

Machine Learning in Genomics: Containerised
tutorials demonstrating best practises, pitfalls,
and reproducibility

Sach Nehal

2024-09-07

Contents

About	5
I Introduction	7
1 Epigenetic Data	9
1.1 What is epigenetic data?	9
1.2 What does epigenetic data look like?	12
1.3 Sources of epigenetic data	15
1.4 UCSC'S Genome Browser	16
2 Pre-processing of bigWig files	17
2.1 Data loaders and simplifying pre-processing	20
II Training models with DNA input	21
3 Loss functions	23
4 Training tricks	25
4.1 Reverse Complements and Sequence Shifts	25
4.2 Hyper-parameter optimisation	27
5 Reproducibility of machine learning models	29
6 Testing	31
7 Software libraries for model building	33
7.1 gReLU	33
7.2 Kipoi	34

III	ML pitfalls in genomics	35
8	Pitfalls overview	37
IV	Using existing models	43
9	Training gReLU models and examining pitfalls	45

About

Applied machine learning utilising vast amounts of data has aided in pattern identification, predictive analytics, and solving complex problems across a multitude of fields. Solving these complex problems within these fields, researchers would find differing answers to the following questions; **what machine learning techniques can we apply to the problem, how do we apply the techniques in the context of this field, and why do we need to apply them in this way?** In any case, applied machine learning requires an interdisciplinary understanding of computing techniques and the field in question.

The aim of this project is to provide you with **a set of reproducible tutorials that include all necessary data, code, and descriptions to replicate key results, along with demonstrations of common pitfalls, in the field of genomics.** It is designed for users with knowledge of machine learning but little or no background in biology as a process to learn about applying machine learning techniques in genomics.

Part I

Introduction

Chapter 1

Epigenetic Data

1.1 What is epigenetic data?

As you may already know, typically all of the cells in your body contain the same DNA. How, then, do we have different cell types in our body? Your DNA contains a script that is able to produce the proteins required for each specific cell in your body. Which proteins, and subsequently which cells are made, depends on gene expression and regulation, i.e. “the way each cell deploys its genome.”¹

Epigenetic data arises from “the study of heritable and stable changes in gene expression that occur through alterations in the chromosome rather than in the DNA sequence.”²

¹Ralston and Shaw [2008]

²Al-Aboud et al. [2023]



commonfund.nih.gov

The image above shows quite simply the basics of genetic structures. Cell nuclei contain 23 chromosomes containing our DNA that dictate cellular function. Several more complex processes are involved during cell replication such as DNA transcription and translation in order to make proteins. A key takeaway in coming closer to understanding gene expression is that Chromatin is a complex structure made up of DNA wound around histone proteins, with some segments of DNA being accessible/inaccessible to further processes that determine gene expression, such as transcription factors³. Euchromatin refers to the accessible state, while Heterochromatin refers to a chromatin state in which DNA cannot be transcribed (inaccessible).⁴ Changes to portions of these structures affect which genes can be expressed and at what levels. There are many different epigenetic modifications that affect chromatin accessibility.

Some common epigenetic modifications include:

1. **DNA Methylation:** Addition of methyl groups to DNA, affecting gene expression regulation⁵.
2. **Histone Modifications:** Chemical changes to histone proteins that DNA wraps around, including acetylation, methylation, or phosphorylation. These changes influence chromatin structure and gene accessibility.⁶
3. **Chromatin Accessibility:** Regions of open chromatin that are accessi-

³tf2 [2014]

⁴Shahid et al. [2023]

⁵Al-Aboud et al. [2023]

⁶T. [2007]

ble to transcription factors (special types of proteins that bind to DNA sequences and regulate gene expression) further dictate which regions of DNA can be expressed⁷.

In studying gene expression and epigenetic modifications, we aim to more closely understand the biological mechanisms that regulate development, disease, and how cells respond to epigenetic factors. For example, certain histone modifications have been understood to regulate the activation or repression of genes that control whether a stem cell becomes a muscle cell, nerve cell, or another specialized type.⁸

1.1.1 What Does DNA Look Like?

As illustrated in the image above, DNA is structured as a double helix, with two complementary strands intertwined to form the characteristic helical shape. DNA consists of an extremely long sequence composed of four types of nucleotides: Adenine (A), Cytosine (C), Thymine (T), and Guanine (G).

According to the National Cancer Institute (USA), nucleotides within the DNA double helix form complementary pairs—Adenine pairs with Thymine, and Guanine pairs with Cytosine⁹. These pairs are commonly referred to as base pairs (bps). For example, if one strand of the double helix has the sequence “ATCGG”, the complementary strand will have the sequence “TAGCC”.

Genes are sequences of DNA located at specific positions on chromosomes and can vary in length. Each gene encodes information necessary for producing proteins or RNA molecules, which are essential for the structure, function, and regulation of an organism¹⁰. The complete set of genetic material in an organism is known as its genome.

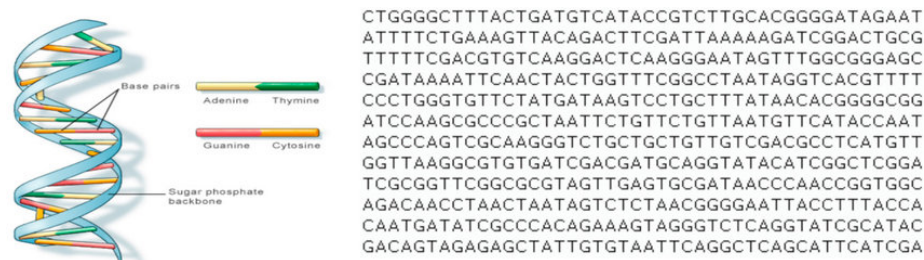


Image highlighting part of a dna sequence and base pairs.¹¹

⁷Kappelmann-Fenzl [2021]

⁸Völker-Albert and et al [2020]

⁹Board

¹⁰Board

¹¹Kanani and Padole [2020]

1.1.2 Common Epigenetic Sequencing Techniques:

1. **ATAC-Seq** (Assay for Transposase-Accessible Chromatin with Sequencing): Measures chromatin accessibility to identify open regions of the genome where transcription factors can bind.
Downstream Output: Peaks indicating accessible chromatin regions.
2. **ChIP-Seq** (Chromatin Immunoprecipitation Sequencing): Used to identify DNA regions bound by specific proteins (e.g., transcription factors, histones with specific modifications).
Downstream Output: Peaks indicating binding sites or modification locations.

1.2 What does epigenetic data look like?

Epigenetic data can be represented in various forms, depending on the type of modification being studied and the methods used to gather the data. **ATAC-Seq** and **ChIP-Seq** are the common methods I will focus on, but there are others that may produce different forms of data, such as WGS (whole-genome sequencing) which produces nucleotide sequencing data, or Bisulfite conversion of DNA producing data on methylation levels across the genome.

1.2.1 Representing epigenetic data

Epigenetic data originates from sequencing methods such as ATAC-Seq or ChIP-Seq experiments. The initial experiments produce raw sequencing reads (fragments), which are then aligned to a reference genome. By aligning these sequences, we can aggregate the reads into regions where they ‘pile up’ to form peaks, indicating areas of significant biological activity or modification. This can be done per base across the genome, or per gene. Additionally, we could also identify variations in single DNA base pairs called SNPs (single nucleotide polymorphisms)¹² by examining mismatches where a read’s base differs from the reference genome’s base. This mismatched information can be recorded in a table showing the position, type of mismatch, and the number of reads supporting each mismatch.¹³

¹²Board

¹³Akalın [2020]

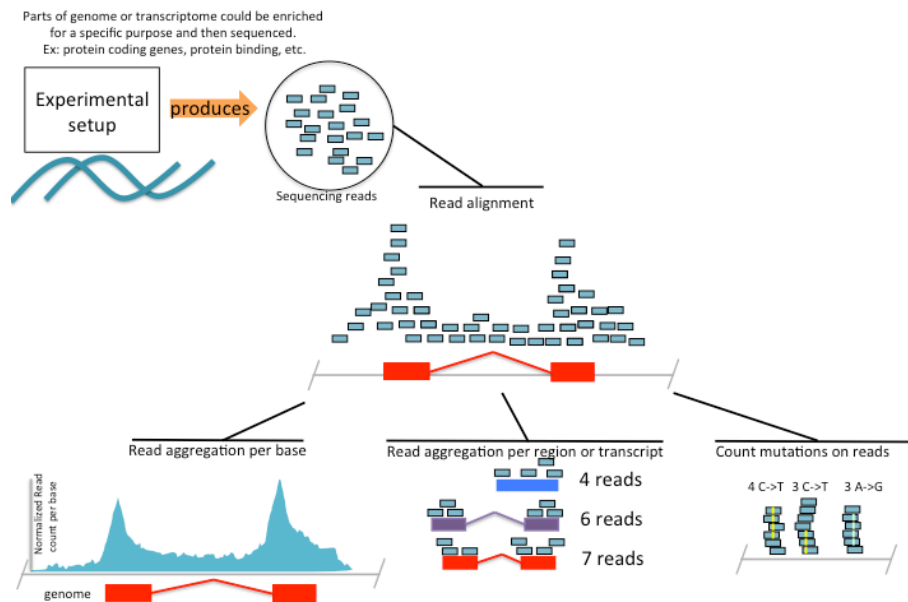


Image showing the sequencing pipeline from high-throughput sequencing methods, and how reads ‘pile up’ to form peaks¹⁴.

1. **Raw Sequence Reads:** These are the basic output of sequencing experiments, such as those from ChIP-Seq or ATAC-Seq. Reads are processed and aligned to a reference genome before undergoing peak calling.

Lets look at what a few lines of raw sequence read data consists of: The data is taken from Encode Experiment ENCSR817LUF (chIP-Seq). The accession ID of the raw sequence read data is ENCFF397NRK. Genomic data comes in many file formats. This specific raw sequence read data is a compressed FASTQ file.

Note: The script I used involved streaming the data directly from a URL using the requests library. While files containing genomic data are generally quite large, for computational efficiency it is recommended that data be downloaded locally.

```
## ID: B091JABXX110402:1:2204:12975:184709
## Sequence: GTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGG
## Quality Scores: [31, 30, 30, 32, 36, 37, 36, 32, 33, 32, 35, 36, 37, 33, 33, 34, 37, 37, 36, 3
##
## ID: B091JABXX110402:1:2205:18641:8399
## Sequence: GGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAG
## Quality Scores: [22, 27, 31, 30, 30, 33, 31, 32, 31, 31, 24, 36, 37, 36, 33, 34, 33, 37, 37, 3
##
```

¹⁴Akalin [2020]

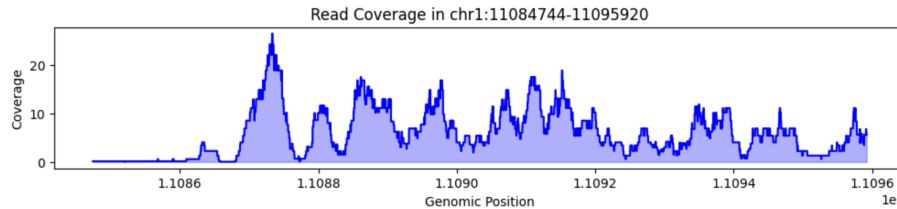
```
## ID: B091JABXX110402:1:1207:12202:100922
## Sequence: AGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTT
## Quality Scores: [33, 33, 36, 37, 31, 34, 34, 36, 37, 36, 29, 33, 35, 36, 39, 37, 32
```

As you can see, each data entry is a DNA sequence (read). While I'm only showing the first three entries, for each experiment there are millions of sequencing reads. The quality scores indicate the confidence of each base call in the sequence. Higher scores suggest higher confidence. The scores are in Phred format, where a score of 20 corresponds to a 99% base call accuracy, 30 corresponds to 99.9%, 40 corresponds to 99.99%, and so on.¹⁵

2. Peak Calling: oA method used to identify regions in the genome where there is significant enrichment of sequencing reads. This indicates the presence of DNA-protein interactions (e.g., transcription factor binding sites or accessible chromatin regions). oPeaks represent areas where epigenetic marks or chromatin accessibility are concentrated.

A common peak calling algorithm is MACS2. Essentially, aligned sequencing reads are aggregated into regions where they 'pile up' to form peaks as a read count per base. The outputs typically include signal p-values or fold change over control values, representing the expectation of a peak. Signal p-values are negative log transformed resulting in $-\log_{10}$ signal p-values. While MACS2 is traditionally used in ChIP-seq experiments, it is also applied to ATAC-seq to identify significant peaks and assess their enrichment.¹⁶ In ChIP-seq, broad peaks often represent histone modifications covering entire gene bodies, while narrow peaks are indicative of transcription factor binding sites. In ATAC-seq, the peaks primarily reflect regions of open chromatin.¹⁷ Peak calling reduces background noise and utilises signal smoothing techniques to more accurately detect peaks.

Representing Peaks: o **P-value or Fold-change:** P-value: Indicates the statistical significance of the peak, helping to distinguish true peaks from background noise. Fold-change: Represents the difference in read density between treated and control samples, indicating the strength of the signal.



This image shows the signal p-value coverage over a small region (11,176bps) in chromosome one from Encode Experiment ENCSR817LUF (The same

¹⁵Green and Ewing

¹⁶Mistry et al. [2022]

¹⁷Wilbanks and Facciotti [2010]

chIP-Seq experiment we saw the raw sequence reads from). For further context, experiment ENCSR817LUF targets the H3K36me3 histone modification in brain tissue. The experiment aims to map the locations where the H3K36me3 histone modification is present along the genome. Therefore the peaks represent regions of the genome where the the H3K36me3 histone modification is enriched compared to the background or control. The accession ID of the signal p-value data is ENCFF601VTB. Genomic data comes in many file formats. This specific signal p-value data is a bigWig file.

- o **Types of Peaks:** Categorical Peaks: Simple yes/no indication of a peak's presence. Continuous Peaks: More nuanced representation that includes the intensity or enrichment level of the peak, often visualized as a signal track. Thresholded/Pseudoreplicated Peaks: Usually categorical, these peaks are of high confidence regions from multiple replicates (experiments) or pseudoreplicates (artificial data splits), to ensure reliability and reproducibility.

Example Data Pipeline

encodeproject.org

This is an example data pipeline from Encode Experiment ENCSR817LUF, the same chIP-Seq experiment we saw the raw sequence reads, and signal p-value coverage from. The yellow bubbles represent downloadable data sets of different types, while the blue boxes represent step types (e.g. peak calling). In the left column are multiple data sets of raw sequence reads, which then undergo data quality steps before being aligned (first blue box) to the reference human genome GRCh38 (denoted by ENCFF110MCL below the reads). The next steps include Peak calling (categorical peaks) and signal generation (continuous peaks) to produce the data we normally use in our machine learning models. This data pipeline process aids in normalisation, noise reduction, and dimensionality reduction of the data.

1.3 Sources of epigenetic data

There are numerous public data banks which contain genomic datasets ready to be downloaded. Blueprint Blueprint's genomic datasets are focused on gene expression in healthy and diseased cells mostly relating to haematopoietic cells (cells which develop into different types of blood cells).

Roadmap The National Institute of Health's Roadmap Epigenomics Project contains sample datasets from multiple experiments as well as reference and mapping datasets.

Encode The Encode Project contains a large amount of publicly available genomic data easily filtered and downloaded. The genomic data used in this markdown book is sourced from Encode.

The largest genomic data bank is the UK Biobank, however they require that you apply for access to their datasets.

1.4 UCSC’S Genome Browser

The UCSC Genome Browser is a tool that allows for the visualisation of many sets of genomic data, especially bigWig files. It offers a large collection of genome assemblies, annotation tracks, and functional data, enabling users to examine gene structures, regulatory elements, and genetic variations. Whether you’re investigating gene functions, exploring genetic variants, or studying comparative genomics, the UCSC Genome Browser can be a good resource to begin understanding complex genomic information. It is also possible to load and visualise genomic data from other sources such as Encode. While the visualisations are extensive, as you can explore below, the browser can be quite overwhelming for first time users.

The following is an example of what the same chIP-Seq data targeting the H3K36me3 histone modification in brain tissue looks like using UCSC’s Genome Browser. The pseudoreplicated peaks represent categorically, the significant locations along the genome where the H3K36me3 histone modification is present.

UCSC Genome Browser

The following is an example of ATAC-Seq data from an experiment on T-helper 17 cells (a type of immune system cell). Recall that the ATAC-Seq method aims to find chromatin regions that are accessible for transcription factor binding. The p-value and fold change graphs show continuous peaks, while the IDR thresholded peaks and pseudoreplicated peaks represent the significant locations of accessible chromatin along the genome.

UCSC Genome Browser

Chapter 2

Pre-processing of bigWig files

In Machine Learning tasks, data used to train models are usually sourced from downstream analysis such as p-values and fold change data for several reasons. These include being aligned to a reference genome, and being of higher quality due to the peak calling process. However, sequencing errors and noise can still be present. When using $-\log_{10}$ p-value and fold change data, base pair averaging is commonly used to further reduce this noise by changing the resolution of the data through aggregation. This is additionally done to prepare the data for the window/region size we intend to examine and predict.¹

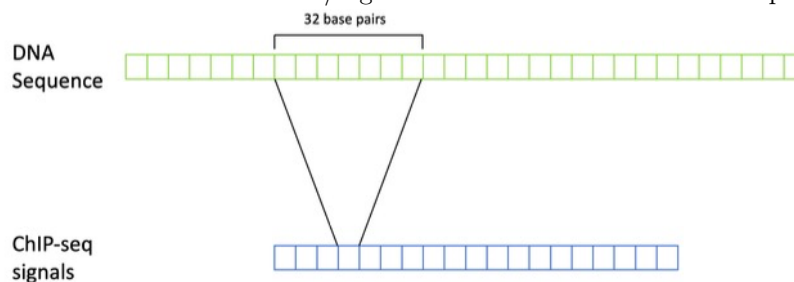


Figure showing a 32 base pair resolution following base pair averaging of ChIP-Seq data²

By applying base pair averaging over a window (e.g. 32bp), you might see that while individual counts vary, there is a broader region where the average read coverage is consistently high, indicating a potential region of interest. The intended effect of base pair averaging is further reducing noise, signal smoothing

¹RY and et al [2022]

²Patel [2024]

as well as reducing computational costs due to binning. You generally do not want to predict at a base pair level as there is not much biological relevance to one base pair as opposed to a region (peaks, binding sites, open chromatin regions etc).

Figure showing some data preprocessing choices in training convolutional neural networks to predict peaks.³

An important point to introduce regarding the input size / input window / input sequence length is this is the genomic ‘context’ your model will use to make predictions. For example, a model trained on an input window of 2048bp could be used to predict the centre 32bp (output window) at a 32bp resolution (prediction resolution). This prediction will be based on the genomic information the model learns from the 2048bp region. Some current models like Basenji and Enformer have input windows in the kbps with prediction resolutions as small as 128bp⁴. The motivation being that the model has a larger ‘context’ in which to learn distal effects/relationships to make predictions. However, the receptive fields of these models need to be large in order to capture these effects. Enformer utilises transformer architecture, while Basenji uses dilated convolutional layers to boost their receptive fields. As you will see in the tutorials, you can re-average your data to a higher base pair average before using it in machine learning models. This can be implemented to decrease dimensionality, reduce computational intensity and tailor the model to understanding regions of a certain scale.

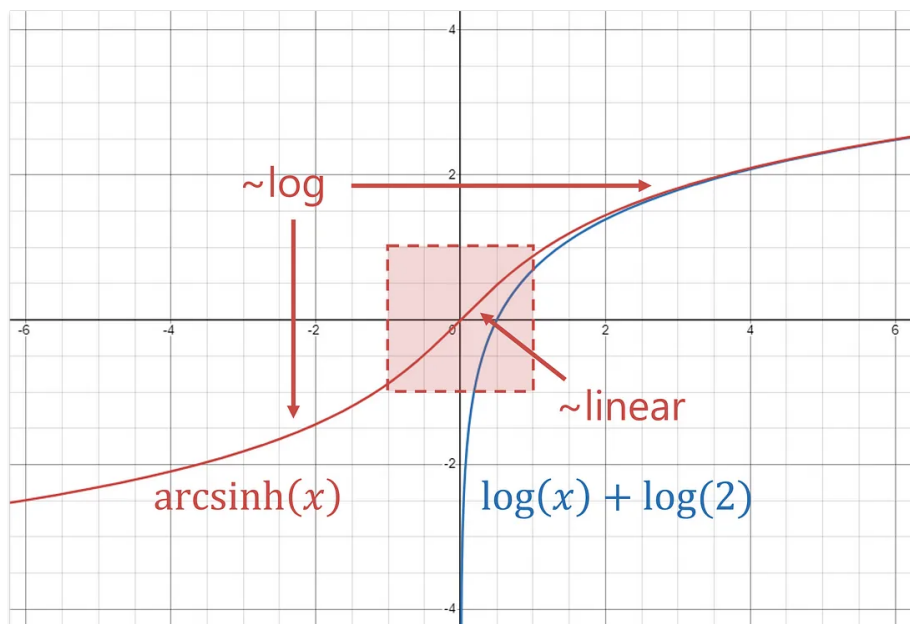
2.0.1 Transformations to stop extreme p-values

When utilising genomic data which incorporates p-values, it is important to consider and deal with extreme p-values. One way this is done is through using an Arcsinh-transformation (inverse hyperbolic sine). $\text{arcsinh}(x) = \ln(x + \sqrt{x^2 + 1})$

The arcsinh-transformation as a logarithmic function helps in reducing the significance of outliers and sequencing errors while maintaining variance by compressing the range of the data. This transformation can be used in the data preprocessing stage. The graph below visualises how the transformation works. While extreme values are transformed logarithmically, the smaller values are barely transformed as the function for smaller values is more linear in nature.

³Toneyan et al. [2022]

⁴Žiga Avsec et al. [2021]



Plot of Arcsinh Transformation compared to a log function, made with Desmos.
Source

In the genomic context, beware of normalising using sample statistics such as the mean or standard deviation, before splitting your data to avoid leaky pre-processing effects. The arcsinh function does not depend on sample statistics in normalising the data.

BigWig files containing signal p-value or fold change data can be quite tricky to deal with. However, libraries such as pyBigWig enable easier access of data. In order to understand how to handle the data pre-processing stage, I have created a jupyter notebook tutorial on Google Colab. The tutorial begins using UCSC's programs to quickly understand the genomic data within BigWigs, before using the pyBigWig library to simply extract BigWig data.

The final part of the tutorial uses the pyBigWig library to load, filter, and split BigWig data into training, validation, and test sets. The data consists of signal p-values from ChIP-seq experiments, processed using the MACS2 tool. We will re-average these signals to a resolution of 32 base pairs. Additionally, we will implement threshold-based filtering and consistent data splits to understand how to ready data for a model.

Tutorial 1: Loading and Pre-Processing Data from bigWigs (interactive)

Tutorial 1: Loading and Pre-Processing Data from bigWigs (nbviewer)

2.1 Data loaders and simplifying pre-processing

Data loaders are scripts/functions to load batches of data into your model. They are crucial in machine learning because they simplify how data is fed into models, making the whole process smoother and more efficient. This becomes especially important with the large datasets used in genomic studies, where managing and processing data manually would be cumbersome. By automating these tasks, data loaders help ensure that data is processed efficiently, allowing for faster and more effective model training. While there are existing github repositories with genomic data loaders, such as “Kipoi Dataloader”, and “Dataloader for BigWig files”.⁵ Depending on the data used and model you build, they won’t cover all of the use cases. When building one yourself, the PyTorch library has its own dataset and dataloader modules, which include creating custom datasets.

⁵Retel et al. [2024]

Part II

Training models with DNA input

Chapter 3

Loss functions

When selecting the optimal loss function for your machine learning model in genomics, the decision should be informed by the nature of the problem and the specific type of data you’re working with. The principles for choosing loss functions in genomics are similar to those in other machine learning contexts. For regression based tasks, while Mean Squared Error (mse) loss functions have been used, models utilising data and making predictions associated with reads or counts use a Poisson loss function. Another two common loss functions include Binary Cross-Entropy loss and Categorical Cross-Entropy loss, when dealing with classification type predictions.¹

Mean Squared Error (MSE): • Use Case: Regression problems where the goal is to predict continuous values, such as gene expression or coverage levels. Example: DeepImpute, a deep neural network-based imputation algorithm that allows for accurate imputation of single-cell RNA-seq data. The model is used to estimate missing or low-quality gene expression values in single-cell RNA sequencing datasets. It uses a “weighted mean squared error (MSE) loss function that gives higher weights to genes with higher expression values. This emphasizes accuracy on high confidence values and avoids over penalizing genes with extremely low values (e.g., zeros)”.²

Poisson Loss: • Use Case: Count data where the number of events (e.g., read counts in chIP-seq data) follows a Poisson distribution. Example: A deep learning architecture called Enformer was used to predict gene expression more accurately in 2021 by integrating long range interactions within the genome. It utilises a poisson negative log-likelihood loss function resulting in a model that was able to integrate information from up to 100 kilobases away.³

Cross-Entropy Loss: • Use Case: Classification problems where the goal

¹Patterson and Gibson [2017]

²Arisdakessian et al. [2019]

³Žiga Avsec et al. [2021]

is to predict binary outcomes. Example: A convolutional neural network was employed to create a software pipeline called CNN-Peaks, designed to categorically detect ChIP-Seq peaks without relying on traditional peak calling methods or manual inspection. The model utilizes a binary cross-entropy loss function, which was weighted to account for the scarcity of peaks in the data.⁴

Categorical Cross-Entropy Loss: • Use Case: Multi-class classification problems. Example: Researchers developed a combined model that integrates cell-free DNA (cfDNA) methylation profile data with ATAC-seq data to enhance cancer detection and tissue-of-origin localization. This approach combines both epigenomic and chromatin accessibility information to improve the accuracy of identifying the specific tissue or organ from which a cancerous signal originates. The model employs a categorical cross-entropy loss function within each component to optimize tissue-of-origin localization, allowing it to effectively determine the most likely source of the cancerous signal.⁵

In the case of this tutorial and running models to predict continuous coverage values from bigwig p-value datasets, I have opted to use a Poisson loss function as the data represents read counts. It is important to remember that “loss functions can penalize the shapes or the magnitudes (for example, the mean squared error (MSE))”⁶ when optimising.

⁴Oh et al. [2020]

⁵Bae et al. [2017]

⁶Toneyan et al. [2022]

Chapter 4

Training tricks

4.1 Reverse Complements and Sequence Shifts

Reverse Complements

As explained in Part 1, DNA has a double helix structure. When we one-hot encode a segment of DNA in our models using a reference genome, we typically represent only one strand of the double helix. The complement of this strand is the opposite strand, where Adenine (A) pairs with Thymine (T), and Cytosine (C) pairs with Guanine (G). The reverse complement of a DNA strand is obtained by first taking its complement and then reading it in the reverse direction.

This figure shows the reverse complement of a DNA sequence¹

Training on DNA sequences and augmenting the data with their reverse complements has been shown to improve model accuracy, prediction, and interpretability in DNA sequence-related models. This approach involves “treating the reverse complement DNA sequence as another sample”². By incorporating reverse complements, the model is exposed to a wider variety of sequence patterns, which helps reduce overfitting and enhances generalization. As a result, models become better at recognizing patterns regardless of strand orientation. Although the logic to obtain the reverse complement of a DNA strand is straightforward, the Bio.Seq module from the Biopython library provides a simple way to do this. Augmenting your dataset with reverse complements is usually done to training sets, but can be applied to validation and test sets as well.

```
from Bio.Seq import Seq
```

```
# Example DNA sequence
```

¹Youens-Clark [2021]

²Cao and Zhang [2019]

```

dna_sequence = Seq("ATGCGTAC")

# Generate the reverse complement
reverse_complement = dna_sequence.reverse_complement()

print("Original Sequence: ", dna_sequence)

## Original Sequence:  ATGCGTAC

print("Reverse Complement: ", reverse_complement)

## Reverse Complement:  GTACGCAT

```

Sequence Shifts

Training on small, random sequence shifts up and downstream by shifting the genomic coordinates of the input sequence is also known as jitter. Jittering adds diversity to the training data by creating slightly different versions of the same sequence. This allows models to be less sensitive to the exact positioning of features, making them more robust to variations in the data. It also allows models to generalise better to unseen data where sites may not always be perfectly aligned. A variation of jittering, called flanking “extends DNA sequences from its midpoint by X base pairs and takes the left, middle and right input windows of the extended sequence as training samples with the same labels, tripling the size of training set.”³

Implementing data augmentations using reverse complements and sequence shifts can be approached in different ways. Similar to the ‘flanking’ example, you can either expand your dataset by adding additional augmented data points or apply a random augmentation strategy, where only some data points are randomly augmented while keeping the total number of points in your dataset unchanged. Data augmentations are usually applied only to training sets, however in the context of computer vision “many research reports have shown the effectiveness of augmenting data at test-time as well”.⁴ When implementing augmentations like reverse complements and sequence shifts, these are typically applied after splitting your data into training, validation, and test sets. When applying sequence shifting, it’s logical to shift the interval before retrieving the nucleotide sequence from the reference genome. The reverse complement should be applied after retrieving the nucleotide sequence but before one-hot encoding it. If you’re using the BioPython library, this works well since BioPython’s reverse complement function operates on string inputs.

In the genomics context, a paper on evaluating deep learning for predicting epigenomic profiles used two convolutional neural networks, Basenji and BPNet, trained on ATAC-seq data, to predict coverage values as a regression. They found that convolutional models trained with augmentations (reverse

³Cao and Zhang [2019]

⁴Shorten and Khoshgoftaar [2019]

complement and sequence shifts), “yielded improved robustness, especially when trained on peak-centered data (BPNet). On the other hand, models that were trained on coverage-threshold data (Basenji) already benefited from the randomly-centered profiles.”⁵ Additionally, while they initially “used a MSE and multinomial NLL loss for BPNet, [they] found that optimization using Poisson NLL yielded better performance.”⁶ This finding is another motivation of using a poisson loss function in subsequent tutorials.

4.2 Hyper-parameter optimisation

Which learning rates are commonly used? How many epochs are typically used to train on?

While the learning rate and number of epochs differ by model and study, based on some of the research cited so far, common learning rates are in the range $1e-4^7$ to $1e-3^8$. Additionally, some studies apply learning rate decay if the loss function shows no improvement over time⁹ while others lower the learning rate for fine tuning.¹⁰

The number of epochs used to train on differs by quite a margin. In training a convolution neural network to explore the effects of genomic data augmentation, 30 epochs were used.¹¹ DeepImpute which constructs multiple sub-neural networks for genotype imputation, trains on a maximum of 500 epochs, while the study involving the Basenji and BPNet models were trained on a maximum of 100 epochs. The clear strategy for these larger models involve the use of early stopping if no improvements are evident after 5-10 epochs.

When hyperparameter optimising, the consensus for achieving the best model performance is to train with a high number of epochs to enable the model to confidently learn features as they apply to labels, starting with a high learning rate, and decreasing over time using a learning rate scheduler. Interestingly, a study on binary peak detection using CNNs on ChIP-Seq data manually tuned their model’s hyperparameters and found that little changes in performance results¹². This highlights the challenges of hyperparameter tuning with larger models, where manually fine-tuning is not ideal. How can hyperparameter tuning on these larger models be done in practice?

Raytune

Raytune “is a Python library for experiment execution and hyperparameter

⁵Toneyan et al. [2022]

⁶Toneyan et al. [2022]

⁷[Arisdakessian et al., 2019], [Žiga Avsec et al., 2021]

⁸[Cao and Zhang, 2019], [Toneyan et al., 2022]

⁹Toneyan et al. [2022]

¹⁰Žiga Avsec et al. [2021]

¹¹Cao and Zhang [2019]

¹²Oh et al. [2020]

tuning at any scale”. It aids in leveraging state of the art hyperparameter optimisation algorithms while simplifying scaling for larger models. Raytune hyperparameter searching can also be scaled to cloud based clusters without the need for large changes in code structure. Additionally, it supports several machine learning frameworks such as pyTorch and TensorFlow¹³. One of the strongest current hyperparameter optimisation algorithms is the Asynchronous Successive Halving Algorithm or **ASHA**. Asha “exploits parallelism and aggressive early-stopping to tackle large-scale hyperparameter optimization problems”¹⁴, allowing for faster optimisation and applicability to the larger models common in genomics. A study on predicting the impact of sequence motifs on gene regulation utilised Raytune and the ASHA algorithm to successfully optimise their model’s hyperparameters¹⁵.

In the genomic context, as a result of complex models using large genomic datasets, hyperparameter tuning using a brute force approach is untenable. Utilising existing libraries such as Raytune and incorporating asynchronous algorithms such as ASHA, has the potential to pave the way forward in improving model performance without unreasonable computational costs.

¹³Liaw et al. [2018]

¹⁴Li [2018]

¹⁵Hepkema et al. [2023]

Chapter 5

Reproducibility of machine learning models

The ability to reproduce machine learning models and results is of extreme importance in the genomic context. According to an article on reproducibility standards for machine learning in the life sciences, the bronze standard for reproducibility includes access to the original data, trained model and source code. The silver standard incorporates computational environment considerations as well as inherent non-determinism, while the gold standard requires automation such that the experiment is reproducible with a single command.¹

Seeding

Seeding runs and setting random number seeds aids in eliminating the inherent non-determinism of creating data splits and training models.

```
# Seeding to ensure reproducibility
import random
import numpy as np
import torch

seed = 42 # or any other integer
random.seed(seed)
np.random.seed(seed)
torch.manual_seed(seed) #CPU
torch.cuda.manual_seed(seed) #GPU
torch.backends.cudnn.deterministic = True # To ensure deterministic behavior
torch.backends.cudnn.benchmark = False # Ensures reproducibility
```

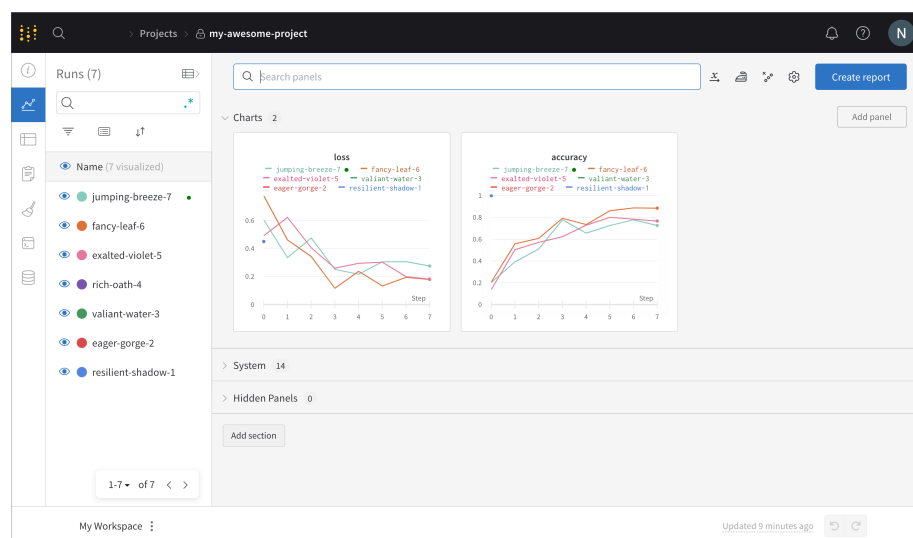
This seeding example shows a seed being used for python's built in random

¹Heil et al. [2021]

library, numpy's random library, and the PyTorch library for both CPUs and GPUs. Additionally, PyTorch's CuDNN library accelerates computations on GPUs. By setting CuDNN to deterministic mode and disabling benchmarking, you ensure that the same algorithm is used in each run. This approach reduces variability due to hardware differences at the cost of computational efficiency.

Dashboarding

Dashboarding can be used to keep track of model performance as well as model and training parameters by visualising logged data. Dashboards such as Weights and Biases (WandB) can be leveraged to keep track of model runs, hyperparameters and even trained models and data through uploading them as artifacts.



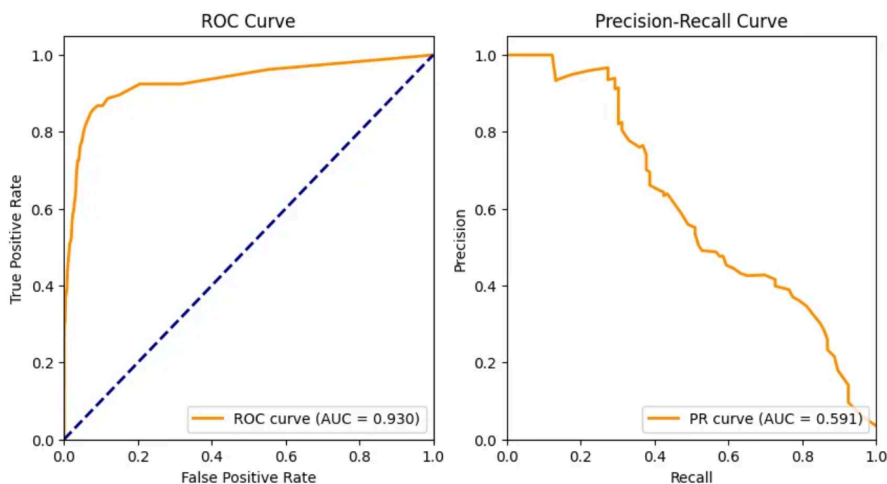
Source: WandB

Real time updates on metrics are important in the genomic context when models can take extended periods of time to train. Furthermore, downloading and sharing specific models, additional artifacts and training logs with information on environment settings becomes easier when utilising dashboards. While not explicitly ensuring or enabling reproducibility, dashboarding helps provide transparency.

Chapter 6

Testing

Testing and evaluating genomic models is similar to other machine learning contexts in that it involves assessing model performance using metrics such as accuracy, precision, recall, and F1-scores. However, the genomic context often requires additional considerations due to the complexity and high-dimensionality of genomic data. There are several pitfalls which when fallen into, inflate model performance and make test metrics untrustworthy. Genomic data can be extremely imbalanced, as we will find out in subsequent tutorials. In these unbalanced classes scenarios, the choice of test metric usually between the area under the receiver operating characteristic (auROC) and the area under the precision-recall curve (auPRC), as only predicting the majority class will result in high accuracy for categorical models.



Source: BlogPost

In genomic situations with extreme imbalance in data, plotting ROC and precision-recall curves and calculating the auROC and auPRC is beneficial to properly test your model’s performance. AuROC and auPRC both provide insight into the model’s ability to distinguish between classes and are both functions of recall (true positive rate). “In many genomics problems, high recall can be achieved at a very low [false positive rate] owing to the large number of negatives in the test set, making it easy to obtain a high auROC even when false positives vastly outnumber true positives”¹. However, auPRC offers a more focused evaluation as the absolute number of false positives is taken into account in precision versus the false positive rate. Therefore, a model trained on imbalanced data that predicts many false positives could have a high auROC and a low auPRC. In cases where we are interested in a minority of positive classes, research suggests using the auPRC as a measure of performance².

In regression settings, classic metrics such as the mean squared error, and mean absolute error are used alongside measures of variance (R^2) and correlation (pearson correlation). It is important to note that differences in the order of magnitude of multiple sets of predictions can affect MSE metrics when evaluated together. For example, if a multi-task model has two prediction sets where one set “has values entirely between 1,000 and 10,000 and the other [set] has values between 0.01 and 0.1, then evaluating the model simply using mean squared error (MSE) across the two tasks will largely ignore the second task.”³

In the genomic context one can also perform marginalisation experiments to ensure models are recognizing biologically relevant patterns. “Marginalisation is a method that requires summing over the possible values of one variable to determine the marginal contribution of another”⁴. Given a model trained on DNA sequences, performing a marginalisation experiment using sequence motifs is possible. “Motifs are short, recurring patterns in DNA that are presumed to have a biological function”⁵. Marginalising with motifs involves comparing a model’s predictions before and after the motif is inserted into a test/prediction set. The difference in predictions allows us to quantify the influence of a motif on a model’s predictions and understand how well our model recognises biological patterns. More specific types of marginalisation experiments include variant effect prediction, which aims to understand the effect of specific genetic variants in sequences on a model’s predictions.

¹Whalen et al. [2022]

²Whalen et al. [2022]

³Whalen et al. [2022]

⁴Brooks-Bartlett [2018]

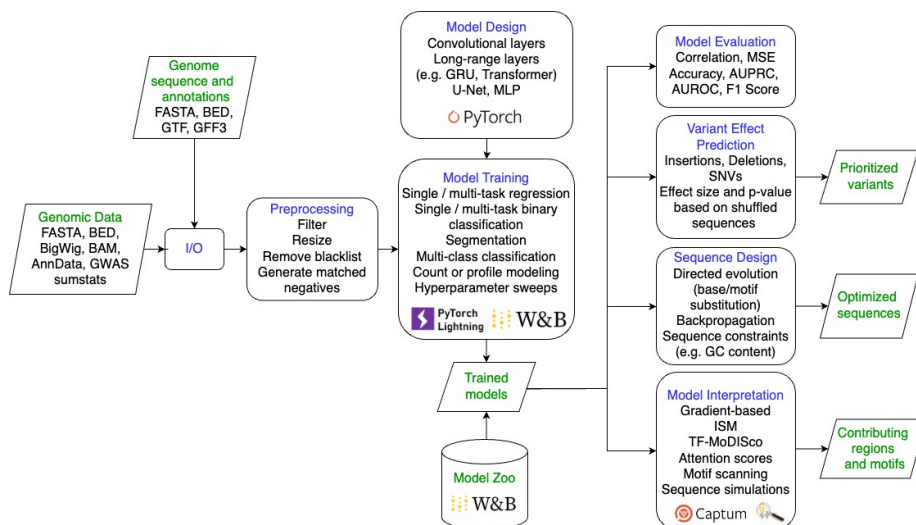
⁵D’haeseleer [2006]

Chapter 7

Software libraries for model building

7.1 gReLU

“gReLU is a Python library to train, interpret, and apply deep learning models to DNA sequences”. The gRelu library contains a model zoo allowing for easy access to several models such as Borzoi, Enformer, or a dilated convolutional model based on the BPnet model architecture. Some of these models can be imported pre-trained. Additionally, simpler base models and convolutional neural networks are also available. On top of access to already built models, the software library allows for designing your own models.

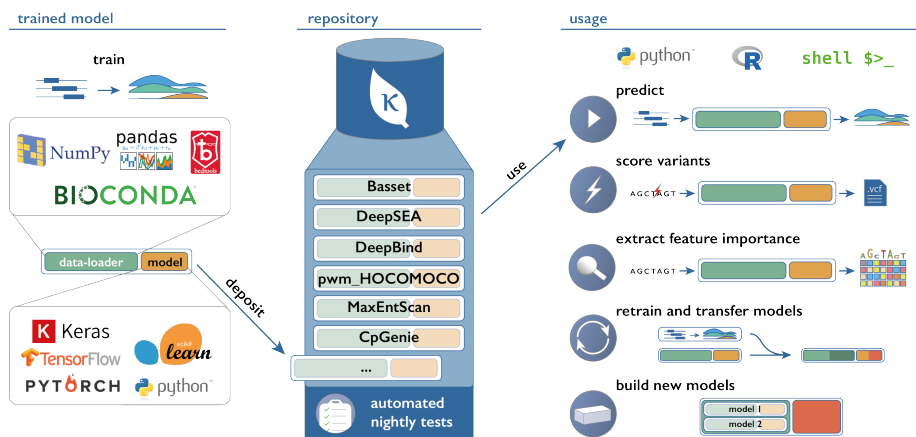


Source: gRelu

Their documentation contains all their available functions and includes tutorials on using gRelu.

7.2 Kipoi

Kipoi is a repository of ready-to-use trained models for genomics. Referring back to the reproducibility of machine learning models, Kipoi contains 2206 different gold standard models, available to be downloaded and tested in a few lines of code.



Source: Kipoi

Similarly to gRelu, they include several use case tutorials and a model zoo. Downloaded models can also be built upon to conduct further research as links to the github source code of each model are provided.

Part III

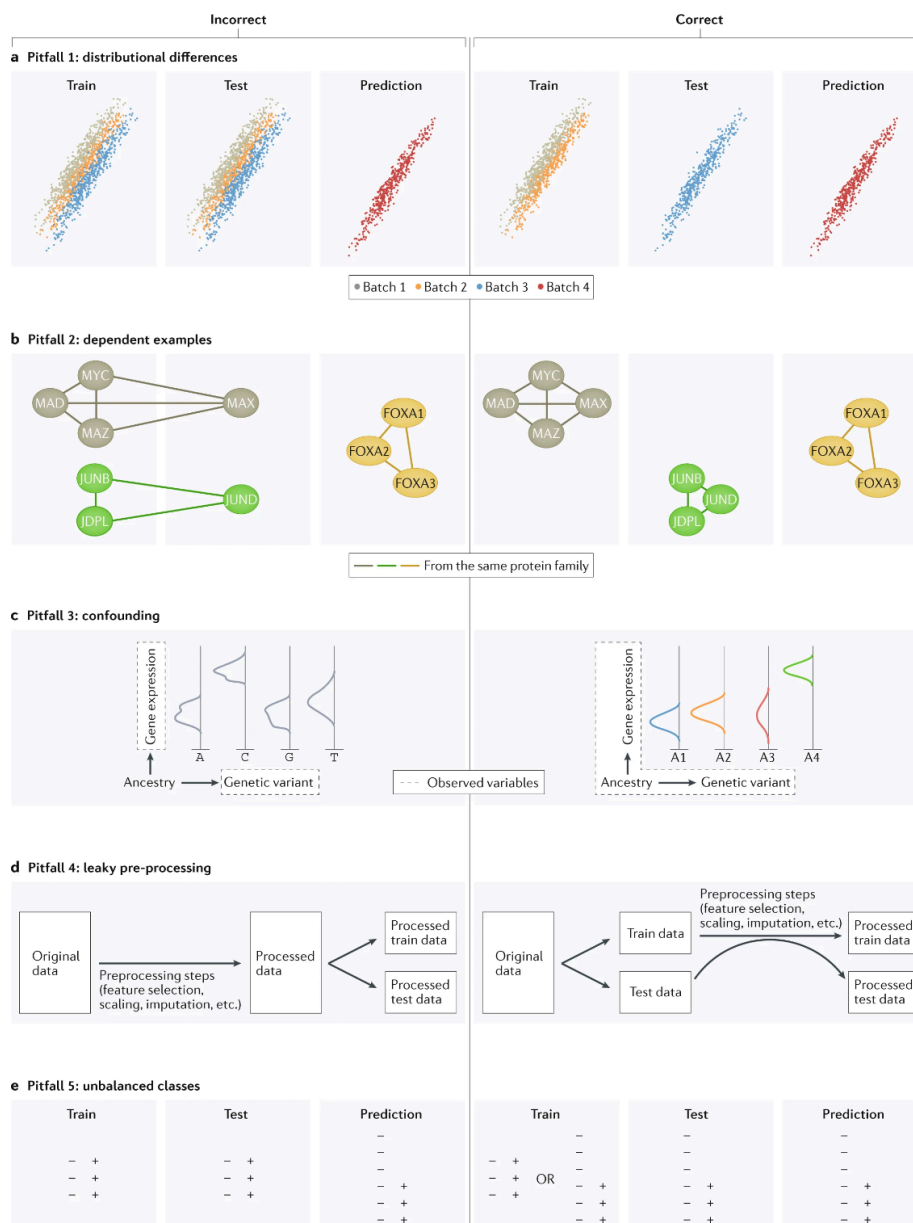
ML pitfalls in genomics

Chapter 8

Pitfalls overview

Applying machine learning in the field of genomics comes with its own challenges. Genomic data is highly complex, featuring high dimensionality, heterogeneity, and noise. It is important to consider the assumptions models make and whether those assumptions hold true within the data. According to research, available here from Whalen et. al., on the pitfalls in machine learning, the most common errors include not taking into account distributional differences between contexts or batches, dependencies within the data, confounding effects distorting true relationships, unbalanced classes and leaking information between datasets.¹ These pitfalls are interconnected where falling into one likely means falling into several. For example leaky preprocessing practices may result in introducing confounding effects. Lets examine the pitfalls mentioned in the research above and methods to mitigate these effects in more detail.

¹Whalen et al. [2022]



Source: Nature Reviews Genetics

Distributional differences

Whalen et. al. discuss the issues that arise when distributions between experiments or batches differ. The example given for distributional differences between experiments involves understanding transcription factor binding sites using experiment data from in vitro assays (out of cell) versus in vivo assays (in

cell). The context in which either experiment was conducted is different introducing unique biases. Training models in one context can perform well within the same context, but performance drops when tested/predicting in the other context. This can also be true when predicting across species. The authors note this is not inherently a bad thing as the differences in distributions can be studied to reveal biological insights. It is important to keep this in mind when attempting to predict across contexts. Distributional differences between batches can occur due to noise or other factors introduced by the specific experiment design. The key point Whalen et. al. conveys as seen in figure 2.9, is in order to train models that generalise well, test sets must be from different batches as to separate these potential distributional differences. Otherwise, test performance will be artificially inflated. Additionally, applying batch effect correction techniques such can help account for these distributional differences.

Dependent examples

Genomics contains many biological dependencies. The main issue arises when data in test sets are correlated or dependent on data in training sets or vice versa. Predictions become easier as the model learns the correlation between dependencies versus the underlying biological factors. The example given by Whalen et. al. argues that a model predicting protein interactions would have an inflated test performance learning dependencies between proteins present in both the training and test set versus novel proteins. Another genomic example of problematic dependency comes from a research paper focused on understanding the pitfalls of neural networks predicting across cell types². They found that several models had been evaluated on test sets comprising of a cell type independent of training, but not independent of chromosomes, leading to inflated performances. That is, gene expression predictions were evaluated on different cells but on the same chromosomes. The inflated performances were due to the general expression levels of test chromosomes being similar across cells. The correct way to evaluate the model's performance would be to test it on a different cell type and chromosome than it was trained on.

Confounding

Confounders are additional variables that affect the variables being studied, resulting in models not capturing the correct relationships in the data.³ According to Whalen et. al. confounders in genomics can come from various factors such as unmodelled environmental factors or population structures in genome wide association studies, with the problem being models learning an association between the outcome and features that is actually driven by a confounding variable but interpreted as a biological insight. In the context of ATAC-Seq and chIP-Seq data, differences in the sequencing depth of the experiment can have a confounding effect on models if not handled. As explained in section 2.2.1, ATAC-Seq and chIP-Seq data consists of reads that are aligned to a reference

²Schreiber and et al [2020]

³Pourhoseingholi and et al [2012]

genome and aggregated. The sequencing depth of the experiment refers to on average how many times a region was sequenced⁴. A higher sequencing depth means a higher overall coverage level. If not accounted for, models using data of different sequencing depths are confounded. Similarly when predicting on a dataset with a different sequencing depth than the training set, the model would be biased. While sequencing depth as a confounder can be more easily corrected by downsampling the raw reads from the dataset with higher sequencing depth, other confounding effects can be harder to account for. However, it is possible to use statistical tools such as PEER⁵, Inter-sample Correlation Emended (ICE) and Surrogate Variable Analysis (SVA)⁶ to understand confounders in your dataset.

Leaky pre-processing

Leaky preprocessing involves information leaks between the training and test sets resulting in altered testing metrics. Pre-processing steps involving the dataset as a whole can introduce dependency. This includes standardisation techniques, supervised feature selection or principle component analysis, before splitting data into test splits. It is ideal to perform these transformations and analysis after data splits or within cross-validation. Whalen et. al. states that leaky preprocessing has been prevalent in various genomic fields, including microarray analysis, DNA methylation, gene expression studies, and more, often leading to misleading results.

Unbalanced classes

Unbalanced datasets can be common in genomics often necessitating the use of statistical methods to validate the few positive instances amid vast amounts of data. In these cases, the pitfall is a model “learns most of the target concepts of the majority class and learns target concepts from the minority class poorly or not at all.”⁷ This is particularly evident in disease related tasks where the focus is on a few disease causing genes or non-coding variants.⁸ Additionally, many experiments that implement conservative significance thresholds to determine true signals involve data imbalance, such as peak detection⁹, gene expression¹⁰, and chromatin accessibility. The extensive size of the human genome exacerbates this issue when the areas of interest are small. Researchers address this imbalance by employing balancing algorithms that oversample the negative class and undersample the majority class. For instance, in training models to predict functional peaks from ChIP-seq or chromatin accessibility data, an approach might involve using all identified peaks along with a matching number of negative regions, thus effectively undersampling the majority class. For datasets with

⁴Sims and et al [2014]

⁵Stegle and et al [2012]

⁶Listgarten and et al [2010]

⁷Haque and et al [2014]

⁸Haque and et al [2014]

⁹Oh et al. [2020]

¹⁰Žiga Avsec et al. [2021]

no such negative regions, researchers have to construct their own. While such imbalances are commonly discussed in classification, they also pose challenges in regression models predicting quantitative outcomes, where performance may be compromised in regions with sparse data, such as genomic areas or genes with low read counts in single-cell genomics studies.¹¹

In Tutorial 1, we utilised thresholding to filter our data to focus on regions with significant coverage. While there were around 300,000 bins across the three chromosomes we looked at, after thresholding our data consisted of roughly 10% or 30,000 bins. Our data does not contain any coverage values below the threshold. In the next tutorials, we will explore how a model performs with and without regions of zero signal.

More general methods for dealing with class imbalances:

- Scikit-learn’s ‘imbalance-learn’ package (Oversampling, Undersampling and Weighting) “Imbalanced-learn (imported as imblearn) is an open source, MIT-licensed library relying on scikit-learn (imported as sklearn) and provides tools when dealing with classification with imbalanced classes.”
- SMOTE (Oversampling) “a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically k=5). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.”
- ADASYN (Oversampling) “similar to SMOTE but it generates different number of samples depending on an estimate of the local distribution of the class to be oversampled.”

¹¹Whalen et al. [2022]

Part IV

Using existing models

Chapter 9

Training gReLU models and examining pitfalls

Continuing from previous tutorials, the next tutorial similarly uses chIP-Seq data from Encode Experiment ENCSR817LUF. In tutorial 2 we use this data to train a gRelu convolutional neural network in order to predict the coverage levels and examine several of the pitfalls mentioned in section 4. An additional tutorial (tutorial1.5) is included to explain how data was pre-processed for tutorial 2. Reading through gRelu's own tutorial on training their models is recommended.

[Tutorial 1.5: Preprocessing Data for gRelu Models \(interactive\)](#)

[Tutorial 1.5: Preprocessing Data for gRelu Models \(nbviewer\)](#)

[Tutorial 2: Training Models with gRelu and Examining Pitfalls \(interactive\)](#)

[Tutorial 2: Training Models with gRelu and Examining Pitfalls \(nbviewer\)](#)

Bibliography

- general transcription factor / transcription factor*. Learn Science at Scitable, 2014. URL <https://www.nature.com/scitable/definition/transcription-factor-167/>.
- Altuna Akalin. *Computational Genomics with R*. Github Pages, 2020. URL <https://compgenomr.github.io/book/>.
- Nora M. Al-Aboud, Connor Tupper, and Ishwarlal Jialal. *Genetics, Epigenetic Mechanism*. National Library of Medicine, 2023. URL <https://www.ncbi.nlm.nih.gov/books/NBK532999/#article-22137.r1>.
- Cédric Arisdakessian, Olivier Poirion, Breck Yunits, Xun Zhu, and Lana Garmire. *DeepImpute: an accurate, fast, and scalable deep neural network method to impute single-cell RNA-seq data*. Springer Nature, 2019. URL <https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1837-6>. <https://doi.org/10.1186/s13059-019-1837-6>.
- Mingyun Bae, Gyuhee Kim, Tae-Rim Lee, and et al. *Integrative modeling of tumor genomes and epigenomes for enhanced cancer diagnosis by cell-free DNA*. Nature Communications, 2017. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10085982/>. <https://doi.org/10.1038/s41467-023-37768-3>.
- PDQ® Cancer Genetics Editorial Board. *genetics-dictionary*. National Cancer Institute USA. URL <https://www.cancer.gov/publications/dictionaries/genetics-dictionary>.
- Jonny Brooks-Bartlett. *Probability concepts explained: Marginalisation*. Towards Data Science, 2018. URL <https://towardsdatascience.com/probability-concepts-explained-marginalisation-2296846344fc>.
- Zhen Cao and Shihua Zhang. *Simple tricks of convolutional neural network architectures improve DNA-protein binding prediction*. Bioinformatics, 2019. URL <https://academic.oup.com/bioinformatics/article/35/11/1837/5142724?login=false>. <https://doi.org/10.1093/bioinformatics/bty893>.
- Patrik D’haeseleer. *What are DNA sequence motifs?* Nat Biotechnol, 2006. URL <https://www.nature.com/articles/nbt0406-423>. <https://doi.org/10.1038/nbt0406-423>.

- Phil Green and Brent Ewing. *Phred - Quality Base Calling*. CodonCode Corporation. URL [https://www.phrap.com/phred/#:~:text=Phred%20is%20a%20base%2Dcalling,%22\)%20to%20each%20base%20call](https://www.phrap.com/phred/#:~:text=Phred%20is%20a%20base%2Dcalling,%22)%20to%20each%20base%20call).
- Muksitul Haque and et al. *Imbalanced Class Learning in Epigenetics*. Journal of Computational Biology, 2014. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4082351/>. PMID: 24798423.
- Benjamin Heil, Michael Hoffman, Florian Markowetz, and et al. *Reproducibility standards for machine learning in the life sciences*. 2021. URL <https://www.nature.com/articles/s41592-021-01256-7>. <https://doi.org/10.1038/s41592-021-01256-7>.
- Jacob Hepkema, Nicholas Lee, Benjamin Stewart, and et al. *Predicting the impact of sequence motifs on gene regulation using single-cell data*. National Library of Medicine, 2023. URL <https://www.ncbi.nlm.nih.gov/pmc/article/s/PMC10426127/>. <https://doi.org/10.1186/s13059-023-03021-9>.
- Pratik Kanani and Mamta Padole. *Improving Pattern Matching performance in Genome sequences using Run Length Encoding in Distributed Raspberry Pi Clustering Environment*. Procedia Computer Science, 2020. URL <https://www.sciencedirect.com/science/article/pii/S1877050920311601?via%3Dihub>. <https://doi.org/10.1016/j.procs.2020.04.179>.
- Melanie Kappelmann-Fenzl. *Design and Analysis of Epigenetics and ChIP-Sequencing Data*. Springer, 2021. URL https://doi.org/10.1007/978-3-030-62490-3_12. ISBN 978-3-030-62490-3.
- Liam Li. *A System for Massively Parallel Hyperparameter Tuning*. Conference on Machine Learning and Systems, 2018. URL <https://arxiv.org/abs/1810.05934>. <https://doi.org/10.1186/s40537-023-00838-w>.
- Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.
- Jennifer Listgarten and et al. *Correction for hidden confounders in the genetic analysis of gene expression*. Proceedings of the National Academy of Sciences of the United States of America, 2010. URL <https://www.pnas.org/doi/full/10.1073/pnas.1002425107>. <https://doi.org/10.1073/pnas.1002425107>.
- Meeta Mistry, Jihe Liu, Radhika Khetani, and et al. *Peak calling with MACS2*. Github Pages, 2022. URL https://hbctraining.github.io/Intro-to-ChIPseq-flipped/lessons/06_peak_calling_mac.html.
- Dongpin Oh, Seth Strattan, Junho Hur, and et al. *CNN-Peaks: ChIP-Seq peak detection pipeline using convolutional neural networks that imitate human visual inspection*. Springer Nature, 2020. URL <https://www.nature.com/articles/s41598-020-64655-4>. <https://doi.org/10.1038/s41598-020-64655-4>.

- Jai Patel. *Leveraging Genetic Diversity With Machine Learning*. Imperial College London, 2024.
- Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. O'Reilly Media, Inc., 2017. URL <https://www.oreilly.com/library/view/deep-learning/9781491924570/>. ISBN: 9781491914250.
- Mohamad Pourhoseingholi and et al. *How to control confounding effects by statistical analysis*. Gastroenterol Hepatol Bed Bench, 2012. URL [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4017459/#:~:text=A%20Confo under%20is%20an%20extraneous,between%20the%20variables%20under%20study](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4017459/#:~:text=A%20Confo under%20is%20an%20extraneous,between%20the%20variables%20under%20study.PMID: 24834204). PMID: 24834204.
- Amy Ralston and Kenna Shaw. *Gene Expression Regulates Cell Differentiation*. Nature Education, 2008. URL [https://www.nature.com/scitable/topicpage/gene-expression-regulates-cell-differentiation-931/#:~:text=All%20of%20 the%20cells%20within,each%20cell%20deploys%20its%20genome](https://www.nature.com/scitable/topicpage/gene-expression-regulates-cell-differentiation-931/#:~:text=All%20of%20 the%20cells%20within,each%20cell%20deploys%20its%20genome.Nature Education 1(1):127). Nature Education 1(1):127.
- Joren Sebastian Retel, Andreas Poehlmann, Josh Chiou, Andreas Steffen, and Djork-Arné Clevert. A fast machine learning dataloader for epigenetic tracks from BigWig files. *Bioinformatics*, 40(1):btad767, January 2024. ISSN 1367-4811. doi: 10.1093/bioinformatics/btad767. URL <https://doi.org/10.1093/bioinformatics/btad767>.
- Li RY and et al. *Comprehensive 100-bp resolution genome-wide epigenomic profiling data for the hg38 human reference genome*. Data Brief, 2022. URL <https://pubmed.ncbi.nlm.nih.gov/36582986/>. PMID: 36582986.
- Jacob Schreiber and et al. *A pitfall for machine learning methods aiming to predict across cell types*. Springer Nature, 2020. URL <https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-02177-y>. <https://doi.org/10.1186/s13059-020-02177-y>.
- Zainab Shahid, Brittany Simpson, Kathleen H. Miao, and Gurdeep Singh. *Genetics, Histone Code*. StatPearls Publishing LLC, 2023. URL <https://www.ncbi.nlm.nih.gov/books/NBK538477/>. PMID: 30860712.
- Connor Shorten and Taghi Khoshgoftaar. *Design considerations for image Data Augmentation*. Journal of Big Data, 2019. URL <https://link.springer.com/article/10.1186/s40537-019-0197-0>? <https://doi.org/10.1186/s40537-019-0197-0>.
- David Sims and et al. *Sequencing depth and coverage: key considerations in genomic analyses*. Nature Reviews Genetics, 2014. URL <https://www.nature.com/articles/nrg3642>. <https://doi.org/10.1038/nrg3642>.
- Oliver Stegle and et al. *Using probabilistic estimation of expression residuals (PEER) to obtain increased power and interpretability of gene expression analyses*. Nature Protocols, 2012. URL <https://www.nature.com/articles/nprot.2011.457>. <https://doi.org/10.1038/nprot.2011.457>.

- Kouzarides T. *Chromatin modifications and their function*. National Library of Medicine, 2007. URL <https://doi.org/10.1016/j.cell.2007.02.005>. PMID: 17320507.
- Shushan Toneyan, Ziqi Tang, and Peter Koo. *Evaluating deep learning for predicting epigenomic profiles*. Springer Nature, 2022. URL <https://www.nature.com/articles/s42256-022-00570-9>. <https://doi.org/10.1038/s42256-022-00570-9>.
- Moritz Völker-Albert and et al. *Histone Modifications in Stem Cell Development and Their Clinical Implications*. PMC PubMed Central, 2020. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7724464/>. PMID: 33296672.
- Sean Whalen, Jacob Schreiber, William Noble, and Katherine Pollard. Navigating the pitfalls of applying machine learning in genomics. 2022. URL <https://www.nature.com/articles/s41576-021-00434-9>. <https://doi.org/10.1038/s41576-021-00434-9>.
- Elizabeth Wilbanks and Marc Facciotti. *Evaluation of Algorithm Performance in ChIP-Seq Peak Detection*. Plos One Journal, 2010. URL <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0011471>. <https://doi.org/10.1371/journal.pone.0011471>.
- Ken Youens-Clark. *Mastering Python for Bioinformatics*. O'Reily Media, 2021. URL <https://www.oreilly.com/library/view/mastering-python-for/9781098100872/ch03.html>. isbn 978-1-098-10088-9.
- Žiga Avsec, Vikram Agarwal, Daniel Visentin, and et al. *Effective gene expression prediction from sequence by integrating long-range interactions*. Springer Nature, 2021. URL <https://www.nature.com/articles/s41592-021-01252-x>. <https://doi.org/10.1038/s41592-021-01252-x>.