MSc in Computer Science 2020-21

Project Dissertation

Project Dissertation title: Developing robust machine learning models for off-target prediction in CRISPR/Cas9 gene editing using ensemble learning and feature space analysis

Term and year of submission: Trinity Term 2021

Candidate Number: 1047337

Word Count: 21,938

Developing robust machine learning models for off-target prediction in CRISPR/Cas9 gene editing using ensemble learning

Candidate Number: 1047337

University of Oxford

A thesis submitted for the degree of

MSc in Computer Science

Trinity 2021

Abstract

CRISPR/CAS9 is the genetic engineering technology bringing about incredible advances in our ability to address today's most serious medical challenges. The technology is built on the capability of guide ribonucleic acids (gRNAs) to identify precisely the parts of the genome to edit. Inadequately designed gRNAs can lead to wrong selection in the genome with often catastrophic effects. The mission for effective guide design has to go beyond preferring guides with high activity at the correct target site but must also include a minimisation of risk when it comes to increasing specificity. While off-target prediction is more challenging than the on-target scenario, we show that the robustness of ensemble learning can provide good predictive performance and outperform existing procedural methods. Furthermore, we develop novel deep learning architectures that can learn from pairs of sequences and identify the levels of off-target activity. We implement classification models that, for the first time, attempt to separate validated experimental off-targets from inactive cases. We also investigate the different features defining the specificity of gRNAs so that future guide design can include these attributes in creating more mistake-resilient guides. All of the investigations were completed on the recently released comprehensive CrisprSQL dataset.

Contents

Introduction

1.1 Motivation

CRISPR/Cas9 (Clustered regularly interspaced short palindromic repeats) is a popular and powerful technology initially derived from the immune system response of bacteria that allows for gene editing mechanisms like the insertion, deletion, or replacement of specific genomic fragments [1]. The technology can be used to edit or remove genes with malignant traits allowing for personalised genetic treatment, prevent harmful mutations in human embryos, and push the limits of research by studying the interplay between phenotype and genetic variation [2, 3]. This gene knockout technology has been applied across various cell types and organisms and several databases have been created to aggregate the data produced by numerous binding and cleavage studies.

CRISPR/Cas9 is based on three components to accomplish gene editing: Cas9 (CRISPR-associated endonuclease) protein, guide RNA (gRNA or sgRNA), and PAM (protospacer adjacent motif) [4]. The steps in the gene-editing process are

visualised in Figure 1.1 below and consist of:

1. The synthetic guide RNA (sgRNA) forms a ribonucleoprotein complex with the Cas9 protein to guide it to the target binding site on the DNA. The sgRNA is user-defined which means that just by changing the sequence of the sgRNA, one can change the genomic target to be edited;

2. Before Cas9 can bind, the target sequence needs to have a complementary PAM site (2-6 bp) downstream to act as a signal to the protein it is binding in the right place. Since different Cas9 proteins have different matching PAM sites, complementarity needs to happen between sgRNA and the DNA on one side, and between Cas9 and PAM on the other. The Cas9 then performs its endonuclease activity by cutting the target DNA 3-4 bp upstream of the PAM site resulting in a double-strand break (DSB);

3. The DSB is repaired by one of two pathways:

   (a) The error-prone non-homologous end joining (NHEJ) causing a few nucleotide deletions or insertions at the DSB site

   (b) The high-fidelity homology-directed repair (HDR) can insert a lot more nucleotides or even fluorescent tags which makes it a highly utilised pathway for insertions

4. After the repair, the edited DNA is once again one molecule with the only difference being the genetic edits just made;

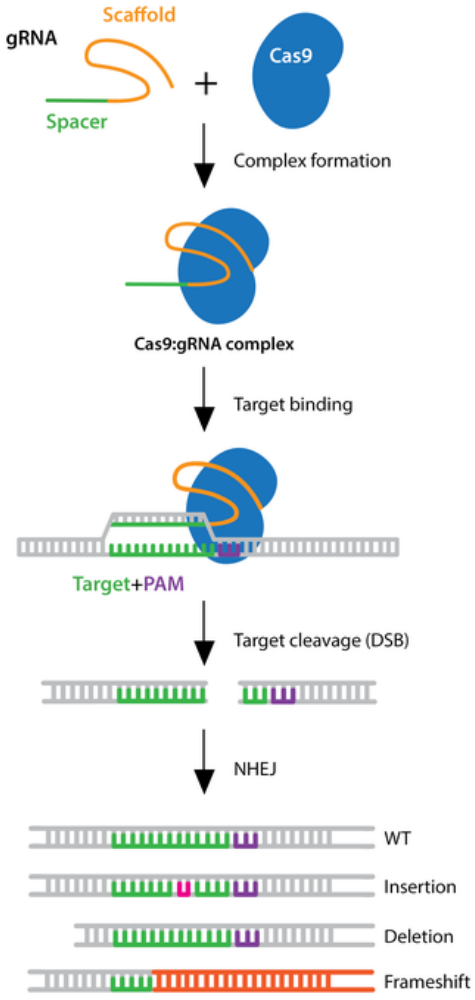

Figure 1.1: Steps for CRISPR/Cas9 Gene Editing [5]

Despite specific DNA sequences being aimed, when the matching occurs between sgRNA and the target DNA, mismatch sites might appear leading to unwanted cleavage of DNA, or off-target cleavage. This behaviour can result in off-target mutations and unintended edits with significant consequences on genomic stability while increasing the risk of disease in the organism which is why it is one of the most pressing challenges in CRISPR/Cas9 clinical applications [6]. Researchers have experimentally struggled with identifying which sgRNA will be the most efficient due to the multitude of factors to be considered. To appropriately address these hurdles,

prediction algorithms can be applied to preemptively detect mismatch sites and loci and help choose specific sgRNA sequences accordingly based on their predicted off-target profiles.

1.2 Aims and Objectives

The development of off-target prediction methods has shown promise in identifying the off-target profile of the sgRNA beforehand to minimise their wrongful effects (high specificity). Some tools predict off-target cleavage efficacy by using scores like CFD and MIT and then comparing them to whole sgRNA off-target detection methods like GUIDE-seq without using any learning-based approaches [7]. Researchers have recently moved to use machine learning and statistical methods to identify the best possible sgRNA matching for the aimed DNA target sequence. Existing work includes CRISPOR, CRISPR ML, DeepCRISPR, and a variety of gradient boosting trees and feed-forward neural networks applied on the mentioned datasets [6, 7].

Because of the exhaustive computational experiments done on these standard datasets and the variations in performance across different models, research needs to move to more comprehensive and updated datasets in the hopes of finding agreement on the factors impacting guide efficiency. Challenges in applying machine learning effectively to this prediction task include using newer and more comprehensive datasets, data heterogeneity, and finding factors that determine sgRNA efficiency to guide more optimal design. With the publishing of crisprSQL, a standardised and comprehensive dataset quantifying off-target effects along several cell lines using ap-

propriate annotations, state-of-the-art cleavage prediction algorithms can be applied to a large and consistent dataset [8]. Furthermore, the choice of feature selection, as well as mapping, can have a sizeable effect on the predictive capabilities of the models, thus identifying optimal subsets of features using ensemble-based methods can not only improve model performance but can also lead to interesting observations in the feature space which can help identify these factors important for good guide sequence design [9].

Thus, the aims of the project are as follows:

1. Data processing of the new CrisprSQL dataset with application of existing procedural models in off-target score prediction on this novel dataset and improvement of predictive performance through ensemble learning;

2. Apply feature importance and explainability methods to these ensembles and observe which procedural scores are more important to specificity;

3. Implement and compare existing state-of-the-art deep learning models on the sequence pairs directly in this dataset with and without using procedural scores and compare them to two of our proposed architectures;

4. Determine the resilience of the different ensemble and deep learning models to experimentally nonvalidated samples and propose a robust model capable of generalisation with up to an order of magnitude more inactive sites in the dataset;

5. The end goal being that given a proposed guide RNA and a random target site,

the hierarchical system proposed will be able to reasonably separate whether the target site will be active, and then, if so, what levels of activity should be expected in the case of off-targets (specificity);

1.3 Structure

The dissertation or thesis is structured into 6 sections according to the guidance provided by the Department of Computer Science, but whose internal structure allows for more thematic and conceptual ordering respectively.

The first section being **Introduction**, where the motivation or problem statement can be found laying out the reasons for pursuing the topic and the relevance of the solutions that will be proposed in the dissertation. The aims and the objectives outline in bullet-point structure the main points that were achieved in the project in the respective conceptual order.

The **Background** section contains the fundamental knowledge needed to understand the complexities of the ideas studied in the dissertation project as well as the key concepts drawing from existing literature and completed work in the area of machine learning for genomics. The section is divided into two parts: genomics and machine learning. This is to create a more systematic approach to two different areas of knowledge while also allowing for discussion on the overlap near the end of the two parts. The genomics part explains terms that will be used throughout the dissertation and that need to be understood for complete awareness of the context

of the problem as well as proposed solutions. This part will also lay out existing efforts that have been achieved by the research community in tackling the off-target effects problem through procedural and not learning-based methods. Finally, since the dissertation built on the relatively recently published CrisprSQL dataset, a short commentary on the background of the database and the collection methods used to gather experimental data is of high importance to understanding subsequent analyses of results obtained from the dataset.

The second part of the background or review section concerns machine learning concepts and tools that have been employed in the fulfillment of the dissertation's aims. The part starts with the overall machine learning problem separation between regression and classification which have been used in this project to solve different aspects of the challenge and are sometimes used in combination in a systems approach, as we will see later. Standard regression and classification methods and their mathematical background will be covered followed by a more involved explanation of neural networks and their components. Then come general machine learning approaches to problem solving that involve training, validation tuning, and crossvalidating results alongside techniques addressing overfitting. Finally, ensemble learning and its conventional implementations (bagging, blending etc.) will be introduced as a more complex method of combining several machine learning models into a larger structure for making predictions, as well as referencing explainability methods like shapley values in estimating feature importance when predicting off-target effects.

The **Methodology** section will go into more details on the methods applied in the project as well as their comparative value and the trajectory taken in designing and implementing several solutions respectively.

The **Results** section contains the performance measures of the different models applied on their respective problems, the comparisons to each other and existing approaches, and several visualisations to aid with prediction results as well as explainability measures.

The next section is the **Discussion** where the results will be interpreted and analysed to the fulfillment of some of the research questions posed in the Introduction. There will also be reflection on the limitation of the results and questions coming from the problem definition and context.

The **Conclusion** is the last questions that will reiterate the important takeaways of the project and its results alongside a few ideas on the future work and direction that can be taken to continue addressing the wider complexities of the challenges studied here as extensions of our work.

The Appendix will contain extraneous detail concerning validation plots of various models and their hyperparameter tuning. Results considered tangential to the central questions of the project will also be found there in table format. The references

are at the end and they follow the IEEE citation format.

Background

2.1 Genomics

The genome is the entire collection of an organism's genetic material where genomics is the science of studying that material, its structure, and properties. The entirety of the genetic material is contained in chromosomes, which consist of chemical objects called deoxyribonucleic acids (DNAs) that through their double helix structure wind around nucleosomal beads allowing them a compact spatial representation into what is chromosomes. The incredible extent to which the packing occurs means that a typical human chromosome averaging about 4 μm contains almost 80 mm of DNA if completely extended, something often called chromatin packing. This implies an average packing ratio of 20,000, meaning that a unique ecosystem exists at the chromosomal level where DNA can not be takes as an isolated unit within the cell's nucleus. Any processes at that level, including transcription, translation, and gene editing, need to consider a multitude of factors that *a priori* would not have been considered as relevant [10, 11].

In a sense, DNA and its sister molecule, ribonucleic acids (RNA) represent genetic material that could be understood as information or even data where translation and transcription are just repeated processes to express that data into functionalisable output, proteins. As with any data, it can be corrupted and it can be edited. Many cellular mechanisms exist to guarantee the fidelity of the DNA code as it plays a crucial importance in an organism's genetic stability but also in its generationally reproductive capabilities. Gene editing, on which more will be said later in a separate subsection, is built on cellular mechanisms that exist to protect genomes from adversarial viruses inserting their RNA into the organisms. Using that same mechanism to remove selectively sections of DNA for either scientific study or prevention of diseases rooted in genetic nature, presents the crux of gene editing technology like CRISPR that still, however, can make mistakes and lead to off-target effects. The study of these effects both in the long and short term has led to the creation of a separate discipline within genomics, underscoring the immense clinical importance of adequate study of this phenomenon [12]. Understanding what leads to an increased chance of these effects, thus, requires a broader understanding of the genetic background of these processes that the subsequent sections will attempt to explore, especially in the context of biological factor importance which might have a direct relationship to the identified feature importance in the machine learning models we later develop [11].

2.1.1 Nucleic Acids

Before continuing to the discussion on the specifics of gene editing technology, it is wise to reference to fundamental concepts like nucleic acids whose abilities to preserve information, be copied, and to mutate are the conceptual building blocks of the technology. Nucleic acids are macromolecules strung together from smaller building blocks called nucleotides that in turn consist of a five carbon sugar, a phosphate group, and an aromatic base. The aromatic bases include adenine (A), guanine (G), cytosine (C), thymine (T), and uracil (U), and depending on the ordering of these bases and the nucleotides in the linear sequences that make up the DNA, the DNA gets its unique genetic code seen often as a representation of the entire molecule. This sequence is what determines the uniquness of the DNA molecule and the genome at large, as well as containing the information used by the cellular mechanisms to create specific proteins and undertake specifically instructed processes. The key differences between DNA and RNA is that the sugar is a deoxyribose in DNA and a ribose in RNA, as well as that RNA has uracil as a aromatic base instead of thymine in DNA [11]. A clearer view of the nucleotide structure can be seen in Figure 2.1 below. The chemical energetic properties of these molecules is what drives the cellular genetic processes and these specific energies can often play a part in gene editing experiments, often included as additional features to the sequences themselves [13].

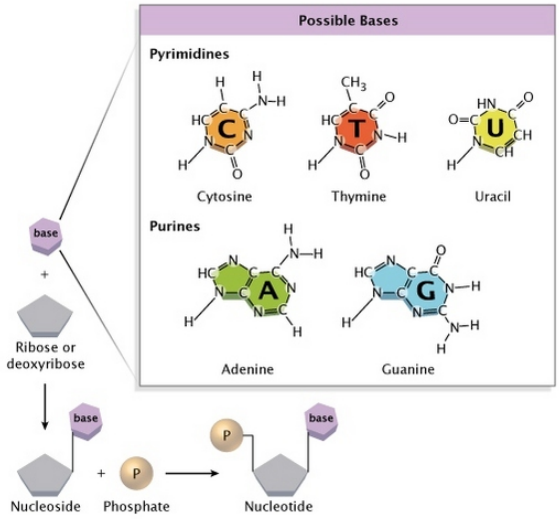

Figure 2.1: Structure of nucleotides [14]

Mutations are permanent changes to the DNA that can occur in various parts, but for the purposes of this thesis, we will restrict ourselves to mutations in the bases of the nucleotide, for example, the change of adenine (A) into guanine (G). Some of these changes end up not having any noticeable effects, while others can prove disease inducing and lethal. Mutations can occur for a variety of reasons, ranging from exposure to environmental factors, through inherited traits from parents, to aging. The types of mutations that exist include single base changes like the one already mentioned called point mutation, deletions/insertions/duplications, translocation or the movement of a segment of DNA in the genome, and inversions or flips [15]. Not too much detail will be expunged on mutations, suffice to say that gene editing is fundamentally either intentional introduction of mutations into the genome or the correction of the same, and that for our limited study of CRISPR in this thesis, we will be mostly working with experimental data of point mutations and deletions/insertions/duplications.

2.1.2 Gene Editing

As alluded to earlier, gene editing is the process by which DNA is edited, inspired and adapted from existing mechanisms in biology sometimes called nuclease platforms. The early types of platforms included zinc fingers which are zinc ions that can be fused to DNA cleavage domains and be used for frame shift or translocation mutations. The more recent example of a nuclease platform is CRISPR–Cas9 which uses a 100-nucleotide long single guide RNA (sgRNA) to direct the Cas9 protein to a specific site in the DNA sequence where it should initiate a cut. This system was adapted from bacterial immune system responses developed to excise viral genetic material parasitising on the bacteria's cell machine [16]. A more detailed look at the mechanism in play can be seen in Figure 1.1 in the Introduction. Through editing or targeted design of the sgRNA, the CRISPR system can recognise any gene or genetic sequence. Once CRISPR induced the cut and results in a DNA break, the break is repaired by either non-homologous end joining (NHEJ) or homologous recombination directed repair (HDR), resulting in indels (INsertions or DELetions). The unintended consequences of using CRISPR like mutations and other off-target effects seem to be more prevalent in cancel cell lines, which further raises the importance of minimising their occurrence for clinical benefit. Much remains to be learned regarding the factors at play influencing the off-target effects but it is clear that more tailored sgRNA design can often help with reducing those negative side-effects [17]. Carefully designing these sgRNAs is sometimes not a deterministic process, but one

that requires finding patterns in experimental data showing which sgRNAs lead to higher specificity. In that sense, powerful data processing methods built on learning can act as effective tools in recognising latent patterns in sequential data of sgRNAs and their targets and, consequently, help tune the design process to create sgRNAs with a reduced likelihood of leading to off-target effects.

To quantify the "off-target effects", a measure is used called target cleavage frequency which is nonnegative, 0 if the binding of the sgRNA and the particular target never happens, and increases depending on how frequently or strongly the sgRNA selected that specific target. Thus, the cleavage frequency is a two variable function, dependent on both the sgRNA and the target sequence which should be as low as possible for all targets except the one specified. The idea of various procedural and learning methods can be reduced to them learning the mapping from sequential and other attributes of the data to this frequency, so that pairs or sgRNAs can be selected that have a priori high cleavage frequency for the intended target, and low for the unintended off-targets. Often, these off-target scores depend on the number and position of the mismatches and mismatches closer to the PAM site like in positions 19 or 20 depicted in Figure 2.2 below can have higher off-target activity.

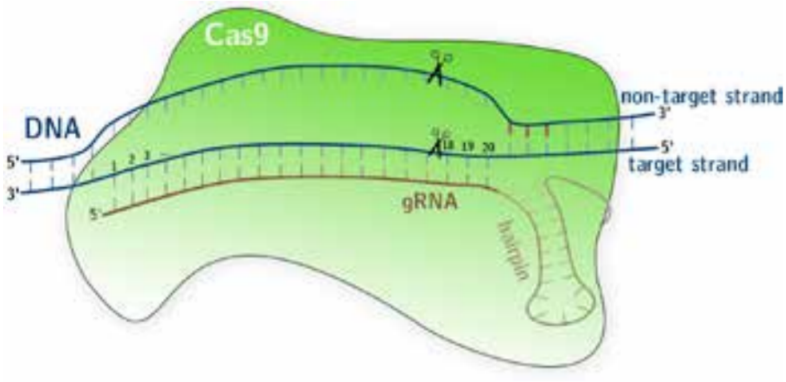

Figure 2.2: CRISPR gene editing at position 18 of target sequence [13]

2.1.3 Procedural Activity Scores

Researchers choose sgRNAs carefully so as to minimise the off-target effects by using several scoring methods to rank sequences according to specificity. These procedural scores are not based on any learning but are rather functions of number of mismatches between the sgRNA and the target sequence, sometimes also being impacted by the proximity and density of the mismatches to the PAM site. Through rigorous experimental testing, weights have been deduced on the effect of mismatches for each possible nucleotide change at each position and different formulas have been proposed to combine these into a score. Depending on the formula used and the distribution of weights as a function of position, different scoring methods have been proposed throughout the years. Some of the most popular off-target scoring tools include MIT, CFD, CROP-IT, and CCTop which offer a variety of scales and frameworks to represent off-target activity. All scoring methods besides CCTop include a penalty for mismatches located close to each other, a density penalty [18]. By using several tools at once, and not just one, one can gain a broader understanding of the specificity of the proposed sgRNA which is why we will introduce each of the

scoring tools and their respective advantages/disadvantages below.

2.1.3.1 MIT

The MIT specificity score used to be obtainable from an online tool that can be used to rank guides according to their potential off-target effects. The site has since been taken down but the underlying methodology can still be reviewed from the original publication. We implemented our algorithm through a combination of heuristics deduced from the original paper and an implementation of the score on the CRISPOR website [18, 19]. It is among the first scoring tools that was developed in 2013 and it relies on simply weighting mismatches depending on the position. The type of mismatch was not taken into account when the weight matrix was proposed. What the score is sensitive to, in particular, is the number of mismatches and their distance to each other (density) [18, 19]. A more detailed look into the 3 components going into the score can be seen in the equation (obtained from [18]):

$$score \propto w \cdot d \cdot \frac{1}{n^2} \tag{2.1}$$

Where w is the product of the weights of the positions where mismatches are detected in the 20bp sequence, d is the penalty for the distance between the mismatches and is inversely proportional to the average distance between mismatches, and n is the total number of mismatches for the entire sequence. The weights for the positional mismatches are greater the closer they are to the PAM site so as to penalise for mismatches closer to the PAM site found experimentally to impact the specificity to

a larger extent than PAM-distal mismatches [19]. The scale of the score is between 0 and 100, with 100 indicating no mismatches between the sgRNA and the target, with maximum specificity.

2.1.3.2 CFD

Similarly to MIT, the Cutting Frequency Determination (CFD) score is calculated from the percent activity values from a matrix of penalties based on mismatches of each possible type at each position within the sequence [20]. The advantage of this score is that it considers the typing of mismatches and not just their position in the sequence. For example, mismatches of A:G and G:C will have very different weights associated with them coming from experimental data indicating correlation between each type of mismatch and a decrease in off-target activity. The full score is then just a multiplication of these weights obtained from the matrix for each mismatch, when there are two or more mismatches, then individual mismatch values are multiplied together. There is no separate term for the distance consideration and number of mismatches like in MIT, it is rather assumed that those factors are implicitly included in the weight matrix. Another difference is that the PAM sites will also have a score of their own to be included and they will also be multiplied with the mismatch score. There is a number of most common PAM sites like NGG and NGA, and each has their own weighting in the specificity score calculation also based on experimentally derived correlation between each of the PAMs and off-target activity. CFD score has been validated with GUIDE-seq and has been shown to be superior to MIT in several experiments including CRISPOR [21]. Perfect match between the

guide and target sequences gets a score of 1 which is the maximum specificity score [22]. Similarly to MIT, there is no equation per se for calculating the score, rather it is just a weighted sum of positional mismatches whose weights have been provided by experimental validation.

2.1.3.3 CROP-IT

CROP-IT was proposed slightly later in 2015 and is meant to outperform the previous 2 scoring methods by including special penalties for the off-target score in the case of consecutive mismatches in the sequence. Furthermore, it considers the chromatin (chromosome material explained in 2.1) structure when assigning specificity scores, going beyond just looking at the genetic sequences themselves. The proposition was that including the global chromatin structure of the cell lines in question would improve the computational performance of the tool and outperform the previous tools in predicting scores. The researchers also noted that when only DNA sequence information is used, a small number of experimentally validated Cas9 bindings sites is predicted showing the limitation of relying on sequence data alone. CROP-IT has been shown to not always be the best performing tool in a variety of experiments but it certainly does have the advantage of considering attributes beyond just the sequence when making the specificity prediction [23].

The underlying logic of the tool relies on a computational model where each position of the gRNA is weighted based on experimental information from multiple sources. The algorithm can be broken down into the following steps [23]:

1. CROP-IT first performs a filtering step whereby only sequences ending with the PAMs 'NGG' or 'NNG' are selected. In this way, separate scoring methodology can be followed depending on the PAM sequence;

2. To score alignment candidates, the first 20bp (sequences without PAM) are divided into three segments of 5, 5 and 10 bp. Different mismatch scores are applied for each of these three segments;

3. Each nucleotide in the 20bp region is compared to the respective nucleotide on the target sequence. If a match, assign a score s_i (where i = 1, 2, 3) based on the segment and if a mismatch, look for a consecutive mismatch. A penalty $-s_i$ is assigned when there are two consecutive mismatches, otherwise penalty $s_i/2$ is assigned for a single mismatch. The penalty assignments are based on observations of the experimental data, where binding intensity was higher for sequences with smaller number of consecutive mismatches. This shows that sequences with smaller number of consecutive mismatches are more likely to be off-target sites and therefore, are given a higher specificity score. The equation for score calculation is then:

$$S = \sum_{i=1}^{3} [(n \times s_i) + (m \times (-s_i) + k \times (-s_i/2))] \tag{2.2}$$

   Where n is the number of mismatches, m the number of consecutive mismatches and k the number of single mismatches.

4. The weights s_i for each of the 3 segments are provided by the researchers and they were obtained by non-parametric optimization and they equal $s_1 = 5$,

$s_2 = 70$, $s_1 = 30$;

5. In the case of consecutive mismatches belonging to separate segments, the mean of the penalties for the two segments is considered;

A perfect maximum specificity score of 650 corresponds to no mismatches and out of all of the tools has the largest scale for the specificity score that will later require standardisation when combined with the other scores in the input. CROP-IT is also available as an online tool for free web use [23].

2.1.3.4 CCTop

CCTop is the most recently proposed out of all of the scores considered having been developed in 2017 and experimentally validated for gene inactivation, non-homologous end-joining, and homology directed repair which the previous methods were not. The model runs on a likelihood framework where for each off-target sequence the probability of the formation of a stable duplex (binding event) is calculated. Based on experimental observations, the researchers concluded that this likelihood decreases the closer the mismatch is to the PAM site, thus they weigh their positional mismatches accordingly. The formula used in the model is (obtained from [24]):

$$\text{score}_{\text{off-target}} = \begin{cases} 224, & \text{if } n_{mismatches} = 0 \\ 224 - \sum_{\text{mismatch}} 1.2^{pos}, & \text{otherwise} \end{cases} \tag{2.3}$$

Where *pos* is the mismatch position as counted from the 5' end with the base of the

power was deduced experimentally [24] The score is then subtracted from a maximum of 224. So, in case of no mismatches, it is: 224-0 (and not 1.2 to the power of 0 because if there are no mismatches the formula is not used). When there are n mismatches, the score is 224-1.2^{pos_n} .

2.1.4 Data: CrisprSQL

It might be clear so far that a variety of prediction tools already proposed have to be experimentally validated and sometimes there is considerable variation in the data resulting in some tools unexpectedly outperforming others. Due to the popularity of the study of off-target effects, a multitude of cleavage experimental datasets have been published but without a common standard of formatting or even result representation. What one research experiment and team designate as "off-target activity" can vary, allowing for "specificity" to be differently measured across different experiments. CrisprSQL combines a selection of datasets into a large and comprehensive interactive database with over 25,000 sequence pairs and other feature entries based on cleavage data from 144 gRNAs [8]. Besides sequence pair information, CrisprSQL also includes epigenetic factors, meant to represent the atomistic parameters of the pair duplex. Furthermore, interaction energies are also included which indicate the levels of free energy of sgRNA folding, which could be another indicator of the propensity of the sgRNA to bind to off-targets built on the connection between thermodynamic stability of the CRISPR system and gene editing efficiency [25].

To be able to develop and train complex machine learning models, one needs to have access to large amounts of diverse data for generalisability purposes which is where CrisprSQL plays a crucial role. Including other features and attributes beyond sequence information might play a deciding role in the prediction of specificity as we have seen in tools like CROP-IT. Another salient attribute of the database is that it only includes data points which have been experimentally validated in cells, greatly narrowing down the prediction problem to only *in vivo* cases [8].

A separate challenge of our project, however, is to investigate the robustness of our models to maintain sufficient predictive power when exposed to **nonvalidated** samples as well, because *a priori*, the investigator does not know which sequence pairs will occur in the cell (validated) and which will not. **Nonvalidated** samples have nothing to do with and are different from the term "validation set" present in machine learning. Here nonvalidated samples represent those data samples whose label is always 0 because they have not been validated experimentally to show any off-target or cleavage activity. We introduce significant and increasing levels of data imbalance by adding a lot more nonvalidated than validated samples in the test reporting set to evaluate the models' generalisability when learned only on validated samples.

The key output feature of interest is the cleavage frequency but as already mentioned, since different experiments had different metrics for the activity, we will

assume that across experiments included in CrisprSQL, cleavage frequency measures the same underlying biological phenomenon. The fundamental concept is that higher specificity corresponds to lower cleavage frequency for off-target sites, so our models will need to be able to predict the cleavage frequency for both on- and off-targets for a given sgRNA, and then allow the user to choose the sgRNA that gives the highest specificity, i.e. lowest cleavage frequency for possible off-targets.

2.2 Machine Learning

Machine learning is today one of the most used tools for data processing and learning, meant to take advantage of the large amounts of data accessible in a variety of applications, and apply powerful algorithms to extraction of patterns for predictive purposes found in the data troves. As Murphy writes in his seminal textbook, "the goal of machine learning is to develop methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data or other outcomes of interest" [26, 27]. Machine learning can be divided into smaller conceptual subspaces, depending on the tasks targeted. Thus, a rough separation of paradigms would be: supervised, unsupervised, and semi-supervised learning. In supervised learning, the data will have pairs consisting of input and output and the task is then to learn a mapping from the former to the latter, which is the learning applied in this project. In unsupervised learning, the data is without outputs, and the goal is to deduce some interesting structure or attributes of the data. Common examples include clustering and principal components analysis. Lastly, in semi-

supervised learning, the data will be mixed, some having output labels and some not, leading to the challenge of optimising a learning algorithm that would be able to learn collaboratively using both types of data [28].

The problems set out in this project will be tackled through supervised learning, i.e. developing automated models to learn a mapping from some input, usually scores or genetic sequences or both, to a real-valued or binary output. If the output is real-valued like predicting target cleavage frequency, then we will resort to regression techniques, but the second type of problem we face is predicting whether samples obtained by online tools and those experimentally validated can be discriminated using classification methods where a binary label of 0 (nonvalidated sample) or 1 (validated sample) is used.

Two key set of machine learning tasks will be undertaken in this project: *regression* and *classification*. Regression methods have been used in statistics for a very long time before the advanced machine learning we have today was even developed. The main idea is that the mapping from input to output that is learned is not a deterministic function but that it includes some random error. It is this error that allows us to model simple linear regression models both deterministically and probabilistically. The response y that is learned can then be represented by an expectation conditioned on the input:

$$y = \mathrm{E}\left(y \mid x_1, \ldots, x_D\right) + \varepsilon = f\left(x_1, \ldots, x_D\right) + \varepsilon \tag{2.4}$$

Depending on the assumed complexity of the function or underlying probability distribution of the error, regression models can be linear or polynomial. If the relationship between the inputs and output is taken as linear, then linear regression will suffice, however, non-linear relationships cannot be captured successfully by such a simple model, thus basis or feature expansion of the features is undertaken to increase the dimensionality of the problem into an input space where it can be modelled as a linear relationship.

Classification is the other type of supervised learning problem, where the mapping from input is now into a categorical or discrete output space. Common methods here include generative models like Naive Bayes Classifiers and Discriminant Analysis, but the popular perceptron-based neural network models are also quite frequently applied and which will be discussed seperately later on.

2.2.1 Data Representation

Another important concept is that of encodings. The independent variables or features do not have to be of the same type, some can be real-valued while other can be categorical. Encodings offer a framework for transforming categorical input into numerical representation so that it can be used as input to a learning model. This will be important for our discussion only when talking about sequential data like the sequences of the targets and sgRNAs. The most common way to encode the sequences is to resort to one-hot encoding like in the Figure 2.3 below:

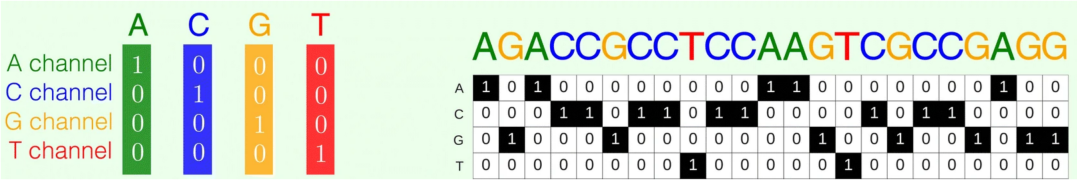

Figure 2.3: Encoding schema for sequences [7]

Sometimes, however, the target and sgRNA sequence are not separately used as input like two separate features but are combined into one feature. An example is to use the OR logical relations between the two encoded sequences which effectively transforms two features into one while maintaining the dimensionality of the sequence input [6]. This is less common but some models, including the ones developed here, utilise this approach to avoid only resorting to siamese networks that would separately learn from each of the sequence pairs.

2.2.2 Classical Machine Learning Models

2.2.2.1 Linear and Polynomial Regression

The most common regression model is linear regression that can be derived in two ways, either the function above is taken to be a linear function of independent variables or regressors, or the expectation can be taken as a mean of the response probability distribution assumed to be gaussian coming from the error taken as gaussian first [29]. Either way, the output is understood as:

$$y = w_0 + x_1 w_1 + \cdots + x_D w_D + \epsilon \tag{2.5}$$

Where D is the number of dimensions of the data, or the number of features describing the input space and the error term is ϵ while the weights of each of the independent variables or features is represented in the w of the equation. The solution to the above equation is an expression for the weights that can be multiplied with any input to provide a prediction. Obtaining the closed-form solution can be done through minimising the residual sum of squares, a square term for the error between prediction and true values, which depends on the invertability of the $\left(\mathbf{X}^\top \mathbf{X}\right)^{-1}$ matrix:

$$\mathbf{w}_{\mathrm{ML}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathrm{y} \tag{2.6}$$

Using the probabilistic approach relies on the maximum likelihood principle maximising the likelihood of the output probability distribution whose prediction $\widehat{y}_{\mathrm{new}}$ will be the same as for the deterministic approach except that now we are able to obtain confidence intervals in our predictions coming from a quantification of the error term as a gaussian with a fixed deviation:

$$\begin{aligned} \widehat{y}_{\mathrm{new}} &= \mathbf{w}_{\mathrm{ML}} \cdot \mathbf{x}_{\mathrm{new}} \\ p\left(y_{\mathrm{new}} \mid \mathbf{x}_{\mathrm{new}}, \mathbf{w}_{\mathrm{ML}}\right) &= \widehat{y}_{\mathrm{new}} + \mathcal{N}\left(0, \sigma^2_{\mathrm{ML}}\right) \\ \sigma^2_{\mathrm{ML}} &= \frac{1}{N}\left(\mathbf{X}\mathbf{w}_{\mathrm{ML}} - \mathbf{y}\right)^\top \left(\mathbf{X}\mathbf{w}_{\mathrm{ML}} - \mathbf{y}\right) \end{aligned} \tag{2.7}$$

When the linear relationship assumption will not hold, more complex assumptions need to be made by expanding the feature space into higher dimensions d. The input space changes, but the overall model is still linear in the weights:

$$\begin{aligned} \boldsymbol{\phi}(x) &= \left[1, x, x^2, \cdots, x^d\right] \\ y &= \mathbf{w}^\top \boldsymbol{\phi}(x) + \epsilon \end{aligned} \tag{2.8}$$

A common problem arises called overfitting if the dimensionality is assumed to large and it fit the training data almost perfectly but the model does not generalise well to unseen data. More will be said on this topic later and the mechanisms deployed to reduce overfitting, but suffice to say that through careful tuning of the dimensionality parameter d, polynomial regression models often offer a good solution to modelling problems too complex to be solved by simple linear regression.

2.2.2.2 Support Vector Machines

Support vector machines were originally conceived as an algorithm for binary classification, however, their versatility also allows using them for regression, often being dubbed support vector regression (SVR) instead like in our case [30]. Building on the discussion from the previous section, the objective function to optimise in linear regression is the Ordinary Least Squares as a measure of the error between the true and predicted value on the regression line. For SVR, instead of just minimising the sum of squares of the error, we have the freedom to choose how much error is acceptable *a priori* and find the appropriate regression line. Instead of minimising the error directly, we instead choose to minimise the l2 norm of the weights w (which implicitly are inversely proportional to the euclidian distance error of the points to the regression line) subject to a constraint that puts an upper limit on the error or tolerance ε [31]. The objective function can then be written as:

$$min\frac{1}{2}||w||^2 \tag{2.9}$$

subject to

$$|y_i - w_i x_i| \leq \varepsilon \tag{2.10}$$

When not all the points can fall into the closest margin determined by the inverse of the weights despite optimisation (non-separable case), the error for those points would be too large and hence an inverse penalty needs to be introduced to the algorithm in the form of slack variables. Assigning slack values ξ_i to all points outside of the region of marginal tolerance measuring their distance to the margin allows us to minimise their occurrence by including a slack term in the objective function itself:

$$min\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}|\xi_i| \tag{2.11}$$

subject to

$$|y_i - w_i x_i| \leq \varepsilon + |\xi_i| \tag{2.12}$$

The parameter C on the sum slack term above determines our tolerance to having points be outside of the margin, i.e. allocated error beyond the threshold. C is a tunable hyperparameter that we will mention more later on when talking about validation and tuning, but serving initial discussion one can say that setting C to 0 forces the slack to go to 0 and an extension of the margin thereby increasing the training error but preventing overfitting on the testing data. In this sense, C has a regularising influence, setting too high of a value can lead to too much slack/narrow

margin in the model with the training data given low error but also having low generalisability since predictions can happen far away from the regression line (overfitting) [32]. Thus, the amount of regularisation and the value of C are inversely proportional and validation tuning is a necessity to minimising generalisation loss.

An advantage of using SVR instead of linear regression is the application of kernels. Kernels allow basis expansion in algorithms like SVR without the computational costs of computing feature expansions. The underlying idea called kernel trick allows doing basis expansion to higher dimensions through a simple dot product of the input data in the original dimension hopefully leading to a more linear relationship easier to model in the higher dimensional space [33]. Many kernels exist beyond just the polynomial including the radial basis function (rbf) kernel:

$$\kappa\left(\mathbf{x}, \mathbf{x}'\right) = \exp\left(-\frac{\left\|\mathbf{x} - \mathbf{x}'\right\|_2^2}{2\sigma^2}\right) \tag{2.13}$$

Using the RBF or polynomial kernels will allow the SVR to model nonlinear relationships keeping in mind that greater complexity can come with larger variance and overfitting where the role of proper tuning of the C parameter comes again into importance.

2.2.2.3 k-Nearest Neighbours

k-Nearest Neighbours (kNN) is a nonparametric learning model that has found wider usage in classification tasks but that can be applied in the regression scheme. The

concept relies on using the neighbourhood of a point to approximate the prediction for that specific point [34]. This 'feature similarity' that is assumed between points uses a distance metric like euclidean distance to find the k most similar/closest points to the point of interest and average the output value of those k points which is then assigned to the specific point. Choosing how many neighbouring points to consider when allocating a prediction is represented by the k parameter and is usually tuned through validation. k can have a regularising effect, because choosing too small a k means that only the closest points are considered which decreases training error but will not generalise well and hence lead to overfitting.

On the other hand, choosing a large k means that our approximation will be less reliable and we will have greater training error, but if tuned right to minimise validation loss, can achieve good generalisation performance [35]. The particular advantage of the algorithms is its nonparametric nature allowing for simpler computation but taking longer for prediction since it needs to get the distances to all the points in the dataset first. For large datasets these costs might be too large to bear but kNN offers a great nonlinear approximation to predictions without the use of kernels or basis expansion that we have seen in the previous models. And finally, kNN do not have inherent interpretability like with the feature weights of linear regression, leaving analysis to be desired when referencing feature importance.

2.2.2.4 Logistic Regression

The previous algorithms introduced will be used mostly in the regression scenario whereas logistic regression is a model for binary classification tasks. We have seen how in linear regression we can model the output as a conditional Gaussian distribution given the input and weights, a similar discussion pertains to logistic regression, the main difference being that the output is now represented as a conditional Bernoulli (discrete binary) distribution. Since Bernoulli requires the input to the function to be probabilistic, the sigmoid (or logit) function will be used to map the product of input and weights (which was also used in linear regression hence shared regression term) into a probability space as seen in the equation below.

$$p(y \mid \mathbf{w}, \mathbf{x}) = \text{Bernoulli}(\sigma(\mathbf{w} \cdot \mathbf{x})) \tag{2.14}$$

$\sigma(\mathbf{w} \cdot \mathbf{x})$ can be understood as the probability that the predicted class label is 1. A point of elaboration is the prediction threshold, i.e. what the value of $\sigma(\mathbf{w} \cdot \mathbf{x})$ needs to be for the point to be classified as 1. The default is usually 0.5 meaning over 50% that the point is classified as 1 but the threshold is adjustable and can be used to optimise the algorithm. Logistic regression tries to find a linear decision boundary to separate the positive from negative class label points in the input space. Like in the probabilistic interpretation of linear regression, the optimisation of logistic regression resorts to maximising the likelihood as represented in the equation above. The optimisation is reduced to minimising the negative log-likelihood derived in the form below.

$$\text{NLL}\left(y_i \mid \mathbf{x}_i, \mathbf{w}\right) = -\left(y_i \log \mu_i + (1 - y_i) \log\left(1 - \mu_i\right)\right) \tag{2.15}$$

Where $\mu_i = \sigma\left(\mathbf{w}^\top \mathbf{x}_i\right)$. The objective function above mimics a cross-entropy between the observation y_i and the probability of the class estimate μ_i. Optimising this objective function can be done using Newton's method by computing the gradient and the Hessian of the function:

$$\begin{aligned} \nabla_{\mathbf{w}} \text{NLL}(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) &= \sum_{i=1}^{N} \mathbf{x}_i \left(\mu_i - y_i\right) = \mathbf{X}^\top (\boldsymbol{\mu} - \mathbf{y}) \\ \mathbf{H}_{\mathbf{w}} &= \mathbf{X}^\top \mathbf{S} \mathbf{X} \end{aligned} \tag{2.16}$$

Since there is no closed-form solution, the weights estimates depend on iteratively computing the Hessian and the update rule:

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - \mathbf{H}_t^{-1} \mathbf{g}_t \\ &= \mathbf{w}_t + \left(\mathbf{X}^\top \mathbf{S}_t \mathbf{X}\right)^{-1} \mathbf{X}^{\mathbf{T}} \left(\mathbf{y} - \boldsymbol{\mu}_t\right) \\ &= \left(\mathbf{X}^{\mathbf{T}} \mathbf{S}_t \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{S}_t \left(\mathbf{X}\mathbf{w}_t + \mathbf{S}_t^{-1} \left(\mathbf{y} - \boldsymbol{\mu}_t\right)\right) \\ &= \left(\mathbf{X}^{\mathbf{T}} \mathbf{S}_t \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{S}_t \mathbf{z}_t \end{aligned} \tag{2.17}$$

For this reason, this method of weight estimation is referred to as iteratively reweighted least squares. Similar regularisation and basis expansion discussion as in linear regression pertains here as well.

2.2.3 Neural Networks

Neural networks have been semantically and conceptually inspired by the neural brain structure found in humans, with the perceptron as the fundamental block of

neural networks meant to be a type of artificial neuron that would fire or produce a 1 once a certain weighted threshold is reached. The artificial neuron operates on a similar procedure as a linear/logistic regression model by taking a linearly weighted input, summing it, and then applying a non-linear activation (different from logistic regression) as a prediction output [36]. If the activation function is just the identity or linear, then the artificial neuron is just a replacement for the linear regression model. A figure representing this building block of neural networks is in Figure 2.4 below.

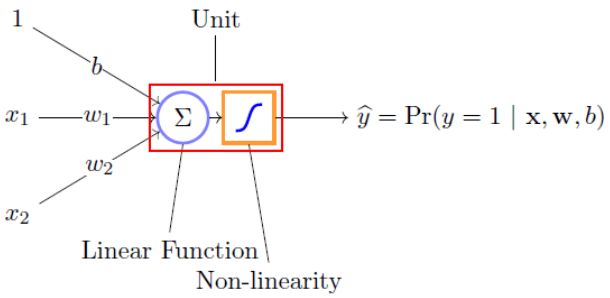

Figure 2.4: Artificial Neuron [37]

It can be shown that these artificial neurons can be used to construct logic gates and further be able to approximate any function that a computer can compute. This often leads to neural networks being referred to universal function approximators which sets them apart from the previous models we discussed in their potential modelling complexity [38]. Using several artificial neurons together in a network allows exploitation of this universal approximation principle where we are no longer learning a linear regression line but rather arbitrarily complex fits. Organising these artificial neurons into structures like layers and referring to them as nodes we arrive at the first type of neural network: a multilayer perceptron.

2.2.3.1 MLP

As the name implies, multilayer perceptrons (MLPs) are just stacks of artificial neurons connected together in layers whose individual neuron components are often called nodes. The number of layers and number of nodes per layer can vary across networks and even within one particular network architecture. The first layer is just the input layer, the inner layers are referred to as hidden layers as they do not interact with the data, and the last layer is the output layer which where the activation function determined whether the neural network is undergoing regression or classification. As for the final activation function, sigmoid is used in binary classification as it places a projection of values on a probability space with a tunable threshold, softmax is used in multiclass scenarios, and the identity or linear activation is used for regression schema [39]. An example of an MLP with one hidden layer and two hidden units is in Figure 2.5 below.

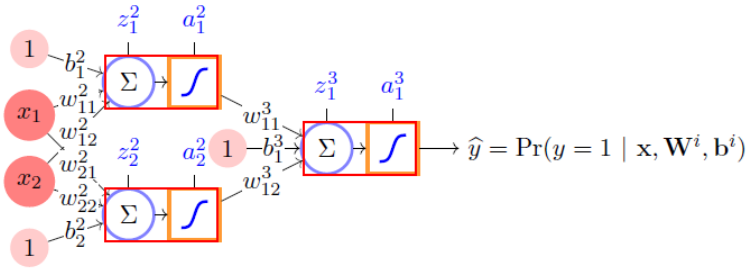

Figure 2.5: An example MLP [37]

If we assume tanh as the activation function in the hidden layers and sigmoid for the output layer, then the following equations representing the network can be posited:

$$
\begin{aligned}
\mathbf{a}^1 &= \mathbf{z}^1 = \mathbf{x} \\
\mathbf{z}^2 &= \mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2 \\
\mathbf{a}^2 &= \tanh\left(\mathbf{z}^2\right) \\
\mathbf{z}^3 &= \mathbf{W}^3\mathbf{a}^2 + \mathbf{b}^3 \\
\mathbf{y} &= \mathbf{a}^3 = \sigma\left(\mathbf{z}^3\right)
\end{aligned}
\tag{2.18}
$$

The non-linear activations are operations on the inputs and subsequent node inputs/outputs that transform those information stimuli akin to basis expansion but without the increase in the number of features. Thus, a hidden layer of the neural network is another layer of feature transformation into a space where linear modelling is more apt, and the network is then learning the features of the model instead of them being provided by humans, often referred to as representational learning.

The parameters of the neural network are learned through a 2 step process: forward and backpropagation. In forward, a single sample, for example, is given to the input layer and propagated all the way to the output, where the weights, bias terms, activation, and pre-activations are calculated and stored for the second step. In backpropagation, the optimisation of the loss function required updating the weights of the model so as to reach a loss minimum. Updating the weights means computing the gradients of the loss with respect to those parameters and making the respective changes from output layer/loss to the input layer, i.e. backwards. The forward propagation is a simple calculation implementing the following equations for each layer:

$$\begin{aligned} \mathbf{z}^l &= \mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l \\ \mathbf{a}^l &= f\left(\mathbf{z}^l\right) \end{aligned} \tag{2.19}$$

The backpropagation derivatives of the parameters must resort to the chain rule due to the dependency of the parameters on the previous layers. The update rule for the weights (here in matrix form for an entire layer) is:

$$\begin{aligned} \frac{\partial \ell}{\partial \mathbf{z}^l} &= \frac{\partial \ell}{\partial \mathbf{z}^{l+1}} \mathbf{W}^{l+1} \frac{\partial \mathbf{a}^l}{\partial \mathbf{z}^l} \\ \frac{\partial \ell}{\partial \mathbf{W}^t} &= \left(\mathbf{a}^{l-1} \frac{\partial \ell}{\partial \mathbf{z}^l} \right)^\top \end{aligned} \tag{2.20}$$

The weights in the model are learned through different optimisation techniques but the most frequently used is gradient descent which can be stochastic or batch. Stochastic means that the weight updates occur one data sample at a time, while batch means that the update happens after a batch number of data samples is processed through the network. While stochastic gradient descent has faster gradient update for large datasets since we do not need to wait for the gradient to be computed after all the data samples have been processed, due to the stochasticity, noise can be present in the optimisation results preventing faster convergence to a loss minimum [40]. Batch gradient descent, on the other hand, can be slow if the batch size is chosen to be too large, and it can also be too noisy and divergent of the batch is chosen too small. Usually, batch size is also a tunable hyperparameter in validation.

2.2.3.2 CNN

Convolutional Neural Networks (CNNs) are a type of neural network that replace the node and artificial neuron paradigm with one of matrix convolutional operations between layers. CNNs have found great success in image analysis and object recognition because they rely on the concept of weight matrices or convolutional layers being applied to different areas of the input image or matrix allowing for successful capture of spatial and temporal dependencies in the data. Once the multiplication of the weights of the convolutional filter and the input matrix is complete, the result is summed, non-linearly activated, and passed as input to the next layer in the network. This allows the model to learn patches in the image or data as it strides the convolutional filter along the data matrix. The learning happens when optimising the loss or objective function for the weights or parameters in these filters while keeping in mind that these weights are shared as the filter moves across the input matrix.

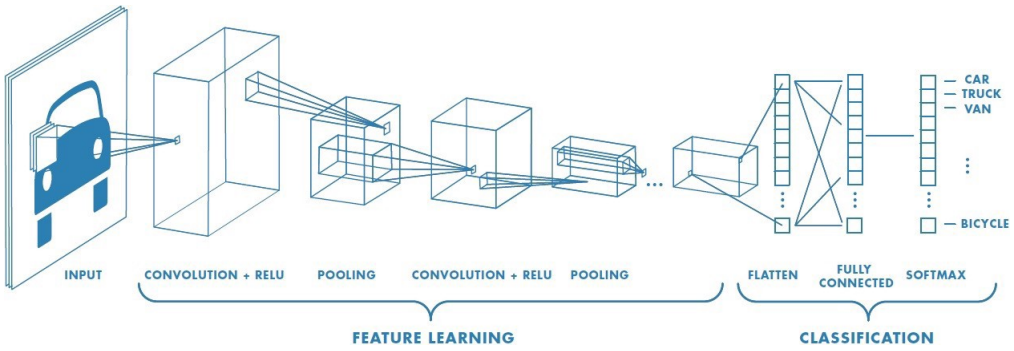

Figure 2.6: An example CNN [41]

Another type of layer is the pooling layer which implements a pooling operation after the convolutional layer to avoid redundancy being captures as the filter strides over possibly overlapping patches in the input matrix. Since we pool the weight

matrix into a smaller matrix, pooling operations also help with increasing sparsity and decreasing the parameter size of easily large CNNs. A visual look into the inner workings of the CNN can be seen in Figure 2.6. As the CNN is also a paradigm of representational learning, its output is a particular set of features which can then be passed into an MLP for learning non-linear combinations of those features, whether for regression or classification with the discussion from the previous section being applicable. In DNA sequence analysis, once one hot encoding is applied, the sequence data is effectively transformed into a matrix resembling an image input which makes it convenient for CNN application [42]. The CNN is then usually used to extract an efficient feature representation from the DNA sequences and passed on to MLPs like we have seen previously with the last layer of the MLP determining whether regression or classification is required.

2.2.3.3 BGRU

We have seen how CNNs are able to extract spatial information in the input data, but when it comes to sequential or ordered data, recurrent neural networks (RNNs) are the standard way to go. When there is dependency between the inputs like, for example, in language modelling, and even perhaps in genetic sequences, RNN design can take advantage of such structure by a simple recursive operation on the gradient update [43]. RNNs model conditional probabilities of any sequence member on the entire history of the preceding members making it seem like an MLP where the hidden layer is a function of both input and the previous hidden layer for the previous member as in Figure 2.7.

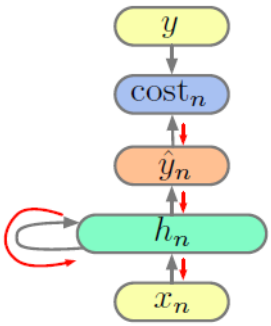

Figure 2.7: An example RNN [37]

The problem, however, is that the dependency on all previous hidden layers for the gradient update leads to a vanishing gradient due to the long multiplications, that is, long-term dependencies can be lost as the gradient shrinks backwards in the sequence for the earlier members. A solution is the gated recurrent unit (GRU) which consists of two parts: the update and reset gate. An example GRU unit can be seen in Figure 2.8 below:

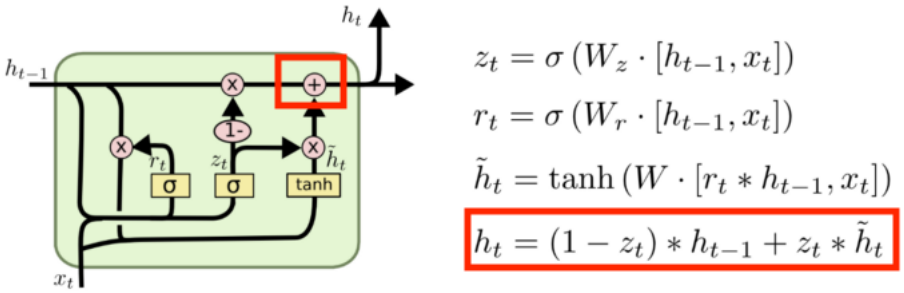

Figure 2.8: An example GRU unit [44]

The h stands for the hidden layer and the subscript *t-1* means it stores information for *t-1* previous layers, and the z variable is the update gate storing previous information and to determine how much of the previous layers needs to be passed. This helps eliminate the vanishing gradient problem encountered earlier. The reset gate r determines how much of the past information is to be discarded based on a similar

formula as the update gate but with a different weight matrix. Current memory content is represented by $\tilde{h}_t$ within which the Hadamard product between the reset gate and the stored information determines the amount that is discarded. The last step is the calculation of h_t which acts as a vector storing information for the current unit and passing it down the network and which uses the update gate Hadamard product with the memory content after reset has been applied to determine how much is passed down. In this sense, GRUs do not just store and pass down information, but they also filter it proactively only keeping relevant information. GRUs have found great success in sequential data processing, including DNA sequences with both word-to-vector encodings and one hot encodings with the latter often not being able to capture the full sequential nature of the data [45].

A specific example of such layers is the bidirectional layers that can be added on top of a GRU layer which has the limitation that the current sequence member prediction only takes into account the previous members. In bidirectional layers, you do not just process sequences from start to end, but in reverse as well. An example of the logic can be seen in Figure 2.9 below. The backward layer just feeds the input from the sequence in the reverse order, starting with in the DNA sequence example from the nucleotide closest to the PAM site. This expands the sequential learning capabilities of the network by analysing the sequences in two orders instead of just one.

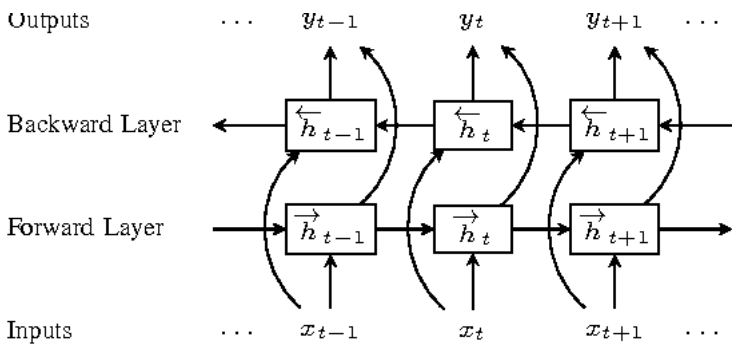

Figure 2.9: Bidirectional layer logic

In the specific case of the GRU, the BGRU layer (Bidirectional Gated Recurrent Unit) consists of two GRU layers stacked up each learning a different direction of the sequence allowing learning dependencies in the past as well as in the future [46]. The use of BGRUs in CRISPR target efficacy prediction has not been extensively researched, and their success with language sequences might imply success with DNA sequence learning as well.

2.2.4 Training

Different types of models mentioned before have different training procedures. In the case of linear and polynomial regression, closed-form solutions exist that allow for deterministic calculation of the weights without an optimisation technique needed (sometimes gradient descent is used, as described further below). In the polynomial case, basis expansion just converts the features to a higher dimensional space while the overall model is still linear allowing for the same solution as in linear regression to apply. Other non-linear regressors like the SVR algorithm introduced have their objective function written as a problem for quadratic programming with linear constraints that is trained through an SMO-type decomposition method assuming

KKT conditions [47]. Usually, non-linear regressors like neural networks resort to gradient optimisation methods to minimise their loss and we have seen that can happen either one sample, several samples, or all samples at a time [48]. What is important when training neural networks is that their objective function is not convex, thus the training process resorts to gradient techniques, and it is hard to know exactly the details of the local minimum reached. Thus, careful validation and tuning of the parameters by extensive hyperparameter sweeps is often necessary to find an appropriate neural network design for each problem.

We have already mentioned that the vanishing gradient occurs in RNNs, but it applies to the general neural network case as well. Looking at the backward propagation equations in 2.20, we can see that the update for the preactivations is a product of many consecutive derivatives of the activation function, i.e. for all the layers after that one. Since these activation functions like in the case of sigmoid can have a very small value and a small slope, multiplying their derivatives several times may cause the gradient update to approach 0. One way to avoid this is to use Rectified Linear Units (ReLUs) instead of sigmoids for the activation functions as they have non-negative derivatives usually [49]. Another important note for neural networks is the initialisation of the weights when the first forward pass is propagated. If all the weights are identically initialised, then they would not be differentiated during training and the neural network would not learn. Initialising weights by sampling from uniform or normal distributions is one way to avoid this [50].

Another common concept for all neural network architectures is that of the optimiser method. It is true that gradient optimisation is the standard, but different methodologies exist to adapt the learning rate, introduce the idea of momentum, and behave differently in the loss optimisation space. A short explanation of these algorithms is below:

1. Gradient Descent: Traditional neural network optimisation technique also used sometimes in linear regression when the computation of the closed-form solution is not feasible. It uses the first order derivatives of the loss function to approach the local minimum but it can get stuck there and its learning rate is fixed. Equation where w are the weights, α is the learning rate, b is the bias, and J is the objective or loss function:

$$\begin{aligned} w &= w - \alpha \nabla_w J \\ b &= b - \alpha \nabla_b J \end{aligned} \tag{2.21}$$

2. Stochastic Gradient Descent: like gradient descent but the parameters are updated after every data sample i, thus more frequent calculations are made and there is more variation in the gradient updates leading sometimes to overshooting a minimum [40]. Equation:

$$w = w - \alpha \nabla_w J\left(x^i, y^i; w\right) \tag{2.22}$$

3. Batch Gradient Descent: like gradient descent but the parameters are updated

after every batch collection of data samples being an improvement on full batch and stochastic gradient descent, keeping the frequent gradient updates but decreasing the variance. Equation where b is now the batch size:

$$w = w - \alpha \nabla_w J\left(x^{\{i:i+b\}}, y^{\{i:i+b\}}; w\right) \tag{2.23}$$

4. ADAptive Moment estimation (Adam): uses the idea of momentum besides just the learning rate like in gradient descent. Momentum speeds up convergence and reduces the variation away from the minimum. Adam uses momentum of the first and second order, allowing for the speed with which we approach a local minimum to be adaptable so as to not overshoot [51]. Equation where the first is the first and second order momentum calculation using the mean and variance of the past gradients and the second is the update rule for the parameters (the other parameters are initialised by default):

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{2.24}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.25}$$

Looking at different ways to optimise the loss function, a short brief is needed on the usual functions representing that loss. In regression, we usually resort to mean squared error (l2 or euclidean norm) between the true and predicted value which is also the starting point of the least squares estimate derivation in the linear regression

case. For binary classification problems which we will tackle when building a discriminator for the nonvalidated samples (preconditioning model), **binary crossentropy** is the appropriate loss function. In classification problems, we are not working with the "value" of the prediction like in regression, but rather the probability that the model gives to each of the classes being allocated to an input sample. Thus, binary crossentropy needs to be able to measure loss with probabilities using the concept of entropy and loglikelihood to quantify this probability-to-loss relationship between two distributions, namely the one for true and predicted values. In the end, we are maximising the probability for the correct class while simultaneously minimising the probability that the incorrect class is assigned [52].

2.2.5 Overfitting

Overfitting occurs when a model fits too well on the training data but does not generalise well to unseen test data. It can be diagnosed by looking at the learning curves of the model, plots of training and testing loss vs fractions of data trained, or in the case of neural networks plots of training and testing loss vs epochs of training. It usually manifests in these curves as decreasing training loss at higher fractions or epochs and an increasing test loss. The model learns the noise or irrelevant features in the training data which do not occur in the test data thereby preventing the usage of the model in real world cases.

There can be multiple sources of overfitting like having a small amount of data in

the dataset where models are forced to learn noise or high complexity of the models which need more data if they are to avoid learning from noise. Overfitting has its foundations in the statistical bias-variance tradeoff [28]. High variance of a model implies that when the model is fit to different sets of data even from the same dataset, the model fits are very different from each other. Overfit models usually have high variance and low bias and improvement is usually observed in terms of test error when more data is added to training.

Other strategies to avoid overfitting when adding more data is not an option is putting a penalty on the magnitude of the weights and/or their number. In linear and polynomial regression, l2 or l1 regularisation implies adding a square or absolute sum term of weights to the objective function that will force the optimisation to balance sparsity and high performance on the training set (see below for l2 regularised regression).

$$\mathcal{L}_{\text{ridge}}(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^{\top}(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \sum_{i=1}^{D} w_i^2 \tag{2.26}$$

For SVR, the parameter C acts as a regulariser whose magnitude is inversely proportional to levels of regularisation. Having a large C forces the margin or regression fit to more stringently fit the training data beyond what is generalisable to unseen points [53]. In the case of kNN, a similar discussion applies. Using a small value of k forces the fit to be much more strict or complex on the training data and the inverse would implicitly add a regularising effect without any added penalty terms for model complexity [28]. As far as neural networks are concerned, l2 regularisation penalty

on the weights is also an option but with a slight difference in implementation from the regression models, while another popular technique is dropout. Dropout makes a certain number of units in the hidden layers it is applied to not be updated at each training step. Which units or nodes get "dropped" is determined randomly at every step but at test time all of the nodes are used with the weights multiplied by the fraction used in dropping. The success of this approach comes from a type of model averaging where every time a set of nodes is dropped, a different model is trained in a way, and at test time, the final model is a sort of average of all of these models thereby helping reduce the variance and the overfitting [36]. Another strategy often applied to the neural scenario is using batch normalisation. In batch normalisation, the internal covariate shifts which represent changes in the distribution of the inputs to deep layers of the network after every batch when the weights are updated causing more challenging learning for the network is stabilised by standardising or normalising the input to a layer for each batch. Normalising the activations of the previous layer means that assumptions made in the next layer about their distribution during the weight update will not change dramatically [54].

2.2.6 Metrics

When working on regression models predicting the off-target cleavage rates, the metric can be the mean squared error, but when it comes to comparing models on their CRISPR target efficiency prediction performance, other metrics are commonly employed like Pearson and Spearman Rank Correlation [55]. The correlation metrics

measure the statistical relationship between our true and predicted values. The metrics output a correlation coefficient between -1 (perfect negative/inverse correlation) and 1 (perfect positive correlation). A coefficient of 0 shows no linear correlation between the true and predicted values and represents the worst performing models since their predictions are not at all related to the true values [56].

The Pearson Correlation Coefficient r measures the linear correlation between two variables and compared to the mean square error takes into account the means of the variables and the sample size. It is computed by dividing the covariance by the product of the two variables' standard deviations:

$$r = \frac{\sum_i (x_i - m_x)(y_i - m_y)}{\sqrt{\sum_i (x_i - m_x)^2}\sqrt{\sum_i (y_i - m_y)^2}} \tag{2.27}$$

Where m is the mean of the respective variable. The Pearson correlation coefficient can be related to the mean square error if the data is mean-centered around 0 and both x and y have been normalized to have an average power of 1:

$$r = \frac{1}{n}\sum_{i=1}^{n} x_i y_i, \quad MSE = 2\left(1 - \frac{1}{n}\sum_{i=1}^{n} x_i y_i\right) = 2(1 - r) \tag{2.28}$$

The Spearman Rank Correlation Coefficient, on the other hand, is a non-parametric measure of rank correlation representing how well the relationship between two variables can be described using a monotonic function. In a monotonic relationship, as compared to a linear one, the rate at which the other variable is affected by changes in the first is not expected to be constant. In that sense, the Pearson coefficient

is a special case of the Spearman coefficient when the relationship is assumed to be linear. Furthermore, the Spearman coefficient works with rank ordered variable data and not with raw data as the Pearson coefficient does [56]. In other words, instead of using covariance and standard deviations of the samples themselves, the statistic is calculated from the relative rank of values on each sample:

$$\rho = 1 - \frac{6 \sum d_i^2}{n\left(n^2 - 1\right)} \tag{2.29}$$

Where ρ is the Spearman rank correlation coefficient, d_i is the difference between two ranks of each sample, and n is the number of samples. The Spearman coefficient is also between the values of -1 to 1 indicating levels of negative to positive rank correlation.

2.2.7 Hyperparameter Tuning

Previously repeatedly mentioned parameters or hyperparameters are important parts of models that determine how fast they converge and what levels of regularisation or complexity are enforced. Since the test data cannot be touched during training, the training data is usually split into training and validation sets with the purpose of, for a specific combination of hyperparameters of that model, the model will be trained on the training set and evaluated on the validation set. The set of hyperparameters that yields the best performance on the validation set is the one to be chosen.

There are different methods to implement different training and validation scenarios

for different hyperparameter combinations including grid search and bayesian optimisation. In grid search, all possible combinations of hyperparameter values are implemented for validation and the one that gives best performance of the model on the validation set is chosen [57]. Grid search does not use any prior information of the success of previous hyperparameter combinations to inform which combinations to focus testing next. In bayesian hyperparameter optimisation, we resort to modelling the probability of the objective function and using it to select the optimal hyperparameters conditioned on the performance of previous combinations. Bayesian optimisation, thus, can be more computationally expensive because the algorithm requires keeping information on the performance of past combinations but it narrows down the search space which is its unique advantage [58].

To help reduce variance in the validation error, KFold crossvalidation can be used to divide the dataset into K folds, and use K-1 for training with the Kth one for validation. This allows the model to train and be validated on all of the data, with the final validation performance being an average of the K individual validation measures. Choosing a larger K can lead to less bias towards overestimating the error with training folds closer to the total dataset, but with higher variance of the error [59]. Some default values for K include 3, 5, and 10 with 10 being used for larger datasets.

2.2.8 Shapley Values

Shapley values is a concept from game theory meant to indicate how much a player in a coalition in cooperative games should get from the shared payoff. The amount a player gets is determined by their estimated contribution to making the coalition "successful" at achieving the payoff, i.e. the importance of each player. Transferring that to the machine learning scenario, features can be taken as players, and the "successful" coalition payoff is the change in the prediction, meaning the prediction is the game. The Shapley value is the average contribution of a feature value to the prediction in different coalitions (sets of features) [60]. Thus, Shapley values of features in machine learning can tell use what each feature's contribution was to the model's prediction. In more mathematical terms, the below equation represents the calculation of Shapley value for feature/player i:

$$\phi_j(val) = \sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_j\}} \frac{|S|!(p-|S|-1)!}{p!} \left(\text{val}\left(S \cup \{x_j\}\right) - \text{val}(S)\right) \tag{2.30}$$

Where p is the number of features, S is a subset of the features used in the model, x is the vector of feature values of the instance to be explained. One interpretation of the equation is to think of a set of features being created one feature at a time with each feature's contribution being the amount the payoff or prediction changes by their addition. For each feature, take the average of its contribution to all possible permutations of sets of features it can be a part of. The Shapley value of a feature is then the average change in the prediction that the subset of features receives when

the feature joins it [61]. Another way to interpret the equation is to say:

> *To calculate the Shapley value of a specific feature j, sets of all possible unions are formed with all p features except feature j. The value of the j-th feature is obtained via calculating the difference between the results of the characteristic function val on the set of all features and S (the subset without feature i). Shapley value of a particular feature j is then calculated by taking the average of the marginal contributions of all possible combinations of the feature unions* [62].

The Shapley value has a particular property called efficiency which states that the feature contributions must add up to the difference of prediction for the value of the feature subset and the average which implies that that difference is fairly distributed among features, an advantage over other explainability methods like LIME [61, 63]. The Shapley value does not have to be computed using the entire test set, but can have contrastive explanations, meaning a prediction could be compared to a subset or even to a single data point of the dataset. The downside of the Shapley value is that it is computationally expensive to compute, as it relies on using all possible coalition and subsets of features. There are ways, however, to limit the number of iterations or by randomly sampling feature subsets but that also increases the variance of the value. Another salient point is that the Shapley value does not measure the impact on model performance or prediction when a feature is removed, rather the contribution of that feature to the difference between the actual prediction and the mean prediction. While these limitations are important to consider, for our explainability purposes and discussion, the Shapley value should suffice in determining

which of our DNA features are of relatively higher relevance to the prediction.

2.2.9 Ensemble Learning

Ensemble learning is the procedure by which several machine learning models are combined to give overall better system performance at a task, either regression or classification prediction. The general idea is that by combining several weaker prediction models together one might achieve better performance. There are several types of ensemble learning with the first usually introduced as bagging. Bagging or bootstrap aggregation is when different training data subsets are randomly drawn from the entire training dataset with each subset used to train a different model of the same type. The output of each of these models is then averaged (in the case of regression) or majority-picked (in the case of classification) as the final model prediction [64]. We introduced a similar concept earlier when talking about dropout in section 2.2.5 which effectively might amount to the same thing which is that the ensemble attempts to reduce the variance when using any one of the weaker models alone. In bagging the models are independently trained from one another and their output is generated in parallel. No correlation or information sharing is expected.

Another example of ensemble learning is stacking or stacked generalisation, where there are two levels or tiers of models. In the first level, different weaker or simpler models are trained on bootstrapped training data independently but then their outputs are used as inputs to the second level or meta-model that can hopefully

learn to leverage the advantages and weaknesses of each different tier 1 model to output a more accurate final prediction. The difference with stacking is that the goal is to decrease the bias of the prediction system and that sometimes comes with a decrease in variance as well while also using different types of models for the first level or tier [65]. Stacking is usually implemented with crossvalidation, that is the level 1 classifiers are trained on K-1 folds with prediction on Kth fold which is then used to train the meta-model. Such an approach will be adapted in this dissertation [64]. An illustrated look at stacking can be seen below:

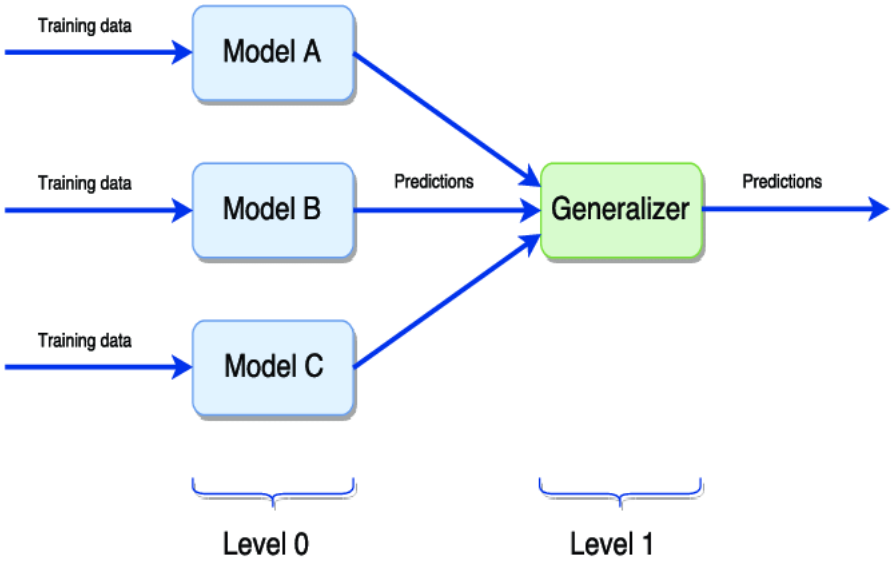

Figure 2.10: Example of a Stacking Architecture [66]

More will be said in methodology on the details of our implementation of ensemble learning methods but it can be stated a priori that, in stacking, a primitive form can include tier 1 models to be procedural or deterministic and not based on learning and optimisation. In that sense, the first tier models are in a way learning how to convert the training data into outputs which are new features for the meta-model.

Methodology

3.1 Datasets for CRISPOR Reproducibility

Our first task was to construct the off-target cleavage dataset used in the CRISPOR integration paper for validation of our procedural algorithms. We manually extracted the sequence pair, cell line, and cleavage frequency data from 8 published genomic studies. The dataset consisted of 650 off-target sequences for 31 different guides after processing while two of the studies did not validate their off-targets with polymerase chain reaction (PCR). One challenge from assembling data from different experiments is that the results have different sensitivity levels. The Hsu and Cho datasets have off-targets with a modification frequency lower than 0.001 % while the others estimated their sensitivity at 0.1–0.2 %. This makes it harder to compare the less frequent off-targets found in the two studies with those from whole-genome assays. Thus, based on a similar approach as in the original paper, we set a minimal threshold to modification frequency of 0.1 % for off-targets [18]. There are two score schema implemented in the integration paper: one looking at only off-targets with up to 4 mismatches, and the other including the few cases that

had 5 and 6 mismatches as well. The cleavage frequency where not provided in some of the studies was calculated by dividing the number of all successful genome insertions or deletions by all observations at the corresponding off-target site [18]. For our purposes, we only needed to show the reliability of our scoring algorithms on one of these schema and we chose the first one because most of the genomic studies have separate supplementals for the second case which would require unnecessary manual work integrating those into the far more numerous first case. A short description into the processing of each of these datasets is below.

3.1.0.1 Hsu et al. 2013

The Hsu study published in 2013 used targeted sequencing of predicted sites to report 37 off-targets with 4 unique guides. They used PCR+ sequencing to validate the targets on the two cell lines, HEK293T and HEK293FT. The reported sensitivity was <0.00001 % meaning that their identified off-targets include rare cases especially compared to those obtained in the other studies. The testing done on this dataset was used to determine the weight matrix for off-target scoring for the MIT score based on positional mismatches identified [19].

3.1.0.2 Cho et al. 2014

The Cho study published in 2014 also used targeted PCR sequencing of predicted sites with 3-8 mismatches adjacent the 5' end to report 77 off-targets with 10 unique guides. They similarly used PCR+ sequencing to validate the targets on the K562

cell line. The reported sensitivity was <0.00001 % meaning that their identified off-targets include rare cases especially compared to those obtained in the other studies like in the Hsu case. The Hsu and Cho data has missing cleavage data which needed to be calculated by dividing number of indels by all observations on the off-target [67].

3.1.0.3 Frock et al. 2015

The Frock study used whole-genome translocation sequencing (HTGTS) of junction sites to report 86 off-targets with 4 unique guides. They did not validate their off-targets which might increase the number of false positives, and they used the HEK293T cell line. Because they did not use targeted sequencing, no sensitivity levels were reported. Most of the 5 and 6 mismatch cases were obtained in this study but because they were not validated their low modification frequencies are not fully reliable. Additionally, they did not directly quantify indels but counted the lentiviral insertions which samples infrequent cleavage events and can overestimate real modification frequencies [18, 68].

3.1.0.4 Tsai et al. 2015

The Tsai study published in 2015 used whole-genome GuideSeq detection of strand breaks to report 403 off-targets with 10 unique guides. They did not validate their off-targets and they used the HEK293 and U2OS cell lines. They did not validate the off-targets but their reported sensitivity was about 0.1 %. This study interest-

ingly shows that some off-targets sites are not predicted by algorithms (nonvalidated samples that we will explore in the last part) especially in the case of targets involving 1bp indels (bulges) [18]. This study as well did not validate off-targets with PCR amplicon sequencing and can include false positives [69].

3.1.0.5 Kim et al. 2015

The Kim study used whole-genome DiGenome-Seq and PCR sequence detection to report 12 off-targets with 2 unique guides. They validated their off-targets with targeted PCR sequencing and they used the HAP1 and K562 cell lines. Their reported sensitivity was about 0.1 % [70].

3.1.0.6 Wang et al. 2015

The Wang study used Lentiviral integration site sequencing to detect viral integration to report 13 off-targets with 2 unique guides. They validated their off-targets with targeted PCR sequencing and they used the HEK293T cell line. Their reported sensitivity was about 0.5 % [71]. This study also showed that some off-targets sites are not predicted by algorithms like CRISPR Design Website and ECrispr including cases like targets involving bulges [18].

3.1.0.7 Ran et al. 2015

The Ran study used whole-genome BLESS (breaks labelling, enrich. on streptavidin, seq.) to detect strand breaks and reported 17 off-targets with 2 unique guides. The study was not investigating off-targets per se, rather demonstrating the efficiency of a novel Cas9 mechanism. They validated their off-targets with targeted PCR sequencing and they used the 293FT cell line. Their reported sensitivity was <0.00001 % [72].

3.1.0.8 Kim et al. 2016

The more recent Kim study used a newer DiGenome-Seq2 + PCR approach to detect 30 off-targets with 10 unique guides with whole-genome sequencing to find CRISPR-induced modifications [18]. They validated their off-targets with targeted PCR sequencing and they used the HAP1 cell line. Their reported sensitivity was the same as in the earlier study: about 0.1 % [73].

3.2 Base Models

3.2.1 Procedural Scores and CrisprSQL Processing

While there are other procedural scores besides the 4 included here, we decided to follow the established methods and results proposed in the integration into CRISPOR paper [18]. The CRISPOR paper showed the relative performance measures of MIT,

CFD, CROP-IT, and CCTop on this selection of datasets described in the earlier section. To test our implementation of these scores, we compiled these genome experimental datasets for off-target prediction and applied our algorithms for procedural score comparing them to the results obtained in the CRISPOR paper. One point of note is that CRISPOR used these procedural scores for classification and not regression. They set a off-target activity threshold for each of the procedural scores above which binding would be classified as on-target and below which as off-target (for example, 0.023 is the cutoff value for CFD scoring) [18]. This is a minor difference since it just choosing a rather artificial threshold to separate values into two classes, whereas we will be working with the real valued activity estimates directly. When replicating the results of the paper, we will be using the classification threshold. A difference between the CRISPOR implementation and our CrisprSQL anaylsis is that the off-target cases considered in the paper and reproduced here only include case of up to 4 mismatches while our implementation of CrisprSQL used example of up to 5 mismatches as those are included in the database.

Because verification of our procedural algorithms and reproducibility of the CRISPOR integration paper is not one of our main results, the implementation details of that part of the dissertation aim will be included here and not in the results section. It is important that our method implementing the procedural scores are validated and their performance compared to existing benchmarks before implementing them on a larger and more comprehensive dataset like CrisprSQL.

Figures 3.1 below has been retrieved from the original integration CRISPOR publication showing the performance benchmarks of the procedural scores on the genomic datasets assembled from the above. Figure 3.2 shows our reproduction of the procedural score results on the genomic datasets which we manually assembled from the original sources and supplementary data.

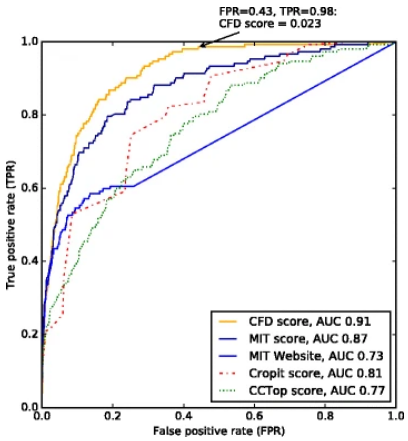

Figure 3.1: ROC of Procedural Scores on A Collection of Genomic Datasets [18]

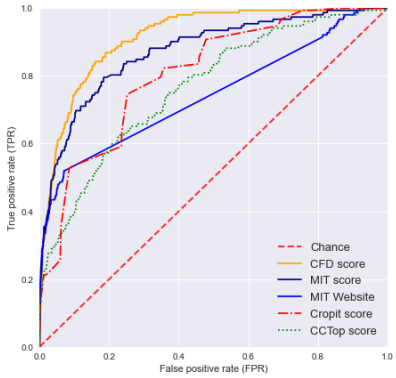

Figure 3.2: ROC of Procedural Scores on A Collection of Genomic Datasets Reproduced

MIT score refers to the MIT off-target score as calculated by the CRISPOR website, MIT Website refers to the MIT off-target score as calculated by the CRISPR Design (MIT) website. The latter values were retrieved from the GitHub repository directly as the website is no longer active [74]. The reason the MIT website score is more diagonal in our reproduction is that we imputed the values using the mean

instead of leaving them as 0 like the original paper did [18]. Otherwise, we can see that our implementation of the procedural scores successfully reproduces the results from the integration paper, thus we have confidence in extending the algorithms to CrisprSQL data calculation.

Once we have verified the fidelity of our procedural score mechanisms by evaluating them on existing genomic datasets, we apply them to the CrisprSQL database. CrisprSQL has 25,632 samples of target and off-target pair information beyond just sequences including energy levels and chromatin information. The label of cleavage rate combines several meaures within it which the researchers classified under the term. Measures subsumed withing the cleavage rate include mutation frequencies, indel frequencies, relative read counts converted by using the read ratio relative to the sum of reads for a given guide, etc. Furthermore, in the CrisprSQL paper, the label is transformed to a Gaussian with zero mean and variance σ=2 using the non-linear Box–Cox transformation based on a similar approach taken by Listgarten et al. [8, 75]. We decided to work with the cleavage frequencies directly as targets and not apply transformations on our labels for regression.

Our initial analysis will focus only on sequence information and extracting relevant features be it through procedural scores or deep learning models. Some target sequences, however, will have triple dashes signifying where the cleavage was done, and those dashes were removed. Single dashes indicate that the nucleotide in question in the target sequence could be any of the 4 nucleotides, thus we decided to

replace all of the single dashes with the corresponding nucleotide on the guide RNA. Such an approach has been utilised previously by CRISPR-Net [76]. Additionally, we only consider the first 20bp of the sequence for analysis besides the PAM of course as most of the procedural scores implemented do so. To maintain consistency among the base models, the same sequence length will be used when training the deep learning models. Finally, as far as missing values are concerned for some of the cleavage rates, instead of removing them, we decided to impute them using the means of the cell line cleavage rates for each sample that is missing. Missing values make up less than 4 percent of the total number of samples, so we do not feel that imputation would affect the result significantly. When it comes to CFD scoring, 1872 samples or target sequences had PAM sites not supported by the CFD scoring thereby missing the weights for those PAM sites and getting a score of 0.

Following the calculation of the scores by the procedural algorithms, we will calculate the Pearson and Spearman correlation coefficients between the scores and the cleavage rate (label, target) for each of the models and compare their relative performance. Since the real values of the scores will be used and not an individual conversion to binary classes, the relative scaling of the scores becomes important for machine learning model input. MIT, for example, is on a scale of 0 to 1 while CROP-IT has values of up to 650. To maintain internal consistency in the features, we applied standard scaling by fitting the training data only and transforming the entire dataset. When using crossvalidation and to avoid leakage, we will be fitting the standard scaler to each iteration of crossvalidation on the training folds only.

Existing efforts of off-target activity prediction like the procedural scores mentioned are standard and usually used for comparison purposes. Their capacity to predict the levels of activity vary between them and their algorithms are not based on learning or optimisation, rather straightforward application of formulas whose weights have been derived experimentally. Thus, machine learning models might be able to outperform these existing tools by learning from sequences directly or by using the procedural scores as features. Our contribution in this dissertation, in part, will be using these established procedural mechanisms as feature extraction tools and as "base models", the first time such an approach has been undertaken in the off-target scenario. We will also apply Shapley value analysis on the machine learning model using the procedural scores as features to identify which procedural mechanisms are most influential in making a prediction, relating back to the underlying differences between what each procedural score measures. More details on the frameworks we decide to develop in integrating these procedural scores as features is in Section 3.3 further below.

Furthermore, deep learning has been applied to the problem of off-target prediction with several noteworthy examples like those in Lin et al., CnnCrispr, and DeepCrispr [6, 4, 7]. Each of the models has a slightly different approach as to how they predict off-target activity which will be described further in the next section. Lin uses both MLP and CNN models on one hot and OR encoded sequences, while DeepCrispr uses a pre-trained encoder in a siamese-type network. Another contribution

we make is to implement BGRU and siamese BGRU (names 2BGRU) architectures and compare them to existing deep learning models like the ones proposed in Lin et al. A step further is to create a larger and more sophisticated ensemble learning model that would combine these deep learning models with procedural scores for fast but accurate prediction of off-target activity levels. Thus, there are two phases of base model implementation, the first is using just the procedural scores combined as features, while the second is using them in conjunction with deep learning models extracting features from the sequences directly.

3.2.2 Deep Learning Models

Several deep learning models have been proposed for different experimental studies of off-target profiles, however, due to the data sizes of the studies and variation across experiments, performance measures have also had high deviation. The hope is that with the combination of experimental data and its standardisation into a large dataset like CrisprSQL, deep learning models applied for the first time to this data will produce more stable results [8, 6]. Lin and Wong proposed in 2018 the first deep learning framework for off-target prediction and their two best performing architectures will be applied here in the case of CrisprSQL as an extension of their work but also as a base of comparison for our own deep learning architectures [6].

3.2.2.1 LinnFNN3 and Linn

These deep learning models developed by Lin and Wong only need sequence pair information represented as a 23 character long string. Each character is a nucleotide base, A, T, C, or G, and the last 3 characters are the PAM site of the respective target sequence. The two sequences for a sample are one-hot encoded to create 2 4x20 matrices of binary code, one for the guide RNA and the other for the target sequence. The two matrices are then OR encoded by performing the OR operation between the mismatches of the two sequence matrices. A more detailed look can be seen in Figure 3.3 below:

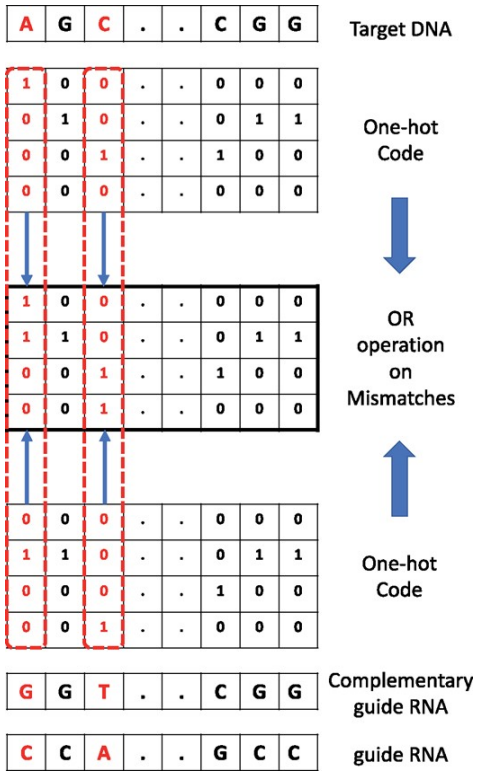

Figure 3.3: OR encoding of sgRNA-target sequence pair [6]

This generates only one 4x20 matrix from both sequences which can be appropriately passed on to neural networks for learning. Depending on the model, Linn or LinnFNN3, since they have a different architecture, the input matrix will have to be

formatted. In the case of Linn, or CNN_std how the classification version of Linn is named in the Lin and Wong paper, the input can be formatted as image input into CNNs of the matrix form 20x4. The Linn architecture we implemented is based on the CNN_std (Figure 3.4) with the following differences: first, we do not consider the PAM, or the last 3 nucleotides of the sequence so our input is 20x4 (the reasons have been stated in the previous section), second, the output layer consists of one node without a softmax activation but rather a linear activation due to us working on a regression and not classification problem. The convolutional layer consists of 40 filters whose dimensions have been adjusted accordingly with a ReLU activation function, and the batch normalisation layer is added to stabilise learning and reduce overfitting. The global max-pooling layer has a window size of 5 and the outputs are flattened before being fed into an MLP. The MLP has two hidden layers, 100 and 23 neurons each, with the second layer preceeding a dropout layer with 0.15 probability connected to a final output node with the linear activation function.

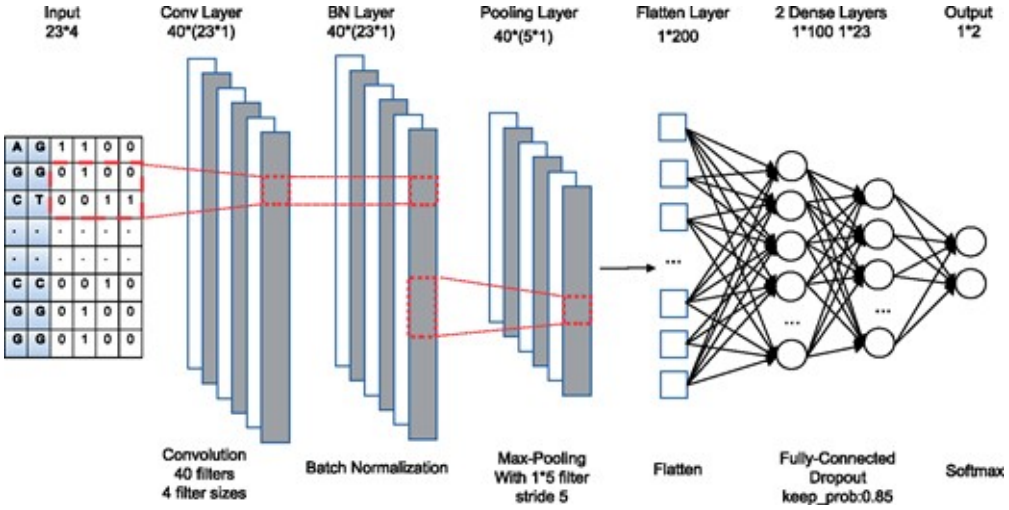

Figure 3.4: CNN_std architecture from which Linn model was adapted [6]

The second architecture adapted was FNN_3layer which we re-dubbed LinnFNN3

in our version. The LinnFNN3 is a relatively simple MLP with 3 hidden layers with 50, 20, and 10 neurons each respectively. The hidden layer activation functions are ReLU and the output node again has a linear activation function. Since MLP require flattened input, we formatted the sequence 4x20 matrix into a 1x80 input vector. Both of the original architectures, CNN_std and FNN_3layer, have outperformed the procedural scores in classification on the CRISPOR dataset as can be seen in Table below.

Model	**Mean AUC**	**Var AUC**
FNN_3layer	0.970	0.005
CNN_std	0.972	0.010
Logistic Regression	0.931	0.018
CFD	0.912	0.027
MIT	0.728	0.063
CROPIT	0.807	0.022
CCTop	0.776	0.029

Table 3.1: Performance of different models under stratified 5-fold cross-validation on CRISPOR dataset [6]

Mean AUC stands for the mean Area Under the Curve of the ROC curve across all 5 folds and Var AUC for the variance. The two models we used as a baseline to develop our own Linn and LinnFNN3 models for off-target activity prediction outperform standard procedural score models. We will investigate whether similar performance ability is maintained in the regression realm on the larger and more varied CrisprSQL dataset.

3.2.2.2 BGRU and 2BGRU

The previous deep learning models of Lin and Wong, including DeepCrispr and CnnCrispr, all tried representing the genetic sequence pair matrix as an image where CNNs will be able to learn the spatial properties of the matrix and be able to extract relevant but abstract features from the OR-paired sequences. A further approach is to go beyond just spatial learning and explore learning the natural sequential nature of the DNA data. In 2020, a paper suggested implementing RNNs and CNNs in a combined effort with the BGRU framework in predicting *on-target* activity [77]. Based on their approach, we decided to design a similar BGRU framework for our problem with the following distinction: first, we are predicting off-target activity score and not on-target, an inherently harder challenge for the models, second, we will not be using any epigenetic features but only learn from sequence data directly not needing a siamese-style network to integrate learning from epigenetic information, third, we will only look at the binding 20bp region without the PAM site, and fourth, we will train on the larger CrisprSQL dataset without being limited by dataset size to have to resort to transfer learning. Our investigation will try to explore whether learning sequential dependencies can aid in the difficult off-target activity prediction problem. The BGRU architecture we implemented can be seen in Figure 3.5 below.

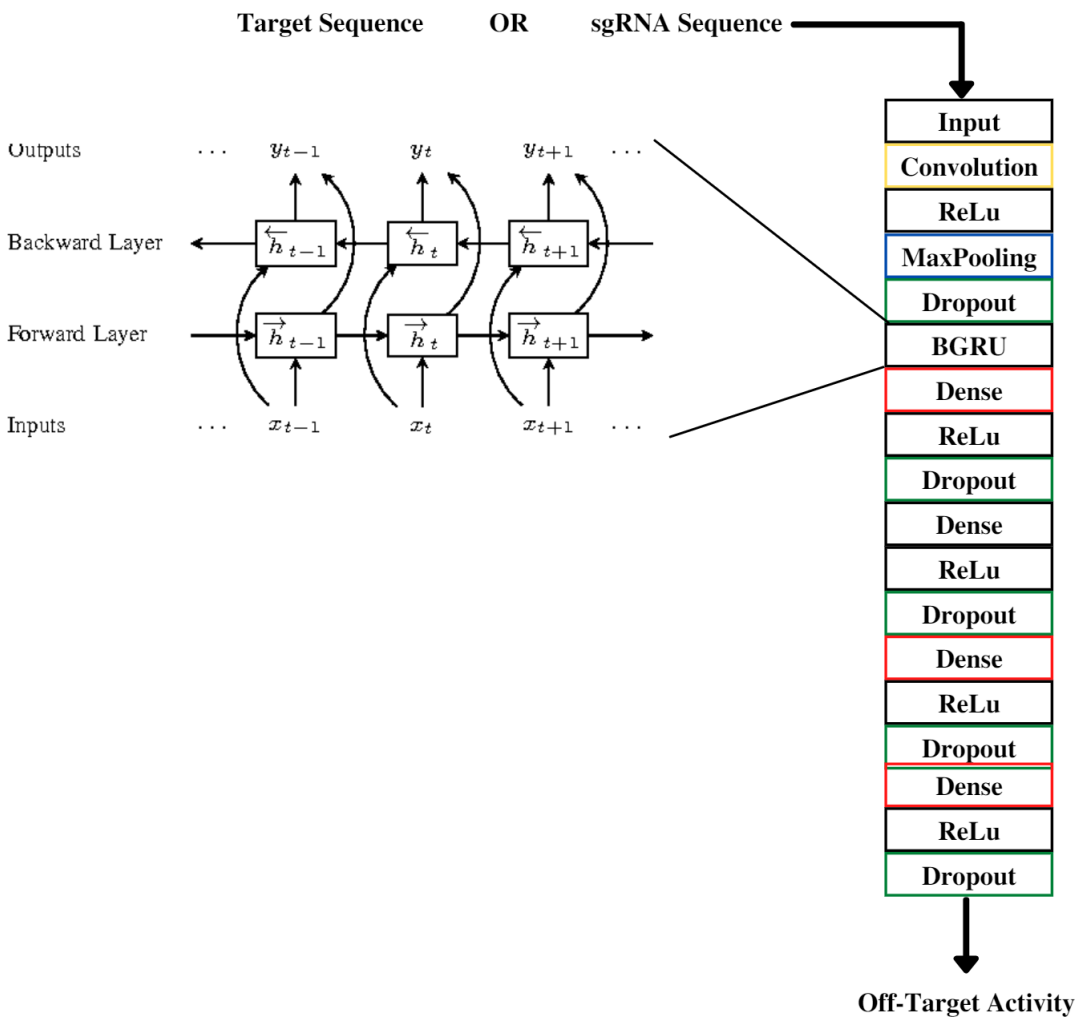

Figure 3.5: Our BGRU Architecture adapted from [77]

As in the Linn model, the input matrix to the convolutional layer has been formatted as a 20x4 matrix from OR encoding where the abstract features will hopefully be extracted. CNNs perform well at capturing local patterns in sequence data by using weight-sharing strategy but not as well at learning sequential correlations [77]. The BGRU layer will be able to learn the sequential dependencies present in the DNA sequence, followed by an MLP with 4 hidden layers (256, 128, 64, 40 hidden units respectively) with the last layer flattened before the output linear activation.

An assumption made in all of the previous deep learning models is that the OR encoding presents a useful way to combine sequence encoding information from pairs of sgRNA and target sequences that will allow the models to learn the underlying patterns that lead to increased off-target activity and decreased specificity. This assumption might be more of a burden than a practical way to format data. Perhaps the OR operation collapses information present in both separately encoded sequences that will be lost or not learned during training on that data. To explore this further, we propose a siamese network architecture that will have two parallel and identical neural networks learning from each of the one-hot encoded sequences (sgRNA and target) directly without needing to apply any additional OR encoding. Our hope is that removing this assumption will lead to more natural learning of each sequence separately and the relevant features before combining that information in making an off-target activity prediction. Since the BGRU network presents a sophisticated approach to learning both local and sequential features, it will be the baseline for the 2BGRU siamese network architecture proposed in Figure 3.6 below.

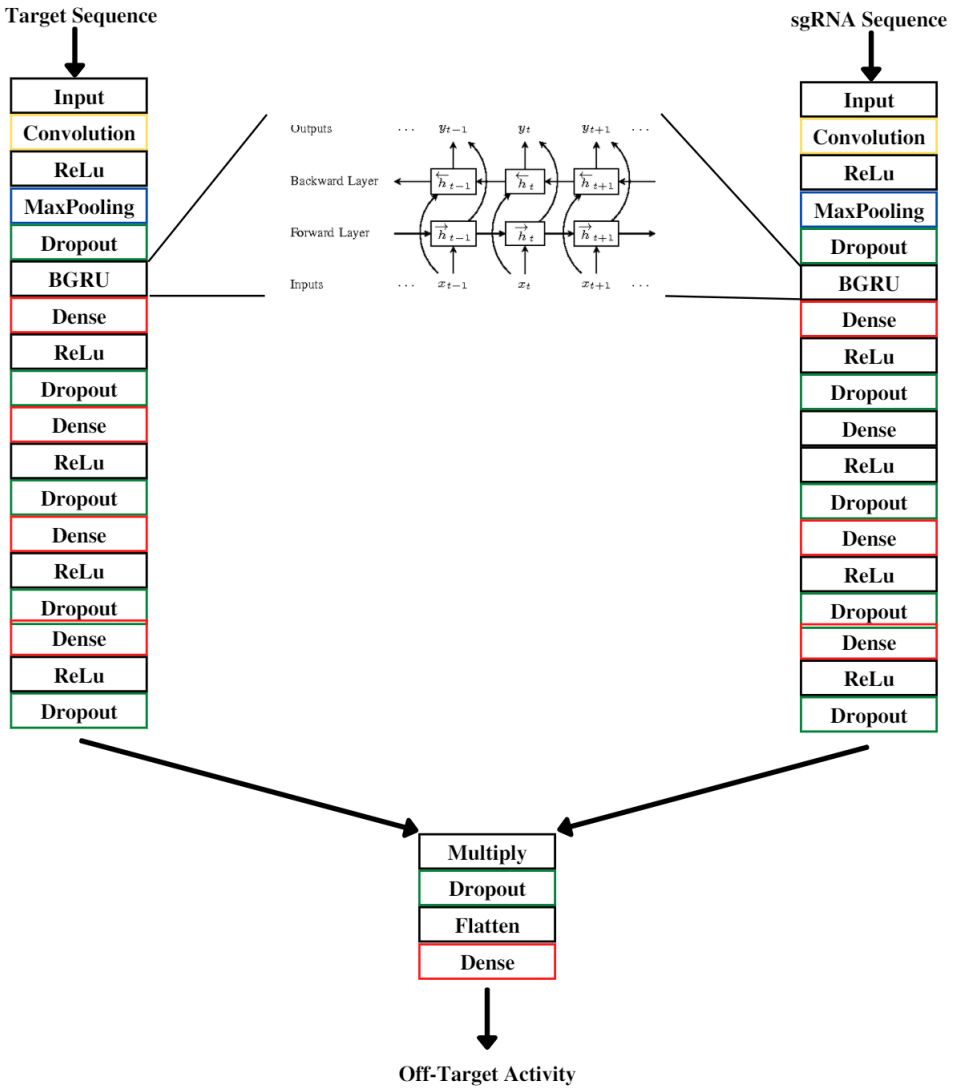

Figure 3.6: Our 2BGRU architecture built on the proposed BGRU model

We could not identify in current published research a similar approach taken to the off-target prediction problem and this architecture represents one of our main contributions in the project. The input data consists of pairs of one-hot encoded sgRNA and target sequences in 20x4 format zipped together and passed to the model where each is used for training by a different twin of the network. The architecture details are similar to the BGRU above, the difference is that the twin network outputs are not flattened but instead each passed to a shared multiplication layer which is then

flattened and the output node is linearly activated.

3.3 Meta-Model Ensemble Selection and Hierarchical Models

The procedural scores obtained from the CrisprSQL dataset will be used as features and input into a meta-model mimicking a simple or primitive ensemble architecture where the base models are not learning models as illustrated in Figure 3.7.

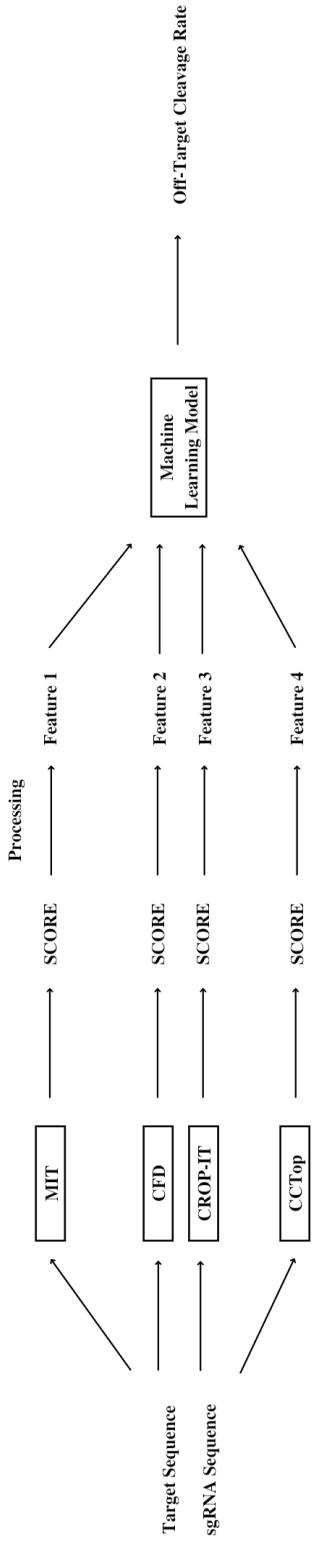

Figure 3.7: Primitive ensemble method for score feature learning

The Pearson and Spearman correlation coefficients will then be calculated between

the prediction of this ensemble and the true values of cleavage rate as well as plots of mean square error loss plotted to diagnose the meta-model's ability to learn to combine these procedural scores. We will implement several model choices for the ensemble meta-model including 4 different MLP neural network architectures as described in the table below, kNN, linear and polynomial regression, and SVR.

Model	**Description**
Architecture I	4 hidden layers and 5 hidden neurons in each
Architecture II	4 hidden layers and 16 hidden neurons in each
Architecture III	10 hidden layers and 5 hidden neurons in each
Architecture IV	10 hidden layers and 16 hidden neurons in each
Linear Regression	Implemented polynomial kernel as well
SVM	SVR for regression
kNN	Tuned for optimal k on validation

Table 3.2: Proposed architectures for learning from procedural scores as features

We have seen how deep learning models have been applied to outperform the predictive power of procedural scores and while that will also be a topic of this dissertation, our aim is to show that by using a simple ensemble with a neural network learning to combine these procedural scores for off-target activity prediction can also help improve their predictive power.

The next step is to learn from sequence data directly without using procedural scores as feature transformers. To that end, combining the deep learning models into the ensemble alongside the procedural scores will follow the basic structure described in

Figure 3.8 below.

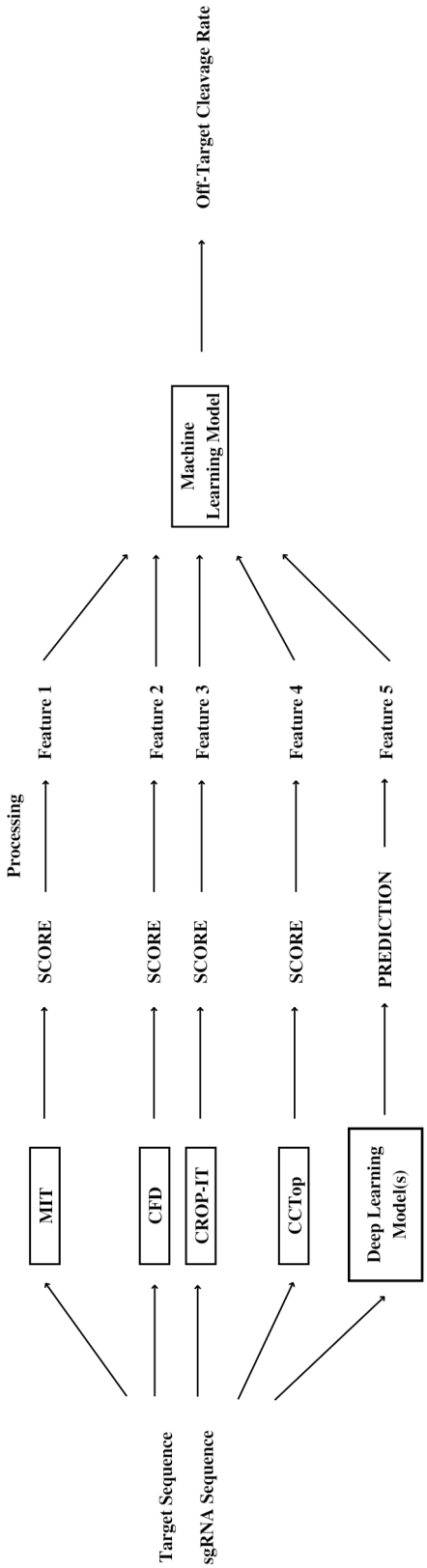

Figure 3.8: Our ensemble design with procedural scores and deep learning sequence models

It will be evident later but of the 4 deep learning models we are implementing, some groupings in terms of prediction distributions between them can be observed. For example, two of the models can tend to overestimate cleavage frequencies below a certain threshold while the other two underestimate. Because of that, having more than one deep learning model in the ensemble that shows a different manner of learning and prediction might help improve the overall predictive power of the ensemble through a best-of-both-worlds approach. Again, the meta-model learning to combine the scores of the procedural methods and the predictions of the deep learning models will either be a neural network or a kNN regressor (after experimenting with several other machine learning model candidates including linear/polynomial regression and SVR).

Furthermore, so far we will be working only with experimentally validated off-targets, i.e. those that were confirmed to occur in genomic studies whereas all the others predicted by algorithms are termed nonvalidated or inactive. A specific challenge is having an off-target activity prediction model that will be affected minimally by including the nonvalidated samples in the validation (not training) data and will be able to separate the active from inactive off-target sites while also giving good activity scores for the active cases. In our regression models, we are learning the complex fits to be able to predict activity scores, but enforcing the task to simultaneously be able to classify or separate the nonvalidates samples (with scores of 0 since they do not occur) might not be successful. To help resolve this, we will resort to a hierarchical system where we develop a separate classification model to

learn to separate the nonvalidated from the validated samples first, to the extent that is possible. In effect, the binary classifier acts as a preconditioning model. After that, the validated samples will proceed on to be learned by the ensemble for off-target activity prediction. Combining this binary classifier with the ensemble consisting of several machine learning models in 2 levels or tiers leads us to propose the hierarchical system in Figure 3.9 below. It is important to note that the term "validated" and "nonvalidated" has nothing to do with whether the samples form parts of the validation or result-reporting set in our experiments as those are two different concepts. Indeed, nonvalidated samples will be part of 5 different validation sets at varying ratios explained further below the figure.

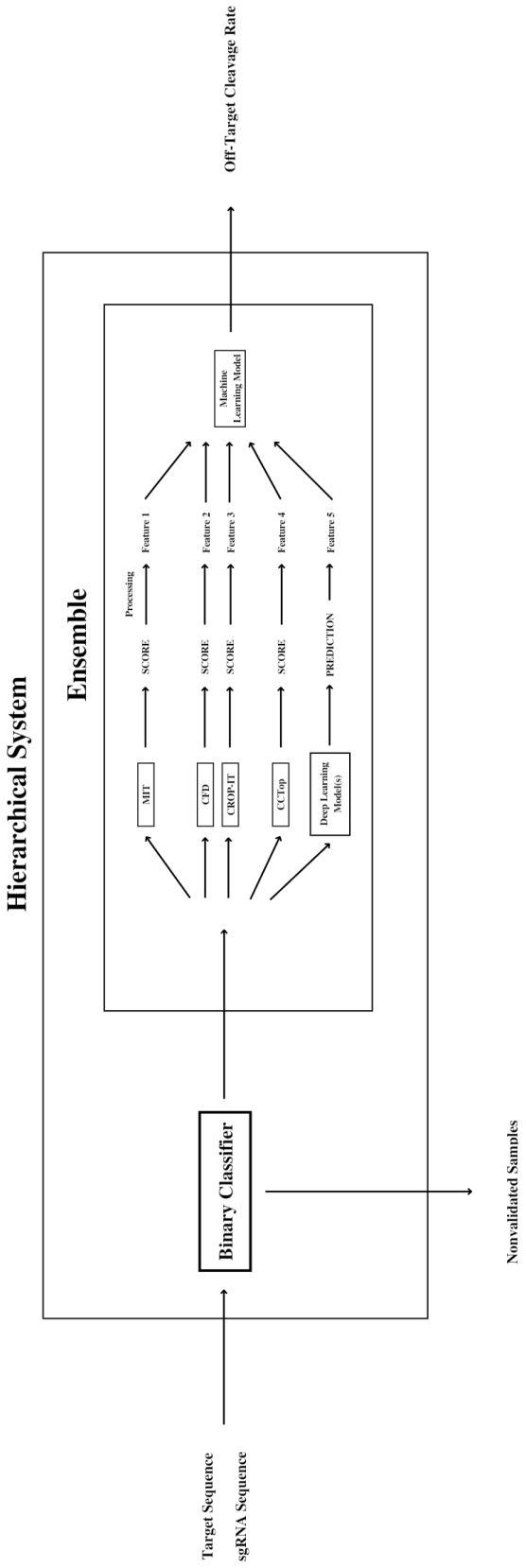

Figure 3.9: Hierarchical system for robust off-target activity prediction

To evaluate the robustness of each model architecture or its resilience to these non-validated samples, 5 evaluation scenarios will be created:

1. 0 which will contain only validated samples from CrisprSQL;
2. 1/1 which will have 1/2 of samples be validated samples from CrisprSQL and 1/2 nonvalidated;
3. 1/10 which will have 1/10 of samples be validated samples from CrisprSQL and 9/10 nonvalidated (almost order of magnitude difference);
4. 1/100 which will have 1/100 of samples be validated samples from CrisprSQL and 99/100 nonvalidated (almost two order of magnitude difference);
5. 1/250 which will have 1/250 of samples be validated samples from CrisprSQL and 249/250 nonvalidated;

The following section will detail the training and validation procedure for these models and systems as well as our mechanism for hyperparameter tuning of each of the learning models.

3.4 Implementation

Python3 was used to both program the procedural scores and test them, as well as all of the machine learning, interpretability, and visualisation methods. Python3 was chosen due to our familiarity with its programming norms, as well as the overwhelming presence of the programming code in the machine learning community.

The packages included in Python3 allow for easy transferability of methods alongside CUDA integration for training. All of the code and implementation was completed on a personal computer. Google Colaboratory was used primarily because of the access to a remote GPU including Nvidia K80s, T4s, P4s and P100s and 13GB of free RAM. These resources proved vital for the completion of the experiments including tuning, training, and visualising the more advanced methods employed here, especially the large 2BGRU model. There were enough experiments that did not require using these resources, so for sustainability purposes, these tasks were performed on the personal computer sporting an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, 16GB of RAM, and GPU Nvidia GeForce 940MX with DirectX 12.

We used Keras and Tensorflow to create, train, and save the machine learning models because of our familiarity with the packages. We regularly depended on Pandas especially for data processing, Numpy, Seaborn and Matplotlib for visualisation, sklearn for crossvalidation and data splits, SciPy for calculation of the correlation metrics, and SHAP for the calculation and visualisation of Shapley values of the features.

For hyperparameter tuning, we used Weights&Biases, an online platform that allows for Bayesian Optimisation (applied here) of listed values for hyperparameters including number of layers in neural networks, hidden units, activation functions, as well as any custom arrangement for a multitude of machine learning models. Validation plots and feature importances as well as details of tuning are included in the

Appendix.

Similar to the approach in Lin and Wong, we will be using 10-fold crossvalidation to display mean and variance values of our Pearson and Spearman correlation results [6]. In the ensemble, we are using the base models for feature extraction, thus they will be given a set of training data for which they will provide predictions, and this set of predictions will be used to train the meta-model of the ensemble. The base models will not have access to the validation data of the meta-model so as to avoid data leakage. The base models and binary classifier will be tuned alongside the meta-model while making sure that when reporting the prediction results, the validation or testing set is not seen by the earlier levels of models.

A plan for experiments can be summarised in the workline below:

1. Combine genomic datasets by manually extracting off-target records from the 8 datasets mentioned in the previous section;

2. Standardise the datasets by computing cleavage frequencies for those that have it missing using indel frequencies as substitute;

3. Export the data into .tab format for easy processing and reproducibility;

4. Implement the procedural methods in Python code and apply them to the combined genomic studies dataset;

5. Obtain the ROC for each of the methods and compare them to the results obtained by CRISPOR integration paper for method validation;

6. Apply data processing to the CrisprSQL dataset (missing values, sequence processing etc.);

7. Implement the validated procedural methods on the CrisprSQL dataset for regression, not converting the scores to binary classes;

8. Compare the scores' relative performance by using Pearson and Spearman Correlation Coefficients;

9. Implement a primitive ensemble by using machine learning models to learn from procedural scores to improve the predictive power for off-target activity;

10. Tune the hyperparameters for the machine learning model;

11. Use Shapley Value analysis to ascertain which procedural methods provide a more insightful look into the specificity of the sgRNA for design optimisation;

12. Design and compare 4 different deep learning architectures (tuned individually) that can learn from sequences directly to predict off-target activity, not needing procedural methods;

13. Integrate 1 or more deep learning models into the primitive ensemble to improve its predictive ability and allow ensemble to learn directly from sequences;

14. Process and standardise the nonvalidates samples dataset and create 5 different experimental scenarios for observing the impact of inactive off-targets on the predictive ability of the models;

15. Include a binary classifier (tuned hyperparameters) to separate validated and

nonvalidated samples so as to increase the robustness of the ensemble to inactive off-targets;

There have been smaller steps and methodology decisions taken in between the methods shortly summarised here, and they will be mentioned where relevant in the results and discussion sections.

Results

Here we will outline the main results of performing experiments as described in the methodology. The segmentation into sections and subsections is to aid in understanding the journey of achieving our aims and is not meant to represent a conceptual fragmentation of the overarching project. Our aims build up on previous questions and answers and they are not separate or independent modes of inquiry. Details on the model selection and hyperparameter tuning will be found in the Appendix as we do not consider them the main results of the dissertation. In the first section, we will outline the fulfillment of the first couple of aims which is implementing procedural scores on CrisprSQL data followed by several model versions for the meta-model in the ensemble learning to combine these procedural scores as features. This will be followed by the analysis of feature importance in terms of these procedural scores with Shapley Value calculation.

The second part concerns implementing 3 existing deep learning architectures and 1 made by us on the CrisprSQL data to learn from sequences directly without using procedural scores. Naturally, the next step is the results from combining both

deep learning and procedural models into an ensemble to investigate the impact on predictive performance. The last part relates to the impact of nonvalidated samples and the robustness of the models we developed in the previous two parts to predicting inactive sites according to 5 scenarios having different ratios of validated/nonvalidated samples. And, finally, observing the efficacy of different binary classifiers in separating validated from nonvalidated samples and including that as a preconditioning model of the ensemble leading to a hierarchical system arrangement.

4.1 Procedural Scores and Ensemble I

We firstly present the results of applying procedural scores on CrisprSQL data to obtain 4 off-target efficiency measurements whose correlation with cleavage frequency will be measured with Pearson and Spearman Coefficients. The models with higher value for Pearson are better linear indicators of cleavage rate, while those with higher Spearman of higher rank correlation between model output and the cleavage rate target variable. Since we used 10-fold crossvalidation, all results represent the mean and variance of the 10 folds so as to decrease the sampling bias and to represent the variance of the results.

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$

Table 4.1: Correlation results for procedural scores

We firstly present the results of applying procedural scores on CrisprSQL data to obtain 4 off-target efficiency measurements whose correlation with cleavage frequency will be measured with Pearson and Spearman Coefficients. The models with higher value for Pearson are better linear indicators of cleavage rate, while those with higher Spearman of higher rank correlation between model output and the cleavage rate target variable. Since we used 10-fold crossvalidation, all results represent the mean and variance of the 10 folds so as to decrease the sampling bias and to represent the variance of the results. We then used all 4 of the procedural scores as features in the different neural network architectures outlined in the methodology in Section 3.3 to predict the cleavage rate and measure the correlation by Pearson and Spearman coefficients.

4.1.1 Architecture I

For architecture I, after hyperparameter search and trained with Adam for 100 epochs, the comparative results obtained can be seen in Table 4.2 below.

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture I	0.381	$1.2e^{-4}$	0.274	$2.0e^{-3}$

Table 4.2: Correlation results for procedural scores and Architecture I as ensemble

4.1.2 Architecture II

Similar procedure as above was repeated for architecture II (Adam/100e) with the following results:

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture I	0.381	$1.2e^{-4}$	0.274	$2.0e^{-3}$
Architecture II	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$

Table 4.3: Correlation results for procedural scores and Architecture I and II as ensemble

The result with batch normalisation is included below and it told us that the method exacerbates the effects of a larger learning rate (sharp drop and saturation of loss)

while also leading to a decrease in performance. We, thus, decided to continue with architecture II in future experiments without batch normalisation.

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture I	0.381	$1.2e^{-4}$	0.274	$2.0e^{-3}$
Architecture II	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$
Architecture II w/ BN	0.453	$1.0e^{-3}$	0.264	$2.0e^{-3}$

Table 4.4: Correlation results for procedural scores and different architectures as ensemble

A short comment on architecture III whose loss behaviour is included in the Appendix which showed the worst performance out of all of the architectures. For these reasons we focused only on architecture IV next.

4.1.3 Architecture IV

Architecture IV is by far the most complex of the architectures with 10 hidden layers with 16 hidden neurons in each. Loss behaviour in Figure 4.1 below shows the effects of overfitting and instability due to the complexity. Thus, we implemented batch normalisation on this architecture as well with training with SGD for 100 epochs with the following results:

Figure 4.1: Training and testing loss for architecture IV

The effects of batch normalisation on the loss of architecture IV can be seen in the Appendix and they do show a more stable learning process alongside the above improved predictive result. The results, however, still cannot compete with architecture II in the combination without batch normalisation which is why that was the model of choice going forwards for the ensemble model. To further capitalise on this model architecture II, we trained it for a longer period of time with 700 epochs and obtained the best predictive result included below:

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture I	0.381	$1.2e^{-4}$	0.274	$2.0e^{-3}$
Architecture II	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$
Architecture II w/ BN	0.453	$1.0e^{-3}$	0.264	$2.0e^{-3}$
Architecture II (700e)	0.501	$4.0e^{-3}$	0.318	$2.5e^{-3}$
Architecture II w/ BN (700e)	0.490	$3.3e^{-3}$	0.260	$2.1e^{-3}$
Architecture IV	0.440	$3.4e^{-3}$	0.265	$2.8e^{-3}$
Architecture IV w/ BN	0.447	$1.1e^{-3}$	0.281	$2.3e^{-3}$

Table 4.5: Correlation results for procedural scores and different architectures as ensemble (bold highlights best)

The loss behaviour of the model architecture II fulfilling the first set of aims and outperforming all procedural scores in both Pearson and Spearman coefficients can be seen in Figure below with the batch normalised loss in the Appendix.

Figure 4.2: Training and testing loss for architecture II trained for 700 epochs

While the loss still shows instability at higher epochs we can confirm this happens as well when batch normalisation is added, thus leading us to conclude that the noise in the higher epochs is more probably due to the crossvalidation procedure and the variation of the data across shuffled folds rather than a strong impact of overfitting.

4.1.4 Shapley Values

Having chosen our best performing model, we implement Shapley Value analysis (using random 100 test samples) on the model with the procedural scores as features and obtain the following Figure 4.3 that shows the relative feature importance of the existing off-target efficiency methods.

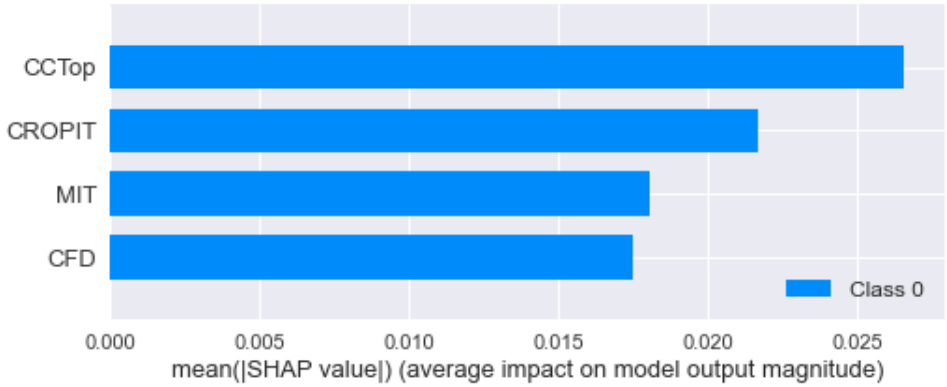

Figure 4.3: Feature importance according to shapley values of procedural scores in ensemble

4.2 Procedural Scores and Ensemble II

Neural networks are not the only mode of learning for regression problems. We investigated also other potential meta-model candidates including linear and polynomial regression, SVR, and kNN models. Starting with regression, we optimised for the polynomial complexity, and achieved best validation performance with degree 7 with the validation plots included in the Appendix. The results of the regression analysis are included in the Table 4.6 below with only the top performing architecture II included going forward for comparison:

Support Vector Regressors are also another framework with which we can learn from procedural scores for the task of regression. The SVR was tuned for 3 different kernels: radial basis function (rbf), linear, and polynomial. The details are included in the Appendix with the best results from validation obtained with the rbf kernel.

The last regression model used for procedural score learning was the kNN algorithm

whose optimal k was found to be 3 and the final results have been compiled below. Because of 3NN being the only model that can compare with architecture II as measured by both Pearson and Spearman results, it will be taken further in experimentation when also adding deep learning models for the final ensemble system.

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture II	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$
Architecture II (700e)	0.501	$4.0e^{-3}$	0.318	$2.5e^{-3}$
Linear Regression	0.297	$1.8e^{-3}$	0.226	$3.5e^{-4}$
Polynomial Regression (d=7)	0.504	$7.8e^{-4}$	0.229	$3.5e^{-4}$
SVR (rbf)	0.314	$4.4e^{-3}$	0.231	$9.0e^{-4}$
3NN	0.609	$2.1e^{-3}$	0.597	$1.9e^{-4}$

Table 4.6: Correlation results for procedural scores and different models including regression as ensemble meta-model

A visual representation of both the architecture II and 3NN models fully fulfilling our aims in successfully learning from procedural scores and outperforming them in both Pearson and Spearman metrics is included in the figure below.

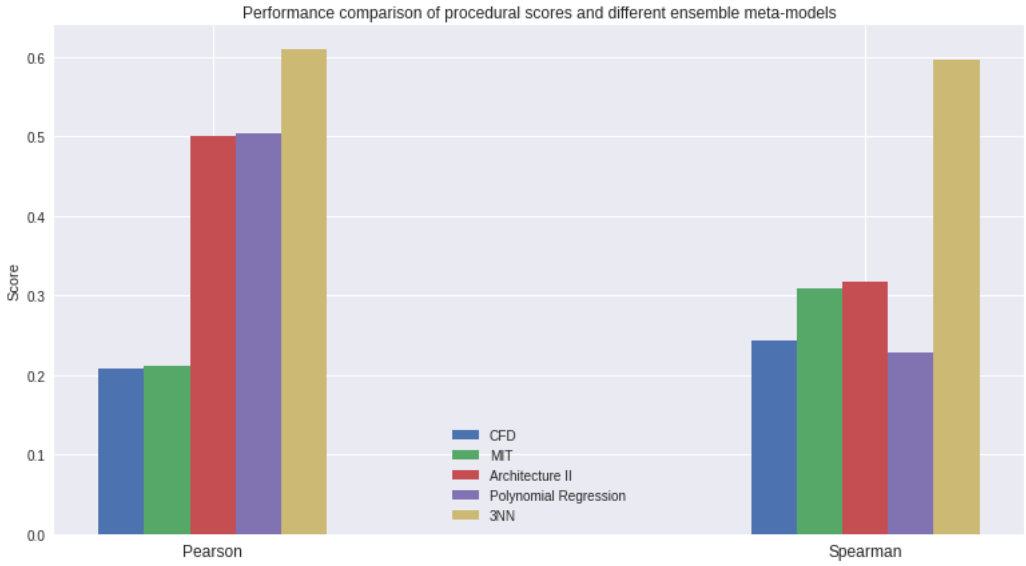

Figure 4.4: Pearson and Spearman results across different ensembles outperforming procedural scores

4.3 Robustness to Nonvalidated Samples I

We have seen the results of the previous models, both procedural and learning, being applied to validated experimental data contained in CrisprSQL. There is, however, still remaining the question as to how robust these different models are to increasing amounts of nonvalidated samples in the validation sets. To reiterate the concept, nonvalidated samples are simply samples in the data that show no off-target activity and whose cleavage frequency is 0. From a machine learning point of view these samples have labels 0 so increasing the fraction of nonvalidated samples is equivalent to increasing the level of data imbalance towards samples with label 0. We implemented the 5 different validation scenarios outlined in the methodology on the procedural scores and the selected architecture and 3NN model to observe their robustness. A visual representation of those results is more appropriate and is included below with a tabular detailed look at the values found in the Appendix. As

a reminder, the 1/0 scenario is the one that corresponds to no nonvalidated samples in the validation sets, while the 1/250 one to 249 nonvalidated samples to every 1 validated sample in the validation sets.

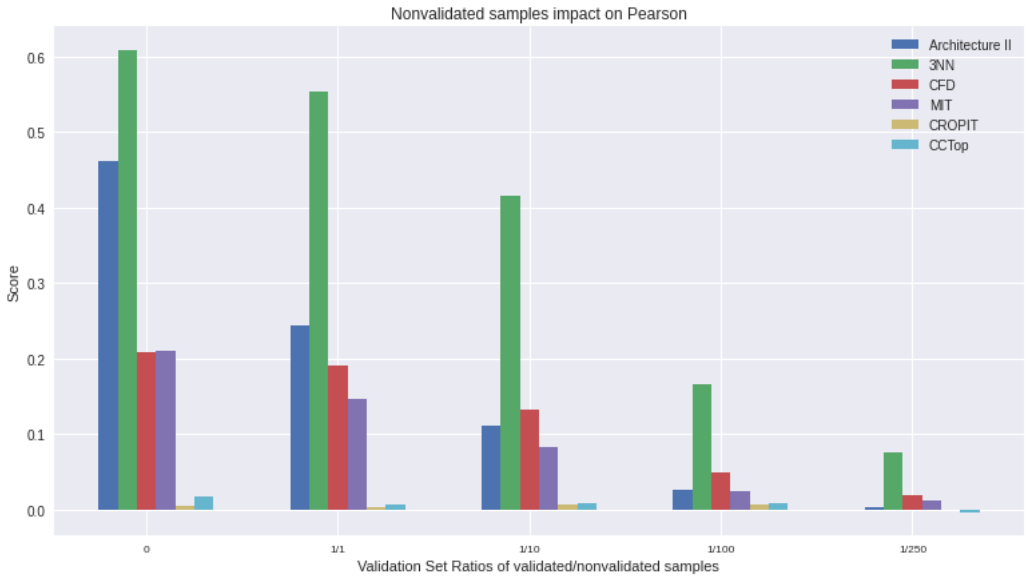

Figure 4.5: Impact of nonvalidated samples on the generalisability or robustness of different models as measured by Pearson coefficient

The same measurement repeated for Spearman coefficient:

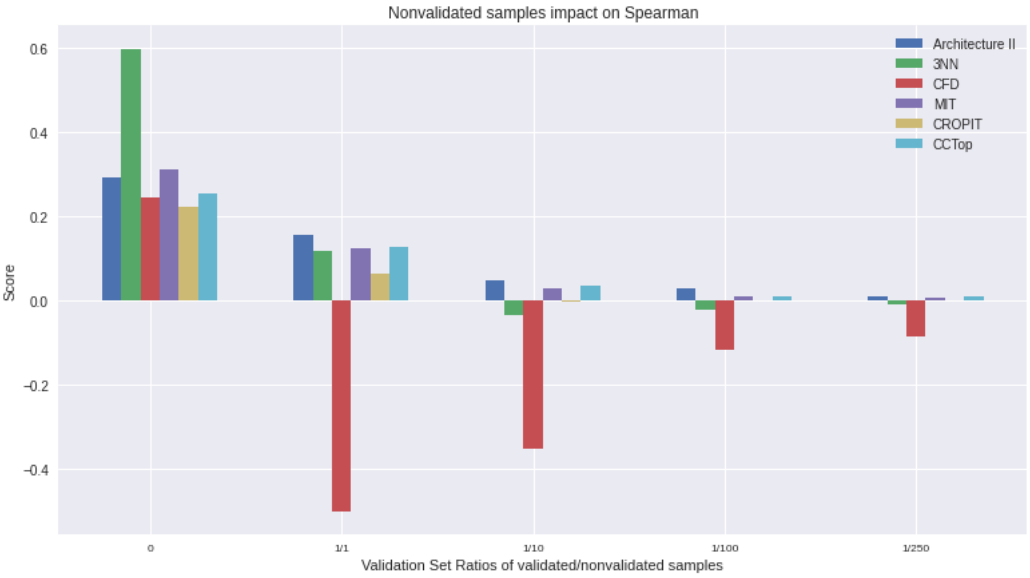

Figure 4.6: Impact of nonvalidated samples on the generalisability or robustness of different models as measured by Spearman coefficient

4.4 Deep Learning Models

So far we have used procedural scores as feature extractors from sequences, but now we will turn to a more popular framework of learning from DNA, which is deep learning models as abstract feature extractors. The first two such models we implemented are the existing Lin and Wong models we adapted to our scenario and termed Linn and LinnFNN3, the former corresponding to the CNN and the latter to the MLP model respectively. After appropriate encoding and data transformation for each of the models, the sgRNA-target pair sequence was used to train (both for 100 epochs) and test the models for off-target cleavage prediction, again measured by Pearson and Spearman coefficients with the following results:

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture II	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$
Architecture II (700e)	0.501	$4.0e^{-3}$	0.318	$2.5e^{-3}$
3NN	0.609	$2.1e^{-3}$	0.597	$1.9e^{-4}$
LinnFNN3	0.646	$4.1e^{-3}$	0.297	$1.7e^{-3}$
Linn	0.646	$7.2e^{-3}$	0.300	$7.7e^{-3}$

Table 4.7: Correlation results for procedural scores and different deep learning and ensemble models

The loss functions for both models are included in the Appendix (Figure A.6, Figure

A.7) but they do show appropriate learning with some overfitting and saturation behaviour that can be clipped by stopping learning at the 25th epoch for both models. If we take a closer look at the predictions of each model, we can see each has its strengths and weaknesses. The prediction plots of predicted vs true value for Linn and LinnFNN3 is below:

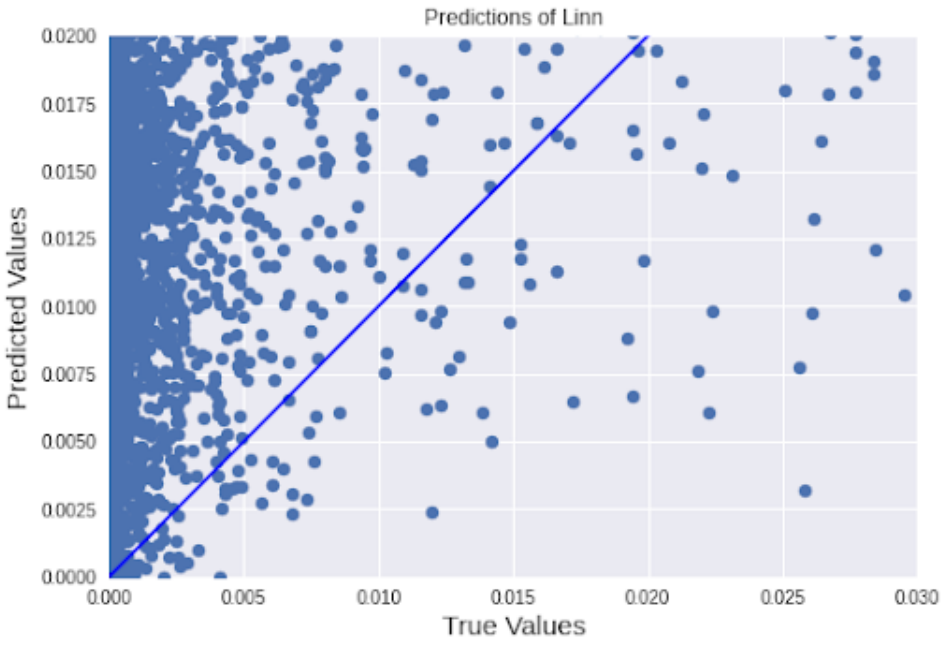

Figure 4.7: Predictions of Linn model with true fit line highlighted

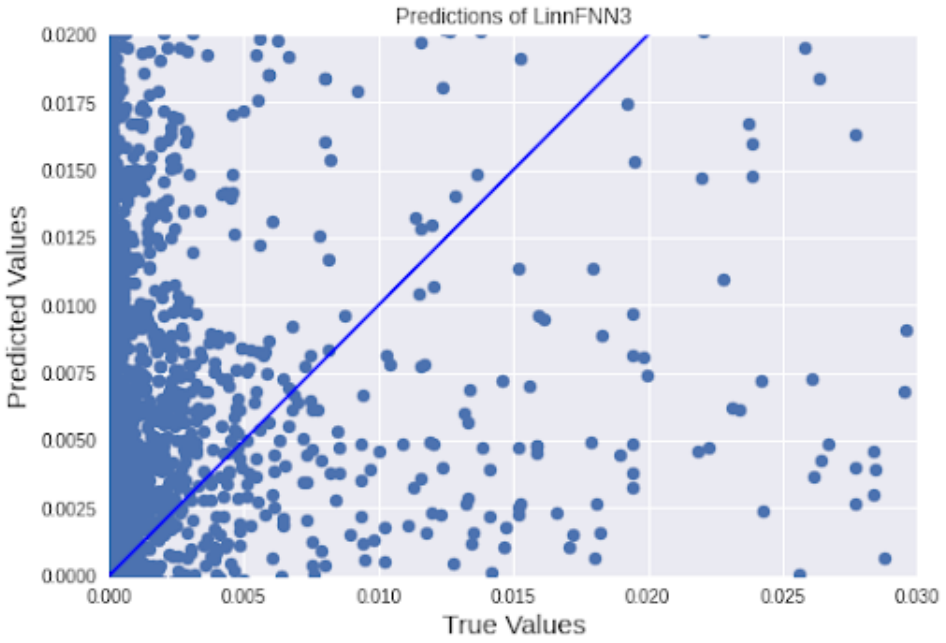

Figure 4.8: Predictions of LinnFNN3 model with true fit line highlighted

We then implemented our own version of the BGRU framework that has shown great success with the on-target problem combining CNNs and RNNs and learning from sequential data. The results are included below:

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture II	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$
Architecture II (700e)	0.501	$4.0e^{-3}$	0.318	$2.5e^{-3}$
3NN	0.609	$2.1e^{-3}$	0.597	$1.9e^{-4}$
LinnFNN3	0.646	$4.1e^{-3}$	0.297	$1.7e^{-3}$
Linn	0.646	$7.2e^{-3}$	0.300	$7.7e^{-3}$
BGRU	0.664	$4.9e^{-3}$	0.316	$2.6e^{-3}$

Table 4.8: Correlation results for procedural scores and different deep learning and ensemble models

The prediction behaviour of the BGRU is also included below:

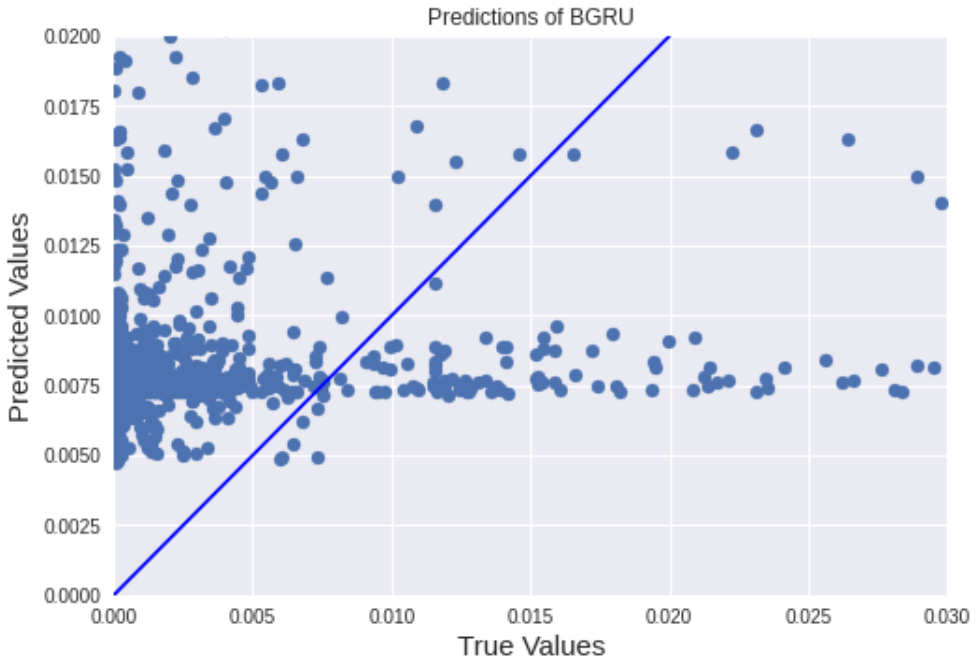

Figure 4.9: Predictions of BGRU model with true fit line highlighted

We finally designed our own 2BGRU architecture that would combine the predictive power of the BGRU framework with the siamese design learning from sequence pairs separately instead of encoding them together into one input matrix before learning. The results of our 2BGRU model are included below:

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture II	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$
Architecture II (700e)	0.501	$4.0e^{-3}$	0.318	$2.5e^{-3}$
3NN	0.609	$2.1e^{-3}$	0.597	$1.9e^{-4}$
LinnFNN3	0.646	$4.1e^{-3}$	0.297	$1.7e^{-3}$
Linn	0.646	$7.2e^{-3}$	0.300	$7.7e^{-3}$
BGRU	0.664	$4.9e^{-3}$	0.316	$2.6e^{-3}$
2BGRU	0.637	$6.2e^{-3}$	0.318	$2.6e^{-2}$

Table 4.9: Correlation results for procedural scores and different deep learning and ensemble models

And the prediction patterns of the 2BGRU model is also included below, with the loss plots for these two models reserved for the Appendix (Figure A.8, Figure A.9).

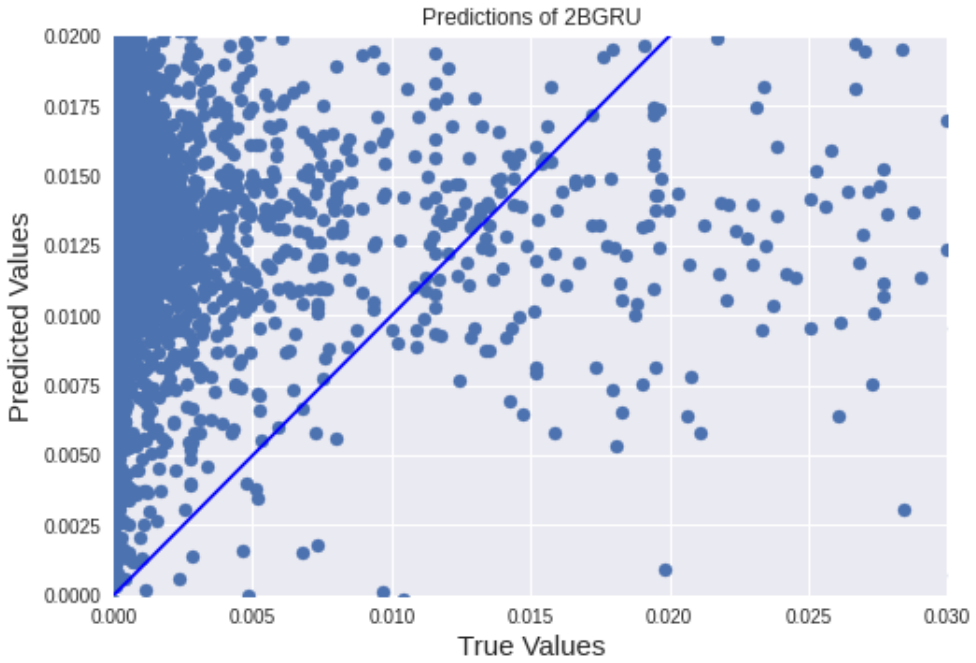

Figure 4.10: Predictions of 2BGRU model with true fit line highlighted

4.5 Deep Learning Models and Ensemble I

The models above learn directly from genetic sequence data and nothing else to make a regression prediction of the cleavage rate, but to answer the question on how well these deep learning models integrate within an ensemble with procedural scores, we obtained the following results. To reiterate, the deep learning models are used similarly as the procedural methods as not just models for off-target activity prediction but also as abstract feature extractors from sequences that will then be used to train a meta-model for final off-target activity prediction. Since the architecture II proved to be the best performing wrapper for such a case, we will continue to use it as the ensemble meta-model and add each of the deep learning models to the system. Those results are included below in Table 4.10.

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture II	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$
Architecture II (700e)	0.501	$4.0e^{-3}$	0.318	$2.5e^{-3}$
3NN	0.609	$2.1e^{-3}$	0.597	$1.9e^{-4}$
LinnFNN3	0.646	$4.1e^{-3}$	0.297	$1.7e^{-3}$
Linn	0.646	$7.2e^{-3}$	0.300	$7.7e^{-3}$
BGRU	0.664	$4.9e^{-3}$	0.316	$2.6e^{-3}$
2BGRU	0.637	$6.2e^{-3}$	0.318	$2.6e^{-2}$
Ensemble w/ LinnFNN3	0.712	$1.4e^{-3}$	0.330	$2.9e^{-3}$
Ensemble w/ Linn	0.706	$2.4e^{-3}$	0.333	$5.9e^{-3}$
Ensemble w/ BGRU	0.714	$2.2e^{-3}$	0.282	$2.3e^{-3}$
Ensemble w/ 2BGRU	0.701	$2.2e^{-3}$	0.293	$6.9e^{-3}$

Table 4.10: Correlation results for procedural scores and different deep learning and ensemble models

A more visual representation of the key results is provided in the figure below comparing the ensemble with the standalone deep learning model results:

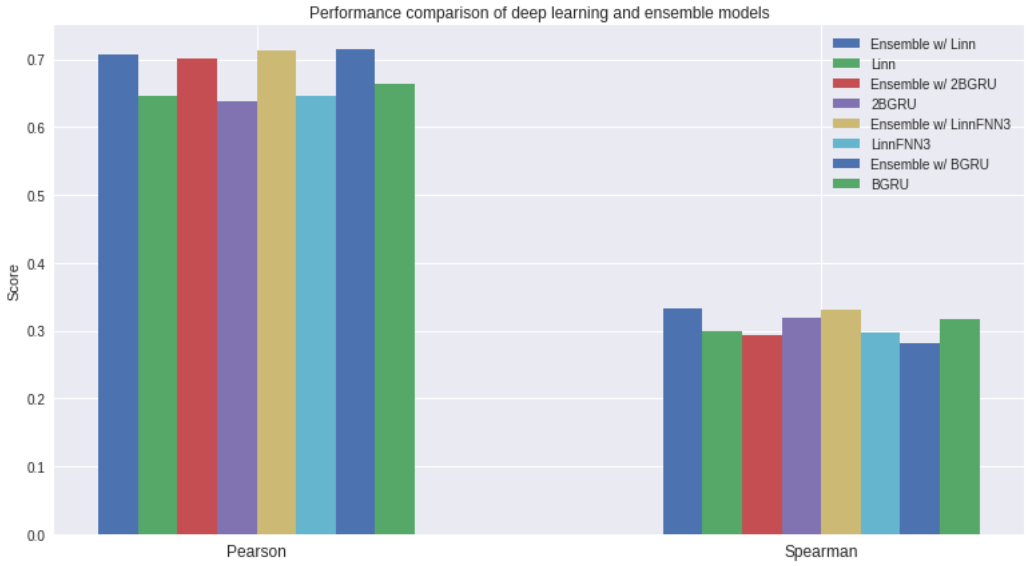

Figure 4.11: Correlation results on deep learning and ensemble models

4.6 Deep Learning Models and Ensemble II

As we are also considering using kNN for the ensemble with considerable success in the procedural case scenario, perhaps a similar framework might provide good results with deep learning scores included as well. Following optimisation and obtaining minimum validation loss for k=3 as in the results above, implementing 3NN on procedural and deep learning scores gives the following results:

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture II	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$
Architecture II (700e)	0.501	$4.0e^{-3}$	0.318	$2.5e^{-3}$
3NN	0.609	$2.1e^{-3}$	0.597	$1.9e^{-4}$
LinnFNN3	0.646	$4.1e^{-3}$	0.297	$1.7e^{-3}$
Linn	0.646	$7.2e^{-3}$	0.300	$7.7e^{-3}$
BGRU	0.664	$4.9e^{-3}$	0.316	$2.6e^{-3}$
2BGRU	0.637	$6.2e^{-3}$	0.318	$2.6e^{-2}$
Ensemble w/ LinnFNN3	0.712	$1.4e^{-3}$	0.330	$2.9e^{-3}$
Ensemble w/ Linn	0.706	$2.4e^{-3}$	0.333	$5.9e^{-3}$
Ensemble w/ BGRU	0.714	$2.2e^{-3}$	0.282	$2.3e^{-3}$
Ensemble w/ 2BGRU	0.701	$2.2e^{-3}$	0.293	$6.9e^{-3}$
3NN w/ LinnFNN3	0.657	$2.5e^{-3}$	0.598	$6.4e^{-5}$
3NN w/ Linn	0.660	$2.1e^{-3}$	0.605	$1.0e^{-4}$
3NN w/ BGRU	0.657	$2.2e^{-3}$	0.605	$8.1e^{-5}$
3NN w/ 2BGRU	0.674	$1.7e^{-3}$	0.606	$4.8e^{-4}$

Table 4.11: Correlation results for procedural scores and different deep learning and ensemble models

Again, a similar visual representation comparing the deep learning and 3NN framework results:

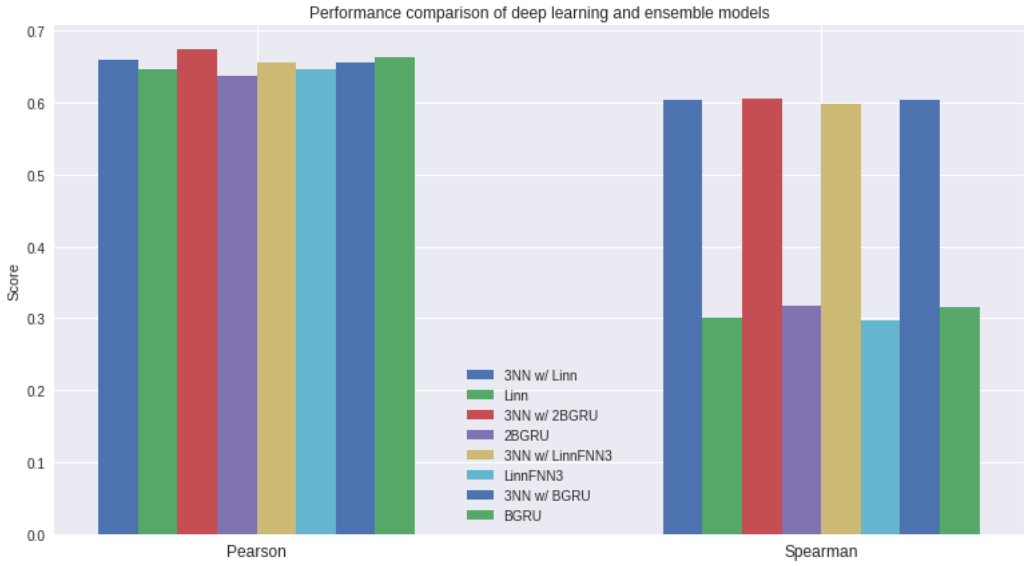

Figure 4.12: Correlation results on deep learning and 3NN ensemble models

4.7 Robustness to Nonvalidated Samples II

Similarly as in the primitive ensemble case, the deep learning models need to be examined for their robustness to nonvalidated samples. We used the same testing scenarios with increasing magnitudes of nonvalidated samples in the validation set without using them in the training set to maintain the independent holdout nature of the generalisation measurement. The below figure indicates the effect increasing amounts of nonvalidated samples has on the performance of the 4 deep learning models as measured by Pearson and Spearman:

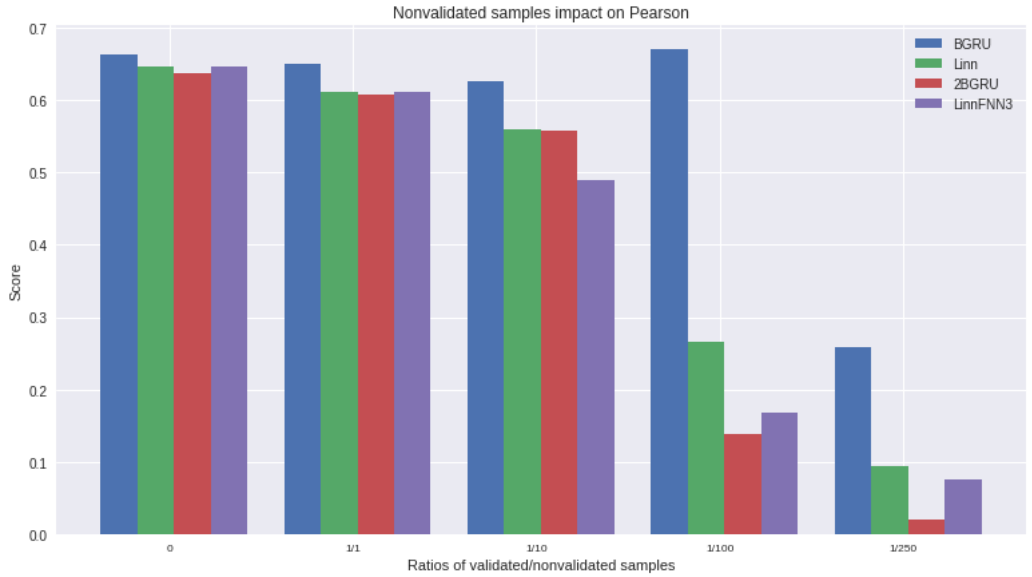

Figure 4.13: Correlation results on deep learning models under inactive sites effect

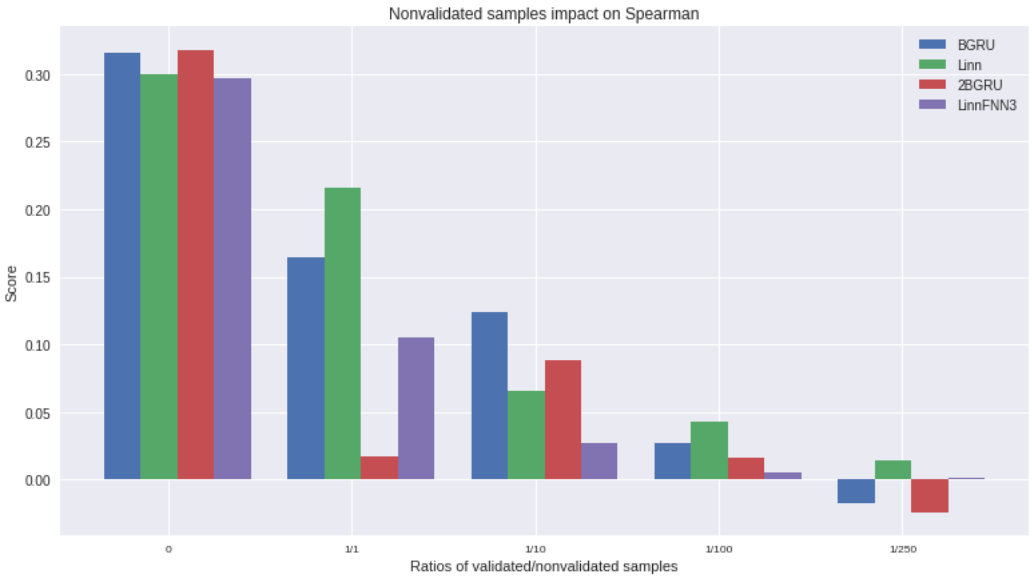

Figure 4.14: Correlation results on deep learning models under inactive sites effect

To investigate the predictive patterns of the different deep learning models when exposed to nonvalidated samples, we overlayed their predictions for non validated samples in the 1/1 scenario. The green dots represent the predictions for nonvalidated samples (their true label is always 0) in the below prediction plots for each deep learning model.

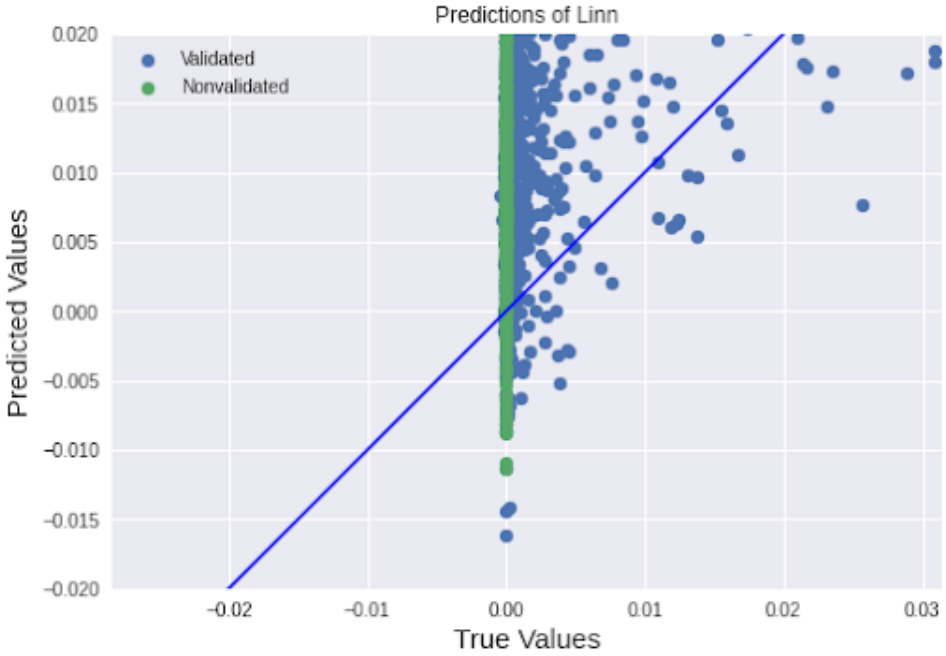

Figure 4.15: Predictions of Linn model with true fit line highlighted including nonvalidated samples

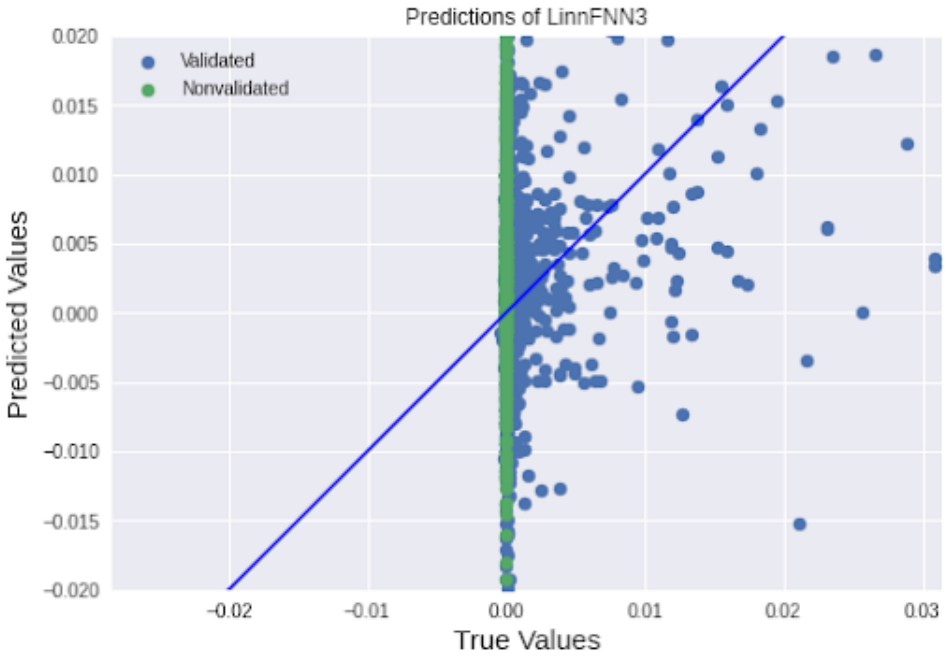

Figure 4.16: Predictions of LinnFNN3 model with true fit line highlighted including nonvalidated samples

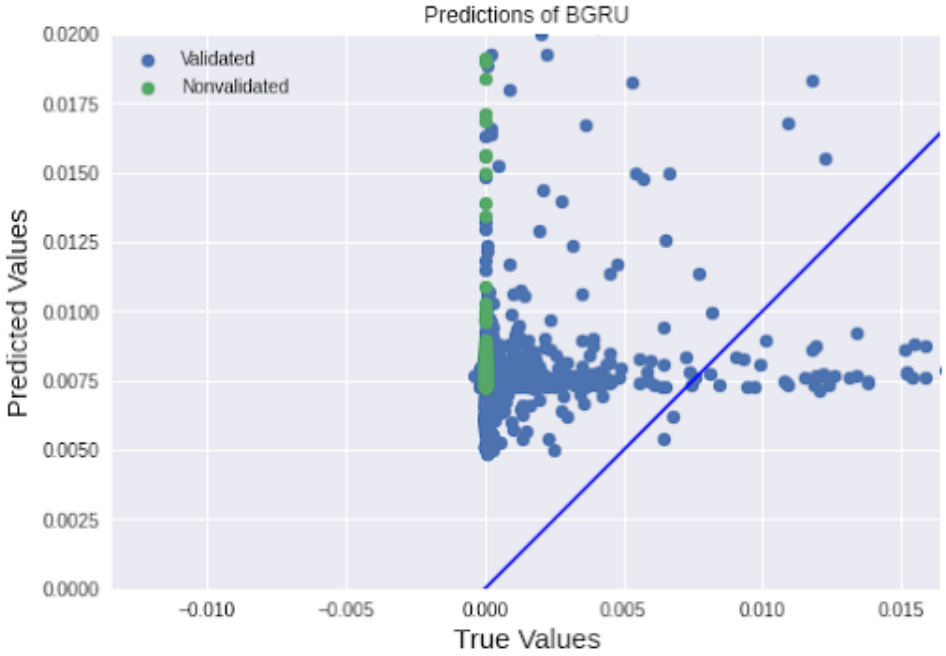

Figure 4.17: Predictions of BGRU model with true fit line highlighted including nonvalidated samples

Figure 4.18: Predictions of 2BGRU model with true fit line highlighted including nonvalidated samples

Going further and exploring a similar impact on the ensemble systems combining the procedural scores and deep learning we obtained the following results (Spearman

results in Appendix Figure A.10):

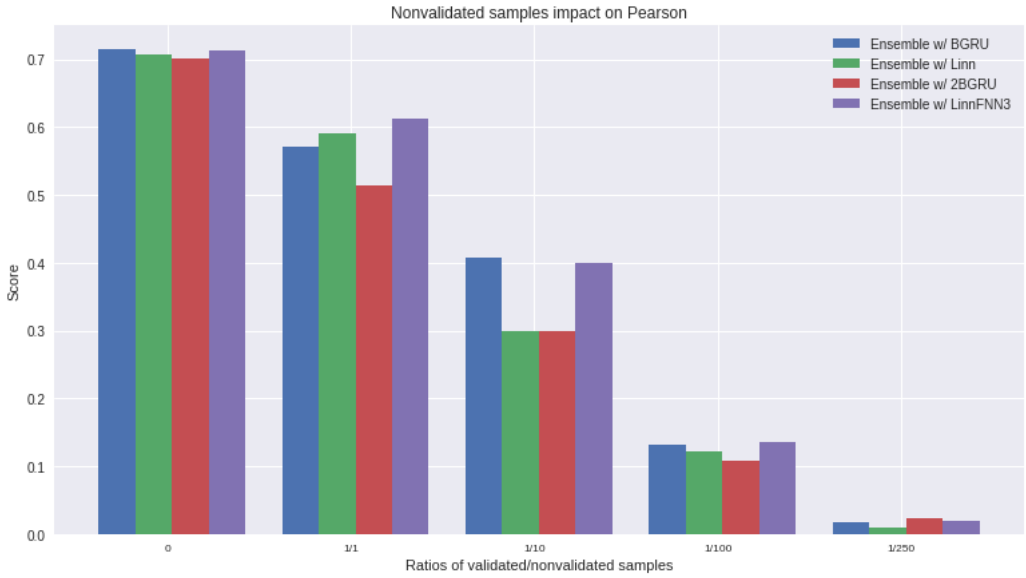

Figure 4.19: Correlation results on ensemble models under inactive sites effect

The 3NN shows a similar behaviour under the inactive sites regime but with important differences with the Spearman figure in the Appendix (A.11):

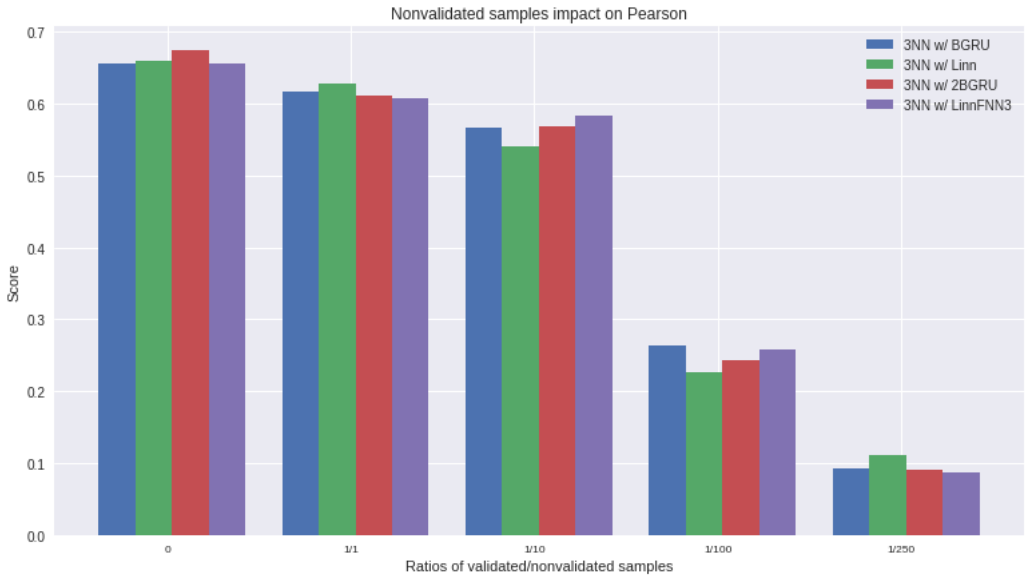

Figure 4.20: Correlation results on 3NN models under inactive sites effect

Since the BGRU model showed considerable resistance to nonvalidated samples in Figure 4.13, and because BGRU and LinnFNN3 show different patterns of predic-

tions highlighted in the last section, a further question to answer is whether an ensemble combining both of these deep learning models might provide a unique advantage. The figures below show the results of the experiment for both the ensemble and 3NN methods (Spearman results are in the Appendix under A.12 and A.13, they show a similar pattern):

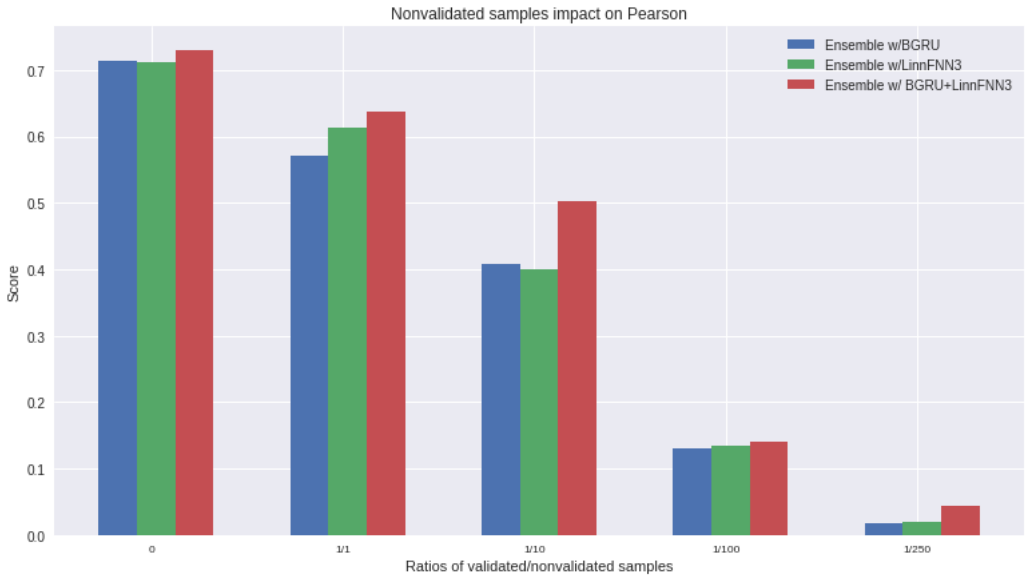

Figure 4.21: Correlation results on ensemble models under inactive sites effect with two deep learning models

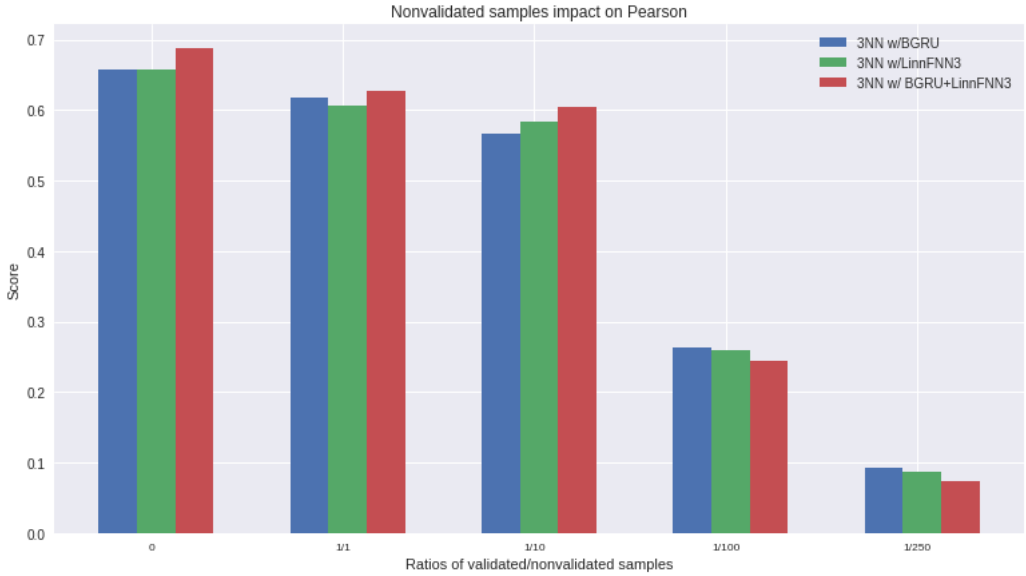

Figure 4.22: Correlation results on 3NN models under inactive sites effect with two deep learning models

Tables with detailed listing of measures can be found in the Appendix (Table A.2, Table A.3, Table A.4).

4.8 Binary Classifier and Hierarchical Model

The impact of the nonvalidated samples on the predictive power of all models is unmistakeable so instead of forcing regression models like the above to learn both a discriminatory decision boundary for validated and nonvalidated samples while also learning a good fit for the validation samples, we broke up the problem into separate challenges. The first is a preconditioning step that would classify the samples as either validated or nonvalidated and then pass the predicted validated samples to the regression models we developed above for off-target activity prediction. The aim is then that this system will dent the curve of the drop in performance with increasing amounts of nonvalidated samples present in the validation set of the regression models.

The search for a reasonably good binary classifier that would be able to separate nonvalidated from validated samples has been quite challenging. Through careful hyperparameter tuning, a neural network regression model has been optimised for the task through Weights and Biases. Logistic regression was optimised through simple grid search and resorting to Weights and Biases was not necessary. The figure below indicates the result from including each of the binary classification models separately as a preconditioning module for the ensemble system in both

single deep learning model and two deep learning model cases. Look to Tables A.5 and A.6 in Appendix for detailed numerical results.

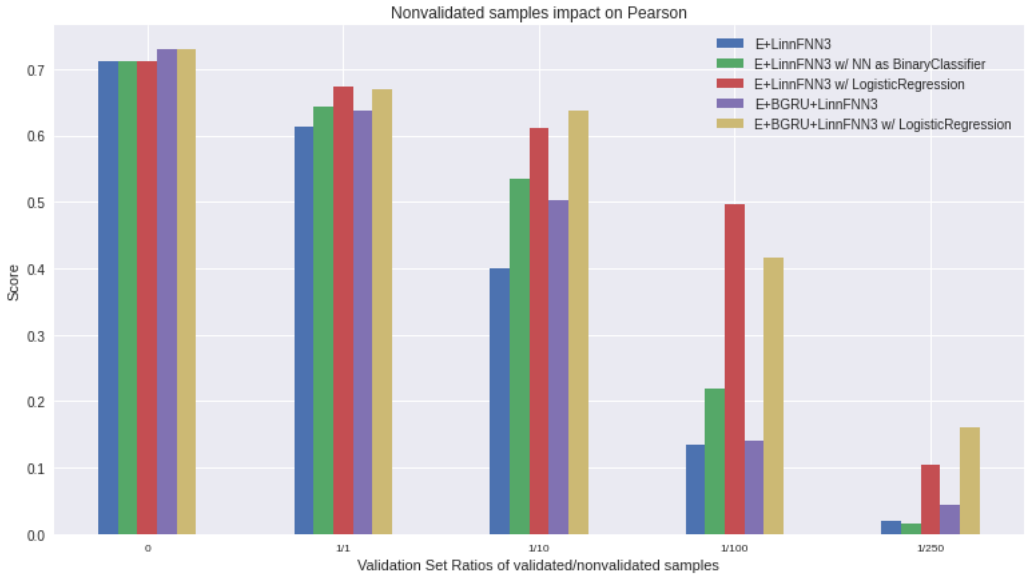

Figure 4.23: Correlation results in Pearson showing mitigation of inactive site effect through use of preconditioning model

Since the logistic regression model provides good preconditioning performance in addition to being less computationally expensive, that is our binary classifier of choice deployed in the final system. With the last hierarchical model, we conclude our main results. There were secondary experiments undertaken throughout the project, some of which are included in the Appendix and which do not represent significant contributions of the project or the fulfillment of the targeted aims. Building on those results to answer further questions will be left to the comments on the future direction outlined in the conclusion.

Discussion

5.1 Procedural Scores and Ensembles

Having successfully applied the procedural scores to the CrisprSQL data as observed in Table 4.1 where MIT is the best performing model across both Pearson and Spearman correlation measurements, we note that CROPIT and CCTop have a much lower Pearson coefficient. This is due to a large number of PAM sites present in CrisprSQL that have not been included in the experimental validation studies used when assigning weights to those two methods. PAM sites that are not 'NGG' or 'NNG' will be classified as outliers according to these two methods and hence their Pearson coefficient will be a lot more affected by this behaviour than Spearman due to its sensitivity to outliers [78]. The MIT score is the foundational procedural method that has been extensively experimentally validated to assign positional mismatch weights to target sequences, thus its generalisability to a dataset like CrisprSQL that contains a multitude of different experimental data is much higher than for the other more recent methods. The following step was to develop a primitive ensemble architecture capable of outperforming each of the procedural

scores when it comes to off-target activity prediction. Out of 4 candidate neural network architectures, architecture II proved to be best suited with great learning performance. The selected model achieved Pearson performance over 2 times higher than the best procedural score while also outperforming MIT as measured by Spearman correlation which can be seen in Table 4.5.

Looking at other substitutes for the ensemble model like regression, SVR, and kNN, we see in Table 4.6 that polynomial regression can compete with architecture II when measured by Pearson. It is important to note that Pearson is measuring the linearity between the true and predicted target value, not between the target and the input which can allow for a nonlinear model like polynomial regression to predict values that are more linearly correlated with the true target values. Its Spearman measure is much lower, however, not being any better than the linear regression model meaning it has a harder time capturing the ordinal or rank correlation between prediction and truth. The 3NN outperforming architecture II might come as a surprise initially but when we think about the predictions of the kNN algorithm being existing values found in the dataset and then averaged versus approximations which neural network predictions are, then higher correlation between those existing values and true values of the dataset are not surprising. The kNN algorithm is a non-parametric model that does not learn the type of analytical function meant to describe the input-target relationship neural networks approximate which can be harder to do in this case. They key evidence will come later when we investigate the robustness of these models to nonvalidated samples which are not part of this

dataset consisting of only validated samples. Since nonvalidated samples need to be given a target of exactly 0 (due to nonvalidated or inactive sites having no off-target or cleavage activity), the 3NN will have an easier time doing so due to it only assigning averages of existing values which includes a lot of 0 targets in the validated samples set. The neural network will be less robust in this case as its analytical function is just an approximation of an assumed relationship between input and target, so it will not be predicting exactly 0 but rather a small value closer or farther from it. This leads to a higher robustness of the 3NN model as confirmed in Figure 4.5 with the Pearson measure more sensitive to those "outliers" the neural network predictions are.

The main takeaway of the 3NN model, however, is its much larger Spearman correlation result and that it significantly helps with obtaining the best result on that metric as seen in Figure 4.12.The inversion of the Spearman correlation for the CFD score indicates that this procedural method is negatively correlated to the rank ordering of off-target activity as seen in Figure 4.6. That could be a consequence of the PAM sites present in the nonvalidated samples that are not supported by the CFD method causing those samples to be allocated a score of 0 when in fact it should be much higher. With subsequent scenarios when the number of nonvalidated samples increases, there is less of those PAM edge cases present and the CFD approaches the behaviour of the other procedural methods. In sum, both procedural and ensemble models are highly sensitive to the presence of nonvalidated samples and despite 3NN being a lot more robust in this regard, it still presents a challenge to dent the

robustness curve and mitigate the negative impacts on performance by nonvalidated samples.

Now that we have a successful learning model able to combine each of the different procedural scores together for improved activity prediction, using Shapley values analysis in Figure 4.3 shows that CCTop and CROP-IT are the most important features affecting the prediction of the neural network ensemble. Since those two methods are meant to be improvements on MIT and CFD that included experimental validation of DNA attributes beyond just sequence information, their large contribution to the model predictions comes as no surprise. CCTop being the most impactful feature supports this argument as it also follows a similar approach where it punishes more when the mismatches are closer to the PAM site (higher positional mismatch weights) and it is also a kind of weighted sum. Both of these methods rely on weights derived experimentally, albeit through two different methodologies, where higher weighting is given to mismatches closer to the PAM site and in the case of CROP-IT also the density of the mismatches i.e. their relative distance to each other on the sequence. This indicates that effective off-target activity prediction can specifically focus on sequence mismatches, proportionally weighting those closer to the PAM site and taking into account the mismatch distribution on the entire sequence. Recent studies also indicate that the most frequent mismatches resulting in off-target activity do occur at the PAM proximal end of the target sequence [79].

5.2 Deep Learning Models

The previous set of aims concerned using the standard procedural methods as feature extractors to a neural network or kNN ensemble model, but deep learning models also exist to directly extract information or abstract features from sequences for activity prediction. We first present the Linn and LinnFNN3 models which are adaptations of the Lin and Wong architectures and which show excellent learning performance when compared to both procedural scores and our primitive ensembles (Table 4.7). Looking at each of the model's prediction patterns in Figures 4.7 and 4.8, we see that Linn is more likely to overestimate predictions while LinnFNN3 underestimates. Moving forward, we implemented the BGRU model for off-target activity prediction for the first time since it was suggested as a model for the on-target activity scheme. It outperforms the other deep learning models as seen in Table 4.8 supporting the idea that a mixture of convolutional and bidirectional sequential learning does offer some advantage to learning from genetic sequences in the off-target problem. The BGRU, similarly to the Linn model, also tends to overestimate the prediction and it struggles with more sensitive predictions closer to 0 or below off-target activity levels of 0.005 (Figure 4.9). And, finally, we implemented our own architecture using the siamese framework with the BGRU's predictive power to achieve the best Spearman result outperforming all previous models while its Pearson result is slightly lower (Table 4.9). Similarly to BGRU, it also overestimates but it does not suffer from the challenge of making predictions closer to 0 so in that respect is more sensitive to lower off-target activity (Figure 4.10).

Implementing these deep learning models as part of an ensemble like the one above with procedural scores leads to an improvement of performance across all models as measured by the Pearson coefficient. The BGRU and 2BGRU do slightly suffer from a decrease in Spearman correlation but it still places them comparatively close to the best-performing models. When we look at using the 3NN model as the ensemble, that problem is alleviated with the Spearman metric vastly outperforming the deep learning model on its own. Figures 4.11 and 4.12 show these results and supports our idea that ensembles with deep learning and procedural models improve the off-target activity prediction capabilities of the system across both Pearson and Spearman correlation metrics.

When we look at the robustness of the deep learning models to nonvalidated samples, the BGRU shows a greater resilience especially on the Pearson correlation metric. This will become important later when we want to design and implement robust ensembles with multiple deep learning models making BGRU an inevitable candidate in those systems. Figure 4.13 also shows that deep learning models, similar to procedural scores and the primitive ensemble, suffer from a lack of generalisability to set containing increasing amounts of nonvalidated samples. This behaviour is repeated when they are part of an ensemble system as well as shown in Figures 4.19 and 4.20. These results suggest there are inherent attributes present in nonvalidated samples that are not captured when learning from validated samples. The prediction patterns of the deep learning models for nonvalidated samples reiterate LinnFNN3's tendency to underestimate while being more confident with predictions closer to 0 than the

other models while Linn, BGRU, and 2BGRU overestimate on nonvalidated samples as well as validated ones. Creating an ensemble system combining LinnFNN3 and one of the other 3 models like in the case of Ensemble+BGRU+LinnFNN3 might offer a predictive edge where those weaknesses of each model might be compensated for by each other. The results in Figures 4.21 and 4.22 show that this approach does work with the LinnFNN3+BGRU system in both neural network and 3NN cases outperforming the individual model ensembles and having a higher robustness to nonvalidated samples up to an entire order of magnitude.

5.3 Robustness to Nonvalidated Samples

We have seen that besides the ensemble model with LinnFNN3 and BGRU, all of the other models suffer largely from the consequences of including nonvalidated samples at levels creating data imbalance. The issue could come from too stringent of a scenario being imposed on these models where they both have to separate nonvalidated and validated samples while also allocating an off-target activity score to the validated samples. Breaking this problem into two parts where the former is to be tackled through a preconditioning model like logistic regression being able to separate at least some of the nonvalidated samples out and increase the robustness of the overall system. This approach proved successful and we obtained the most significant results of the project in Figure 4.23 which shows that with the system consisting of the Ensemble+BGRU+LinnFNN3 as a regressor and logistic regression as a preconditioning module, the robustness to nonvalidated samples up to an entire

order magnitude can be achieved. The drop in Pearson for that large range is around 0.08 with the highest achieved Pearson correlation yet for the validated samples of 0.73. A similar pattern is observed for the Ensemble+ LinnFNN3 case across both Pearson and Spearman measures. One could argue both models also show sufficiently good performance even for two orders of magnitude more nonvalidated samples (1/100 scenario) with the Ensemble+LinnFNN3 maintaining a stable 0.5 Pearson correlation and 0.16 Spearman correlation which is about a half of the best performing model under the scenario of no nonvalidated samples.

Conclusions

As the preceding discussion shows, the results go beyond our original aims but here we will reiterate how our experiments indicate the fulfillment of our investigative objectives. Firstly, we originally applied 4 procedural methods to the entire CrisprSQL dataset and compared them respectively as well as proposed a simple neural network and kNN model that can combine these procedural scores as features into a more powerful system capable of predicting off-target activity better than either of the procedural methods as measured by both Pearson and Spearman correlation coefficients. The 3NN model showed excellent results capable of comparing across both metrics to deep learning models that learn from sequences directly. We then undertook Shapley value analysis to obtain the most insightful procedural scores in this system allowing us to make deductions on what areas or attributes of the sequence and the DNA contribute more to the off-target activity levels without looking at the sequences directly.

Secondly, we implemented two existing off-target deep learning models to the CrisprSQL dataset alongside our adaptation of the BGRU on-target framework to the off-target

scenario while comparing all of these models to our own 2BGRU siamese network architecture. The 2BGRU showed similar predictive capability as the other more standard models and its predictions did not suffer from an inability to make predictions for smaller off-target activity levels like the BGRU did. All of the above models are greatly impacted by an increasing data imbalance in favor of nonvalidated samples which is a so far unresolved problem in the field. To attempt to provide a robust solution to this challenge, we designed a hierarchical model with a simple logistic regression model as a preconditioner to greatly increase the robustness of our models with the Ensemble+BGRU+LinnFNN3 indicating minimal negative impact on performance up to even an entire order of magnitude more nonvalidated samples.

We have been positively surprised by the great predictive performance of simpler, less computationally expensive, machine learning models showed on this set of problems. The key to our ensemble design and getting the final hierarchical system was looking at the prediction patterns of the different models and combining them in a manner where one of the model's weaknesses could be compensated for by another's like in the case of the BGRU and LinnFNN3. Our results obtaining an average 0.64 Pearson correlation on a heavily imbalanced dataset with 10 times as many nonvalidated samples outperforms a similar study done by researchers last year without having to resort to bias-inducing artificial oversampling techniques [80]. Further work can be done on using more advanced preconditioning strategies to attempt to separate the nonvalidated samples better before off-target activity prediction by regression models but the conclusion is that models can be constructed to achieve

this robustness without applying additional restraints on the data.

In all of the studies presented here we restricted ourselves to using only nucleotide sequences to make predictions, whether it be for procedural methods or deep learning models. Future work can try to incorporate energy measurements, chromatin attributes, and PAM regions as features into the learning process that might allow the models to achieve even higher predictive capability. Our brief look at explainability when it comes to procedural scores indicated that positional mismatches and their density might play a bigger role than those sequence-extraneous features but further study is required. We also only scratched the surface of utilising siamese networks to learn from sequence pairs and thereby circumnavigating the need for OR encoding. Future experiments can try with simpler siamese networks that will tend less to overfit and will be of a much smaller parameter size to allow for more efficient computation. The study of specificity of CRISPR/Cas9 is of high importance to designing safe and scalable gene editing technology and by leveraging the different facets of machine learning together, we can develop robust tools to aid in that design.

Appendix

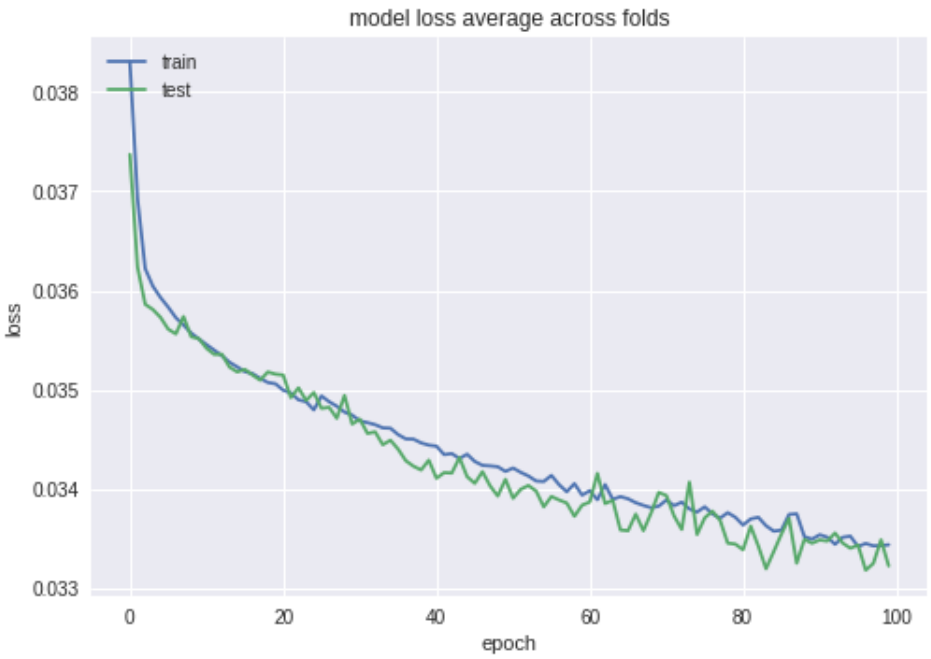

Figure A.1: Training and testing loss for architecture III

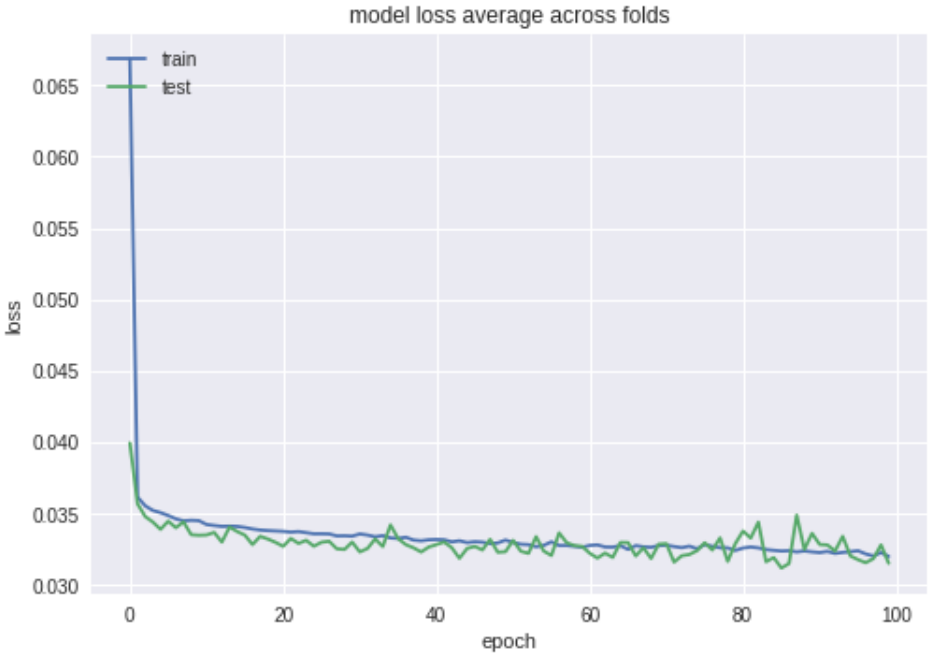

Figure A.2: Training and testing loss for architecture IV with batch normalisation

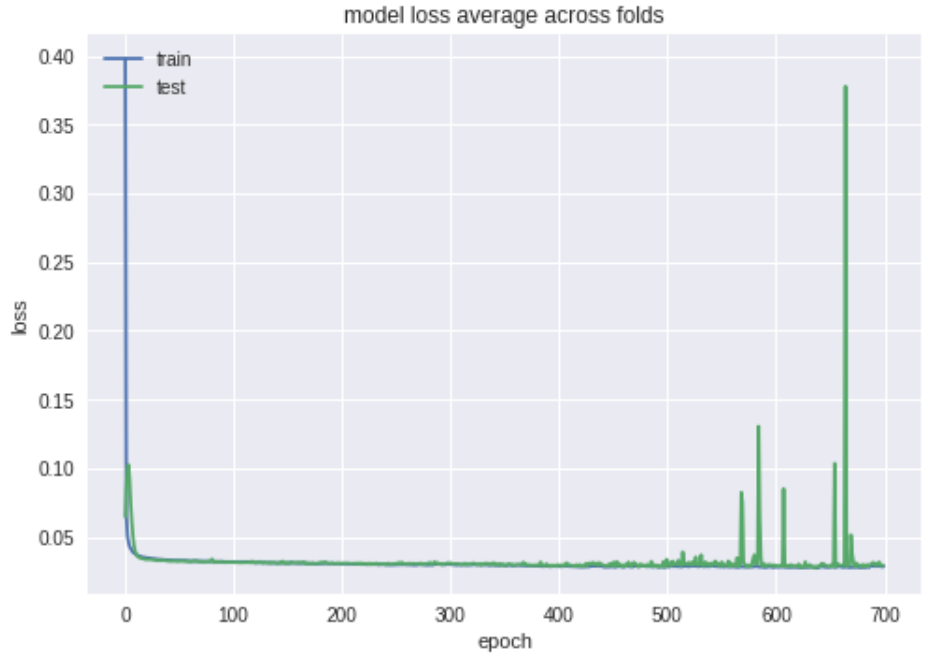

Figure A.3: Training and testing loss for architecture II with batch normalisation trained for 700 epochs

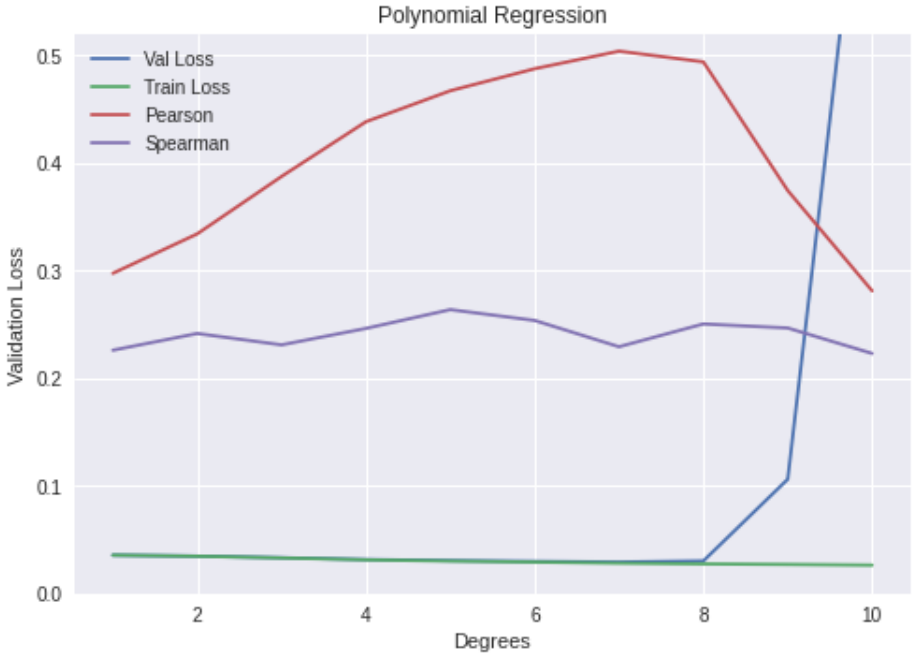

Figure A.4: Validation for degree search of polynomial regression (optimal is d=7)

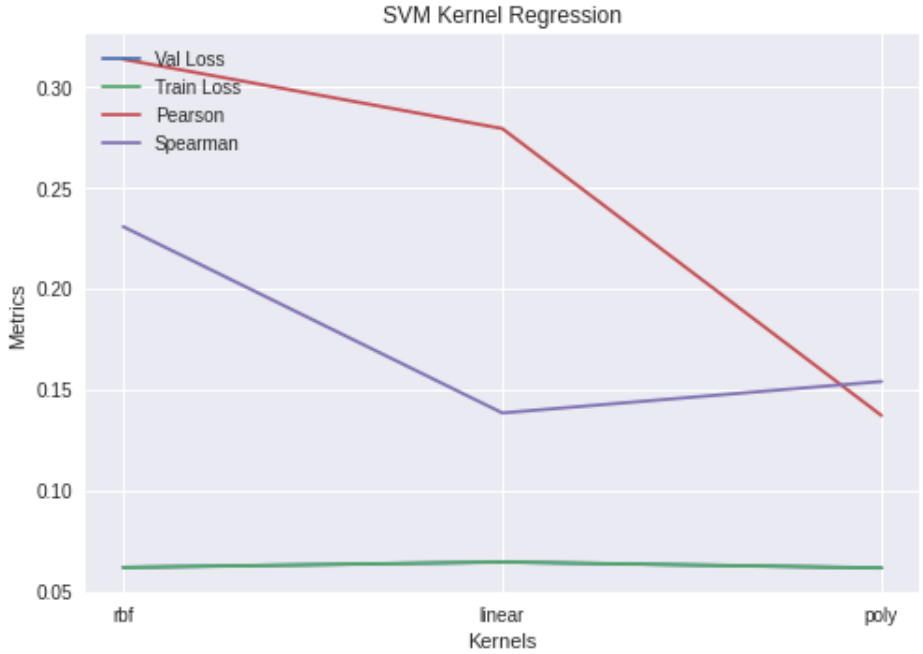

Figure A.5: Validation for degree search of SVR (optimal is rbf)

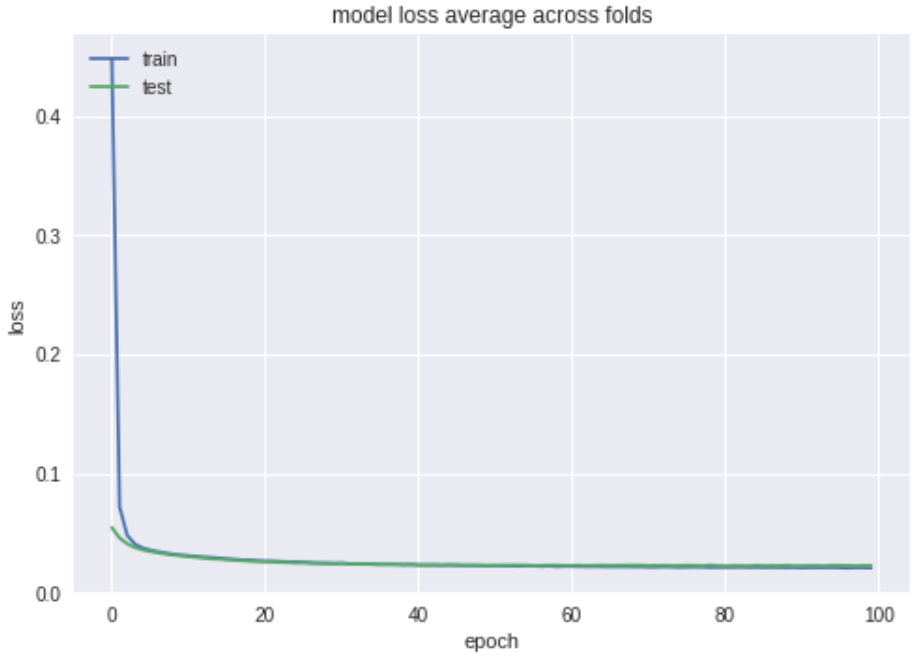

Figure A.6: Loss behaviour of Linn model

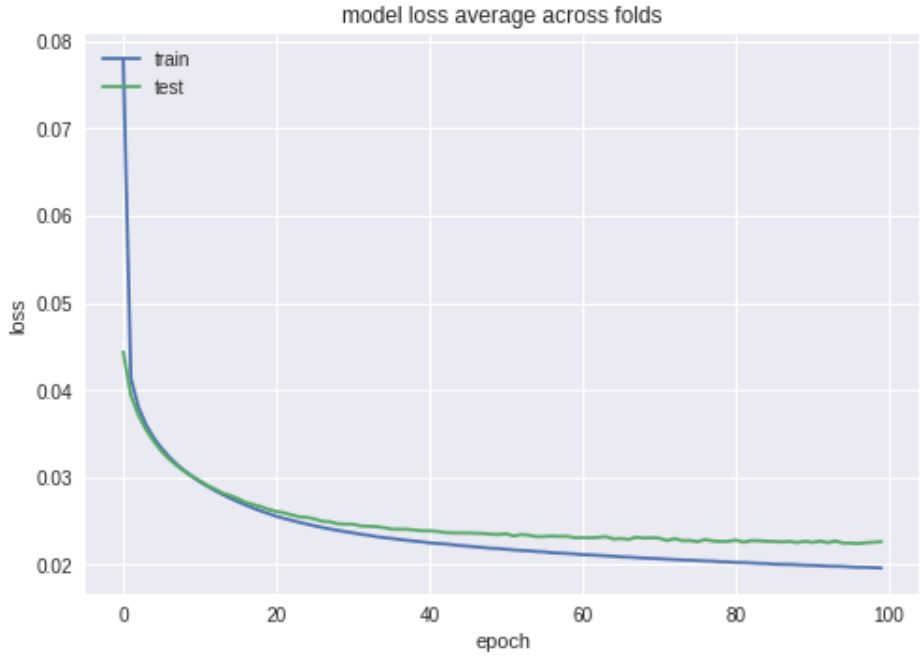

Figure A.7: Loss behaviour of LinnFNN3 model

Figure A.8: Loss behaviour of BGRU model

Figure A.9: Loss behaviour of 2BGRU model

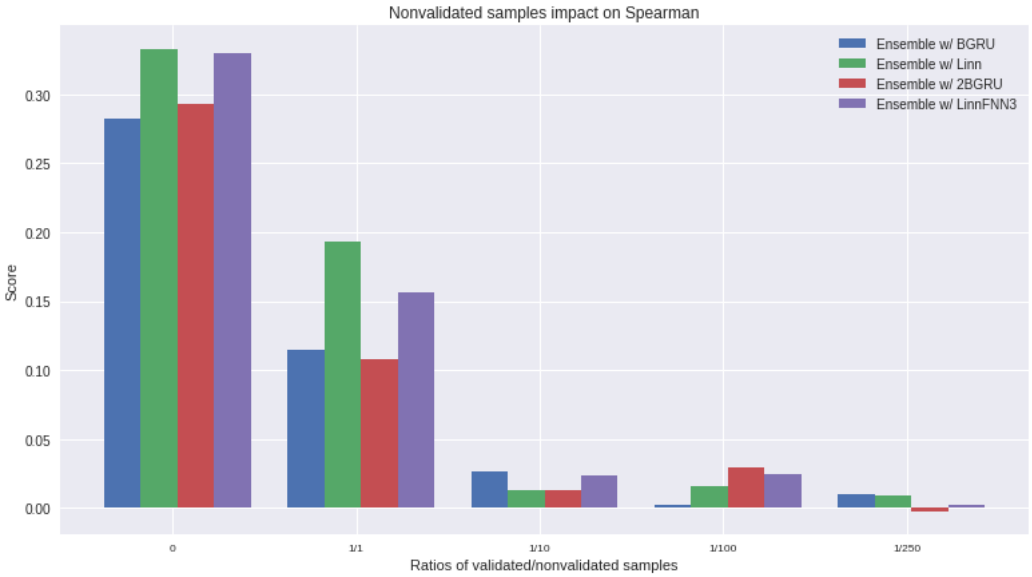

Figure A.10: Correlation results for Spearman under ensemble scheme under inactive site effect

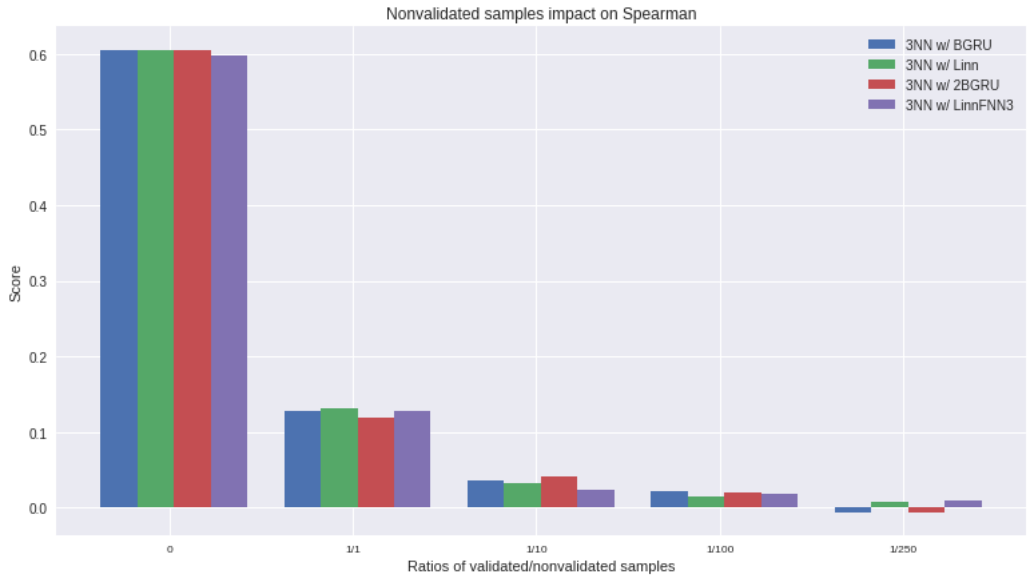

Figure A.11: Correlation results for Spearman under 3NN scheme under inactive site effect

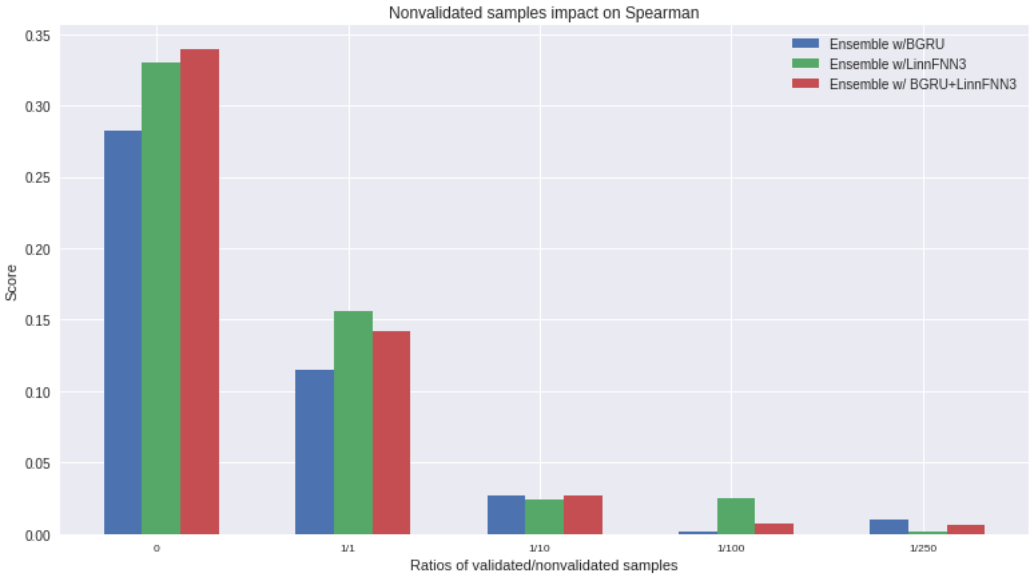

Figure A.12: Correlation results for Spearman under ensemble scheme under inactive site effect with two deep learning models

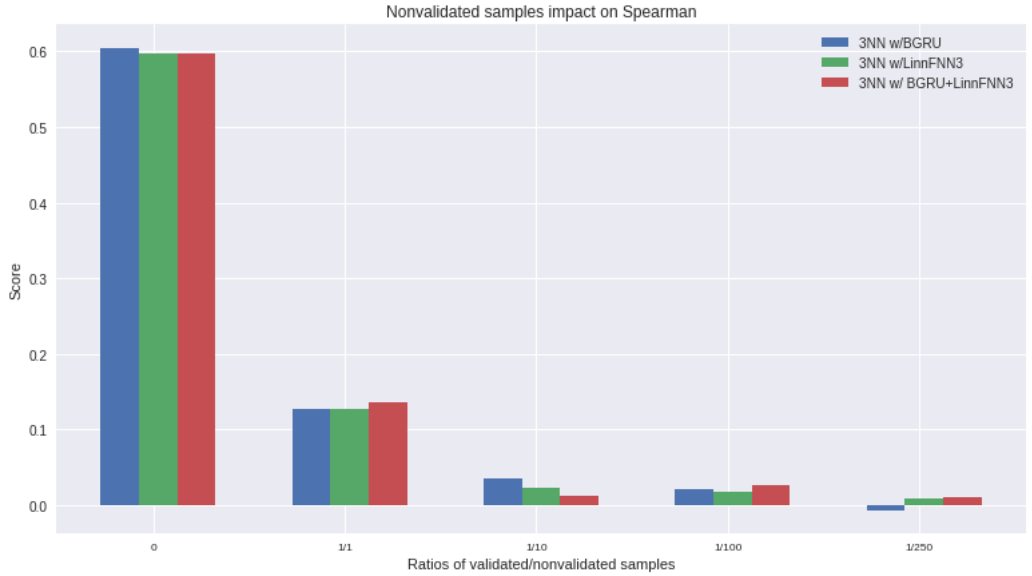

Figure A.13: Correlation results for Spearman under 3NN scheme under inactive site effect with two deep learning models

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
CFD (1/250)	0.019	$1.0e^{-2}$	-0.085	$3.4e^{-4}$
MIT (1/250)	0.012	$1.0e^{-3}$	0.005	$1.4e^{-4}$
CROPIT (1/250)	-0.001	$5.8e^{-4}$	0.000	$3.2e^{-4}$
CCTop (1/250)	-0.003	$6.8e^{-4}$	0.009	$2.3e^{-4}$
CFD (1/100)	0.050	$1.2e^{-2}$	-0.116	$7.3e^{-4}$
MIT (1/100)	0.024	$9.0e^{-4}$	0.009	$2.9e^{-4}$
CROPIT (1/100)	0.007	$5.3e^{-4}$	-0.001	$7.8e^{-4}$
CCTop (1/100)	0.008	$4.4e^{-4}$	0.008	$9.0e^{-4}$
CFD (1/10)	0.132	$9.4e^{-3}$	-0.352	$4.8e^{-4}$
MIT (1/10)	0.082	$3.1e^{-3}$	0.029	$1.0e^{-4}$
CROPIT (1/10)	0.006	$1.0e^{-3}$	-0.002	$5.8e^{-4}$
CCTop (1/10)	0.008	$9.6e^{-4}$	0.035	$6.3e^{-4}$
CFD (1/1)	0.190	$3.1e^{-3}$	-0.499	$3.2e^{-4}$
MIT (1/1)	0.146	$2.0e^{-3}$	0.122	$2.6e^{-4}$
CROPIT (1/1)	0.003	$9.6e^{-4}$	0.064	$6.8e^{-4}$
CCTop (1/1)	0.007	$5.2e^{-4}$	0.128	$6.8e^{-4}$
CFD (1/0)	0.208	$1.6e^{-3}$	0.244	$1.0e^{-3}$
MIT (1/0)	0.211	$1.9e^{-3}$	0.309	$1.4e^{-4}$
CROPIT (1/0)	0.005	$4.8e^{-4}$	0.222	$2.3e^{-4}$
CCTop (1/0)	0.018	$3.2e^{-4}$	0.254	$1.7e^{-4}$
Architecture II (1/250)	0.004	$1.4e^{-4}$	0.009	$1.7e^{-4}$
Architecture II (1/100)	0.027	$9.0e^{-4}$	0.030	$4.0e^{-4}$
Architecture II (1/10)	0.111	$7.2e^{-3}$	0.049	$2.8e^{-3}$
Architecture II (1/1)	0.243	$5.6e^{-3}$	0.156	$1.4e^{-2}$
Architecture II (1/0)	0.462	$1.2e^{-3}$	0.290	$2.1e^{-3}$
3NN (1/250)	0.076	$1.2e^{-2}$	-0.010	$1.9e^{-4}$
3NN (1/100)	0.166	$2.0e^{-2}$	-0.021	$1.9e^{-4}$
3NN (1/10)	0.416	$3.4e^{-2}$	-0.036	$6.7e^{-4}$
3NN (1/1)	0.553	$2.9e^{-3}$	0.117	$7.3e^{-4}$
3NN (1/0)	0.609	$2.1e^{-3}$	0.597	$1.9e^{-4}$

Table A.1: Impact of nonvalidated samples on predictive performance of different methods

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
LinnFNN3 (1/250)	0.075	$1.8e^{-2}$	0.001	$5.8e^{-4}$
LinnFNN3 (1/100)	0.168	$2.1e^{-2}$	0.005	$1.7e^{-3}$
LinnFNN3 (1/10)	0.490	$2.3e^{-2}$	0.027	$5.5e^{-3}$
LinnFNN3 (1/1)	0.612	$9.0e^{-3}$	0.105	$4.8e^{-3}$
LinnFNN3 (1/0)	0.646	$4.1e^{-3}$	0.297	$1.7e^{-3}$
Linn (1/250)	0.094	$2.4e^{-2}$	0.014	$2.9e^{-4}$
Linn (1/100)	0.266	$7.6e^{-2}$	0.043	$1.5e^{-3}$
Linn (1/10)	0.560	$3.5e^{-2}$	0.065	$1.0e^{-2}$
Linn (1/1)	0.612	$9.0e^{-3}$	0.216	$6.0e^{-3}$
Linn (1/0)	0.646	$7.2e^{-3}$	0.300	$5.8e^{-4}$
BGRU (1/250)	0.259	$1.3e^{-1}$	-0.018	$5.8e^{-4}$
BGRU (1/100)	0.510	$4.8e^{-2}$	0.027	$9.0e^{-4}$
BGRU (1/10)	0.626	$2.3e^{-2}$	0.124	$1.7e^{-2}$
BGRU (1/1)	0.650	$8.6e^{-3}$	0.164	$1.4e^{-2}$
BGRU (1/0)	0.664	$4.9e^{-3}$	0.316	$2.6e^{-3}$
2BGRU (1/250)	0.020	$1.0e^{-3}$	-0.024	$7.8e^{-4}$
2BGRU (1/100)	0.138	$6.2e^{-2}$	0.016	$1.1e^{-3}$
2BGRU (1/10)	0.558	$5.6e^{-2}$	0.088	$7.4e^{-3}$
2BGRU (1/1)	0.608	$1.3e^{-2}$	0.017	$1.8e^{-2}$
2BGRU (1/0)	0.637	$6.2e^{-3}$	0.318	$2.6e^{-2}$

Table A.2: Impact of nonvalidated samples on predictive performance of different methods including deep learning models

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
Ensemble w/ LinnFNN3 (1/250)	0.020	$1.8e^{-2}$	0.002	$5.8e^{-4}$
Ensemble w/ LinnFNN3 (1/100)	0.135	$2.1e^{-2}$	0.025	$1.7e^{-3}$
Ensemble w/ LinnFNN3 (1/10)	0.400	$2.3e^{-2}$	0.024	$5.5e^{-3}$
Ensemble w/ LinnFNN3 (1/1)	0.613	$9.0e^{-3}$	0.156	$4.8e^{-3}$
Ensemble w/ LinnFNN3 (1/0)	0.712	$4.1e^{-3}$	0.330	$1.7e^{-3}$
Ensemble w/ Linn (1/250)	0.010	$4.8e^{-4}$	0.009	$6.8e^{-4}$
Ensemble w/ Linn (1/100)	0.122	$2.7e^{-2}$	0.016	$9.0e^{-4}$
Ensemble w/ Linn (1/10)	0.300	$3.0e^{-2}$	0.013	$5.2e^{-3}$
Ensemble w/ Linn (1/1)	0.590	$6.2e^{-3}$	0.193	$2.0e^{-2}$
Ensemble w/ Linn (1/0)	0.706	$2.4e^{-3}$	0.333	$5.9e^{-3}$
Ensemble w/ BGRU (1/250)	0.017	$1.0e^{-3}$	0.010	$5.3e^{-4}$
Ensemble w/ BGRU (1/100)	0.131	$4.2e^{-2}$	0.002	$9.6e^{-4}$
Ensemble w/ BGRU (1/10)	0.407	$3.1e^{-2}$	0.027	$8.8e^{-3}$
Ensemble w/ BGRU (1/1)	0.571	$1.1e^{-2}$	0.115	$8.6e^{-3}$
Ensemble w/ BGRU (1/0)	0.714	$2.2e^{-3}$	0.282	$2.3e^{-3}$
Ensemble w/ 2BGRU (1/250)	0.024	$5.8e^{-4}$	-0.002	$5.8e^{-4}$
Ensemble w/ 2BGRU (1/100)	0.108	$1.5e^{-2}$	0.029	$5.2e^{-3}$
Ensemble w/ 2BGRU (1/10)	0.300	$3.0e^{-2}$	0.013	$5.2e^{-3}$
Ensemble w/ 2BGRU (1/1)	0.514	$1.8e^{-2}$	0.108	$1.7e^{-2}$
Ensemble w/ 2BGRU (1/0)	0.701	$2.2e^{-3}$	0.293	$6.9e^{-3}$
Ensemble w/ BGRU+LinnFNN3 (1/250)	0.044	$8.6e^{-3}$	0.006	$7.8e^{-4}$
Ensemble w/ BGRU+LinnFNN3 (1/100)	0.141	$3.5e^{-2}$	0.007	$1.0e^{-3}$
Ensemble w/ BGRU+LinnFNN3 (1/10)	0.502	$4.7e^{-2}$	0.027	$8.1e^{-3}$
Ensemble w/ BGRU+LinnFNN3 (1/1)	0.637	$1.0e^{-2}$	0.142	$2.5e^{-2}$
Ensemble w/ BGRU+LinnFNN3 (1/0)	0.729	$2.1e^{-3}$	0.339	$4.5e^{-3}$

Table A.3: Impact of nonvalidated samples on predictive performance of different methods including deep learning ensemble models

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
3NN w/ LinnFNN3 (1/250)	0.087	$1.9e^{-2}$	0.009	$2.6e^{-4}$
3NN w/ LinnFNN3 (1/100)	0.259	$4.0e^{-2}$	0.018	$3.6e^{-4}$
3NN w/ LinnFNN3 (1/10)	0.583	$3.7e^{-2}$	0.024	$4.8e^{-4}$
3NN w/ LinnFNN3 (1/1)	0.607	$4.5e^{-3}$	0.127	$7.3e^{-4}$
3NN w/ LinnFNN3 (1/0)	0.657	$2.5e^{-3}$	0.598	$6.4e^{-5}$
3NN w/ Linn (1/250)	0.111	$1.8e^{-2}$	0.008	$2.6e^{-4}$
3NN w/ Linn (1/100)	0.226	$4.9e^{-2}$	0.015	$4.0e^{-4}$
3NN w/ Linn (1/10)	0.540	$3.4e^{-2}$	0.032	$3.2e^{-4}$
3NN w/ Linn (1/1)	0.628	$5.8e^{-3}$	0.132	$4.8e^{-4}$
3NN w/ Linn (1/0)	0.660	$2.1e^{-3}$	0.605	$1.0e^{-4}$
3NN w/ BGRU (1/250)	0.092	$2.3e^{-2}$	-0.006	$3.2e^{-4}$
3NN w/ BGRU (1/100)	0.264	$3.0e^{-2}$	0.021	$3.2e^{-4}$
3NN w/ BGRU (1/10)	0.566	$3.4e^{-2}$	0.035	$7.3e^{-4}$
3NN w/ BGRU (1/1)	0.618	$6.9e^{-3}$	0.127	$6.8e^{-4}$
3NN w/ BGRU (1/0)	0.657	$2.2e^{-3}$	0.605	$8.1e^{-5}$
3NN w/ 2BGRU (1/250)	0.090	$1.5e^{-2}$	-0.006	$2.0e^{-4}$
3NN w/ 2BGRU (1/100)	0.244	$4.1e^{-2}$	0.019	$2.6e^{-4}$
3NN w/ 2BGRU (1/10)	0.569	$4.1e^{-2}$	0.041	$4.4e^{-4}$
3NN w/ 2BGRU (1/1)	0.612	$4.8e^{-3}$	0.118	$4.8e^{-4}$
3NN w/ 2BGRU (1/0)	0.674	$1.7e^{-3}$	0.606	$1.2e^{-4}$
3NN w/ BGRU+LinnFNN3 (1/250)	0.073	$2.5e^{-2}$	0.011	$2.6e^{-4}$
3NN w/ BGRU+LinnFNN3 (1/100)	0.244	$5.1e^{-2}$	0.027	$4.4e^{-4}$
3NN w/ BGRU+LinnFNN3 (1/10)	0.604	$2.7e^{-2}$	0.013	$7.8e^{-4}$
3NN w/ BGRU+LinnFNN3 (1/1)	0.627	$4.9e^{-3}$	0.136	$7.8e^{-4}$
3NN w/ BGRU+LinnFNN3 (1/0)	0.687	$1.8e^{-3}$	0.597	$4.9e^{-5}$

Table A.4: Impact of nonvalidated samples on predictive performance of different methods including deep learning 3NN ensemble models

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
3NN w/ LinnFNN3 (1/250)	0.079	$2.6e^{-2}$	0.012	$9.6e^{-4}$
3NN w/ LinnFNN3 (1/100)	0.250	$5.6e^{-2}$	0.021	$5.8e^{-4}$
3NN w/ LinnFNN3 (1/10)	0.624	$3.8e^{-2}$	0.001	$1.5e^{-2}$
3NN w/ LinnFNN3 (1/1)	0.596	$5.5e^{-3}$	0.134	$1.6e^{-3}$
3NN w/ LinnFNN3 (1/0)	0.657	$2.5e^{-3}$	0.598	$6.4e^{-5}$
Ensemble w/ Linn FNN3 (1/250)	0.032	$3.0e^{-3}$	0.001	$1.3e^{-3}$
Ensemble w/ Linn FNN3 (1/100)	0.219	$7.8e^{-2}$	0.012	$2.9e^{-4}$
Ensemble w/ Linn FNN3 (1/10)	0.535	$6.2e^{-2}$	0.104	$6.6e^{-3}$
Ensemble w/ Linn FNN3 (1/1)	0.644	$1.0e^{-2}$	0.163	$9.2e^{-3}$
Ensemble w/ LinnFNN3 (1/0)	0.712	$4.1e^{-3}$	0.330	$1.7e^{-3}$

Table A.5: Impact of nonvalidated samples on predictive performance of different ensemble models using neural network as binary classifier for inactive site separation

	Pearson		Spearman	
Model	**Mean**	**Var**	**Mean**	**Var**
3NN w/ LinnFNN3 (1/250)	0.232	$1.1e^{-1}$	0.021	$9.0e^{-4}$
3NN w/ LinnFNN3 (1/100)	0.398	$1.8e^{-1}$	0.032	$3.2e^{-3}$
3NN w/ LinnFNN3 (1/10)	0.628	$3.2e^{-2}$	0.056	$2.9e^{-2}$
3NN w/ LinnFNN3 (1/1)	0.608	$5.9e^{-3}$	0.091	$6.3e^{-2}$
3NN w/ LinnFNN3 (1/0)	0.657	$2.5e^{-3}$	0.598	$6.4e^{-5}$
Ensemble w/ Linn FNN3 (1/250)	0.104	$3.0e^{-2}$	0.001	$5.2e^{-3}$
Ensemble w/ Linn FNN3 (1/100)	0.497	$1.1e^{-1}$	0.150	$5.2e^{-2}$
Ensemble w/ Linn FNN3 (1/10)	0.611	$3.5e^{-2}$	0.063	$2.1e^{-1}$
Ensemble w/ Linn FNN3 (1/1)	0.674	$4.8e^{-3}$	0.177	$1.7e^{-2}$
Ensemble w/ LinnFNN3 (1/0)	0.712	$4.1e^{-3}$	0.330	$1.7e^{-3}$
Ensemble w/ BGRU+LinnFNN3 (1/250)	0.161	$6.2e^{-2}$	0.058	$3.0e^{-2}$
Ensemble w/ BGRU+LinnFNN3 (1/100)	0.416	$8.8e^{-2}$	0.097	$4.2e^{-2}$
Ensemble w/ BGRU+LinnFNN3 (1/10)	0.638	$2.3e^{-2}$	0.041	$1.6e^{-1}$
Ensemble w/ BGRU+LinnFNN3 (1/1)	0.670	$6.7e^{-3}$	0.043	$2.2e^{-1}$
Ensemble w/ BGRU+LinnFNN3 (1/0)	0.729	$2.1e^{-3}$	0.339	$4.5e^{-3}$

Table A.6: Impact of nonvalidated samples on predictive performance of different ensemble models using logistic regression as binary classifier for inactive site separation

Models	**Optimiser**		**Architecture**
Architecture I (MLP)	Adam 1e-2 512 100e	Input Latents Network	4. 5. FC 4. ReLU activation.
Architecture II (MLP)	Adam 1e-2 512 100e	Input Latents Network	4. 16. FC 4. ReLU activation.
Architecture III (MLP)	Adam 5e-3 64 100e	Input Latents Network	4. 5. FC 10. ReLU activation.
Architecture IV (MLP)	Adam 5e-3 128 100e	Input Latents Network	4. 16. FC 10. ReLU activation.
LinnFNN3	Adam 1e-3 100 100e	Input Latents Network	80 (flattened 4x20). 50, 20, 10. FC 3. ReLU activation.
Linn	Adam 1e-3 100 100e	Input Latents Network	4x20. Conv 10x1x() 4x, MP 1x5, 100, 23. FC 2. ReLU activation.
BGRU	Adam 1e-3 256 100e	Input Latents Network	20x4. Conv 256, BGRU 256, 128, 64, 40, 1. FC 4. ReLU activation.
2BGRU	Adam 5e-3 512 100e	Input Latents Network	20x4. 2x(Conv 256, BGRU 256, 128, 64, 40), 1. FC 2x(4). ReLU activation.

Table A.7: Model Architecture Details

Models	**Optimiser**
Architecture II / LinnFNN3	Adam 5e-2/1e-3 256/100 100e/25e
Architecture II / Linn	Adam 5e-2/1e-3 512/100 100e/25e
Architecture II / BGRU	Adam 1e-2/1e-3 512/256 100e/25e
Architecture II / 2BGRU	Adam 1e-2/5e-3 512/512 100e/25e
Architecture II / BGRU / LinnFNN3	Adam 1e-2/1e-3/1e-3 512/256/100 100e/25e/25e
Logistic Regression	l2 C = 1 liblinear threshold = 0.917

Table A.8: Model Architecture Details (continued)

List of Figures

References

[1] L. Cong, F. A. Ran, D. Cox, S. Lin, R. Barretto, N. Habib, P. D. Hsu, X. Wu, W. Jiang, L. A. Marraffini, *et al.*, "Multiplex genome engineering using crispr/cas systems," *Science*, vol. 339, no. 6121, pp. 819–823, 2013.

[2] H. Ma, N. Marti-Gutierrez, S.-W. Park, J. Wu, Y. Lee, K. Suzuki, A. Koski, D. Ji, T. Hayama, R. Ahmed, *et al.*, "Correction of a pathogenic gene mutation in human embryos," *Nature*, vol. 548, no. 7668, pp. 413–419, 2017.

[3] R. S. Shapiro, A. Chavez, C. B. Porter, M. Hamblin, C. S. Kaas, J. E. DiCarlo, G. Zeng, X. Xu, A. V. Revtovich, N. V. Kirienko, *et al.*, "A crispr–cas9-based gene drive platform for genetic interaction analysis in candida albicans," *Nature microbiology*, vol. 3, no. 1, pp. 73–82, 2018.

[4] Q. Liu, X. Cheng, G. Liu, B. Li, and X. Liu, "Deep learning improves the ability of sgrna off-target propensity prediction," *BMC bioinformatics*, vol. 21, no. 1, pp. 1–15, 2020.

[5] "Addgene: Crispr guide." https://www.addgene.org/guides/crispr/. (Accessed on 02/08/2021).

[6] J. Lin and K.-C. Wong, "Off-target predictions in crispr-cas9 gene editing using deep learning," *Bioinformatics*, vol. 34, no. 17, pp. i656–i663, 2018.

[7] G. Chuai, H. Ma, J. Yan, M. Chen, N. Hong, D. Xue, C. Zhou, C. Zhu, K. Chen, B. Duan, *et al.*, "Deepcrispr: optimized crispr guide rna design by deep learning," *Genome biology*, vol. 19, no. 1, pp. 1–18, 2018.

[8] F. Störtz and P. Minary, "crisprsql: a novel database platform for crispr/cas off-target cleavage assays," *Nucleic Acids Research*, vol. 49, no. D1, pp. D855–D861, 2021.

[9] N. Hoque, M. Singh, and D. K. Bhattacharyya, "Efs-mi: an ensemble feature selection method for classification," *Complex & Intelligent Systems*, vol. 4, no. 2, pp. 105–118, 2018.

[10] "Genomics — britannica." https://www.britannica.com/science/genomics. (Accessed on 06/08/2021).

[11] J. Hardin, G. P. Bertoni, and L. J. Kleinsmith, *Becker's World of the Cell, eBook*. Pearson Higher Ed, 2017.

[12] D. C. Wang and X. Wang, "Off-target genome editing: A new discipline of gene science and a new class of medicine," 2019.

[13] F. Alkan, A. Wenzel, C. Anthon, J. H. Havgaard, and J. Gorodkin, "Crispr-cas9 off-targeting assessment with nucleic acid duplex energy parameters," *Genome biology*, vol. 19, no. 1, pp. 1–13, 2018.

[14] L. Pray, "Discovery of dna structure and function: Watson and crick," *Nature Education*, vol. 1, no. 1, 2008.

[15] T. Kirkwood, "Dna, mutations and aging," *Mutation Research/DNAging*, vol. 219, no. 1, pp. 1–7, 1989.

[16] R. O. Bak, N. Gomez-Ospina, and M. H. Porteus, "Gene editing on center stage," *Trends in Genetics*, vol. 34, no. 8, pp. 600–611, 2018.

[17] P. Liang, Y. Xu, X. Zhang, C. Ding, R. Huang, Z. Zhang, J. Lv, X. Xie, Y. Chen, Y. Li, *et al.*, "Crispr/cas9-mediated gene editing in human tripronuclear zygotes," *Protein & cell*, vol. 6, no. 5, pp. 363–372, 2015.

[18] M. Haeussler, K. Schönig, H. Eckert, A. Eschstruth, J. Mianné, J.-B. Renaud, S. Schneider-Maunoury, A. Shkumatava, L. Teboul, J. Kent, *et al.*, "Evaluation of off-target and on-target scoring algorithms and integration into the guide rna selection tool crispor," *Genome biology*, vol. 17, no. 1, pp. 1–12, 2016.

[19] P. D. Hsu, D. A. Scott, J. A. Weinstein, F. A. Ran, S. Konermann, V. Agarwala, Y. Li, E. J. Fine, X. Wu, O. Shalem, *et al.*, "Dna targeting specificity of rna-guided cas9 nucleases," *Nature biotechnology*, vol. 31, no. 9, pp. 827–832, 2013.

[20] "sgrna scoring help." https://portals.broadinstitute.org/gpp/public/software/sgrna-scoring-help. (Accessed on 07/08/2021).

[21] G. Liu, Y. Zhang, and T. Zhang, "Computational approaches for effective crispr guide rna design and evaluation," *Computational and structural biotechnology journal*, vol. 18, pp. 35–44, 2020.

[22] J. G. Doench, N. Fusi, M. Sullender, M. Hegde, E. W. Vaimberg, K. F. Donovan, I. Smith, Z. Tothova, C. Wilen, R. Orchard, *et al.*, "Optimized sgrna design to maximize activity and minimize off-target effects of crispr-cas9," *Nature biotechnology*, vol. 34, no. 2, pp. 184–191, 2016.

[23] R. Singh, C. Kuscu, A. Quinlan, Y. Qi, and M. Adli, "Cas9-chromatin binding information enables more accurate crispr off-target prediction," *Nucleic acids research*, vol. 43, no. 18, pp. e118–e118, 2015.

[24] M. Stemmer, T. Thumberger, M. del Sol Keyer, J. Wittbrodt, and J. L. Mateo, "Cctop: an intuitive, flexible and reliable crispr/cas9 target prediction tool," *PloS one*, vol. 10, no. 4, p. e0124633, 2015.

[25] S.-J. Chen, "Minimizing off-target effects in crispr-cas9 genome editing," 2019.

[26] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–16, 2016.

[27] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[28] T. M. Mitchell *et al.*, "Machine learning," 1997.

[29] L. Fahrmeir, T. Kneib, S. Lang, and B. Marx, *Regression*. Springer, 2007.

[30] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[31] M. Martin, "On-line support vector machine regression," in *European Conference on Machine Learning*, pp. 282–294, Springer, 2002.

[32] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[33] M. Hofmann, "Support vector machines-kernels and the kernel trick," *Notes*, vol. 26, no. 3, pp. 1–16, 2006.

[34] F. Burba, F. Ferraty, and P. Vieu, "k-nearest neighbour method in functional nonparametric regression," *Journal of Nonparametric Statistics*, vol. 21, no. 4, pp. 453–469, 2009.

[35] S. Kohli, G. T. Godwin, and S. Urolagin, "Sales prediction using linear and knn regression," in *Advances in Machine Learning and Computational Intelligence*, pp. 321–329, Springer, 2021.

[36] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[37] P. Blunsom and A. Calinescu, "Machine learning." https://www.cs.ox.ac.uk/teaching/courses/2021-2022/ml/. (Accessed on 10/08/2021).

[38] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," *Applied and Computational Harmonic Analysis*, vol. 43, no. 2, pp. 233–268, 2017.

[39] L. Noriega, "Multilayer perceptron tutorial," *School of Computing. Staffordshire University*, 2005.

[40] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *International Conference on Machine Learning*, pp. 1675–1685, PMLR, 2019.

[41] S. Saha, "A comprehensive guide to convolutional neural networks — the eli5 way — by sumit saha — towards data science." https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. (Accessed on 11/08/2021).

[42] A. S. Mubarak, A. Süleyman, O. Mehmet, *et al.*, "Development of cnn model for prediction of crispr/cas12 guide rna activity," in *International Conference on Theory and Application of Soft Computing, Computing with Words and Perceptions*, pp. 697–703, Springer, 2019.

[43] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[44] sey kh, "Gru recurrent network." https://gist.github.com/sey-kh/34cd13a4139b30776ff697b431c3c370. (Accessed on 11/08/2021).

[45] Z. Shen, W. Bao, and D.-S. Huang, "Recurrent neural network for predicting transcription factor binding sites," *Scientific reports*, vol. 8, no. 1, pp. 1–10, 2018.

[46] Y. Deng, L. Wang, H. Jia, X. Tong, and F. Li, "A sequence-to-sequence deep learning architecture based on bidirectional gru for type recognition and time location of combined power quality disturbance," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 8, pp. 4481–4493, 2019.

[47] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[48] H. Guo and S. B. Gelfand, "Analysis of gradient descent learning algorithms for multilayer feedforward neural networks," in *29th IEEE Conference on Decision and Control*, pp. 1751–1756, IEEE, 1990.

[49] C. Bauckhage and D. Speicher, "Lecture notes on machine learning neurons with non-monotonic activation functions," *cal*, vol. 5, p. 4, 1943.

[50] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *1990 IJCNN International Joint Conference on Neural Networks*, pp. 21–26, IEEE, 1990.

[51] I. K. M. Jais, A. R. Ismail, and S. Q. Nisa, "Adam optimization algorithm for wide and deep neural network," *Knowledge Engineering and Data Science*, vol. 2, no. 1, pp. 41–46, 2019.

[52] U. Ruby and V. Yendapalli, "Binary cross entropy with deep learning technique for image classification," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 10, 2020.

[53] S. S. Durbha, R. L. King, and N. H. Younan, "Support vector machines regression for retrieval of leaf area index from multiangle imaging spectroradiometer," *Remote sensing of environment*, vol. 107, no. 1-2, pp. 348–361, 2007.

[54] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[55] S. Abadi, W. X. Yan, D. Amar, and I. Mayrose, "A machine learning approach for predicting crispr-cas9 cleavage efficiencies and patterns underlying its mechanism of action," *PLoS computational biology*, vol. 13, no. 10, p. e1005807, 2017.

[56] L. Myers and M. J. Sirois, "Spearman correlation coefficients, differences between," *Encyclopedia of statistical sciences*, vol. 12, 2004.

[57] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[58] T. T. Joy, S. Rana, S. Gupta, and S. Venkatesh, "Hyperparameter tuning for big data using bayesian optimisation," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2574–2579, IEEE, 2016.

[59] G. Jiang and W. Wang, "Error estimation based on variance analysis of k-fold cross-validation," *Pattern Recognition*, vol. 69, pp. 94–106, 2017.

[60] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777, 2017.

[61] C. Molnar, "5.10 shapley values — interpretable machine learning." https://christophm.github.io/interpretable-ml-book/shapley.html. (Accessed on 12/08/2021).

[62] L. Ibrahim, M. Mesinovic, K.-W. Yang, and M. A. Eid, "Explainable prediction of acute myocardial infarction using machine learning and shapley values," *IEEE Access*, vol. 8, pp. 210410–210417, 2020.

[63] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan, "L-shapley and c-shapley: Efficient model interpretation for structured data," *arXiv preprint arXiv:1808.02610*, 2018.

[64] R. Polikar, "Ensemble learning," in *Ensemble machine learning*, pp. 1–34, Springer, 2012.

[65] M. Autenrieth, R. A. Levine, J. Fan, M. A. Guarcello, *et al.*, "Stacked ensemble learning for propensity score methods in observational studies," *Journal of Educational Data Mining*, vol. 13, no. 1, pp. 24–189, 2021.

[66] F. Divina, A. Gilson, F. Goméz-Vela, M. García Torres, and J. F. Torres, "Stacking ensemble learning for short-term electricity consumption forecasting," *Energies*, vol. 11, no. 4, p. 949, 2018.

[67] S. W. Cho, S. Kim, Y. Kim, J. Kweon, H. S. Kim, S. Bae, and J.-S. Kim, "Analysis of off-target effects of crispr/cas-derived rna-guided endonucleases and nickases," *Genome research*, vol. 24, no. 1, pp. 132–141, 2014.

[68] R. L. Frock, J. Hu, R. M. Meyers, Y.-J. Ho, E. Kii, and F. W. Alt, "Genome-wide detection of dna double-stranded breaks induced by engineered nucleases," *Nature biotechnology*, vol. 33, no. 2, pp. 179–186, 2015.

[69] S. Q. Tsai, Z. Zheng, N. T. Nguyen, M. Liebers, V. V. Topkar, V. Thapar, N. Wyvekens, C. Khayter, A. J. Iafrate, L. P. Le, *et al.*, "Guide-seq enables genome-wide profiling of off-target cleavage by crispr-cas nucleases," *Nature biotechnology*, vol. 33, no. 2, pp. 187–197, 2015.

[70] D. Kim, B. Langmead, and S. L. Salzberg, "Hisat: a fast spliced aligner with low memory requirements," *Nature methods*, vol. 12, no. 4, pp. 357–360, 2015.

[71] X. Wang, Y. Wang, X. Wu, J. Wang, Y. Wang, Z. Qiu, T. Chang, H. Huang, R.-J. Lin, and J.-K. Yee, "Unbiased detection of off-target cleavage by crispr-cas9 and talens using integrase-defective lentiviral vectors," *Nature biotechnology*, vol. 33, no. 2, pp. 175–178, 2015.

[72] F. A. Ran, L. Cong, W. X. Yan, D. A. Scott, J. S. Gootenberg, A. J. Kriz, B. Zetsche, O. Shalem, X. Wu, K. S. Makarova, *et al.*, "In vivo genome editing using staphylococcus aureus cas9," *Nature*, vol. 520, no. 7546, pp. 186–191, 2015.

[73] D. Kim, S. Kim, S. Kim, J. Park, and J.-S. Kim, "Genome-wide target specificities of crispr-cas9 nucleases revealed by multiplex digenome-seq," *Genome research*, vol. 26, no. 3, pp. 406–415, 2016.

[74] M. Haeussler, "maximilianh/crisporwebsite: All source code of the crispor.org website." https://github.com/maximilianh/crisporWebsite. (Accessed on 14/08/2021).

[75] J. Listgarten, M. Weinstein, B. P. Kleinstiver, A. A. Sousa, J. K. Joung, J. Crawford, K. Gao, L. Hoang, M. Elibol, J. G. Doench, *et al.*, "Prediction of off-target activities for the end-to-end design of crispr guide rnas," *Nature biomedical engineering*, vol. 2, no. 1, pp. 38–47, 2018.

[76] J. Lin, Z. Zhang, S. Zhang, J. Chen, and K.-C. Wong, "Crispr-net: A recurrent convolutional network quantifies crispr off-target activities with mismatches and indels," *Advanced Science*, vol. 7, no. 13, p. 1903562, 2020.

[77] G. Zhang, Z. Dai, and X. Dai, "C-rnncrispr: Prediction of crispr/cas9 sgrna activity using convolutional and recurrent neural networks," *Computational and structural biotechnology journal*, vol. 18, pp. 344–354, 2020.

[78] C. R. Pernet, R. R. Wilcox, and G. A. Rousselet, "Robust correlation analyses: false positive and power validation using a new open source matlab toolbox," *Frontiers in psychology*, vol. 3, p. 606, 2013.

[79] A. Newman, L. Starrs, and G. Burgio, "Cas9 cuts and consequences; detecting, predicting, and mitigating crispr/cas9 on-and off-target damage: Techniques for detecting, predicting, and mitigating the on-and off-target effects of cas9 editing," *BioEssays*, vol. 42, no. 9, p. 2000047, 2020.

[80] Y. Gao, G. Chuai, W. Yu, S. Qu, and Q. Liu, "Data imbalance in crispr off-target prediction," *Briefings in bioinformatics*, vol. 21, no. 4, pp. 1448–1454, 2020.