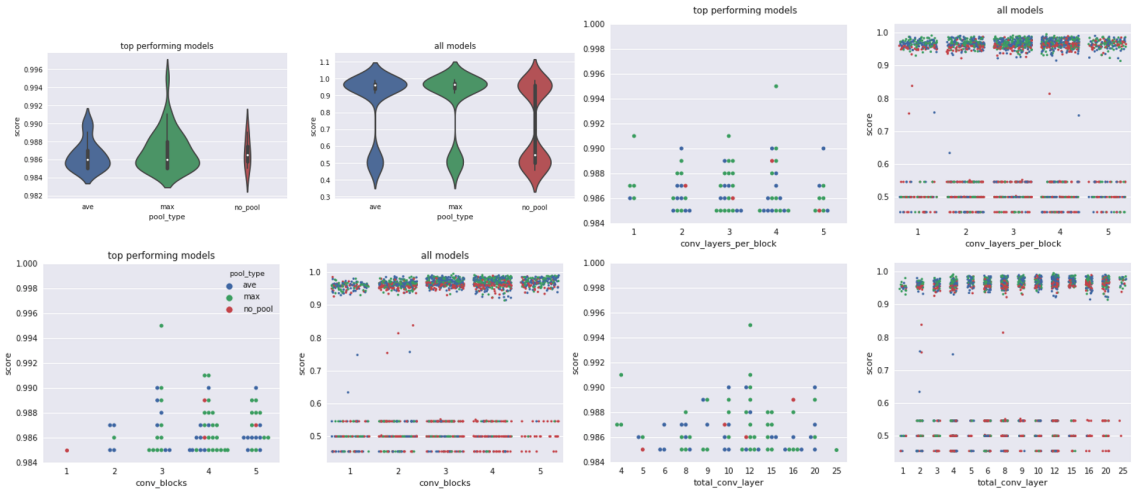


Figure 3.13: Model accuracy compared across pooling type (top left two panels), convolutional blocks (bottom left two panels), convolutional layers per block (top right two panels), and total convolutional filters (bottom right two panels). 5,000 models were created with the top models, defined as those with accuracy of 98.5% or greater when classifying unseen data (left panels of the sets of two), compared to all models (right panels of the sets of two).

contained four of these blocks, though the singular best model had three. While the number of convolutional layers per block is fairly well spread across two, three, and four, the total number of convolutional layers showed best performance at 12 layers - including the top performing model (Fig. 3.13). Filter size (also called kernel size) showed a wide range of performances without a consensus as to the best performer. Similarly, no obvious trend surfaced with the number of filters. The top performer had a filter size of four base pairs with a total of 40 filters, but good results were seen with the full range of two to 10 bp sizes and 20 to 100 filters (Fig. 3.14). Finally, 5-fold cross validation was used for the top 10 performing models. The top model structure achieved a mean 99.89% accuracy, and its structure and hyperparameters were used for subsequent work with the wide and deep model design (Fig. 3.15).

Testing was done to see if an autoencoder could be used to pretrain the model weights for the CNN. Because balanced datasets generally improve performance for deep learning classification tasks, a strategic implementation of sampling was used such that every peak labelled was retained and roughly the same number of noise regions were sampled. Because enriched noise regions with the great-

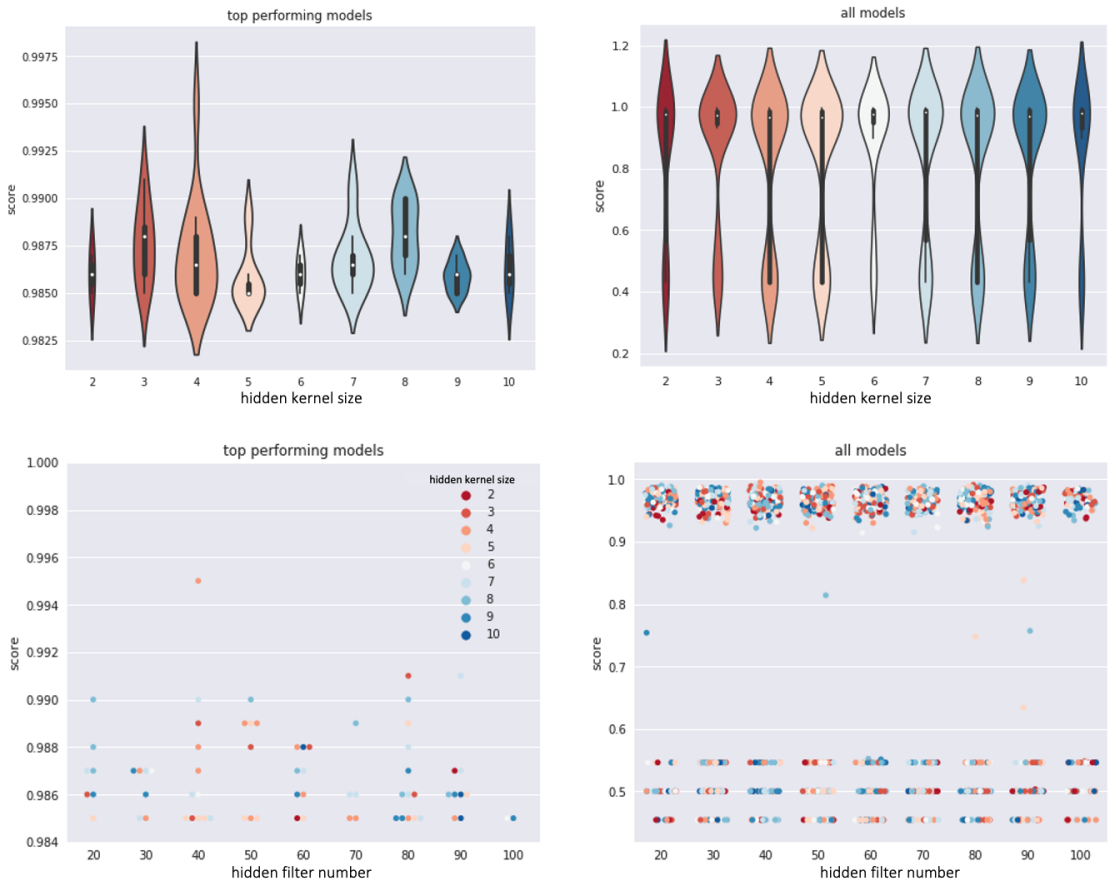


Figure 3.14: Model accuracy compared against filter (kernel) size (top) and number (bottom). 5,000 models were created with the top models, defined as those with accuracy of 98.5% or greater when classifying unseen data (left), compared to all models (right).

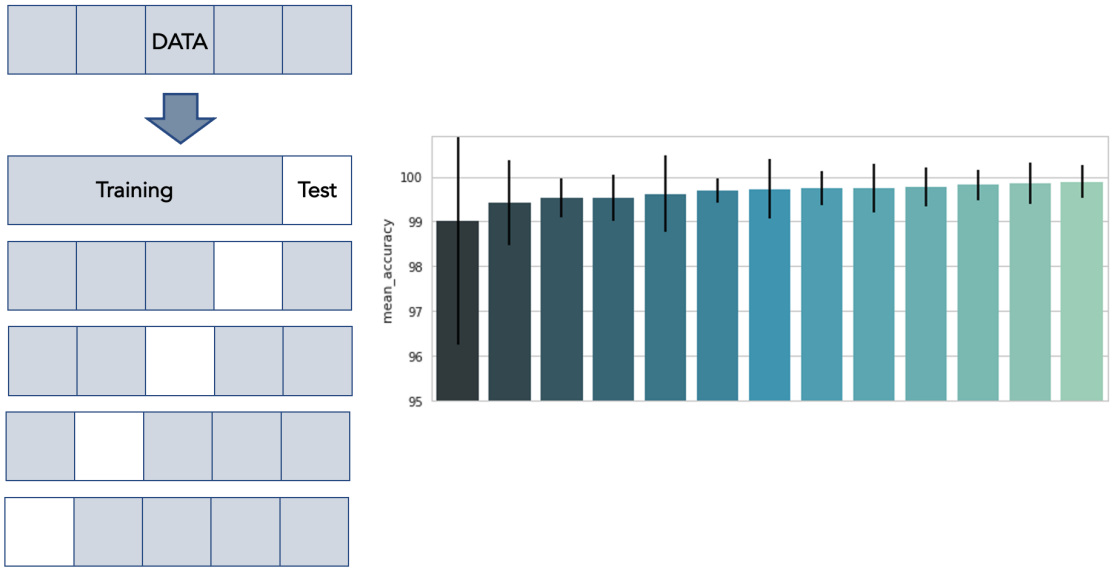


Figure 3.15: Five fold cross validation strategy for training and testing (left), and the accuracy of the top 10 models found from hyperparameter testing. The structure and hyperparameters from the top performing model (99.89% average accuracy) were used for subsequent work using the wide and deep model design.

est maximum height are most prone to lead to false positives, all labelled noise regions at the 25th percentile or greater were included in the dataset and equal numbers below the 25th percentile were randomly sampled. A great deal of effort was put forth in labelling data, but unfortunately the vast majority of these samples were never used. Balancing the datasets was done to match the task at hand, i.e. trying to predict the right class a region belongs to. However when the task is predicting the input, as it is in an autoencoder, all regions are relevant since class information is not used in the process. This training strategy was implemented using the entire labelled dataset in an autoencoder whose first-half structure was a CNN with the exact same configuration as the wide and deep model used (Fig. 3.16). The autoencoder was trained, with the weights from each layer of the neural network saved. Instead of using random initialisation for the learned parameters, these weights were transferred over from the autoencoder. Unfortunately the autoencoder pretraining lowered classification accuracy. Many samples that were used in pretraining but excluded in the hand labelled dataset were low quality noise regions, and included them may have caused the reduced performance. Pretraining was therefore dropped from further use, though fortunately even without implementing this strategy the model still achieved over 99% accuracy.

3.2.5 Enhancing ChIPmentation signal

ChIPmentation is an experiment used to detect TF and histone binding (like ChIP-seq), and does so using Tn5 transposases (like ATAC-seq) (Schmidl et al., 2015). The standard to which ChIPmentation is compared is ChIP-seq; here, a GAN was built to test if ChIPmentation signal (i.e. the pattern of sequenced reads in coverage tracks) could be transformed to ChIP-seq signal, thus upscaling it. Traditionally, a GAN's generator would take in random noise, and create an output that could conceivably be a real input. Instead of random noise, the generator in this model used an encoder-decoder to transform the ChIPmentation data to be more

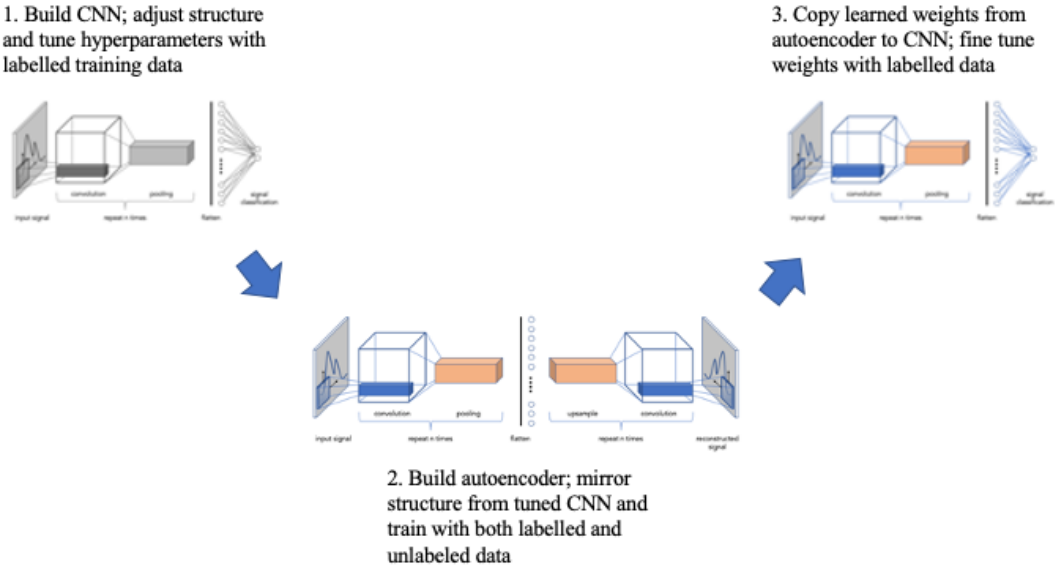


Figure 3.16: Autoencoder pretraining strategy. The structure of the CNN (left) was used to design the autoencoder (middle), perfectly replicating it on the input side and of the autoencoder while using a mirrored structure for the output side. Once the autoencoder is trained, the weights it learns can be transferred to the CNN (right). This strategy can be employed to include additional data not used to train the CNN.

ChIP-seq-like. Working much like an autoencoder, signal from ChIPmentation is compressed into a fraction of the initial input features. Where an autoencoder would reconstruct the initial input signal from the compressed representation, the encoder-decoder here instead constructed signal from the coordinate matched region of the ChIP-seq track. The discriminator's job was to determine which signal was the original ChIP-seq, and which was the transformed ChIPmentation for the same coordinate matched region. The generator and discriminator were trained separately, with the intention that the generator makes more accurate transformations, and the discriminator is able to detect increasingly finer details.

To measure the effect of the GAN, peaks were called for H3K4me3 ChIP-seq in mouse. Roughly 20k peaks were found there, and the corresponding regions were extracted from the matched ChIPmentation track. To establish a baseline of correlation between the tracks, the mean squared error (MSE) was calculated for the called peaks between the two experiment types. Without any modification, the MSE between corresponding regions (N.B. not the entire genome) between ChIP-

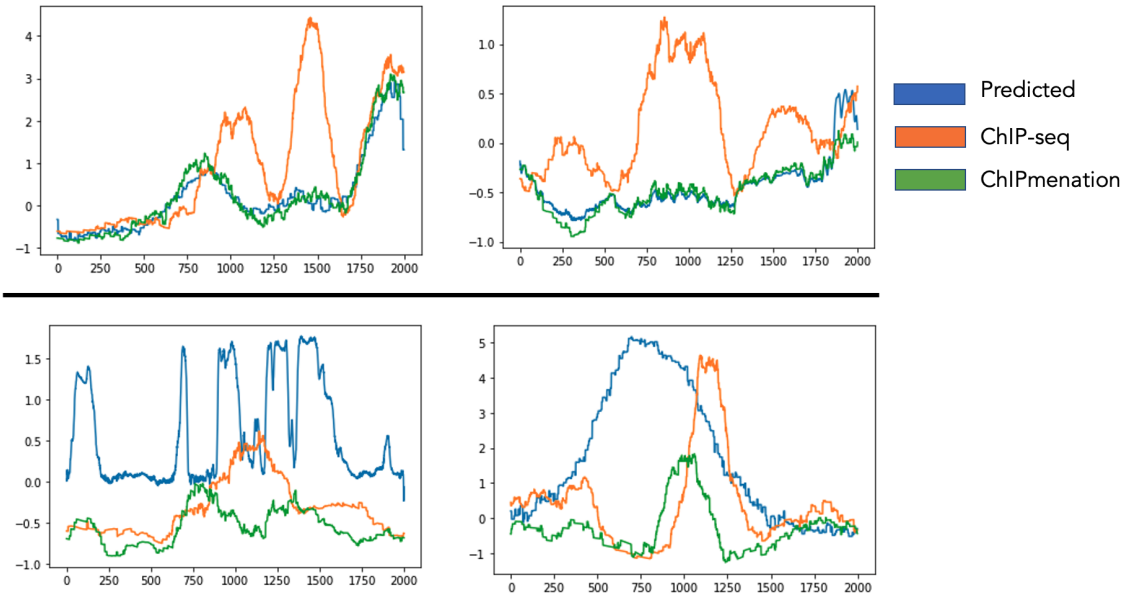


Figure 3.17: Examples of predicted ChIP-seq signal using a generative adversarial network to transform ChIPmentation signal, as well as the original ChIPmentation and ChIP-seq coverage tracks for the corresponding areas. While the correlation improved overall using the GAN, many regions were only very slightly modified (top) to improve signal, while others were grossly transformed (bottom).

seq and ChIPmentation was determined to be 77.6%. Using the GAN to transform the regions of the ChIPmentation track, the MSE was reduced to 65.8%. While this represents a remarkable improvement, the way in which these gains were made left something to be desired. Nearly all of the regions were only very slightly altered from the original signal yielding small reductions in error, however some regions were dramatically altered, heavily distorting the original signal in a way which did not correspond to the ChIP-seq track it was attempting to replicate nor the ChIPmentation track it used as input. The cumulative effect of the small corrections helped reduce the overall MSE, but made no appreciable effect on the interpretation of the signal, while the cumulative effect of large and incorrect transformations was small, but this signal was skewed to such a degree that it could be misinterpreted; ultimately the "improvements" that were made had no practical benefit and a small number of regions were actively made worse (Fig. 3.17).

3.3 Discussion

In this chapter I tried multiple avenues to optimise the performance of a deep learning peak classifier, including designing CNN input zoom levels, convolutional autoencoders pretraining, and GAN upsampling. Some of these strategies proved not to be useful in and of themselves, but led to the development of thoughtfully designed deep learning architecture. A highly effective model was arrived at by understanding the shortcomings of these approaches, and ultimately it has been demonstrated that deep learning, in various forms, can be used to accurately analyse the signal from chromatin profiling assays.

For the CNN only model, the structure of the input data was three channels, each with a different zoom level creating multiple perspectives of a region. Each of these zoom levels was composed of 2000 elements, and for the first channel that meant one element covered exactly one base pair, but for the next channel an element covered a variable number of bp depending on the size of the region, and in the final channel an element covered 10 bp. When a convolutional filter passes over a region of pixels, it passes over each channel simultaneously. The most common usage for a three channel system of input data is in image classification of RGB pictures, where each pixel has an associated red, green, and blue value. In RGB pictures there is a positional connection naturally linking each channel. This is because a filter passes over the same region of the image, just measured in different colour/signal values; the weights learned by the convolutional filters can be optimised to detect patterns by combining information from the different channels. However the CNN model developed here does not utilise positional linkage. For example if a 200 bp region on chromosome 13, centred at position 500,000, was being assessed, then the initial convolutional filter (kernel size three) would be simultaneously optimising the weights of considering the following regions: chr13:499,500-499,503; chr13:499,900 (repeated three times); chr13:490,000-490,030. Unfortunately it is unlikely that any meaningful positional information can be found across these coordinates. These different viewpoints

are still incredibly useful however, because they provide the necessary context to measure enrichment at different levels. Ultimately this structuring of inputs results in some inefficiencies, rather than a detriment to performance. The CNN will learn patterns from each channel separately, meaning that more filters are needed to compensate, costing computational time but still capable of delivering good results.

Results from the CNN demonstrated that biologically important information can be extracted from shapes present in signal. This information has great potential to be found across a range of different genomics experiments, and through collaboration I was able to test this on 3C data from the MCC technique. Experimentally this is a massive change, but thanks to the increased resolution of the MCC method the coverage tracks produced closely resemble the training data used by the constructed neural networks. One change that was uniquely used in the 3C analysis was the implementation of Bayesian updating for combining replicates. Bayes' rule can be easily used because the false positive and false negative rates can be measured directly from training data and its application makes for more informed probability predictions. Here, the error rates calculated were based on binary classifiers - i.e. error rate of a region being predicted as a peak if its true class was noise and vice versa. Using distributions this concept has the potential to be taken even further. For instance, defining a different error rate if the model predicts a peak region with a 55% certainty versus a 95% certainty. This approach, though only utilised for the MCC experiments, could prove useful for other applications of deep learning probability predictions for biological data composed of multiple replicates.

Good results were seen using the convolutional neural network on the 3C experiments, despite never using these data for model training. Because interaction frequencies are being measured, it is expected that promoters or CTCF sites are found to be most associated with other regions within the same TAD. This was observed in this work, providing biological evidence alongside significance mea-

surements. While consensus regions were identified for nearly all samples using each different method, other regions of interest were also identified by the various methods. These were less significantly enriched, and the coordinates did not overlap across the three peak calls. Both the Poisson model and MACS2, which rely on the Poisson distribution, often reported the most significant P-values for the same identified regions. Additionally, these methods also reported higher values of regions unconfirmed using the other models. High false positive rates using Poisson-base peaks callers has been well established in ChIP-seq analysis, and with their similar coverage pattern this may extend to MCC analysis as well. The properties of the Poisson distribution for identifying regions was detailed in Chapter 2, and these ideas will be expanded further to overall peak caller performance in Chapter 4.

Convolutional autoencoders are powerful models, utilising the analysis capabilities of CNNs without the tedious data labelling. Because the CNN model performed well working with ATAC-seq, ChIP-seq, and DNase-seq, autoencoders were identified as a potential method of use - especially because a nearly unlimited amount of unlabelled data could be generated. Unfortunately, clustering the outputs of the most condensed latent layer was unable to separate peak and noise classes. Using other clustering methods may provide better results, and can be followed up as future work.

Autoencoders were also considered to assist with model training. This was in part because the labelled dataset consisted of all of the identified peaks, but only a small sampling of the noise regions. Arguably the most important noise regions were in the dataset, however a massive 724,421 regions labelled were disregarded. Autoencoder pretraining was a method to utilise these data, though here it ultimately degraded performance. This is perhaps because the technique relies on the supposition that the same features learned when rebuilding a sample in an autoencoder are also the best features to predict peaks from noise in a classification task. While this does seem plausible, it may in fact not be the case.

In addition, the same issues that made the downsampling necessary for the labelled dataset could have affected the autoencoder as well. This is because this strategy works under the assumption that the model was agnostic to the class of region, which may have been false. It is conceivable that reconstructing signal from peak regions via autoencoder may in fact be different from reconstructing the noise regions, and because noise regions outnumbered peaks 145 to one, the model may have never learned to understand peaks. As described in Chapter 2, many of the peaks left out of the final dataset were single reads in an area with no surrounding signal - these samples may simply not have been useful for classification CNNs or autoencoders. Pretraining does not replace the training phase, but ideally it can be used as a way to have model weights initialised close to their optimal values, thus requiring fewer examples and passes through the data. Here however, this process did not require less resources nor did it improve performance.

The GAN model succeeded in improving correlation between matched ChIPmentation and ChIP-seq tracks, but did not do so in a useful manner. It should be considered however if this “upscaling” from ChIPmentation to ChIP-seq is indeed necessary. As the bench techniques mature, the signal produced is increasingly clean - while ChIP-seq was treated as the gold standard, recent ChIPmentation tracks produced by members of our lab appear remarkably similar.

The autoencoder training strategy did not improve performance for the wide and deep model, though given the model’s high performance without pretraining there was little room for improvement with the test data. This high performance was achieved by separating enrichment and peak shape. Linearly combining p-values showed improved accuracy over singular enrichment measurements in previous work. The CNN-only model was capable of using peak shape to improve performance, but made mistakes when enrichment information was held back. It also relied on multiple viewpoints to understand the context of the region. While the shape of the surrounding area was encoded across 2000 elements, if

the information extracted from this context was simply relative enrichment this information could be represented as a single value without loss of accuracy. Each approach has limitations, but combining them helps to counteract their individual weaknesses. The subsequent wide and deep model made sense conceptually, however constructing a relatively new type of model to utilise data never analysed in such a way remained a challenge. While perhaps lacking elegance, the brute force method of building many, many models proved to arrive at an acceptable solution.

With an accuracy at 99.89%, the top wide and deep model obviously excelled at classifying the unseen testing data. This accomplishment was the product of understanding the data analysed, finding the shortcomings of p-values, and learning from the mistakes of early stage model building. Achieving high accuracy on testing data is a challenge, however this often fails to translate into real world high performance. Several decisions were made in consideration of of this models use in practical settings. For instance, enrichment values used as inputs to the logistic regression model and the multilayer perceptron were calculated as p-values. While this is an internal model parameter, and not used for significance thresholding, it was decided to use the p-value because of the increased interpretability, though numerous enrichment metrics could have been used to yield similar results. These p-values can also returned as an additional calculated measurement; indeed the results of a traditional statistical peak caller could be mimicked by simply using p-values as the sole filtering metric. In the next chapter model testing is taken further, looking at multiple datasets - internally, externally, and algorithmically labelled data are all included. Besides this, motif enrichment, transcription start sites, and active genomic elements are used as biological markers to provide further support showing the regions classified by this model are true peaks. Finally this testing is done not as an accuracy measurement analysis, but instead a direct comparison to the most commonly used tool in the field, MACS2.

CHAPTER

FOUR

LANCEOTRON

4.1 Introduction

The culmination of my work to develop candidate peak calling, and DL based peak classification models was to integrate the most effective of the approaches to create a usable, flexible, and fully featured peak caller. This meant that beyond technical performance, I not only needed to consider how such a tool would be used, but also how biologists would effectively interact with its output.

The result was the development of a tool called LanceOtron, composed of three modules tailored for different use cases. Each module takes a coverage file as input (in the form of a bigwig file) and returns enriched regions with associated scores as output. Module one, Find and Score Peaks, first labels enriched regions as candidate peaks, then scores them using LanceOtron's deep learning model. Module two, Find and Score Peaks with Input, performs the same function as the first module but additionally calculates the p-values of regions based on enrichment compared to a separate input control track. Module three, Score Peaks uses the DNN to score genomic locations provided as an additional file.

The first two modules, Find and Score Peaks and Find and Score Peaks with Input, employ LanceOtron's candidate peak calling algorithm. This works

by applying a 25-way enrichment test, consisting of different smoothing window-threshold combinations (Chapter 2). This allows for various ways for a region to be considered enriched, with the aim of generating an overcomplete set of all possible areas of interest to present to the neural network for assessment. This is important because it means the ethos of LanceOtron is different from existing peak callers, which include or exclude peaks from the output based on parameters and cut-offs. Instead, LanceOtron's identifies all potentially enriched regions, scores these using machine learning, and returns the complete dataset in a manner that can be examined and queried in its entirety.

The final module, Score Peaks, uses LanceOtron's neural network component in isolation from the candidate peak identifier. This allows users to analyse the quality of peak calls from other tools, publications or databases.

The core of LanceOtron's peak scoring algorithm is a customised wide and deep model (Chapter 3). Briefly, local enrichment measurements are taken from the maximum number of overlapping reads in a peak compared to its surroundings - chromosome-wide as well as 10 kilobases (kb) to 100 kb regions in 10 kb increments. These measurements are used in a logistic regression model, which produces an enrichment score. A base pair resolution view of the signal over a 2 kb window, centred on each peak, is then encoded and input into LanceOtron's CNN. The CNN uses the relationship between the number of overlapping reads at all 2,000 points, i.e., the shape, to determine if the region is a peak arising from a biological event or noise. Finally, a multilayer perceptron combines the outputs from the CNN and logistic regression computations, as well as the 11 local enrichment measurements to produce the overall peak score.

Benchmarking is essential in demonstrating the capabilities of a peak caller or any bioinformatic approach. As this model is based on a supervised machine learning algorithm, training data is required to provide examples of the shapes and enrichment for the peak and noise regions. For this I used 50 datasets from open chromatin, transcription factor and chromatin modification ChIP-seq exper-

iments reaching a total of 736,753 labelled regions covering 499 Mb of genome (Chapter 2). In the field of deep learning, it is common to divide datasets into three unequal parts. Data is randomised, and generally 80% is used as training data to optimise the weights of the DNN. When the goal of the model is to predict classes from unseen examples, this training data cannot provide an accurate assessment for the DNN's performance and so the remaining 20% of the dataset is set aside for evaluation purposes. These evaluation examples are divided into the validation set, which is referenced consistently throughout training to test for overfitting but never used to optimise weights, and the test set which is reserved for the final model performance evaluation exclusively. While this dataset breakdown is good practice generally for DL and was carried out for the underlying DNN used, there are several additional considerations to take in account with peak caller evaluation.

The data used for evaluation, as well as its randomisation strategy, is important to assessing the performance of peak caller. In DL, complete randomisation of the testing data means the model is evaluated across a range of different samples and has the benefit of being robust to idiosyncrasies in labelling or local biases. However this strategy means quintessential class types (i.e. near perfect peaks or blatantly obvious noise) are over represented in these data and may obscure region types that do not appear frequently. I employed several strategies in order to retain the robustness of the DL strategy, but in such a way as to measure the performance expected in real use cases. This strategy was applied across five different benchmarking datasets: ATAC-seq in MCF-7, CTCF ChIP-seq in primary spleen cells, H3K27ac ChIP-seq in HAP-1 cells, H3K4me3 ChIP-seq in MG63 cells, and DNase-seq in A549 cells. First, I hand labelled continuous 1 MB regions, marking every region that was a peak or noise to ensure the full variety of shapes associated with each class. Sequencing coverage can vary by chromosome, so multiple chromosomes were randomly sampled from each dataset - three chromosomes each for the open chromatin tracks, and ten chromosomes

each for the ChIP-seq tracks; this meant labelling a full 36 MB by hand. All of the random chromosomes across the five datasets represents a variety of unseen data, and using continuous regions meant a great breadth of signal shapes were included.

Labelling peaks by visual inspection is highly accurate, and was the gold standard I used for benchmarking. The merit of this technique was first shown by Rye et al. in 2011 during the process of constructing a dataset to use for analysing ChIP-seq peak callers (Rye et al., 2011). Here they compared regions annotated by peak callers as well as humans in three TF ChIP-seq experiments, and attempted to recover the TFs' binding motifs *de novo* from the labelled regions. All three sets of human labelled regions recovered the canonical motif as the best scoring motif found, while none of the peak callers tested could identify all three. Hocking et al. produced work complementing these conclusions when they generated a dataset of visually inspected regions to test peak caller parameters (Hocking et al., 2016). They showed that while labelling peaks by visual inspection is a seemingly subjective process, the classified regions were highly consistent across different individuals, and demonstrated that changing the labeller did not have a significant effect on their testing results. An even stronger claim was put forward by Oh et al., who showed that regions labelled were consistent between non experts, and even stated that labels produced by non-experts were consistent with their experts (Oh et al., 2020).

To account for any internal biases with my labelling process, I also used two annotated datasets published from an outside lab - H3K27ac ChIP-seq in GM12878 cells and H3K4me3 ChIP-seq in K562 cells. Even combining internal and external sources, the painstaking nature of hand labelling regions means only a fraction of a human genome can be manually annotated at a time, and therefore I sought out other methods of evaluating the peak calls genome wide. For ChIP-seq, the assays' targets offer other means for assessment. A ChIP-seq dataset for CTCF was selected not only for its importance to genome regulation by af-

fecting boundary elements, but also because of its well described binding motif. Motifs bind at TFBSs, however the specificity of a TFBS is variable and generally a perfect sequence match to the motif is not required for binding. Therefore this does not a providing a perfect one-to-one correlation (i.e. an enriched region is a true peak if it overlaps a motif and is a false positive otherwise), but still more motifs would be expected to be found under true peaks, and finding which peaks are over a CTCF motif is bioinformatically feasible for the entire genome. An additional property that makes CTCF a useful target for evaluation purposes is it is often associated with promoters and enhancers (Bastiaan Holwerda & de Laat, 2013), and a similar bioinformatic strategy can be applied for these CREs as with motifs.

Histones are associated with DNA across the genome, and do not have any associated sequence or binding motif that can be searched out. Instead because the histone modifications H3K27ac and H3K4me3 are associated with TSSs, I calculated the percentage of these found under peaks. Due to the frequency with which TSSs are found in the genome, I restricted the analysis to the top 5,000 peaks called for each peak caller and normalised the regions' size to 1 kb. This has the added benefit of being resilient to peak caller parameter changes, as the top peaks are unlikely to change based on parameters. Like motifs this is not a perfect metric, but once again there should be an increased rate of TSSs under true peaks. Following the same logic this was also repeated with the open chromatin datasets for ATAC-seq and DNase-seq, but additionally we used published annotations for open chromatin in the cell type used to further confirm the results.

4.2 Results

LanceOtron's performance was benchmarked against the ENCODE recommended peak caller MACS2. Peak calls were compared from transcription factor ChIP-seq, histone ChIP-seq, and the open chromatin assays ATAC-seq and DNase-

seq - both with and without input control tracks when available (For a numerical listing of performance benchmarks for labelled datasets see Table 4.2).

MACS2 was selected amongst the plethora of available peak callers because it is far and away the most commonly used peak caller currently, garnering over 9,900 citations to date (interestingly I made a note of the current number of citations earlier during my research, 4865, which was a figure I gave during a presentation in September 2018) (Y. Zhang et al., 2008). For LanceOtron, a Peak Score threshold of 0.5 was selected, defining "peaks" as those regions with a 50% probability or greater of being a peak as scored by the deep learning model. While this may seem like a low threshold, in practice LanceOtron has the benefit of scoring the vast majority of regions as either peaks or noise with high probability. For all candidate peaks from the five in-house datasets here, LanceOtron scored only 12% of them a Peak Score of 0.1 to 0.9 (Fig. 4.1). Because of this polarisation, the difference between lax and stringent thresholds is relatively small in terms of number of regions whose classification changes from peak to noise or vice versa, and thus makes LanceOtron somewhat invariant to parameter changes. This is in contrast to MACS2's performance, which has been shown to be highly dependent on parameters selected (Hocking et al., 2016), a trait that could also be observed on the datasets here (Supplementary Figure A.2).

4.2.1 Reanalysis of existing peak call

The most basic module, Score Peaks, assesses previously called peak by taking in a coverage track and a list of peak coordinates. Using this reanalysis capability, we have found that publicly available peak calls, even following the strictest guidelines, may contain large numbers of low-quality peaks. To demonstrate this, LanceOtron was used to analyse peak calls from ENCODE ChIP-seq for H3K27ac from 22Rv1 prostate cancer epithelial cells (ENCSR391NPE). As part of the ENCODE pipeline, two biological replicates were independently peak called and only peaks present in both were included - resulting in 46,030 double

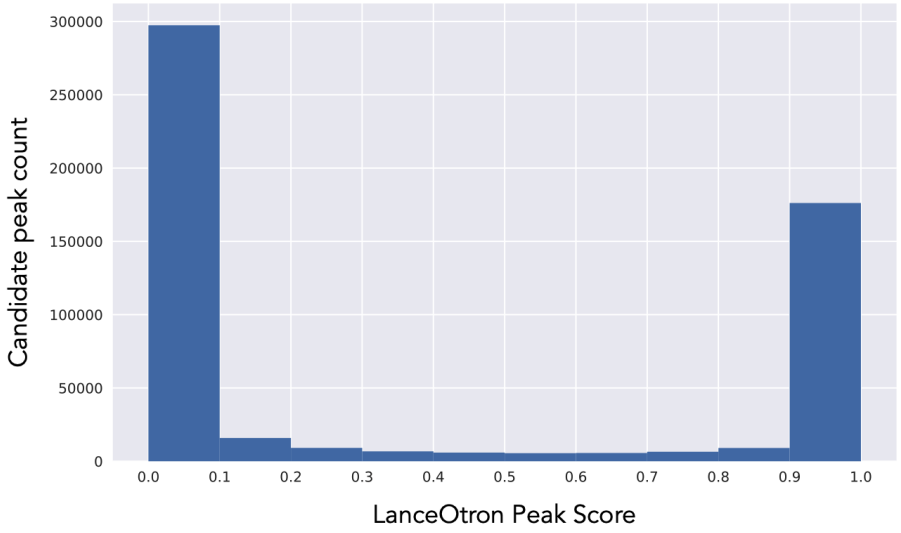


Figure 4.1: Histogram showing the total number of candidate peaks called by LanceOtron and their associated Peak Score over all in-house datasets . LanceOtron's deep learning model tends to score enriched regions being either peak or noise at high probabilities.

called peaks. Of these 18,889 regions (41.0%) had less than a 50% probability of being a real peak using LanceOtron's scoring; 15,162 (32.9%) regions had less than a 10% probability. Using LanceOtron's deep learning based scoring in combination with clustering and visualisation tools, it can be visually confirmed that many very low quality peaks remain in the datasets despite ENCODE requiring independent calls (LanceOtron 22Rv1 H3K27ac project)(Fig. 4.2).

4.2.2 Transcription factor ChIP-seq

Our transcription factor dataset was CTCF in spleen primary cells, downloaded from ENCODE (ENCSR692ILH). 10 Mb of the dataset was hand labelled, resulting in 109 region annotations. When no input control track was used, both LanceOtron and MACS2 achieved perfect sensitivity, detecting all labeled peaks in the dataset, but MACS2 had far lower selectivity and overall F1 score. With input, LanceOtron outperformed MACS2 in precision, recall/sensitivity, selectivity, and F1 score. Comparing across peak call types, LanceOtron without input actually achieved higher scores than MACS2 with input across all metrics (LanceOtron spleen CTCF projects: without input; with input)(Fig. 4.3).

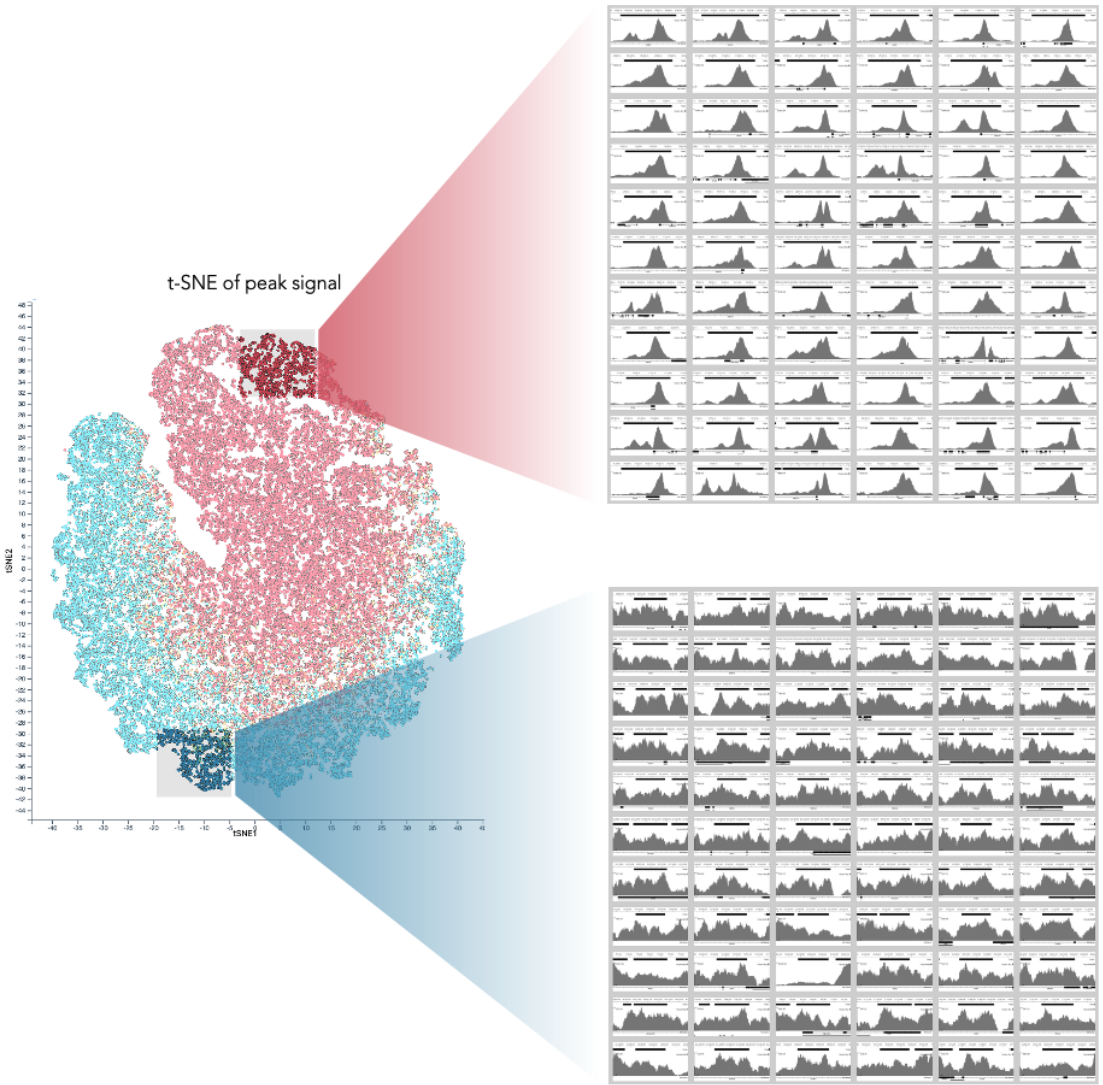


Figure 4.2: Reanalysis of ENCODE peak call using Score Peaks module. Signal from each peak is reduced to 2D using t-SNE and coloured using LanceOtron's Peak Score, with thumbnail images selected from areas with a high Peak Score and a low Peak Score.

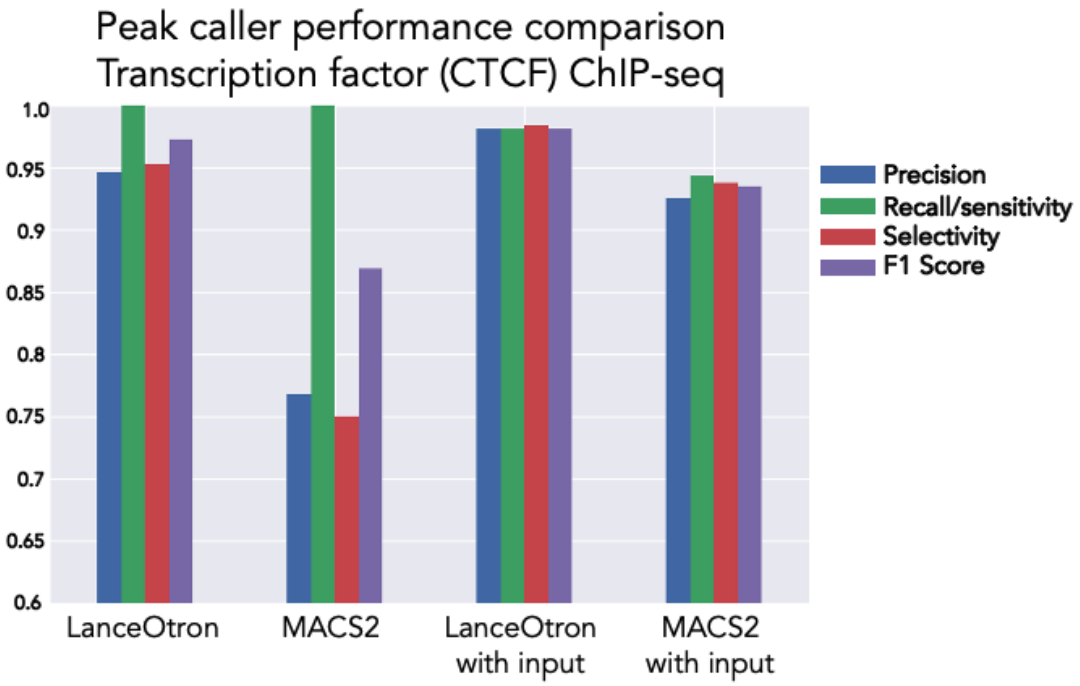
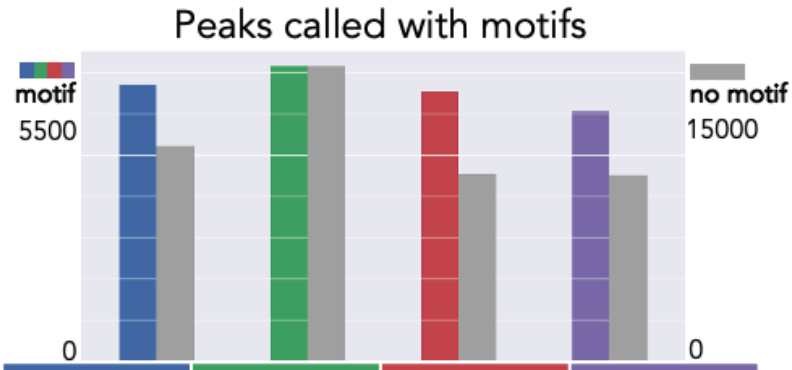


Figure 4.3: Comparison of peak calling for CTCF ChIP-seq

To gain insight into the peak calls genome wide, I performed motif analysis. The number of peaks called were similar between the different methods, though MACS2 without input was slightly higher: LanceOtron, 19,291; MACS2, 24,534; LanceOtron with input, 17,398; MACS2 with input, 16,901. Without input, LanceOtron called fewer peaks with motifs than MACS2 but called fewer peaks in total, resulting in a larger percentage of the overall peak call containing motifs: 32.3% for LanceOtron versus 27.0% for MACS2. When inputs were used, LanceOtron had both a larger count of peaks containing CTCF motifs as well as a larger percentage of the peak call with motifs: 34.9% versus 33.2% (Fig. 4.4).

I further investigated the differences between LanceOtron with input and MACS2 with input peak calls, finding 1,970 LanceOtron only and 1,471 MACS2 only regions. In assessing CTCF for association with promoters and enhancers, I found 87.6% of peaks found exclusively with LanceOtron overlapped with promoters or enhancers compared to just 45.8% of MACS2 only peak calls (Fig. 4.5).

When visualising the top enriched regions called exclusively by each peak caller, LanceOtron's peaks have strikingly more signal than MACS2 (Fig. 4.6 A).



	LanceOtron	MACS2	LanceOtron with input	MACS2 with input
Peaks called	19,291	24,534	17,398	16,901
Peaks with motifs	6,222	6,627	6,066	5,607
% with motif	**32.3%**	**27.0%**	**34.9%**	**33.2%**

Figure 4.4: Comparison of CTCF motifs found for peak calls

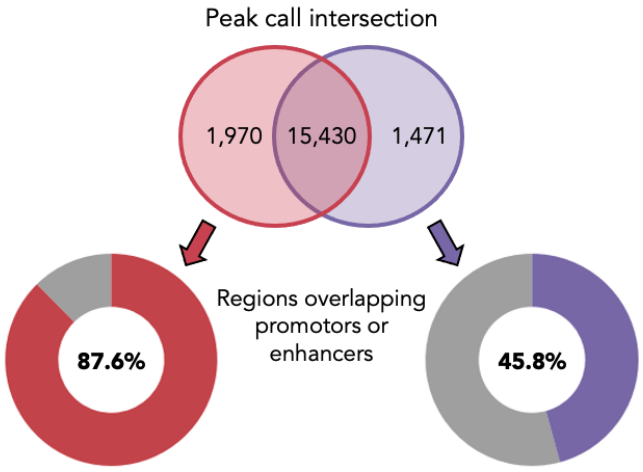


Figure 4.5: Venn diagram of CTCF peak calls from LanceOtron and MACS2. Regions which did not intersect were assessed for overlap with promoters or enhancers.

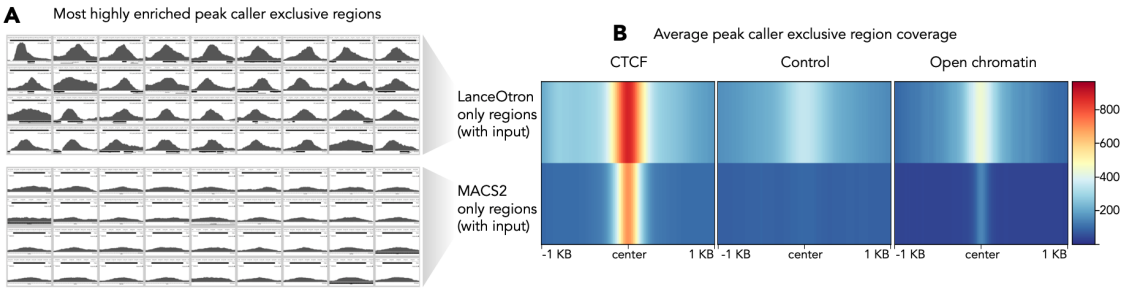


Figure 4.6: A, Thumbnail images from the most highly enriched regions called exclusively by either LanceOtron (top) or MACS2 (bottom). B, Average coverage of the regions called exclusively by either LanceOtron (top) or MACS2 (bottom) for CTCF experimental track, control track, and DNase-seq open chromatin track.

Indeed, this trend holds when inspecting the average signal of the exclusive peak calls; MACS2 only regions were found in regions with less surrounding signal, containing peaks which were narrower and with very low enrichment compared to LanceOtron only peaks (Fig. 4.6 B).

4.2.3 Histone ChIP-seq

Our histone ChIP-seq datasets were H3K27ac in HAP-1 cells (ENCSR131DVD) and H3K4me3 in MG63 cells (ENCSR579SNM). For H3K27ac, the top sensitivity was achieved with three peak calls: LanceOtron, both with and without input, and MACS2 without input. LanceOtron outperformed MACS2 in the remaining metrics of precision, selectivity, and F1 score. The same performance was achieved both with and without input for the LanceOtron peak calls, highlighting the power of its deep neural network (LanceOtron HAP-1 H3K27ac projects: without input; with input)(Fig. 4.7 A). In the H3K4me3 dataset, specificity was equal between LanceOtron and MACS2 with input, and LanceOtron outperformed MACS2 across all peak call types for the remaining metrics (LanceOtron MG63 H3K4me3 projects: without input; with input)(Fig. 4.7 B).

I also tested LanceOtron's performance against published datasets for H3K27ac ChIP-seq in GM12878 cells and H3K4me3 in K562 cells. LanceOtron achieved the highest overall F1-score for both histone datasets, besting CNN-peaks (Oh

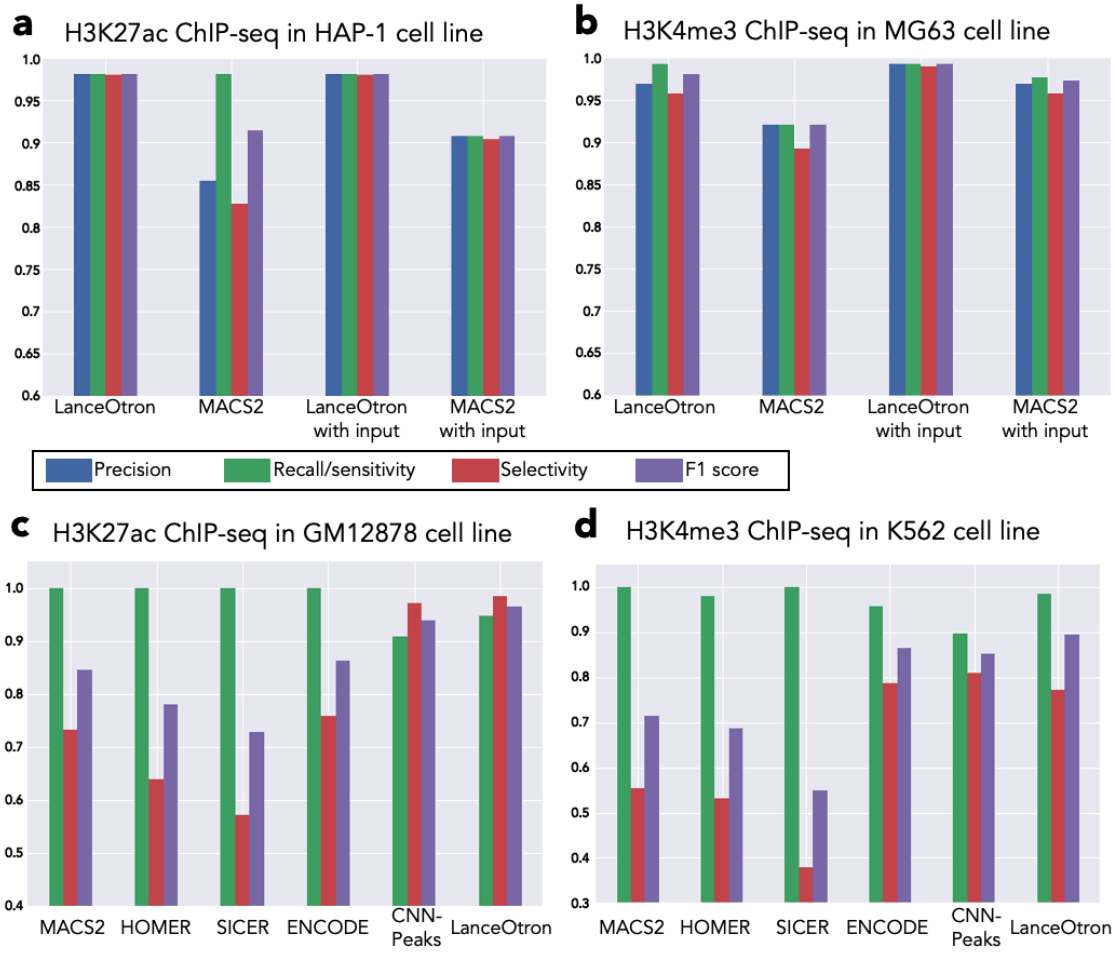


Figure 4.7: Benchmarking LanceOtron against MACS2 for peak calling histone ChIP-seq. A, Model performance metrics using 10 Mb of labelled genomic regions of ENCODE ChIP-seq datasets for H3K27ac in HAP-1 cell line, and B, H3K4me3 in MG63 cell line. C, ChIP-seq dataset labelled by Oh et al. for H3K27ac in GM12878 cell line and D, H3K4me3 in K562 cell line.

et al., 2020), MACS2 (Y. Zhang et al., 2008), HOMER (Heinz et al., 2010), SICER (Zang et al., 2009), and ENCODE's IDR method (Q. Li et al., 2011). These results were consistent with our in-house labeled data, with legacy peak callers performing slightly better on sensitivity, and LanceOtron outperforming on selectivity and F1 score (LanceOtron GM12878 H3K27ac project)(Fig. 4.7 C) and H3K4me3 data (LanceOtron K562 H3K4me3 project)(Fig. 4.7 D).

To investigate the histone mark ChIP-seq peak calls genome wide, I counted the number of transcription start sites (TSSs) overlapping with the returned regions. For this I restricted the analysis to the top 5,000 peaks called for each peak caller and normalised the regions' size to 1 kb. For H3K27ac, LanceOtron

Top enriched peaks intersecting TSSs		
	LanceOtron	**MACS2**
% top H3K27ac ChIP-seq in HAP-1 peaks overlapping TSSs (count)	**56.1%** (2,806 / 5,000)	**48.6%** (2,428 / 5,000)
% top H3K4me3 ChIP-seq in MG63 peaks overlapping TSSs (count)	**69.4%** (3,472 / 5,000)	**66.4%** (3,318 / 5,000)
% top ATAC-seq in MCF-7 peaks overlapping TSSs (count)	**44.4%** (2,218 / 5,000)	**21.9%** (1,096 / 5,000)
% top DNase-seq in A549 peaks overlapping TSSs (count)	**43.3%** (2,164 / 5,000)	**22.7%** (1,133 / 5,000)
	LanceOtron with input	**MACS2 with input**
% top H3K27ac ChIP-seq in HAP-1 peaks overlapping TSSs (count)	**56.2%** (2,812 / 5,000)	**52.1%** (2,607 / 5,000)
% top H3K4me3 ChIP-seq in MG63 peaks overlapping TSSs (count)	**70.0%** (3,501 / 5,000)	**49.8%** (2,491 / 5,000)
GM12878 open chromatin peaks intersecting active regions		
	LanceOtron	**MACS2**
% ATAC-seq peaks in active regions (count)	**12.6%** (628 / 5,000)	**7.3%** (366 / 5,000)
% DNase-seq peaks in active regions (count)	**12.1%** (607 / 5,000)	**9.5%** (476 / 5,000)

Table 4.1: LanceOtron and MACS2 peak call comparison for transcription start sites (TSSs), and for active regions in open chromatin. Percentages and counts of peaks intersecting TSSs or active areas of the genome are given for 5,000 regions of LanceOtron and MACS2 peak calls, selected for being most enriched (highest Peak Score or q-value for LanceOtron and MACS2 respectively).

performance was very similar with and without input, increasing from 2,806 to 2,812 peaks when the input track was included. Both LanceOtron peak calls had more overlap with TSSs than MACS2, which had 2,367 and 2,591 with input. I observed similar results for the H3K4me3 data, with LanceOtron finding 3,472 peaks intersecting TSSs, increasing slightly to 3,501 with input control. MACS2 had better performance without input, though not reaching LanceOtron levels, at 3,335 and decreasing down to 2,589 with input (Table 4.1).

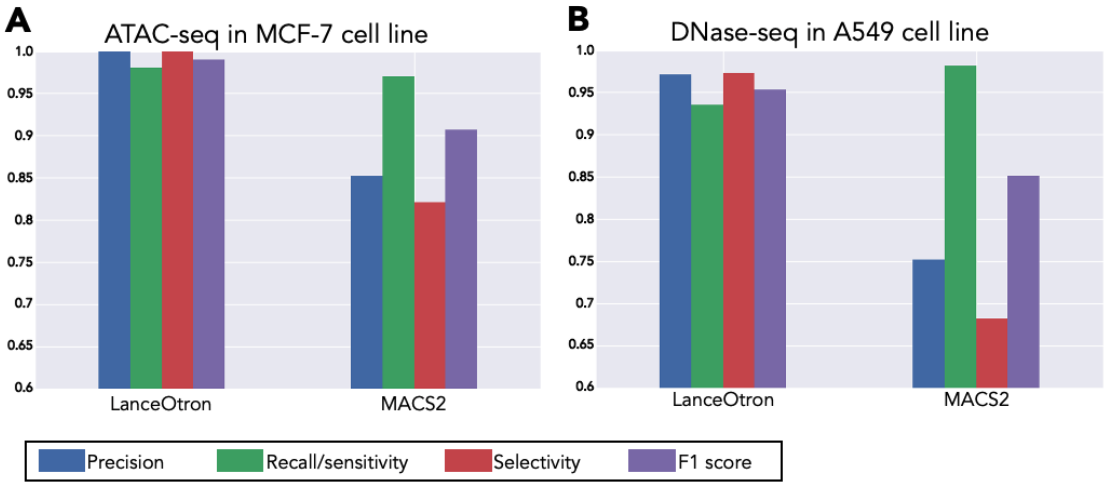


Figure 4.8: Benchmarking LanceOtron against MACS2 for calling open chromatin. A, Model performance metrics using labelled genomic regions of an ENCODE ATAC-seq dataset in MCF-7 cell line and B, DNase-seq in A549 cell line.

4.2.4 ATAC-seq and DNase-seq

In-house labelling for ATAC-seq consisted of 196 regions in the MCF-7 cell line from ENCODE (ENCSR422SUG). LanceOtron outperformed MACS2 across all metrics (LanceOtron MCF-7 ATAC-seq project)(Fig. 4.8 A). Results were similar for our in-house DNase-seq data, consisting of 224 labelled regions in the A549 cell line from ENCODE (ENCSR000ELW). MACS2 outperformed LanceOtron for recall/sensitivity but had a very high false positive rate. Consequently, LanceOtron outperformed MACS2 on precision, sensitivity, and F1 score (LanceOtron A549 DNase-seq project)(Fig. 4.8 B).

As with the histone datasets, I also intersected the top 5,000 peaks, normalised to 1 kb, with TSSs. LanceOtron's top peaks had nearly double the number of intersections with TSSs compared with MACS2 for DNase-seq (2,164 versus 1,151) and over double for ATAC-seq peaks (2,218 versus 1,086) (Table 4.1).

I also compared peak calling performance on GM12878 cells for ATAC-seq (ENCFF576DMC) and DNase-seq (ENCSR000EMT). Here I used published annotations from Tarbell and Liu (2019), whereby they defined active areas of the genome using enhancer and promoter data with the software GenoSTAN (Zacher et al., 2017). Following the same procedure as the TSS analysis, the top 5,000

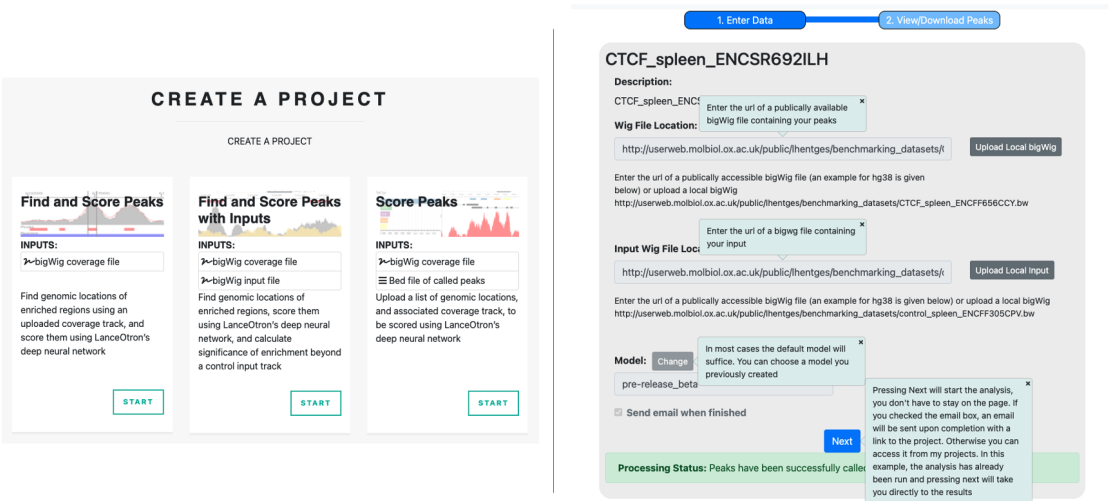


Figure 4.9: Screenshots taken from the LanceOtron webtool. Three module options are given depending on analysis needed (left). Required data fields and coverage track upload (tutorial page shown, right).

peaks were selected from both LanceOtron and MACS2, normalised for size, and intersected with the published annotations. Once again, LanceOtron's top peaks were found more often in active regions for both ATAC-seq (12.6% vs 7.3%) and DNase-seq (12.1% vs 9.5%) datasets (LanceOtron GM12878 projects: ATAC-seq; DNase-seq) (Table 4.1).

4.3 Discussion

Technical features, described in detail in previous chapters, as well as the separate modules for different use cases constituted a large focus of the peak caller development, but built into LanceOtron are several other software features that improve its function. Crucially, the type of input file I chose to use has a large effect on the possible utilities. LanceOtron extracts genomic data from a bigwig track, which has the benefits of being both compact and readily visualised. With widely used peak callers (including MACS2), assessing the quality of results cannot be done directly, rather the user must upload their output to a genome browser. This is somewhat restrictive for judging the quality of a peak call, in that

CTCF ChIP-seq in spleen				
	LanceOtron	**MACS2**	**LanceOtron w/ input**	**MACS2 w/ input**
Precision	0.946	0.768	**0.981**	0.926
Recall/sensitivity	**1**	**1**	0.981	0.943
Selectivity	0.953	0.75	**0.984**	0.938
F1 score	0.972	0.869	**0.981**	0.935
H3K27ac ChIP-seq in HAP-1				
	LanceOtron	**MACS2**	**LanceOtron w/ input**	**MACS2 w/ input**
Precision	**0.981**	0.854	**0.981**	0.907
Recall/sensitivity	**0.981**	**0.981**	**0.981**	0.907
Selectivity	**0.981**	0.827	**0.981**	0.904
F1 score	**0.981**	0.914	**0.981**	0.907
H3K27ac ChIP-seq in GM12878				
	LanceOtron	MACS2		
Precision	**0.937**	0.862		
Recall/sensitivity	0.961	**0.974**		
Selectivity	**0.932**	0.836		
F1 score	**0.949**	0.915		
H3K4me3 ChIP-seq in MG63				
	LanceOtron	**MACS2**	**LanceOtron w/ input**	**MACS2 w/ input**
Precision	0.969	0.92	**0.992**	0.968
Recall/sensitivity	**0.992**	0.92	**0.992**	0.976
Selectivity	0.957	0.892	**0.989**	0.957
F1 score	0.98	0.92	**0.992**	0.972
H3K4me3 ChIP-seq in K562				
	LanceOtron	MACS2		
Precision	**0.821**	0.753		
Recall/sensitivity	0.986	**1**		
Selectivity	**0.773**	0.652		
F1 score	**0.896**	0.859		
ATAC-seq in MCF-7				
	LanceOtron	**MACS2**		
Precision	**1**	0.852		
Recall/sensitivity	**0.98**	0.97		
Selectivity	**1**	0.821		
F1 score	**0.99**	0.907		
DNase-seq in A549				
	LanceOtron	**MACS2**		
Precision	**0.971**	0.752		
Recall/sensitivity	0.935	**0.981**		
Selectivity	**0.973**	0.681		
F1 score	**0.953**	0.851		

Table 4.2: Performance benchmarking comparing LanceOtron with MACS2.

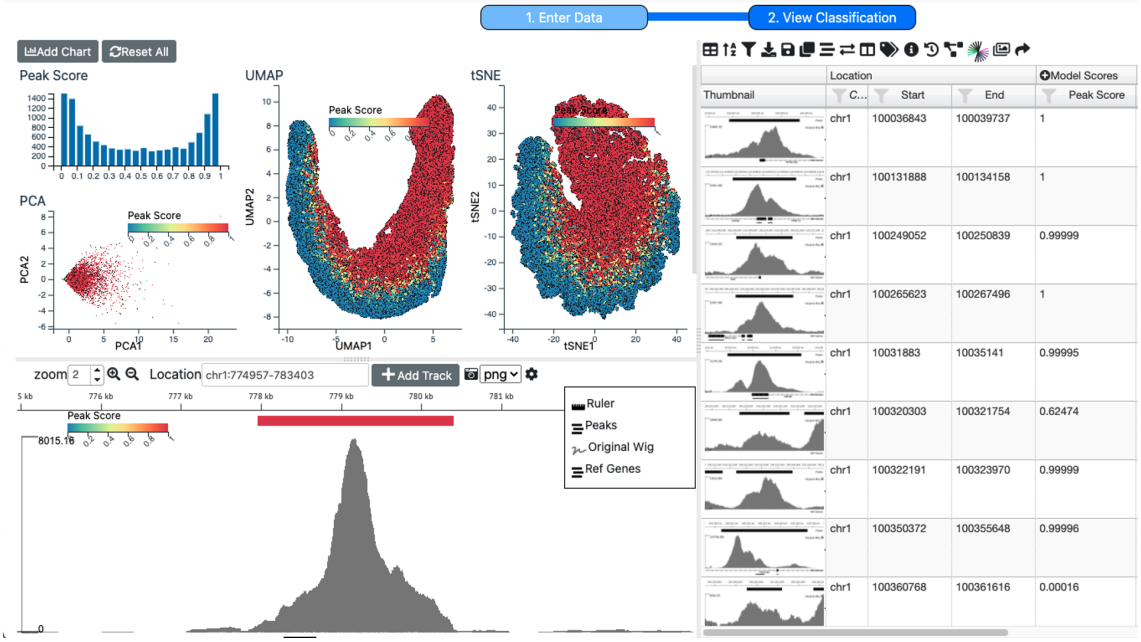


Figure 4.10: Screenshots taken from the LanceOtron webtool. Completed analysis showing interactive charts, built in genome browser, and quick navigate bed file with thumbnail previews.

the output file and genome browser are disconnected, meaning users are limited to haphazardly scanning some genomic regions to see if their results are sensible. To address this LanceOtron is built on the powerful MLV genome visualization software (Sergeant et al., 2021), which allows users to sort and filter results, as well as visualising peaks and their metadata en masse. Additionally, clustering peaks based on shape and quality is built-in via the unsupervised machine learning techniques PCA (Jollife & Cadima, 2016), t-SNE (van der Maaten & Hinton, 2008), and UMAP (McInnes et al., 2018). This allows for rapid assessment of data quality, structure, and the appropriateness of the output of the algorithm for the current dataset (Supplementary video 1 and 2). The combination of visualisation plus dimensionality reduction works well with LanceOtron's overcomplete candidate peak calling strategy, allowing users to explore and filter a comprehensive listing of enriched regions. This is also extremely useful for the Score Peaks module, since the quality of peak calls made outside of LanceOtron become immediately obvious. As in the assessment of the ENCODE 22rv1 H3K27ac ChIP-seq, large amounts of similarly low-quality peaks can be identified in many other

public data sets based on similar statistical peak calling approaches.

The combination of the gold standard, hand labelled dataset plus the genome wide features, informed by the biology of the tracks, performed across five datasets provided a comprehensive set of tests to judge real life performance. Over the course of these tests, LanceOtron managed top marks in nearly every category with very consistent performance across datasets. MACS2 however, had large variations when evaluated. Though outperformed, MACS2 actually achieved its second-best overall performance score on the CTCF dataset (F1 score, first best on H3K4me3). This is perhaps not surprising as MACS2 was designed for peak calling transcription factor binding experiments. While the narrow binding pattern of H3K4me3 closely resembles the distribution of read coverage seen in transcription factor binding, MACS2 performance suffered when the genomic signal deviated from this pattern, especially true in ATAC-seq and DNase-seq. When no input track was available, MACS2 overall performance further declined. This is in contrast to LanceOtron, whereby performance was only slightly lower without an input track, and outperformed MACS2 with input for F1 score on every test where this comparison was available.

Though not perfect metrics, motif analysis for CTCF ChIP-seq and TSS analysis for the histone ChIP-seq and open chromatin datasets provided additional support to the results seen from the hand labelled datasets on a larger scale. Examining the differences between the peaks called exclusively by both LanceOtron and MACS2 also provided insight into their genome wide performance. It seems that the MACS2 exclusive regions are a sporadic sampling of the numerous peaks close to noise found throughout the genome, however the peaks that MACS2 missed are relatively strongly enriched. These missing peaks are excluded by MACS2 because of the increase in control signal, however some increased signal from the control track is expected when the region is found in areas of open chromatin (Auerbach et al., 2009), which can be seen associated with the LanceOtron only peaks. Using the outcome of a statistical test as the sole criteria of categoris-

ing genomic regions means striking a balance between calling false positives and false negatives. While the stringent cut-off set by MACS2 helps reduce false positives in some areas of the genome, here it does so at the cost of false negatives. Because LanceOtron additionally uses the shape of the peak, the statistical threshold can be relaxed, thus preventing these false negatives without trading them for a plethora of false positives.

During this process much time and effort was put into developing the use cases, functionality, and user experience for LanceOtron. Through collaboration I worked with an extremely talented JavaScript developer to bring transform these ideas into a webtool. The vision was for a tool actually useful for biologists. Deep learning, probabilities, and statistics can be difficult to interpret on their own, but with visualisation these concepts can be made accessible. This is crucial because there is no hard cutoff that can be applied universally, and the best that can be done is offer all of the information possible in a way that is fully transparent and easily interpreted, and let scientists make the best choice they can for their given situation.

CHAPTER

FIVE

METHODS

5.1 Data and Candidate Peaks - Chapter 2 methods

5.1.1 Data acquisition

The data used to train the neural network was obtained from ENCODE. To generate a complete list of experiments which met quality specifications ENCODE's REST API (Davis et al., 2018) was used to query and filter their database (scripts and outputs available on GitHub). The filters used were as follows: "released" status at the time of search inquiry; aligned to human reference genome hg38 as BAM files; the availability of a corresponding control track (for H3K27ac, H3K4me3, and transcription factor ChIP-seq experiments); metadata indicating sequencing type (single-end or paired-end).

ENCODE's database is organised using experiments, which may themselves include multiple individual ChIP-seq or open chromatin experiments. This means experiments matching the filters may not actually include individual ATAC, ChIP-seq, or DNase-seq files meeting each of the criteria. To accommodate this, the data extraction script used a four stage process: (1) search ENCODE for list of experiments meeting criteria, (2) extract experiment metadata, and filter for presence of controls (ChIP-seq only), (3) iterate through files of experiment, filter, and

categorise into paired-end or single-end sequenced, (4) write accession numbers of specific experiments to file.

For ATAC-seq, DNase-seq, H3K27ac ChIP-seq, and TF ChIP-seq, 10 accession numbers were chosen at random for each experiment (50 total) to use for labelling. Paired end sequenced datasets were prioritised over single end sequenced, and for ATAC-seq, DNase-seq, H3K27ac ChIP-seq, and TF ChIP-seq all randomly selected datasets were paired end. For H3K4me3 ChIP-seq however, only six paired end datasets were available. All of the paired end H3K4me3 ChIP-seq datasets were used, and four single end datasets were selected at random.

Each BAM file was downloaded directly from ENCODE, along with the corresponding control BAMs for H3K27ac, H3K4me3, and transcription factor ChIP-seq experiments. If multiple replicates of the control experiments existed, only the first listed in ENCODE's database was used for analysis. BAM files were sorted and indexed using Samtools (H. Li et al., 2009) 1.3 (samtools sort filename.bam and samtools index filename.bam.sorted commands respectively).

Bigwig file coverage maps were created from the BAM files using deepTools (Ramírez et al., 2014) version 3.0.1 commands: bamCoverage –bam filename.bam.sorted -o filename.bw –extendReads -bs 1 –normalizeUsing RPKM for paired-end sequenced experiments. For single-end sequenced experiments the average fragment length was obtained from ENCODE and used with the –extendReads flag, making the command: bamCoverage –bam filename.bam.sorted -o filename.bw –extendReads averageFragmentLength -bs 1 –normalizeUsing RPKM.

5.1.2 Hand-labelling peaks

Genomic regions for labelling were selected by using enrichment detection algorithms. Putative peak calls were carried out on all datasets, followed by classification as either peak or noise based on visual inspection. For each processed bigwig a dictionary of suitable chromosomes was made (excluding mitochondria,

Epstein-Barr virus or alternative mappings) and candidate peaks were found. An enrichment map, composed of an array of the sequencing coverage, was made and then smoothed by applying a rolling mean of a small window. Two preliminary background enrichment maps were made in a similar way, but using larger windows for the rolling mean: 5 kb and 10 kb. The final background enrichment map found the maximum value - either from the two preliminary maps or a scalar multiplied by the mean signal of the chromosome - at every coordinate. This process was repeated using different values for the enrichment map smoothing windows: 100 bp, 200 bp, 400 bp, 800 bp, 1600 bp, and scalars used for finding maximum value in the background map: 1, 2, 4, 8, 16. The coordinates where the enrichment exceeded background for 50 contiguous base pairs or more were recorded as local area enrichment candidate peaks. Additionally, chromosome mean enrichment candidate peaks were recorded, but only used the background enrichment as chromosome mean multiplied by the scalar, again repeating with the same smoothing windows and scalars as the local area enrichments. Finally the following command was used for the MACS2 peak caller: macs2 callpeak -t filename.bam.sorted -c controlFilename.bam.sorted -n sampleLabel -f BAM -g hs -B -q 0.01 for H3K27ac, H3K4me3, and transcription factor ChIP-seq datasets. For ATAC-seq and DNase-seq, which lack control tracks, the following command was used: macs2 callpeak -t filename.bam.sorted -n sampleLabel -f BAM -g hs -B -q 0.01.

From each dataset a 1 Mb continuous region was selected at random for each chromosome. If the start of the randomly selected region was near the end of the chromosome, the area considered was from that point to the chromosome end, then from the chromosome start extending out until a full 1 Mb was covered. Candidate peaks called from all 3 methods starting within the random region were made available for labelling. For both the local area enrichment and chromosome mean enrichment methods, it was possible for candidate peaks to overlap. A python implementation of BEDTools (Quinlan & Hall, 2010) (pybedtools) was used

to find overlapping peaks, and only one selected at random was considered for visual inspection.

Visual inspection was carried out using MLV (Sergeant et al., 2021), with control tracks overlaid when available. Only candidate regions which were clearly peaks (consistent enrichment which smoothly increases and decreases around a summit) or noise (sporadic or sometimes "blocky" enrichment) were labelled as such. Entire 1 Mb regions were assessed (no early stopping), with the order of chromosomes randomised. Chromosomes were inspected one at a time, until at least 100 verified peaks were found for the dataset or all of the chromosomes were assessed.

Additional labels were generated using the following algorithm. First the raw signal was smoothed using the rolling mean for the surrounding 400 bp, and any coordinate where the signal was at least 4-fold*mean chromosome signal was marked as enriched. If the size of an enriched region was between 50 bp and 2 kb it was considered a candidate peak. Regions smaller than 50 bp were discarded, and regions above 2 kb were recursively re-evaluated at a 1-fold higher threshold until the region size was between 50 bp and 2 kb, or the region was greater than 20-fold enriched, at which point the 20-fold enrichment start and end coordinates were recorded. If these candidate peaks intersected with the previously labelled peaks, these regions were then automatically labelled peaks, resulting in an additional 3,447 labels for a total of 8,463 peaks (ATAC-seq: 1,926; DNase-seq: 2,097; H3K27ac ChIP-seq: 1,651; H3K4me3 ChIP-seq: 1,806; transcription factor ChIP-seq: 983).

Noise regions were down sampled with prioritisation given to regions with the highest signal. The maximum number of sequenced reads within the regions, i.e. the region heights, were unevenly distributed (the majority of which were in the 0th percentile), and caused the imputed percentiles to break from the number of regions present. Here all noise regions with a max height in the 25th percentile or greater were used (3,658), and equal numbers below the 25th percentile were

randomly sampled.

These regions were used as training data for the wide and deep model (see below), then scored by the model to identify outliers. The model misclassifications were reassessed, and any identified falsely labelled samples were rectified.

5.1.3 Candidate peak comparison

An algorithm for identifying enriched regions using a Savitsky-Golay (SG) filter was implemented in Python utilising the SciPy package (Virtanen et al., 2020). The SG function was used to quickly calculate the second derivative of a n-degree polynomial (n=2 for all work here) fitted to the signal of the coverage track smoothed over a fixed window size. Because the sites where the second derivative equals zero indicate where the slope of the signal changes from increasing or decreasing, all of these coordinates were collected. Sites where the slope changed to increasing (i.e. changed to positive slope) were used as the starting coordinate of an identified enrichment region and its matched decreasing slope location as the ending coordinate. Chromosome edge cases where a coverage track started or ended in an area of enrichment, and therefore lacked a starting or ending coordinate, were accounted for by inserting the starting or ending coordinate as the nearest end of the chromosome.

Optimal window size, that is finding the number of surrounding bases to average over such that noise is reduced without signal distortion, was calculated based on the work of Vivó-Truyols and Schoenmakers, 2006. This process involves first finding the lag-1 autocorrelation coefficient of unenriched regions. Unenriched regions were defined as regions which were not greater than 2-fold enriched above the chromosome mean signal after applying a 400 bp smoothing window. Next, the SG filter was applied at different window sizes: 51 bp, 101 bp, 201 bp, 401 bp, and 801 bp. The lag-1 autocorrelation coefficient was then calculated for the smoothed signal and compared to the noise coefficient, and the window size closest to this was selected.

The enrichment consensus candidate peak calling algorithm was designed to optimise resources and also centre the area around the highest signal to improve performance with deep learning models. For each chromosome an enrichment map is built using sequencing coverage smoothed by applying a rolling mean. An array of zeros the same length as the chromosome is initialised, and the smoothed signal is compared to the mean chromosome signal multiplied by a scalar. If the signal from the coverage map is greater than the scaled mean chromosome signal, then one is added to the corresponding coordinate of the initialised array. This process is repeated with five different windows for smoothing the coverage map: 100 bp, 200 bp, 400 bp, 800 bp, and 1600 bp, and five different scalars multiplied to the chromosome mean: 2, 4, 8, 16, and 32, adding one to the corresponding coordinate for each of the 25 permutations. A final enrichment thresholding is carried out, recording coordinates on the array at or above a count of five for 50 or more contiguous base pairs. Regions enriched over 2 kb are reevaluated at a higher threshold, increasing by one, and repeated until all internal regions found are all 2 kb or less or the highest enrichment level (25) is achieved.

To compare the candidate peak calls found using the enrichment consensus method with MACS2's peak call of the same data, BEDTools function intersect was used with default parameters.

5.1.4 P-value comparison

P-values were calculated using the Poisson distribution, whereby signal at a region of interest N was compared to background signal λ:

$$P = 1 - pPois(N, \lambda) \tag{5.1}$$

where $pPois$ is the Poisson cumulative distribution function:

$$\sum_{i=0}^{N} \frac{e^{-\lambda}\lambda^{i}}{i!} \tag{5.2}$$

Different λ values from the average signal from the sequencing coverage track were compared, ranging from 10 kb surrounding the coordinate N was located, up to 400 kb in 10 kb intervals; λ equal to the entire chromosome mean was also included. For these comparisons, N was determined by finding the maximum height of the region assessed.

The package Scikit-learn was used for the p-value comparison work (Pedregosa et al., 2011). Receiver operator characteristic (ROC) curves as well as the area under said curves were calculated using Scikit, as well as generating two dimensional PCA and t-SNE plots (for creating the UMAP plots the UMAP-learn package was used (McInnes et al., 2018)); the logistic regression model was also created using this package.

The logistic regression model was trained on the hand labelled peaks and enriched noise regions described above. Output from this model was a probability, and the cutoff for the peak class was 0.5 or above. P-values were once again calculated from equations 5.1 and 5.2. Numerous versions of the logistic regression model were built for comparison. The range of region sizes used to calculate the average signal used as λ (up to 50 kb, 100 kb, 200 kb, and 400 kb) was varied, as was the number of p-values calculated in the range (10, 20, or 40). For this the λ values were calculated from equally spaced divisions within the size range. For example to calculate 10 p-values with a background region size range of 100 kb, λ would be calculated as the average signal from 10 kb, 20 kb, 30 kb, 40 kb, 50 kb, 60 kb, 70 kb, 80 kb, 90 kb, and 100 kb - these would then be used to calculate the p-values, which would go on to be used as the inputs of the logistic regression model, which would return the probability of the region belonging to the peak class.

Different ways of measuring N were also used - this was measured either as the maximum height of the region, the average height, or maximum height after

a 400 bp smoothing window was applied. This resulted in 36 different logistic regression models, which were then compared by measuring the accuracy of the predictions on the testing data.

5.2 Deep Learning - Chapter 3 methods

5.2.1 CNN model

The CNN model was built using Keras (Chollet et al., 2015) with TensorFlow (Abadi et al., 2015) backend. For each region the model classifies, three channels of 2000 input signal (3x2000 matrix) are extracted from its corresponding normalised bigwig track. Normalisation is defined here as chromosome mean equal to zero, with a standard deviation equal to one. The first channel is the region compressed or expanded into 2000 bins. The second channel is 2000 bp of signal, determined by finding the centre coordinate of the region and extracting 999 bp upstream and 1000 bp downstream. The final channel works like a combination of the two, first finding the centre coordinate, extracting 9999 bp to the left and 10 kb to the right, then compressing this into 2000 bins. These matrices were used as the input to a Keras sequential model.

The first layer of the model is a one dimensional convolutional layer with 32 filters, kernel size two with a stride length of one. All convolutional layers in this model used the ReLU activation function. Convolution is followed by a batch normalisation layer, with this convolution-batch normalisation repeated (with the same hyperparameters) for a total of five times. This was followed by a size two max pooling layer and a 25% dropout layer. This fed into another convolutional layer with 128 filters, filter size 32, stride length one. Batch normalisation, pooling, and dropout were repeated as before, except the dropout rate was increased to 50%. Two fully connected layers followed, the first with 1000 outputs and a ReLU activation function, the second with six outputs (one for each class) and a softmax activation function. The optimiser used was Adam (Kingma & Ba, 2017)

with default hyperparamters, at a learning rate of 0.00001 using the loss function categorical cross entropy.

ROC, AUC, confusion matrices were generated using Scikit learn.

5.2.2 Chromatin capture conformation interactions

Sequencing data for the chromatin capture conformation assay Micro-Capture-C were aligned with Bowtie2 (Langmead et al., 2009) and prepared as described in Hua et al., 2021.

Peak calls were made using three different methods: the CNN model described above, a Poisson model, and MACS2. The CNN as well as the Poisson model were presented with candidate peaks, found using a basic smoothing window, 400 bp, with enrichment thresholding (4-fold chromosome mean).

The Poisson model used the maximum value of the enriched regions as N, and used background signal from the associated input control track for λ. The average signal of the surrounding 1 kb, 5 kb, 10 kb, and input chromosome from the centre coordinate was calculated from the control track and the highest value was used as background. This meant that the cumulative distribution function remained the same (equation 5.2), but the p-value calculation was modified slightly:

$$P = 1 - pPois(N, \lambda = max[\lambda_{1kb}, \lambda_{5kb}, \lambda_{10kb}, \lambda_{chrom}]) \tag{5.3}$$

Because there were multiple samples, significance scores could be combined. For the CNN this meant combining probabilities of class predictions, and was carried out using Bayesian updating. The initial prior was set at 0.5. A function was built to loop through the associated classification probabilities. If this probability was greater than or equal to 0.5, the prior of the region belonging to the peak class was updated by multiplying it by the model's accuracy for predicted peaks. The prior of the region belonging to the noise class was also updated, by multiplying it by (1 - noise accuracy). These were reversed in the case where the

classification probability was less than 0.5 (i.e the peak prior was 1 - peak accuracy and the noise prior was multiplied by noise accuracy). The posterior was calculated for the region belonging to the peak class by dividing the peak prior by the peak prior added to the noise prior.

For combining p-values (P) for the Poisson model and MACS2, Fisher's combined probability test (Mosteller & Fisher, 1948) was used.

$$-2\sum_{i=1}^{n} ln(P_i) \tag{5.4}$$

5.2.3 Autoencoder

The autoencoder model resembled the CNN model described above. Here convolutional layers, size two with 32 filters alternated with pooling layers, forming a convolutional block. Five convolutional blocks were used, with pooling size two except for the fifth layer, which had a pooling size of five. This represented the bottleneck point, the output of which was 25 x 32. Starting with a convolutional layer the structure was mirrored from here (i.e. five more convolutional blocks), however instead of max pooling, a one dimensional upsampling layer was used instead (size five for the first instance and size two for the remaining four). The optimiser used was adadelta (Zeiler, 2012) and loss was binary crossentropy.

A Scikit implementation of t-SNE was once again used as was their Bayesian Gaussian mixture model for clustering.

5.2.4 Wide and deep model

The wide and deep architecture (Cheng et al., 2016) used combines enrichment values, logistic regression, and a CNN. The logistic regression model takes as inputs the enrichment values, while the CNN uses the 2 kb of signal centred on the region of interest. The outputs of these two models, along with 11 enrichment values, are input into a multilayer perceptron, which outputs a peak quality metric

(called Peak Score) with values ranging from 0 to 1.

The 11 enrichment values consisted of Poisson-based p-values (equations 5.1 and 5.2), using region maximum height as N and λ values calculated from the average signal of 10kb to 100kb regions in 10kb increments as well as whole chromosome average signal.

Training data used were the hand labelled regions described above. The logistic regression model was trained separately (using the same data), and all coefficients and model parameters saved. The wide and deep model was then trained with the logistic regression component locked, using a loss distributed 70:30 to wide-and-deep-output:CNN-only-output.

To determine the optimal structure and hyperparameters, a brute force method of building many models with different configurations was carried out. Model performance was assessed by measuring the number of correctly predicted classifications of enriched regions from data unseen to the model. This was done in two stages using the RandomSearch function from the python package Keras Tuner (O'Malley et al., 2019). First, 5000 models were built varying the following components: the first convolutional layer's filter number (50-200) and filter size (2-10), the number of convolutional blocks (1-5), number of convolutional layers per block (1-5), convolutional filters used per hidden convolutional layer (20-100) and their size (2-10), pooling type (max, average, or none), dropout rate (0.0-0.5), first dense layer size (10-100), number of additional dense layers (0-3), additional dense layer size (10-100), and learning rate (0.0001-0.01). Convolutional blocks were composed of first convolutional layers, then batch normalisation, followed by an optional pooling layer. Performance was robust across a range of configurations. This included models with less complexity, which required less time and computational resources. Given this several architectural choices were made: fixing the number of convolutional blocks to four, using one convolutional layer per block, and using max pooling; the model used two additional dense layers. The search process was repeated, building another 500 models which explored the

following range of component variables: the first convolutional layer's filter number (50-150) and filter size (2-10), convolutional filters used per hidden convolutional layer (50-200) and their size (2-10), dropout rate (0.0-0.5), first dense layer size (10-100), additional dense layers size (10-100), and learning rate (0.0001-0.01). The top 10 performing models were then subjected to 5-fold cross validation, and the architecture from the top performer was used.

5.2.5 Enhancing ChIPmentation signal

Coverage maps from ChIPmentation assays were transformed with the purpose of more closely resembling the related (and often considered to be the higher resolution) experiment ChIP-seq. For this, an encoder-decoder model was built and simultaneously trained with a discriminator. The encoder used 2000 bp of signal centred around enriched ChIPmenation regions, with the decoder target being coordinate matched regions of ChIP-seq - both assays were used for determining the histone H3K4me3 binding locations. These two experiments were done internally by an associated lab group in mice, aligned using Bowtie to the mm9 genome, and the coverage map was made using the deepTools bamCoverage function.

The encoder was composed of three convolutional layers and three pooling layers, starting with convolution and alternating between layer types. The convolutional layers all had kernel size three, stride one, and SELU activation functions (Klambauer et al., 2017). ReLU activation functions generally offer faster model training times with SELU having slightly better performance. Because of the proof of principle nature of this model and the difficulty often associated with training GANs, I opted to trade speed for potential performance. The number of filters for the first, second, and third convolutional layers were 16, 32, and 64 respectively. The type of pooling was maximum value for all layers, and the pooling size was five for the first, two for the second, and two for the third layer. Similar to the autoencoder described above, the second half of the model, the decoder, mirrored

the first half with the exception that upsampling layers replaced pooling layers.

The discriminator followed a typical CNN structure, with 8 convolutional layers, kernel size three, stride size one, and SELU activation. Filter sizes were as follows: 64, 64, 128, 128, 256, 256, 512, 512. Each convolutional layer was separated by batch normalisation. The convolutional layers were fed into a dense layer with 1024 outputs and SELU activation, followed by one last dense layer with final, singular output using sigmoid activation.

Keras was used by first compiling encoder-decoder and discriminator as separate sequential models, then combining these two models into a single sequential model. There were no built in training methods for GANs as part of Keras at the time this model was built, and so a custom protocol was used. Here, the number of epochs and batch size were taken as variables, and for each batch the encoder-decoder would transform the ChIPmentation signal. The discriminator would then take this, as well as the coordinate matched ChIP-seq signal, and train its parameters. Next these weights of the discriminator were locked (specifically the Keras argument trainable=False), and only the ChIPmentation samples were fed through the entire model, but with the target labels as ChIP-seq. This forced the generator to adjust its weights such that its outputs would more closely match a signal classified as ChIP-seq.

5.3 LanceOtron - Chapter 4 methods

LanceOtron used the consensus enrichment candidate peak calling algorithm combined with the wide and deep neural network; both of these components are described above. Besides this, an additional calculation was performed when LanceOtron used an input control track. Here the Poisson based p-value was calculated, using the maximum number of sequenced reads as N and λ taken as either the average signal of the coordinate matched region of the control track, or the average signal of the coordinate matched region of the control track plus an

additional 1 kb - whichever was more stringent

$$P = 1 - pPois(N, \lambda = max[\lambda_{region}, \lambda_{\pm 1kb}]) \tag{5.5}$$

with the cumulative distribution function equation 5.2

A region was defined as a peak by LanceOtron if the overall Peak Score was 0.5 or greater, and in the cases of an input track being used, if the p-value calculated from input was greater than 10 (-log).

5.3.1 Test data and model performance

Testing datasets were also obtained from ENCODE but were not used in LanceOtron's training data (Supplementary Table 1). Each track was downloaded as a BAM file and converted to bigwig using the same deepTools commands given above for in training data preparation. Chromosomes were shuffled (mitochondrial and alternative mapping chromosomes were excluded), and 1 Mb was labelled for peaks or noise; regions which were not clearly either were excluded. For CTCF, H3K27ac, and H3K4me3 ChIP-seq datasets, 10 chromosomes each were labelled in this manner for a total of 122 labels (55 positive peaks and 67 noise regions), 101 labels (45 positive peaks and 56 noise regions), and 224 labels (129 positive peaks and 95 noise regions) respectively. For ATAC-seq and DNase-seq, 3 chromosomes each were labelled, resulting in 196 ATAC labels (101 positive peaks and 95 noise regions) and 224 DNase labels (114 positive peaks and 110 noise regions).

True positives, false positives, true negatives, and false negatives were determined by intersecting peak calls from LanceOtron and MACS2 with these labelled data using BEDTools. True positives were found by using the command bedtools intersect -a peakCall.bed -b labeledPeaks.bed -u -wa. False negatives used bedtools intersect -a peakCall.bed -b labeledPeaks.bed -v -wa. True negatives used the command bedtools intersect -a peakCall.bed -b labeledNoise.bed

-v -wa, while false positives used bedtools intersect -a peakCall.bed -b labeled-Noise.bed -u -wa.

For peaks which were exclusively found with LanceOtron or MACS2, BEDTools was also used to find intersections which occurred at promoter or enhancer regions, as well as TSSs. The bed files listing the coordinates of the promoters or enhancers were from GenoSTAN (Zacher et al., 2017), and for TSSs we used RefTSS (Abugessaisa et al., 2019).

The heat map of the coverage was made using the deepTools command: computeMatrix reference-point -S CTCFSpleenENCFF656CCY.bw -R CTCF-spleenLoT-only-peaks.bed CTCF-spleenMACS2-only-peaks.bed –referencePoint center -a 1000 -b 1000 -out CTCF-spleenLoT-and-MACS2matrix.tab.gz. This was followed by the command: plotProfile -m CTCF-spleenLoT-and-MACS2matrix.tab.gz -out CTCF-spleenLoT-and-MACS2.png –samplesLabel “Peak caller exclusive regions” –regionsLabel “LanceOtron only” “MACS2 only” –plotType=heatmap.

5.3.2 Motif analysis

The transcription factor CTCF was intentionally selected for benchmarking because of its well studied binding motif and associated regulatory elements. For motif analysis, the CTCF position weight matrix (PWM) was downloaded from JASPAR (Fornes et al., 2020). JASPAR gathers their human data from CistromeDB (Zheng et al., 2019), who in turn use a uniform processing pipeline utilising MACS2 for peak calling. To eliminate any potential biases this may introduce, a custom motif matching script was developed. This used a simple Python regex function to find motifs composed of nucleotides in the CTCF PWM whose frequency occurred at 75% or higher, and with conserved spacing.

With the bed file of the motif coordinates made, once again BEDTools was employed to find intersections with the peak calls.

CHAPTER

SIX

DISCUSSION

The field of genomics produces completely overwhelming amounts of data - even single experiments can easily exceed the amount of information a human could understand unaided. That means the development of tools to extract meaning are not just helpful extras, they are critical to the scientific process itself. This project set out to build a method to improve the peak calling process, a goal I believe was achieved (Hentges et al., 2021). However this work also demonstrated something larger: CNNs are actually capable of extracting biologically important information from shapes present in the signal. Chromatin binding experiments lended themselves well to this type of analysis, and were the logical place to start. But as demonstrated with the 3C data from the MCC experiments, information rich patterns are more universal and found outside the data the models were trained on. This means this approach to data extraction and analysis could potentially be applied to any genomics method, as they all contain specific patterns in the data that relate to biological function. With only minor adjustments and retraining I am confident that they could be applied to transcription start sites identification from short, capped RNA-seq assays, or identifying specific nucleosomal patterns in MNase-seq Data - even Raman spectroscopy produces a signal remarkably similar to ChIP-seq. Just as optical character recognition has had countless applica-

tions beyond solely identifying hand-written digits, the ability to identify, classify and model these biological patterns in signal using DNN approaches could be the necessary step-change required for our extraction of biological insight for our ever accumulating mass of genome based information.

6.1 Summary

6.1.1 LanceOtron

In this work I developed a deep learning-based peak caller that surpasses the current gold standard peak caller MACS2. This was demonstrated by comparing performance metrics like sensitivity, selectivity, recall, and F1 scores; it was also shown by using biological evidence like motif analysis, TSS locations, and intersections with enhancers and promoters. These performance gains were demonstrated in tracks including CTCF, open chromatin tracks, and the more difficult to call histone marks H3K27ac and H3K4me3. Performance was robust even in the absence of an input control track, potentially eliminating the need for a costly, time consuming and even confounding (MACS2) additional experiment. Beyond this, a fully featured webtool with built-in genome browser, interactive charts, unsupervised learning assessments, and quick navigate bed files was implemented, with workflows developed for both new analyses or reanalysis of previous outside peak calls. This was co-developed with the DNN approaches to intuitively put the output of these complex analyses in the hands of the biologists and scientists generating the data and forming the hypotheses; an important step that is often overlooked.

6.1.2 Labelling data

The primary goal of this project was to create a peak caller that used deep learning for evaluation, and to do this required a large number of labelled samples

for training the model. Peak callers work by solving two subproblems: finding candidate peaks, and evaluating their significance (Thomas et al., 2016). So when designing the process for hand labelling regions, algorithmically labelling enriched regions as candidate peaks, then inspecting their quality mimicked how peak callers function.

ENCODE is an incredibly important resource for accessing incredible amounts of genome regulation data. Central to ENCODE's mission was the development of flexible ways to query and filter the ENCODE database, and here this was used to facilitate the creation of a tight list of very high quality ATAC-seq, ChIP-seq, and DNase-seq tracks. The scripts created for accessing the database have continued utility, and could be used to recreate or expand the datasets for other big data projects.

6.1.3 Candidate peak calling

A machine learning model's success is driven by large amounts of well balanced, unbiased datasets. 79% of data scientists' time is spent on cleansing and improving training data (*Data Science Report*, 2016). Bias, unfairness and inconsistencies are commonplace: 42% of published ML papers contain significant testing errors (Shepperd et al., 2019). Candidate peak calling was used as a method of helping to alleviate this. Labelling all of the algorithmically determined enriched regions of a randomly selected area, was thought to try and reduce my personal bias when classifying regions. LanceOtron's high performance was consistent with both internally and externally labelled datasets, suggesting that inconsistencies between labelling methodologies were minimal.

I believe it is possible to remain unbiased while selecting the start and end coordinates as long as it is a comprehensive labelling process covering a reasonably large region. In retrospect, if I were to repeat the candidate peak labelling process, I would have labelled regions by scanning and highlighting coordinates. The candidate peak process was accurate and relatively quick, but quite ineffi-

cient. A low enrichment threshold was necessary to ensure true peaks were not missed, but a consequence of this was that for every region labelled “peak”, 145 more were labelled “noise”. What I discovered when labelling regions to benchmark LanceOtron was that it was actually considerably faster to simply scan the genome over a coordinate range and just label peak and noise start and end coordinates. Given the resources and time, multiple people could label the same dataset which would also reduce potential bias. Along those same lines, Oh et al., 2020 found labelling to be consistent between experts and non-experts, so another potential solution could be to use a service like Zooniverse where the peak identification process is crowdsourced.

Unsupervised learning may provide an improved method of unbiased data annotation. Here this was attempted through the use of autoencoders. Unfortunately this attempt did not provide the clean class separation needed for accurate labelling. Because a large enough dataset had already been constructed, this was not pursued further. However it is possible that different autoencoder architectures, other unsupervised learning techniques, or a different set of engineered features could improve performance. This could also address human biases as well - even beyond disagreements over the classification of region being peak or noise. We as humans decide what is relevant, and what is most likely to be a biologically important signal. The experiments we explore are proxies for biological events, and we can never quite get all the way down to ground truth. There could be some low level enrichment pattern distinctly different from peaks normally called, but are regularly dismissed due to lack of sufficient coverage. If unsupervised learning techniques are honed, this may provide more detailed classifications for known binding events or even open the possibility of discovering new binding patterns.

Much of the work to establish methodology for candidate peak calling for the purpose of labelling translated well to peak caller functionality. Early in this project, two unexpected developments occurred. Firstly, and most importantly, a

proof of principle CNN was able to classify peaks based on simulated coverage tracks with incredible, almost unbelievable accuracy. And secondly, moving from simulated data to real experiments complicated the problem immensely. Some of this was expected. Creating a large representative dataset and working through sample labelling is the price of admission for deep learning. However challenges surfaced downstream of this process: how do you feed the model regions from real experiments in a way that it has all the necessary information for classification? This issue could be sidestepped by just analysing the whole genome in multiple overlapping sections. Then a score for every base pair could be compiled and a highly accurate map could be created. However if this were implemented in LanceOtron, with overlapping sections of 500 bp, for a human genome there would be roughly 6.6 million regions scored by the neural network. This will no doubt be feasible in the future, however the goal was to not only provide a proof of principle of the effectiveness of DNNs for peak calling, but to also develop a practical tool ready for use and publicly hosted, so we had to work within the current technological limitations. As such a great deal of time was devoted to candidate peak calling, and resulted in an efficient algorithm with little sacrifice of accuracy.

6.1.4 Measuring enrichment

How enriched does something have to be in order to be considered important? Often this question is answered with a p-value, but how an area of interest is measured or what it is being compared against are easily manipulated to affect the calculation. Exploring the various ways of p-value calculation showed that individual methods had some weaknesses, but when used in combination they could be strong. Even still, some regions simply could not be recognised using individual p-values or combinations. For this reason multiple, simplistic enrichment definitions formed the basis of the candidate peak calling algorithm taken forward for use in LanceOtron. Parameters exist for this algorithm, but in practice they provided good results across numerous datasets. A parameter-free solution

would be optimal for reproducibility, and this was explored using the Savitzky-Golay method of candidate peak calling, but unfortunately this method, although it identified noise at a high rate, performed poorly on signal. Future work could utilise both methods, identifying candidate peaks with the consensus definition algorithm and pinpointing the exact starting or ending position with Savitzky-Golay methods.

6.1.5 CNN design

Much of the groundwork for building the final wide and deep model came from building the first CNN. It is simpler to select network structure, number and size of convolutional filters, as well as the other numerous hyperparameters using a randomised grid search approach. Because of the immensely complicated way these variables interact, it will often lead to better results as well. However, by intentionally designing the CNN it forced me to contemplate the process and really consider how convolution works with genome sequencing experiments. Early on I tried normalising out all of the enrichment information, but this decreased performance. While the shape can be assessed using convolution, there was not enough context for the model to interpret the region compared to the wider area; when it was normalised to the mean chromosome signal, it could infer enrichment level. This led to the development of a model that accounts for enrichment directly, while allowing the CNN to focus on shape alone.

6.1.6 ChIPmentation

ChIPmentation analysis was another attempt to broaden the scope of genomic signal extraction and interpretation. While the MCC experiments explored the limits of the training data and pattern recognition, ChIPmentation analysis used GANs to explore universal features shared between experiment types through model design. The encoder reduced the ChIPmentation signal to its fundamental

features, and the decoder attempted to build ChIP-seq signal out of those features - all the while the discriminator of the GAN was forcing the model to develop and improve these processes. Ultimately while improving the correlation between these experiment types, this process did not improve interpretability, though this technique could be promising for the future.

6.1.7 Wide and deep model

The wide and deep model took lessons learned from p-value analysis and the CNN classifier and combined them in a way that focused on their strengths. Logistic regression is useful for measuring enrichment, which is a substantial portion of peak calling. But enrichment can arise from real biological events or undesirable sources like noise or mapping errors, and this model is completely indifferent to the source of enrichment. This is why the CNN is so useful, it is capable of matching the patterns found in the coverage tracks to their source, and compensating for the simplistic categorisation that happens with purely enrichment based models. Achieving an accuracy of 99.87% on testing data, this was the model taken forward for use in the LanceOtron package.

6.2 Contribution to the field

At some point the decision was made to take a working model, which was a solid proof of concept in its own right, and turn it into a fully featured analytical platform. This practical goal therefore meant that a substantial portion of my time also included exploring effective "use cases" for the software and developing these with my colleague, Dr. Martin Sergeant. He worked to build the JavaScript front end to marry with the computational processes and machine learning core. Considerable efforts were exerted connecting the front end web server to the computational cluster, dealing with hardware compatibility issues, working out sensible resource allocations with systems administrators, or reworking protocols to meet

the University's security guidelines. Users then went on to break my code in truly unfathomable ways (one user tried to call peaks exclusively on chromosomes that did not exist in their coverage track, another tried to upload a 4k video instead of a bigwig file). I believe expending the time to fully develop the platform and provide an interpretable result to the scientists doing these experiments was a worthwhile and indeed important endeavour. This ensured that the output was maximally transparent and exploitable for my colleagues in the field.

LanceOtron offers two main benefits to users. Firstly, the model used for peak calling improves upon the long-time gold standard peak caller MACS2. The best possible performance one could hope to get from MACS2 is when calling transcription factor binding with a narrow binding pattern producing punctate peaks, and even then LanceOtron outperforms it. Secondly, the graphical display offers features that command line only programs cannot. How results are displayed affects the usefulness of said results. Fine tuning specific parameters of a statistical model, understanding assumptions involved in p-value calculations (and more importantly where the assumptions break down), hyperparameter optimisation in deep learning models are all factors that have dramatic impacts on results. These are also often obscured from the user, however I wanted to display LanceOtron's output in a completely transparent way, so even if the finer points of these aspects are not understood, it was obvious how they work. This is why LanceOtron's candidate peak calling is only a loose filter and all enriched regions are scored and returned. There are numerous reasons why certain regions are of particular interest to individual researchers, countless definitions of enrichment, and ultimately no model is perfect, so to be able to see and interact with your data, and to set your own thresholds based on criteria that make sense to your experiment were central to going beyond simply having a good model and actually building a good peak caller.

Given that the reproducibility crisis continues to be a problem, it is useful if tools can make it easier to check if results are sensible. Using the classic com-

bination of MACS2 for calling peaks and the UCSC genome browser for inspecting them means either spending time side scrolling through large portions of the genome, or users are left without knowing if the regions examined are representative of the peak call as a whole. The LanceOtron webtool reduces the number of steps for quality control and provides a simple way to eliminate blatantly bad analyses; the thumbnail view means thousands of peaks can be quickly scanned. It also allows better quality control for scientists who intend to publish as well as reviewers of data who need to assess the quality of a data submission for an archive or for a publication.

6.3 Future

A potential further direction of research in deep learning peak calling is to incorporate the sequence along with the coverage track information. As a test, I developed such a model, and indeed it did achieve very good results. However I do not believe this is ultimately a good approach if the goal is to find chromatin binding sites. Consider a completely novel transcription factor, which binds readily at two motifs, one conventional and one unconventional. A ChIP-seq experiment is carried out and the exact same number of reads are pulled down at each of the motifs, forming two identical peaks - should the site with the conventional motif be considered more important? If these experiments are used for discovery, then pulling in sequence biases the experiment to find results that are more expected. This does not mean however that this metric lacks any usefulness in its own right.

I have focused here on problems of data extraction - how can biological insights be found in data? Moving forward the next challenge is data integration. Once a transcription factor binding site is discovered, then what is the underlying sequence? How does the histone modification landscape help predict the region's function and underlying biology? How can these different modes of information be combined to create a more detailed picture of what is occurring in the cell and

how does sequence variation affect it? The human genome project established the sequence, defining the first layer of complexity. These sequences can be transcribed and translated to protein, but this process is controlled by numerous regulatory elements. Picking apart how CREs, TFs, boundary elements, etc. behave in particular environments is important work and continues to progress. At each step the complexity of the problem increases, and integrating all of these data from the regulome is poised to be the next big challenge facing genomics. Once again this shows the continued relevance of the Bartha Knoppers quote, "As the radius of knowledge gets longer, the circumference of the unknown increases even more".

APPENDIX

A

APPENDIX

A.1 ENCODE training and testing data

Training Data

Assay	Target	Tissue	ENCODE accession
ATAC-seq	Open chromatin	Breast epithelium	ENCSR955JSO
ATAC-seq	Open chromatin	Tibial artery	ENCSR630REB
ATAC-seq	Open chromatin	Foreskin keratinocyte	ENCSR290YMN
ATAC-seq	Open chromatin	Adrenal gland	ENCSR113MBR
ATAC-seq	Open chromatin	Foreskin keratinocyte	ENCSR158XTU
ATAC-seq	Open chromatin	Foreskin keratinocyte	ENCSR677MJF
ATAC-seq	Open chromatin	Transverse colon	ENCSR668VCT
ATAC-seq	Open chromatin	Sigmoid colon	ENCSR548QCP
ATAC-seq	Open chromatin	Tibial nerve	ENCSR831KAH
ATAC-seq	Open chromatin	Thyroid gland	ENCFF710ELD
ChIP-seq	H3K27ac	RWPE1	ENCSR203KEU
ChIP-seq	H3K27ac	SKNSH	ENCSR564IGJ
ChIP-seq	H3K27ac	Bipolar neuron	ENCSR905TYC
ChIP-seq	H3K27ac	GM23338	ENCSR729ENO

ChIP-seq	H3K27ac	C42B	ENCSR279KIX
ChIP-seq	H3K27ac	22Rv1	ENCSR391NPE
ChIP-seq	H3K27ac	Foreskin keratinocyte	ENCSR709ABP
ChIP-seq	H3K27ac	Foreskin keratinocyte	ENCSR709ABP
ChIP-seq	H3K27ac	Epithelial cell of prostate	ENCSR910PDW
ChIP-seq	H3K27ac	RWPE2	ENCSR987PNT
ChIP-seq	H3K4me3	SKNSH	ENCSR975GZA
ChIP-seq	H3K4me3	SKNSH	ENCSR975GZA
ChIP-seq	H3K4me3	NCIH929	ENCSR082NQB
ChIP-seq	H3K4me3	NCIH929	ENCSR082NQB
ChIP-seq	H3K4me3	Bipolar neuron	ENCSR849YFO
ChIP-seq	H3K4me3	Bipolar neuron	ENCSR849YFO
ChIP-seq	H3K4me3	Muscle of leg	ENCSR128QKM
ChIP-seq	H3K4me3	Heart right ventricle	ENCSR107RDP
ChIP-seq	H3K4me3	Gastrocnemius medialis	ENCSR098OLN
ChIP-seq	H3K4me3	OCILY3	ENCSR548PZS
ChIP-seq	NR2C1	GM12878	ENCSR784VIQ
ChIP-seq	EP300	Ovary	ENCSR696LQU
ChIP-seq	NFXL1	GM12878	ENCSR746XEG
ChIP-seq	MXI1	Neural cell	ENCSR934NHU
ChIP-seq	ZNF318	K562	ENCSR334HSW
ChIP-seq	CREB1	HepG2	ENCSR112ALD
ChIP-seq	CTCF	RWPE1	ENCSR303GFI
ChIP-seq	RFX1	MCF7	ENCSR788XNX
ChIP-seq	CTCF	Ascending aorta	ENCSR960MDF
ChIP-seq	E4F1	K562	ENCSR731LHZ
DNase-seq	Open chromatin	Left arm bone	ENCSR976XOY
DNase-seq	Open chromatin	A673	ENCSR346JWH
DNase-seq	Open chromatin	T-helper 1 cell	ENCSR000EQC

DNase-seq	Open chromatin	Retina	ENCSR820ICX
DNase-seq	Open chromatin	Uterus	ENCSR129BZE
DNase-seq	Open chromatin	NAMALWA	ENCSR301OGM
DNase-seq	Open chromatin	SKMEL5	ENCSR000FEK
DNase-seq	Open chromatin	ELF1	ENCSR678ILN
DNase-seq	Open chromatin	Myocyte	ENCSR000EPD
DNase-seq	Open chromatin	Pancreas	ENCSR828FVZ

Testing Data

Assay	Target	Tissue	ENCODE accession
ATAC-seq	Open chromatin	MCF-7	ENCSR422SUG
ChIP-seq	CTCF	Spleen	ENCSR692ILH
ChIP-seq	H3K27ac	HAP-1	ENCSR131DVD
ChIP-seq	H3K4me3	MG63	ENCSR579SNM
DNase-seq	Open chromatin	A549	ENCSR000ELW

Table A.1: ENCODE training and testing data.

A.2 Additional LanceOtron benchmarking

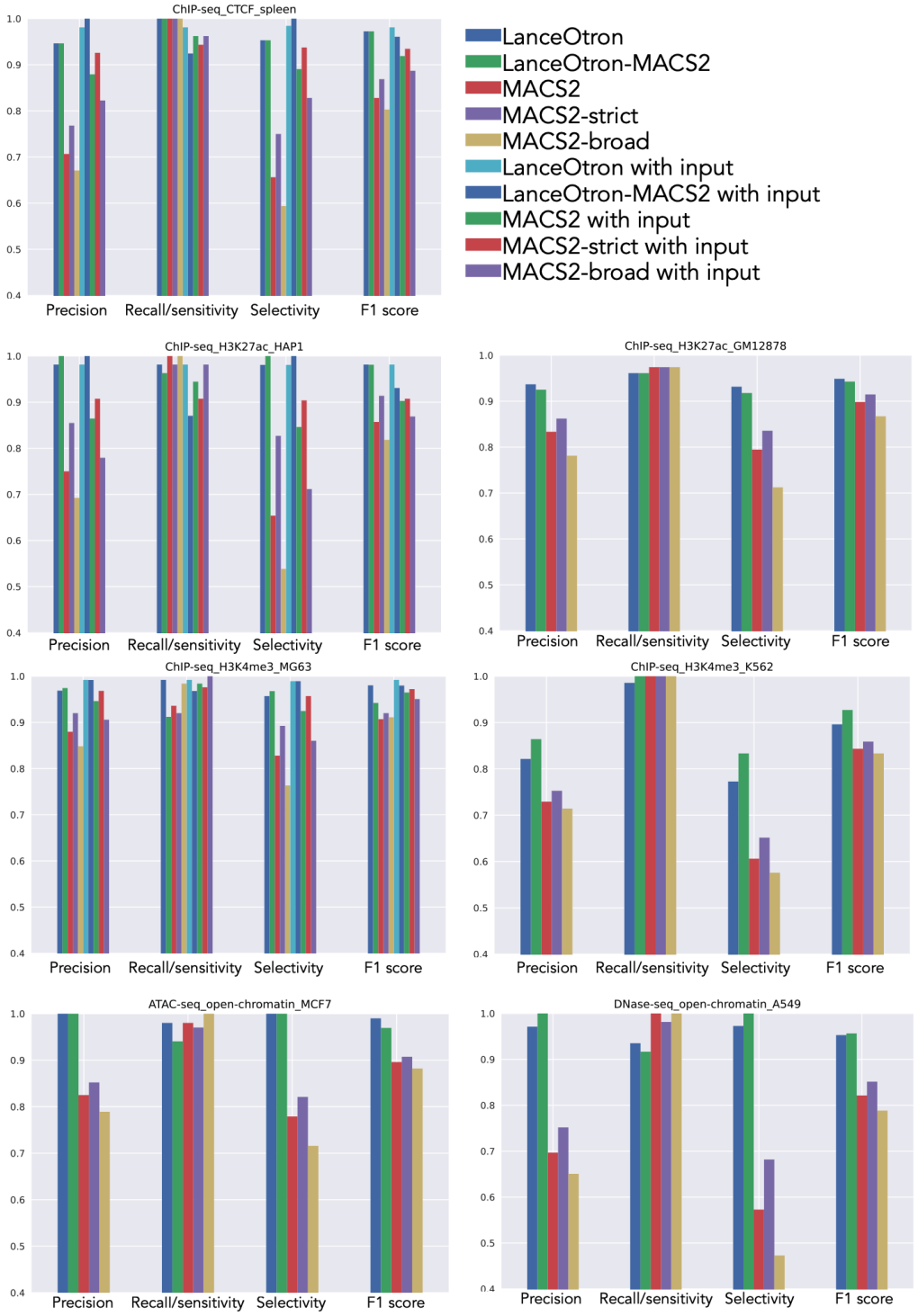


Figure A.1: Performance benchmarks of LanceOtron compared to MACS2 with various parameters. MACS2 performance scores are obtained using a default q-value of 0.05, while MACS2-strict uses a q-value of 0.01 (the same value as Chapter 4); MACS2-broad uses the broad parameter. LanceOtron-MACS2 uses LanceOtron's deep learning model to score MACS2 peak calls, using them as candidate peaks.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., & Brevedo, E. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/

Abugessaisa, I., Noguchi, S., Hasegawa, A., Kondo, A., Kawaji, H., Carninci, P., & Kasukawa, T. (2019). refTSS: A Reference Data Set for Human and Mouse Transcription Start Sites. *Journal of Molecular Biology*, *431*(13), 2407–2422. https://doi.org/10.1016/j.jmb.2019.04.045

Alkuraya, F. S. (2021). How the human genome transformed study of rare diseases. *Nature*, *590*(7845), 218–219. https://doi.org/10.1038/d41586-021-00294-7

Amemiya, H. M., Kundaje, A., & Boyle, A. P. (2019). The ENCODE Blacklist: Identification of Problematic Regions of the Genome. *Scientific Reports*, *9*(1), 9354. https://doi.org/10.1038/s41598-019-45839-z

Ashoor, H., Hérault, A., Kamoun, A., Radvanyi, F., Bajic, V. B., Barillot, E., & Boeva, V. (2013). HMCan: A method for detecting chromatin modifications in cancer samples using ChIP-seq data. *Bioinformatics*, *29*(23), 2979–2986. https://doi.org/10.1093/bioinformatics/btt524

Auerbach, R. K., Euskirchen, G., Rozowsky, J., Lamarre-Vincent, N., Moqtaderi, Z., Lefrançois, P., Struhl, K., Gerstein, M., & Snyder, M. (2009). Mapping accessible chromatin regions using Sono-Seq. *Proceedings of the National Academy of Sciences of the United States of America*, *106*(35), 14926–14931. https://doi.org/10.1073/pnas.0905443106

Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, *533*(7604), 452–454. https://doi.org/10.1038/533452a

Bardet, A. F., Steinmann, J., Bafna, S., Knoblich, J. A., Zeitlinger, J., & Stark, A. (2013). Identification of transcription factor binding sites from ChIP-seq data at high resolution. *Bioinformatics*, *29*(21), 2705–2713. https://doi.org/10.1093/bioinformatics/btt470

Barski, A., Cuddapah, S., Cui, K., Roh, T.-Y., Schones, D. E., Wang, Z., Wei, G., Chepelev, I., & Zhao, K. (2007). High-Resolution Profiling of Histone Methylations in the Human Genome. *Cell*, *129*(4), 823–837. https://doi.org/10.1016/j.cell.2007.05.009

Bastiaan Holwerda, S. J., & de Laat, W. (2013). CTCF: The protein, the binding partners, the binding sites and their chromatin loops. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *368*(1620). https://doi.org/10.1098/rstb.2012.0369

Boeva, V., Surdez, D., Guillon, N., Tirode, F., Fejes, A. P., Delattre, O., & Barillot, E. (2010). De novo motif identification improves the accuracy of predicting transcrip-

tion factor binding sites in ChIP-Seq data analysis. *Nucleic Acids Research*, *38*(11), 1–9. https://doi.org/10.1093/nar/gkq217

Boyle, A. P., Davis, S., Shulha, H. P., Meltzer, P., Margulies, E. H., Weng, Z., Furey, T. S., & Crawford, G. E. (2008). High-Resolution Mapping and Characterization of Open Chromatin across the Genome. *Cell*, *132*(2), 311–322. https://doi.org/10.1016/j.cell.2007.12.014

Boyle, A. P., Guinney, J., Crawford, G. E., & Furey, T. S. (2008). F-Seq: A feature density estimator for high-throughput sequence tags. *Bioinformatics*, *24*(21), 2537–2538. https://doi.org/10.1093/bioinformatics/btn480

Buenrostro, J. D., Giresi, P. G., Zaba, L. C., Chang, H. Y., & Greenleaf, W. J. (2013). Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nature Methods*, *10*(12), 1213–1218. https://doi.org/10.1038/nmeth.2688

Cheng, H.-T., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., Shah, H., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., & Chai, W. (2016). Wide & Deep Learning for Recommender Systems. *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems - DLRS 2016*, 7–10. https://doi.org/10.1145/2988450.2988454

Chitpin, J. G., Awdeh, A., & Perkins, T. J. (2019). RECAP reveals the true statistical significance of ChIP-seq peak calls. *Bioinformatics*, *35*(19), 3592–3598. https://doi.org/10.1093/bioinformatics/btz150

Chollet, F. et al. (2015). Keras.

Chollet, F. (2016). How convolutional neural networks see the world. https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

Chung, D., Park, D., Myers, K., Grass, J., Kiley, P., Landick, R., & Keleş, S. (2013). dPeak: High Resolution Identification of Transcription Factor Binding Sites from PET and SET ChIP-Seq Data. *PLoS Computational Biology*, *9*(10), 9–11. https://doi.org/10.1371/journal.pcbi.1003246

Collins, F. S., & McKusick, V. A. (2001). Implications of the human genome project for medical science. *Journal of the American Medical Association*, *285*(5), 540–544. https://doi.org/10.1001/jama.285.5.540

Craig Venter, J., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., Smith, H. O., Yandell, M., Evans, C. A., Holt, R. A., Gocayne, J. D., Amanatides, P., Ballew, R. M., Huson, D. H., Wortman, J. R., Zhang, Q., Kodira, C. D., Zheng, X. H., Chen, L., … Zhu, X. (2001). The sequence of the human genome. *Science*, *291*(5507), 1304–1351. https://doi.org/10.1126/science.1058040

Data Science Report (tech. rep.). (2016). CrowdFlower. https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf

Davies, J. O., Telenius, J. M., McGowan, S. J., Roberts, N. A., Taylor, S., Higgs, D. R., & Hughes, J. R. (2015). Multiplexed analysis of chromosome conformation at vastly improved sensitivity. *Nature Methods*, *13*(1), 74–80. https://doi.org/10.1038/nmeth.3664

Davis, C. A., Hitz, B. C., Sloan, C. A., Chan, E. T., Davidson, J. M., Gabdank, I., Hilton, J. A., Jain, K., Baymuradov, U. K., Narayanan, A. K., Onate, K. C., Graham, K., Miyasato, S. R., Dreszer, T. R., Strattan, J. S., Jolanki, O., Tanaka, F. Y., & Cherry, J. M. (2018). The Encyclopedia of DNA elements (ENCODE): Data portal update. *Nucleic Acids Research*, *46*(D1), D794–D801. https://doi.org/10.1093/nar/gkx1081

Dekker, J., Rippe, K., Dekker, M., & Kleckner, N. (2002). Capturing Chromosome Conformation. *Science*, *295*(5558), 1306–1311. https://doi.org/10.1126/science.1067799

Dunham, I., Kundaje, A., Aldred, S. F., Collins, P. J., Davis, C. A., Doyle, F., Epstein, C. B., Frietze, S., Harrow, J., Kaul, R., Khatun, J., Lajoie, B. R., Landt, S. G., Lee, B. K., Pauli, F., Rosenbloom, K. R., Sabo, P., Safi, A., Sanyal, A., … Lochovsky, L. (2012). An integrated encyclopedia of DNA elements in the human genome. *Nature*, *489*(7414), 57–74. https://doi.org/10.1038/nature11247

Elo, L. L., Kallio, A., Laajala, T. D., Hawkins, R. D., Korpelainen, E., & Aittokallio, T. (2012). Optimized detection of transcription factor-binding sites in ChIP-seq experiments. *Nucleic Acids Research*, *40*(1), 1–7. https://doi.org/10.1093/nar/gkr839

Fejes, A. P., Robertson, G., Bilenky, M., Varhol, R., Bainbridge, M., & Jones, S. J. (2008). FindPeaks 3.1: A tool for identifying areas of enrichment from massively parallel short-read sequencing technology. *Bioinformatics*, *24*(15), 1729–1730. https://doi.org/10.1093/bioinformatics/btn305

Feng, X., Grossman, R., & Stein, L. (2011). PeakRanger: A cloud-enabled peak caller for ChIP-seq data. *BMC Bioinformatics*, *12*. https://doi.org/10.1186/1471-2105-12-139

Fornes, O., Castro-Mondragon, J. A., Khan, A., Van Der Lee, R., Zhang, X., Richmond, P. A., Modi, B. P., Correard, S., Gheorghe, M., Baranašić, D., Santana-Garcia, W., Tan, G., Chèneby, J., Ballester, B., Parcy, F., Sandelin, A., Lenhard, B., Wasserman, W. W., & Mathelier, A. (2020). JASPAR 2020: Update of the open-Access database of transcription factor binding profiles. *Nucleic Acids Research*, *48*(D1), D87–D92. https://doi.org/10.1093/nar/gkz1001

French, J. D., & Edwards, S. L. (2020). The Role of Noncoding Variants in Heritable Disease. *Trends in Genetics*, *36*(11), 880–891. https://doi.org/10.1016/j.tig.2020.07.004

Fudenberg, G., Imakaev, M., Lu, C., Goloborodko, A., Abdennur, N., & Mirny, L. A. (2016). Formation of Chromosomal Domains by Loop Extrusion. *Cell Reports*, *15*(9), 2038–2049. https://doi.org/10.1016/j.celrep.2016.04.085

Gates, A. J., Gysi, D. M., Kellis, M., & Barabási, A. L. (2021). A wealth of discovery built on the Human Genome Project — by the numbers. *Nature*, *590*(7845), 212–215. https://doi.org/10.1038/d41586-021-00314-6

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly.

Gogol-Döring, A., & Chen, W. (2012). An Overview of the Analysis of Next Generation Sequencing Data. *Methods in molecular biology* (pp. 249–257). https://doi.org/10.1007/978-1-61779-400-1{_}16

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. *Advances in Neural Information Processing Systems*, *2017-Decem*, 4089–4099. http://arxiv.org/abs/1406.2661

Guo, Y., Mahony, S., & Gifford, D. K. (2012). High Resolution Genome Wide Binding Event Finding and Motif Discovery Reveals Transcription Factor Spatial Binding Constraints (S. Aerts, Ed.). *PLoS Computational Biology*, *8*(8), e1002638. https://doi.org/10.1371/journal.pcbi.1002638

Haberle, V., & Stark, A. (2018). Eukaryotic core promoters and the functional basis of transcription initiation. *Nature Reviews Molecular Cell Biology*, *19*(10), 621–637. https://doi.org/10.1038/s41580-018-0028-8

Halbritter, F., Vaidya, H. J., & Tomlinson, S. R. (2012). GeneProf: Analysis of high-throughput sequencing experiments. *Nature Methods*, *9*(1), 7–8. https://doi.org/10.1038/nmeth.1809

Hansen, P., Hecht, J., Ibrahim, D. M., Krannich, A., Truss, M., & Robinson, P. N. (2015). Saturation analysis of ChIP-seq data for reproducible identification of binding peaks. *Genome Research*, *25*(9), 1391–1400. https://doi.org/10.1101/gr.189894.115

Heinz, S., Benner, C., Spann, N., Bertolino, E., Lin, Y. C., Laslo, P., Cheng, J. X., Murre, C., Singh, H., & Glass, C. K. (2010). Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. *Molecular Cell*, *38*(4), 576–589. https://doi.org/10.1016/j.molcel.2010.05.004

Heinz, S., Romanoski, C. E., Benner, C., & Glass, C. K. (2015). The selection and function of cell type-specific enhancers. *Nature Reviews Molecular Cell Biology*, *16*(3), 144–154. https://doi.org/10.1038/nrm3949

Hentges, L. D., Sergeant, M. J., Downes, D. J., Hughes, J. R., & Taylor, S. (2021). LanceOtron: a deep learning peak caller for ATAC-seq, ChIP-seq, and DNase-seq. *bioRxiv*. https://doi.org/10.1101/2021.01.25.428108

Higgs, D., Vickers, M., Wilkie, A., Pretorius, I., Jarman, A., & Weatherall, D. (1989). A review of the molecular genetics of the human alpha-globin gene cluster. *Blood*, *73*(5), 1081–1104. https://doi.org/10.1182/blood.V73.5.1081.1081

Hindorff, L. A., Sethupathy, P., Junkins, H. A., Ramos, E. M., Mehta, J. P., Collins, F. S., & Manolio, T. A. (2009). Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. *Proceedings of the National Academy of Sciences*, *106*(23), 9362–9367. https://doi.org/10.1073/pnas.0903103106

Hocking, T. D., Goerner-Potvin, P., Morin, A., Shao, X., Pastinen, T., & Bourque, G. (2016). Optimizing ChIP-seq peak detectors using visual labels and supervised machine learning. *Bioinformatics*, *33*(4), btw672. https://doi.org/10.1093/bioinformatics/btw672

Hower, V., Evans, S. N., & Pachter, L. (2011). Shape-based peak identification for ChIP-Seq. *BMC Bioinformatics*, *12*. https://doi.org/10.1186/1471-2105-12-15

Hsieh, T. H. S., Fudenberg, G., Goloborodko, A., & Rando, O. J. (2016). Micro-C XL: Assaying chromosome conformation from the nucleosome to the entire genome. *Nature Methods*, *13*(12), 1009–1011. https://doi.org/10.1038/nmeth.4025

Hua, P., Badat, M., Hanssen, L. L. P., Hentges, L. D., Crump, N., Downes, D. J., Jeziorska, D. M., Oudelaar, A. M., Schwessinger, R., Taylor, S., Milne, T. A., Hughes, J. R., Higgs, D. R., & Davies, J. O. J. (2021). Defining genome architecture at base-pair resolution. *Nature*, *595*(7865), 125–129. https://doi.org/10.1038/s41586-021-03639-4

Hughes, J. R., Roberts, N., Mcgowan, S., Hay, D., Giannoulatou, E., Lynch, M., De Gobbi, M., Taylor, S., Gibbons, R., & Higgs, D. R. (2014). Analysis of hundreds of cis-regulatory landscapes at high resolution in a single, high-throughput experiment. *Nature Genetics*, *46*(2), 205–212. https://doi.org/10.1038/ng.2871

Ibrahim, M. M., Lacadie, S. A., & Ohler, U. (2015). JAMM: A peak finder for joint analysis of NGS replicates. *Bioinformatics*, *31*(1), 48–55. https://doi.org/10.1093/bioinformatics/btu568

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *33*(2), 81–87. https://doi.org/10.1088/1751-8113/44/8/085201

Jalili, V., Matteucci, M., Masseroli, M., & Morelli, M. J. (2015). Using combined evidence from replicates to evaluate ChIP-seq peaks. *Bioinformatics*, *31*(17), 2761–2769. https://doi.org/10.1093/bioinformatics/btv293

Japkowicz, N. (2000). The Class Imbalance Problem: Significance and Strategies. *Proceedings of the 2000 International Conference on Artificial Intelligence*, 111–117.

Ji, H., Jiang, H., Ma, W., Johnson, D. S., Myers, R. M., & Wong, W. H. (2008). An integrated software system for analyzing ChIP-chip and ChIP-seq data. *Nature Biotechnology*, *26*(11), 1293–1300. https://doi.org/10.1038/nbt.1505

Johnson, D. S., Mortazavi, A., & Myers, R. M. (2007). Protein-DNA Interactions. (June), 1497–1503.

Jollife, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2065). https://doi.org/10.1098/rsta.2015.0202

Joyner, M. J., & Paneth, N. (2019). Promises, promises, and precision medicine. https://doi.org/10.1172/JCI126119

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., & Haussler, a. D. (2002). The Human Genome Browser at UCSC. *Genome Research*, *12*(6), 996–1006. https://doi.org/10.1101/gr.229102

Kim, N. K., Jayatillake, R. V., & Spouge, J. L. (2013). NEXT-peak: A normal-exponential two-peak model for peak-calling in ChIP-seq data. *BMC Genomics*, *14*(1), 349. https://doi.org/10.1186/1471-2164-14-349

Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-Normalizing Neural Networks. https://doi.org/1706.02515

Kornberg, R. (1974). Chromatin Structure : A Repeating Unit of Histones and DNA Chromatin structure is based on a repeating unit of eight. *Science*, *184*, 868–871.

Kuan, P. F., Chung, D., Pan, G., Thomson, J. A., Stewart, R., & Keleş, S. (2011). A statistical framework for the analysis of ChIP-Seq data. *Journal of the American Statistical Association*, *106*(495), 891–903. https://doi.org/10.1198/jasa.2011.ap09706

Kumar, V., Muratani, M., Rayan, N. A., Kraus, P., Lufkin, T., Ng, H. H., & Prabhakar, S. (2013). Uniform, optimal signal processing of mapped deep-sequencing data. *Nature Biotechnology*, *31*(7), 615–622. https://doi.org/10.1038/nbt.2596

Lambert, S. A., Jolma, A., Campitelli, L. F., Das, P. K., Yin, Y., Albu, M., Chen, X., Taipale, J., Hughes, T. R., & Weirauch, M. T. (2018). The Human Transcription Factors. *Cell*, *172*(4), 650–665. https://doi.org/10.1016/j.cell.2018.01.029

Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., Fitzhugh, W., Funke, R., Gage, D., Harris, K., Heaford, A., Howland, J., Kann, L., Lehoczky, J., Levine, R., McEwan, P., … Chen, Y. J. (2001). Initial sequencing and analysis of the human genome. *Nature*, *409*(6822), 860–921. https://doi.org/10.1038/35057062

Landt, S. G., Marinov, G. K., Kundaje, A., Kheradpour, P., Pauli, F., Batzoglou, S., Bernstein, B. E., Bickel, P., Brown, J. B., Cayting, P., Chen, Y., DeSalvo, G., Epstein, C., Fisher-Aylor, K. I., Euskirchen, G., Gerstein, M., Gertz, J., Hartemink, A. J., Hoffman, M. M., … Snyder, M. (2012). ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Research*, *22*(9), 1813–1831. https://doi.org/10.1101/gr.136184.111

Langmead, B., Trapnell, C., Pop, M., & Salzberg, S. (2009). 2C- Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, *10*(3), R25. https://doi.org/10.1186/gb-2009-10-3-r25

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., & Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, *25*(16), 2078–2079. https://doi.org/10.1093/bioinformatics/btp352

Li, Q., Brown, J. B., Huang, H., & Bickel, P. J. (2011). Measuring reproducibility of high-throughput experiments. *Annals of Applied Statistics*, *5*(3), 1752–1779. https://doi.org/10.1214/11-AOAS466

Li, Y., Umbach, D. M., & Li, L. (2014). T-KDE: A method for genome-wide identification of constitutive protein binding sites from multiple ChIP-seq data sets. *BMC Genomics*, *15*(1). https://doi.org/10.1186/1471-2164-15-27

Long, H. K., Prescott, S. L., & Wysocka, J. (2016). Ever-Changing Landscapes: Transcriptional Enhancers in Development and Evolution. *Cell*, *167*(5), 1170–1187. https://doi.org/10.1016/j.cell.2016.09.018

Lund, E., Oldenburg, A. R., & Collas, P. (2014). Enriched domain detector: A program for detection of wide genomic enrichment domains robust against local variations. *Nucleic Acids Research*, *42*(11). https://doi.org/10.1093/nar/gku324

Maurano, M. T., Humbert, R., Rynes, E., Thurman, R. E., Haugen, E., Wang, H., Reynolds, A. P., Sandstrom, R., Qu, H., Brody, J., Shafer, A., Neri, F., Lee, K., Kutyavin, T., Stehling-Sun, S., Johnson, A. K., Canfield, T. K., Giste, E., Diegel, M., … Stamatoyannopoulos, J. A. (2012). Systematic Localization of Common Disease-Associated Variation in Regulatory DNA. *Science*, *337*(6099), 1190–1195. https://doi.org/10.1126/science.1222794

McGowan, S. J., Hughes, J. R., Han, Z. P., & Taylor, S. (2013). MIG: Multi-Image Genome viewer. *Bioinformatics*, *29*(19), 2477–2478. https://doi.org/10.1093/bioinformatics/btt406

McInnes, L., Healy, J., Saul, N., & Großberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, *3*(29), 861. https://doi.org/10.21105/joss.00861

Micsinai, M., Parisi, F., Strino, F., Asp, P., Dynlacht, B. D., & Kluger, Y. (2012). Picking ChIP-seq peak detectors for analyzing chromatin modification experiments. *Nucleic Acids Research*, *40*(9). https://doi.org/10.1093/nar/gks048

Mikkelsen, T. S., Ku, M., Jaffe, D. B., Issac, B., Lieberman, E., Giannoukos, G., Alvarez, P., Brockman, W., Kim, T. K., Koche, R. P., Lee, W., Mendenhall, E., O'Donovan, A., Presser, A., Russ, C., Xie, X., Meissner, A., Wernig, M., Jaenisch, R., … Bernstein, B. E. (2007). Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature*, *448*(7153), 553–560. https://doi.org/10.1038/nature06008

Mosteller, F., & Fisher, R. A. (1948). Questions and Answers. *The American Statistician*, *2*(5), 30. https://doi.org/10.2307/2681650

Muiño, J. M., Kaufmann, K., van Ham, R. C., Angenent, G. C., & Krajewski, P. (2011). ChIP-seq Analysis in R (CSAR): An R package for the statistical detection of protein-bound genomic regions. *Plant Methods*, *7*(1), 1–8. https://doi.org/10.1186/1746-4811-7-11

Nakato, R., Itoh, T., & Shirahige, K. (2013). DROMPA: Easy-to-handle peak calling and visualization software for the computational analysis and validation of ChIP-seq data. *Genes to Cells*, *18*(7), 589–601. https://doi.org/10.1111/gtc.12058

Narlikar, L., & Jothi, R. (2012). ChIP-Seq Data Analysis: Identification of Protein–DNA Binding Sites with SISSRs Peak-Finder. https://doi.org/10.1007/978-1-61779-400-1{_}20

Newkirk, D., Biesinger, J., Chon, A., Yokomori, K., & Xie, X. (2011). AREM: Aligning short reads from ChIP-sequencing by expectation maximization. *Journal of Computational Biology*, *18*(11), 1495–1505. https://doi.org/10.1089/cmb.2011.0185

Nuzzo, R. (2014). Scientific method: Statistical errors. *Nature*, *506*(7487), 150–152. https://doi.org/10.1038/506150a

Oh, D., Strattan, J. S., Hur, J. K., Bento, J., Urban, A. E., Song, G., & Cherry, J. M. (2020). CNN-Peaks: ChIP-Seq peak detection pipeline using convolutional neural networks that imitate human visual inspection. *Scientific Reports*, *10*(1), 7933. https://doi.org/10.1038/s41598-020-64655-4

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al. (2019). Keras Tuner. https://github.com/keras-team/keras-tuner

Park, P. J. (2009). ChIP–seq: advantages and challenges of a maturing technology. *Nature Reviews Genetics*, *10*(10), 669–680. https://doi.org/10.1038/nrg2641

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

Qin, Z. S., Yu, J., Shen, J., Maher, C. A., Hu, M., Kalyana-Sundaram, S., Yu, J., & Chinnaiyan, A. M. (2010). HPeak: An HMM-based algorithm for defining read-enriched regions in ChIP-Seq data. *BMC Bioinformatics*, *11*. https://doi.org/10.1186/1471-2105-11-369

Quinlan, A. R., & Hall, I. M. (2010). BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, *26*(6), 841–842. https://doi.org/10.1093/bioinformatics/btq033

Ramírez, F., Dündar, F., Diehl, S., Grüning, B. A., & Manke, T. (2014). deepTools: a flexible platform for exploring deep-sequencing data. *Nucleic Acids Research*, *42*(W1), W187–W191. https://doi.org/10.1093/nar/gku365

Rashid, N. U., Giresi, P. G., Ibrahim, J. G., Sun, W., & Lieb, J. D. (2011). ZINBA integrates local covariates with DNA-seq data to identify broad and narrow regions of enrichment, even within amplified genomic regions. *Genome Biology*, *12*(7). https://doi.org/10.1186/gb-2011-12-7-r67

Robertson, G., Hirst, M., Bainbridge, M., Bilenky, M., Zhao, Y., Zeng, T., Euskirchen, G., Bernier, B., Varhol, R., Delaney, A., Thiessen, N., Griffith, O. L., He, A., Marra, M., Snyder, M., & Jones, S. (2007). Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nature Methods*, *4*(8), 651–657. https://doi.org/10.1038/nmeth1068

Robinson, J. T., Thorvaldsdóttir, H., Winckler, W., Guttman, M., Lander, E. S., Getz, G., & Mesirov, J. P. (2011). Integrative genomics viewer. *Nature Biotechnology*, *29*(1), 24–26. https://doi.org/10.1038/nbt.1754

Rozowsky, J., Euskirchen, G., Auerbach, R. K., Zhang, Z. D., Gibson, T., Bjornson, R., Carriero, N., Snyder, M., & Gerstein, M. B. (2009). PeakSeq enables systematic scoring of ChIP-seq experiments relative to controls. *Nature Biotechnology*, *27*(1), 66–75. https://doi.org/10.1038/nbt.1518

Rye, M. B., Sætrom, P., & Drabløs, F. (2011). A manually curated ChIP-seq benchmark demonstrates room for improvement in current peak-finder programs. *Nucleic Acids Research*, *39*(4), e25–e25. https://doi.org/10.1093/nar/gkq1187

Savitzky, A., & Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, *36*(8), 1627–1639. https://doi.org/10.1021/ac60214a047

Schmidl, C., Rendeiro, A. F., Sheffield, N. C., & Bock, C. (2015). ChIPmentation: Fast, robust, low-input ChIP-seq for histones and transcription factors. *Nature Methods*, *12*(10), 963–965. https://doi.org/10.1038/nmeth.3542

Schwessinger, R., Gosden, M., Downes, D., Brown, R. C., Oudelaar, A. M., Telenius, J., Teh, Y. W., Lunter, G., & Hughes, J. R. (2020). DeepC: predicting 3D genome folding using megabase-scale transfer learning. *Nature Methods*, *17*(11), 1118–1124. https://doi.org/10.1038/s41592-020-0960-3

Sergeant, M. J., Hughes, J. R., Hentges, L., Lunter, G., Downes, D. J., & Taylor, S. (2021). Multi Locus View: an extensible web-based tool for the analysis of genomic data. *Communications Biology*, *4*(1), 1–7. https://doi.org/10.1038/s42003-021-02097-y

Shepperd, M., Guo, Y., Li, N., Arzoky, M., Capiluppi, A., Counsell, S., Destefanis, G., Swift, S., Tucker, A., & Yousefi, L. (2019). The Prevalence of Errors in Machine Learning Experiments. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11871 LNCS*, 102–109. https://doi.org/10.1007/978-3-030-33607-3{_}12

Shlyueva, D., Stampfel, G., & Stark, A. (2014). Transcriptional enhancers: From properties to genome-wide predictions. *Nature Reviews Genetics*, *15*(4), 272–286. https://doi.org/10.1038/nrg3682

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, 1–14. https://doi.org/10.2146/ajhp170251

Song, Q., & Smith, A. D. (2011). Identifying dispersed epigenomic domains from ChIP-Seq data. *Bioinformatics*, *27*(6), 870–871. https://doi.org/10.1093/bioinformatics/btr030

Spyrou, C., Stark, R., Lynch, A. G., & Tavaré, S. (2009). BayesPeak: Bayesian analysis of chip-seq data. *BMC Bioinformatics*, *10*, 299. https://doi.org/10.1186/1471-2105-10-299

Stanton, K. P., Jin, J., Lederman, R. R., Weissman, S. M., & Kluger, Y. (2017). Ritornello: high fidelity control-free chromatin immunoprecipitation peak calling. *Nucleic Acids Research*, (October), 1–18. https://doi.org/10.1093/nar/gkx799

Tarbell, E. D., & Liu, T. (2019). HMMRATAC: a Hidden Markov ModeleR for ATAC-seq. *Nucleic acids research*, *47*(16), e91. https://doi.org/10.1093/nar/gkz533

Taskesen, E., Hoogeboezem, R., Delwel, R., & Reinders, M. J. (2013). Hypergeometric analysis of tiling-array and sequence data: Detection and interpretation of peaks. *Advances and Applications in Bioinformatics and Chemistry*, *6*(1), 55–62. https://doi.org/10.2147/AABC.S51271

Thomas, R., Thomas, S., Holloway, A. K., & Pollard, K. S. (2016). Features that define the best ChIP-seq peak calling algorithms. *Briefings in Bioinformatics*, *18*(3), bbw035. https://doi.org/10.1093/bib/bbw035

Valouev, A., Johnson, D. S., Sundquist, A., Medina, C., Anton, E., Batzoglou, S., Myers, R. M., & Sidow, A. (2008). Genome-wide analysis of transcription factor binding sites based on ChIP-Seq data. *Nature Methods*, *5*(9), 829–834. https://doi.org/10.1038/nmeth.1246

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

Vega, V. B., Cheung, E., Palanisamy, N., & Sung, W. K. (2009). Inherent signals in sequencing-based Chromatin-ImmunoPrecipitation control libraries. *PLoS ONE*, *4*(4). https://doi.org/10.1371/journal.pone.0005241

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Vivó-Truyols, G., & Schoenmakers, P. J. (2006). Automatic selection of optimal Savitzky-Golay smoothing. *Analytical Chemistry*, *78*(13), 4598–4608. https://doi.org/10.1021/ac0600196

Wang, C., Xu, J., Zhang, D., Wilson, Z. A., & Zhang, D. (2010). An effective approach for identification of in vivo protein-DNA binding sites from paired-end ChIP-Seq data. *BMC Bioinformatics*, *11*. https://doi.org/10.1186/1471-2105-11-81

Wang, J., Lunyak, V. V., & King Jordan, I. (2013). BroadPeak: A novel algorithm for identifying broad peaks in diffuse ChIP-seq datasets. *Bioinformatics*, *29*(4), 492–493. https://doi.org/10.1093/bioinformatics/bts722

Wang, R., Hsu, H. K., Blattler, A., Wang, Y., Lan, X., Wang, Y., Hsu, P. Y., Leu, Y. W., Huang, T. H., Farnham, P. J., & Jin, V. X. (2013). LOcating Non-Unique matched Tags (LONUT) to Improve the Detection of the Enriched Regions for ChIP-seq Data. *PLoS ONE*, *8*(6), 1–10. https://doi.org/10.1371/journal.pone.0067788

Wilbanks, E. G., & Facciotti, M. T. (2010). Evaluation of Algorithm Performance in ChIP-Seq Peak Detection (G. J. C. Veenstra, Ed.). *PLoS ONE*, *5*(7), e11471. https://doi.org/10.1371/journal.pone.0011471

Wittkopp, P. J., & Kalay, G. (2012). Cis-regulatory elements: Molecular mechanisms and evolutionary processes underlying divergence. *Nature Reviews Genetics*, *13*(1), 59–69. https://doi.org/10.1038/nrg3095

Wu, H., & Ji, H. (2014). Polya peak: Detecting transcription factor binding sites from Ch IP-seq using peak shape information. *PLoS ONE*, *9*(3). https://doi.org/10.1371/journal.pone.0089694

Zacher, B., Michel, M., Schwalb, B., Cramer, P., Tresch, A., & Gagneur, J. (2017). Accurate promoter and enhancer identification in 127 ENCODE and roadmap epigenomics cell types and tissues by GenoSTAN. *PLoS ONE*, *12*(1), 1–25. https://doi.org/10.1371/journal.pone.0169249

Zang, C., Schones, D. E., Zeng, C., Cui, K., Zhao, K., & Peng, W. (2009). A clustering approach for identification of enriched domains from histone modification ChIP-Seq data. *Bioinformatics*, *25*(15), 1952–1958. https://doi.org/10.1093/bioinformatics/btp340

Zaret, K. S. (2020). Pioneer Transcription Factors Initiating Gene Network Changes. *Annual Review of Genetics*, *54*, 367–385. https://doi.org/10.1146/annurev-genet-030220-015007

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. http://arxiv.org/abs/1212.5701

Zhang, X., Robertson, G., Krzywinski, M., Ning, K., Droit, A., Jones, S., & Gottardo, R. (2011). PICS: Probabilistic Inference for ChIP-seq. *Biometrics*, *67*(1), 151–163. https://doi.org/10.1111/j.1541-0420.2010.01441.x

Zhang, Y., Liu, T., Meyer, C. A., Eeckhoute, J., Johnson, D. S., Bernstein, B. E., Nussbaum, C., Myers, R. M., Brown, M., Li, W., & Liu, X. S. (2008). Model-based Analysis of ChIP-Seq (MACS). *Genome Biology*, *9*(9), R137. https://doi.org/10.1186/gb-2008-9-9-r137

Zheng, R., Wan, C., Mei, S., Qin, Q., Wu, Q., Sun, H., Chen, C. H., Brown, M., Zhang, X., Meyer, C. A., & Liu, X. S. (2019). Cistrome Data Browser: Expanded datasets and new tools for gene regulatory analysis. *Nucleic Acids Research*, *47*(D1), D729–D735. https://doi.org/10.1093/nar/gky1094

Zhou, J., & Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, *12*(10), 931–934. https://doi.org/10.1038/nmeth.3547