Skolkovo Institute of Science and Technology

UNCERTAINTY QUANTIFICATION AND NEURAL NETWORK INTERPRETATION FOR STUDYING CRISPR MECHANICS

Doctoral Thesis
by

BOGDAN KIRILLOV

DOCTORAL PROGRAM IN LIFE SCIENCES

Supervisor
Assistant Professor, Maxim Panov

Moscow - 2023

I hereby declare that the work presented in this thesis was carried out by myself at Skolkovo Institute of Science and Technology, Moscow, except where due acknowledgement is made, and has not been submitted for any other degree.

Candidate (Bogdan Kirillov)

Supervisor (Assistant Professor Maxim Panov)

Abstract

Genome editing facilitated by CRISPR-Cas is a powerful tool for both fundamental investigations of biological reality and industrial applications of biotechnology. Despite the illusory simplicity of a CRISPR-associated (Cas) protein cleavage experiment, limitless amounts of experimental setups utilize Cas proteins as tools to generate an immense pool of data that requires statistical analysis. Each step of any experimental pipeline, starting from amplification to Next Generation Sequencing, increases the amount of noise in resulting data sets which leads to the need for extensive applications of advanced mathematical methods. This dissertation is devoted to an investigation of CRISPR mechanisms and the development of tools ready to use in biotechnological applications made possible by the careful employment of Deep Neural Networks, Explainable Machine Learning, and Uncertainty Quantification. While researching for the two projects in this dissertation (potential off-target event detection and cleavage efficiency estimation), several state-of-the-art models were developed for estimating the cleavage efficiency of a gRNA-Cas protein complex. The accuracy of the models received additional biological validation through an unsupervised rediscovery of a well-known factor that determines cleavage efficiency (seed region importance). This rediscovery was made possible through recent developments in the fields of Explainable Machine Learning and Neural Network Interpretation. Via the addition of prediction uncertainty as a novel axis for off-target cleavage efficiency analysis, a previously unknown diversity of possible off-target cleavage behavior was discovered, allowing for a new approach to select optimal gRNA for gene editing experimentation. Additionally, this dissertation presents a set of mathematical methods for supervised anomaly detection that helps accurately recognize rare biological events. These methods were applied to CRISPR off-target recognition and passed a careful comparison study. Due to their superior accuracy and efficiency, these methods are a good addition to the tool set of an experimental biologist.

Publications

Main author

1. B. Kirillov, E. Savitskaya, M. Panov, A. Y. Ogurtsov, S. A. Shabalina, E. V. Koonin, and K. V. Severinov. Uncertainty-aware and interpretable evaluation of Cas9–gRNA and Cas12a–gRNA specificity for fully matched and partially mismatched targets with Deep Kernel Learning. Nucleic acids research, 50(2), e11-e11. 01 2022. [1]

2. B. Kirillov and M. Panov. Measuring internal inequality in capsule networks for supervised anomaly detection. Scientific reports, 12(1), 1-11. 08 2022. [2]

Conference presentations

1. B. Kirillov and M. Panov. Deterministic DNA embeddings with no predefined k-mer length. MCCMB 2021, Moscow, Russian Federation, 2021.

2. B. Kirillov, A. Stepakov, P. Muzyukina, K. Severinov. Explainable Clostridium difficile variant classification based on CRISPR array contents. CRISPR 2021, online, 2021.

3. B. Kirillov, A. Vasileva, O. Fedorov, M. Panov, K. Severinov. The use of Active Machine Learning for Protospacer-Adjacent Motif recovery in Class 2 CRISPR-Cas systems. BelBI 2023, Belgrade, Republic of Serbia, 2023.

4. I. Sharov, B. Kirillov. Bioinformatical search for new CRISPR-associated proteins using deep neural networks. MCCMB 2023, Moscow, Russian Federation, 2023. (in Russian)

Contents

List of symbols, Abbreviations

CRISPR Clustered Regularly Interspaced Short Palindromic Repeats

Cas CRISPR-associated

gRNA guide RNA

PAM Protospacer-Adjacent Motif

DKL Deep Kernel Learning

HOM Hit-Or-Miss

CN Capsule Networks

MSE Mean Squared Error

ELBO Evidence Lower BOund

GP Gaussian Process

LSTM Long Short Term Memory

GRU Gated Recurrent Unit

GFP Green Fluorescent Protein

UQ Uncertainty Quantification

ML Machine Learning

SVM Support Vector Machine

GPCR G Protein Coupling Receptors

RNN Recurrent Neural Network

CNN Convolutional Neural Network

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

A number of valuable biotechnological tools starting from precise *in vitro* [6], *ex vivo* [7] and *in vivo* [8] gene editing to DNA computing [9] could be achieved through employing experimental setups derived from a kind of prokaryotic acquired immunity system called CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) – Cas (CRISPR-associated). In these applications, an RNA-Guided Nuclease (RGN) is used to bind (and, in many cases, cleave) a target DNA via the guidance of a special guide RNA (gRNA). This binding and subsequent DNA cleavage are a stochastic process and the exhaustive study of this process requires application of statistical analysis and Machine Learning. CRISPR-Cas experiments are a source of big data, so a researcher would benefit from Deep Learning to fully utilize the information stored in its results.

The strength of base-pairing interactions can modulate the RGN binding affinity and reduce off-target effects [10, 11]. Improvement in specificity, on-target cleavage efficiency, and reduction of off-target cleavage can be achieved through directed engineering of the RGN or sgRNA, as well as modification of Cas-sgRNA delivery methods. Improvements in each of these directions are crucial for the full realization of the enormous potential of the RGN-based technologies [10, 11, 12], but this dissertation is devoted purely to improvements in computational methods.

Overall, accurate estimation of Cas-protein cleavage efficiency is crucial for the successful application of CRISPR-based technologies. Computational models can provide valuable insight into the cleavage efficiency of a given gRNA-Cas complex. In

recent years, deep learning methods have been applied to the estimation of CRISPR cleavage efficiency [13]. By training the neural network on a large dataset of known target sites and their corresponding cleavage efficiencies, it is possible to develop a model that can accurately predict the cleavage efficiency for a given target site.

One advantage of using deep learning methods is that they can consider a wide range of factors that may influence cleavage efficiency. For example, the model can take into account the sequence context of the target site [14], the composition and structure of the guide RNA [15], as well as chromatin accessibility [16]. This can provide a more accurate prediction of cleavage efficiency than is possible using simpler methods. The importance of sequence and non-sequence factors can be highlighted with Explainable Machine Learning [17] which helps in rediscovery of known and identification of novel biological results.

Another option to improve CRISPR-Cas experimental design is to employ advanced Machine Learning methods that go beyond point estimates of cleavage efficiency. One of such ways is Uncertainty Quantification [18] – analysis of Deep Learning prediction errors. Addition of information regarding uncertainty is crucial for careful planning of experiment because it allows to take the expected deviation of real results from the prediction into account.

The results of the current dissertation are presented in three parts. First one (Chapter 5) describes potential off-target event detection through investigation of inequalities in coupling coefficients within Capsule Networks with dynamic routing. This part arose mostly from experiments with off-target classification that lead to decision to opt for routing-less Capsule Networks and hybrid Gaussian Process – Hit-or-Miss capsule systems that are described in the second part (Chapter 6). The third part (Chapter 7) presents the diversity of an off-target cleavage space that was discovered through application of Uncertainty Quantification. Discussion of the results and conclusion are presented in Chapter 8.

Chapter 2

Background

2.1 Structure and evolution of CRISPR-Cas system

Clustered Regularly Interspaced Short Palindromic Repeats is a multifaceted system present in over 50% of known bacteria and 90% of known archaea [19]. The system is believed to be a part of bacterial acquired immunity because it provides the functions of immunity memory and pathogen elimination [20]. The system consists of a region of bacterial chromosome and/or plasmid, called CRISPR array, a set of short non-coding RNAs and a set of proteins, called CRISPR-associated (Cas) proteins. CRISPR array of a well-studied bacterium *Staphilococcus pyogenes* is shown in Figure 2-1.

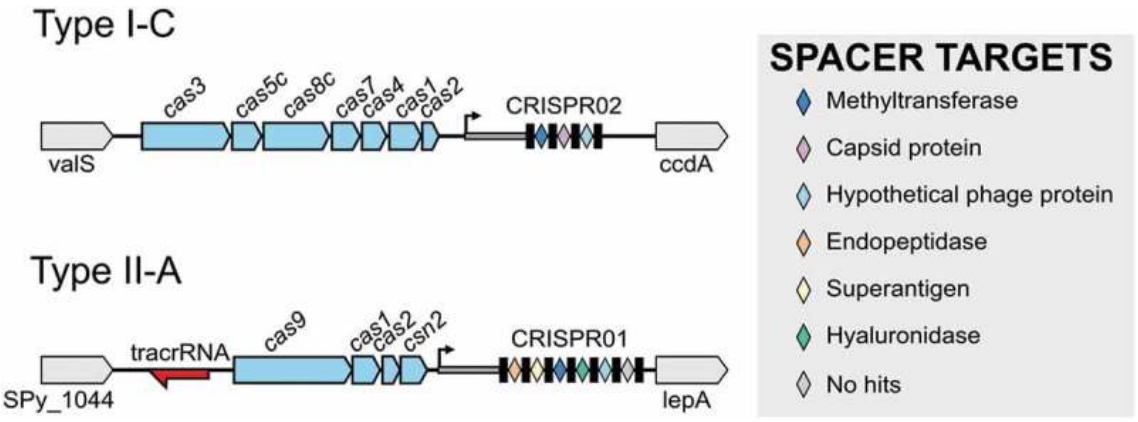

Figure 2-1: Genomic organization of type I-C and type II-A CRISPR-Cas loci in *S. pyogenes* SF370 (image source [21]).

Evolutionary classification of CRISPR-Cas systems is an area of ongoing re-

search. A best way to immerse oneself into its history is to read the series of review papers by Evolutionary Genomics Research Group [22, 23, 24, 25, 26] which is considered currently the main source for a birds-eye view on basic concepts of CRISPR-Cas evolution.

According to the latest version of the classification [26], CRISPR-Cas systems consist of two large classes, six types and over 30 subtypes with new subtypes being discovered all the time (as shown in Figure 2-2). The current section is an attempt to give a concise introduction into the topic and does not aim to cover all possible aspects of CRISPR-Cas evolution, only highlight a number of interesting facets within it.

Palindromic repeats (shown in Figure 2-1 as small black rectangles) that separate spacers from each other form a hairpin loop that helps in the biogenesis of gRNAs [27]. Each spacer in the CRISPR array (shown in Figure 2-1 as diamonds of various colors) correspond to a target sequence in the genome of some infectious agents, for example, a phage or a plasmid. Such target sequence is called protospacer. Spacers can come from various sources, namely, genes encoding capsid proteins or methyltransferases (as Figure 2-1 shows). Interestingly, the proportion of spacers that match the protospacers from the open access genomic databases exactly or near-exactly is quite small [28]. It means that for the majority of spacers in a spacerome for a bacterium of interest, the question of spacer origin remains open.

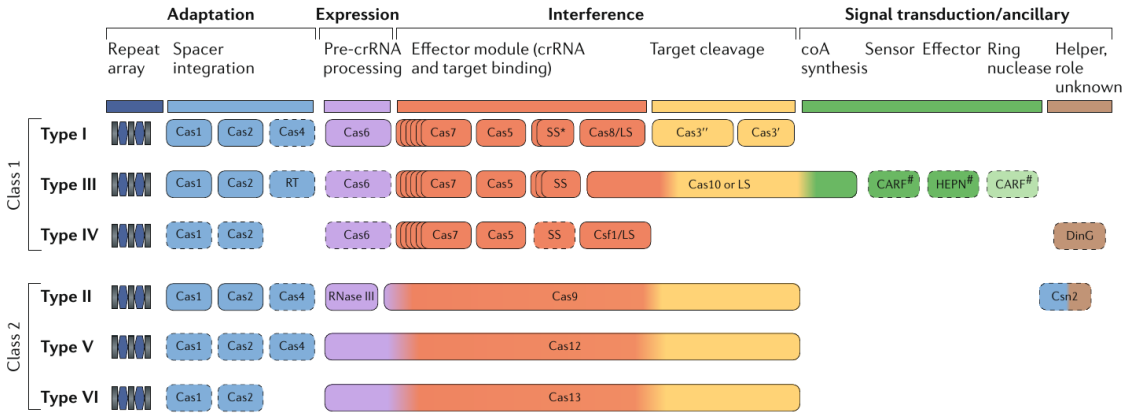

Figure 2-2: Schematic representation of CRISPR-Cas evolutionary classification, extracted from Figure 1 of [26].

CRISPR arrays can contain spacers that target the genome of the bacterium

itself. Self-targeting tends to be avoided [29, 30] but in some cases it may be a form of gene expression regulation [31, 32] which shows that CRISPR systems have functionality beyond acquired immunity [33, 34]. And indeed, a lot of exaptation cases were found, such as a minimal type I-F CRISPR system recruited for propagation of Tn7-like transposon [35] or a damaged variant of type I-E system involved in signaling [36].

A special case of CRISPR-Cas system exaptation is dubbed "guns for hire" [37] - a usage of Cas proteins, adaptation or interference modules, crRNAs and CRISPR arrays by not only bacteria and archaea for defense but also by adversarial mobile genetic elements (e.g. transposons, viruses, plasmids and so on) for attack. For example, some viruses use mini-CRISPR arrays containing spacers that target related viruses present in the same environment to stove off the competition [38]. There is a substantial gene flow between bacteria and MGEs in both directions which leads to a rich and complex arms race [39, 40].

Evolutionary classification of CRISPR systems largely depends on the type of its interference unit [26]. There are two large classes: class 1 and class 2. For class 1 CRISPR systems, there are a number of Cas proteins needed to construct the interference unit (see Figure 2-2 for the shortlist). In class 2, there is usually only one multifunctional effector (Cas9. Cas12 or Cas13) that operates not only during interference, but also participates in crRNA expression.

The machinery for three main stages (adaptation, expression and interference, described in details in the next section) aside, CRISPR-Cas systems possess also a number of accessory genes with functionality still under investigation, for example CARF and non-CARF (CRISPR-associated Rossman Fold – a structural motif responsible for DNA or RNA recognition [41]) accessory genes [42]. Some of such proteins with unknown functions are predicted to be membrane-associated [43] which is quite interesting because it might highlight another new function of CRISPR-Cas system.

This dissertation is concerned with class 2 CRISPR-Cas systems that use Cas9 and Cas12a as interference proteins. There are a lot of known Cas9 orthologs that have unique properties like activity at a high temperature, long PAM, small size and

so on [44]. Cas12a was discovered more recently and is not studied as thoroughly as Cas9. Class 2 systems were chosen for this research mostly due to their extreme importance for biotechnological applications of CRISPR [45].

2.2 Immune response of CRISPR-Cas system

CRISPR system performs an immune response in three main stages – adaptation, generation of CRISPR RNAs (crRNAs) and interference [46]. Different types of CRISPR systems use different machinery for the immune response but all known CRISPR systems follow this three-stage scheme:

Spacer acquisition or adaptation. Adaptation involves integration of foreign DNA fragments into the CRISPR array [47]. During this phase, the parts of a pathogen (virus or plasmid) DNA are incorporated into the CRISPR array which acts as a storage facility. This allows further use of genetic information for threat recognition and elimination in the future. There are two kinds of adaptation – naive and primed [47]. Naive spacer acquisition is running, when there is no suitable spacers already present in the CRISPR array, when the pathogen is not known. Primed spacer acquisition is running when there is already an interference process ongoing. On a molecular level, naive adaptation involves the usage of Cas1-Cas2 protein complex [47] which consist of the most conserved CRISPR-associated proteins Cas1 and Cas2 that occur virtually in every CRISPR system studied [48]. In addition to Cas1-Cas2, adaptation heavily involves RecBCD protein and other genome maintenance machinery [49, 50, 51]. Naive adaptation also requires RecBCD, but not as a nuclease, but rather as a helicase [49], unlike in primed adaptation, where both helicase and nuclease activity of Cas3 is needed [47]. The scheme for construction of adaptation module is shown in Figure 2-3.

While naive adaptation uses doublestrand breaks produced by different events like stalled replication forks or work of restriction endonucleases (Figure 2-3A), primed adaptation uses doublestrand breaks produced by nuclease activity of Cas3 (Figure 2-3B) or Cas9 (Figure 2-3C). In class 2 systems, in addition to Cas9, acces-

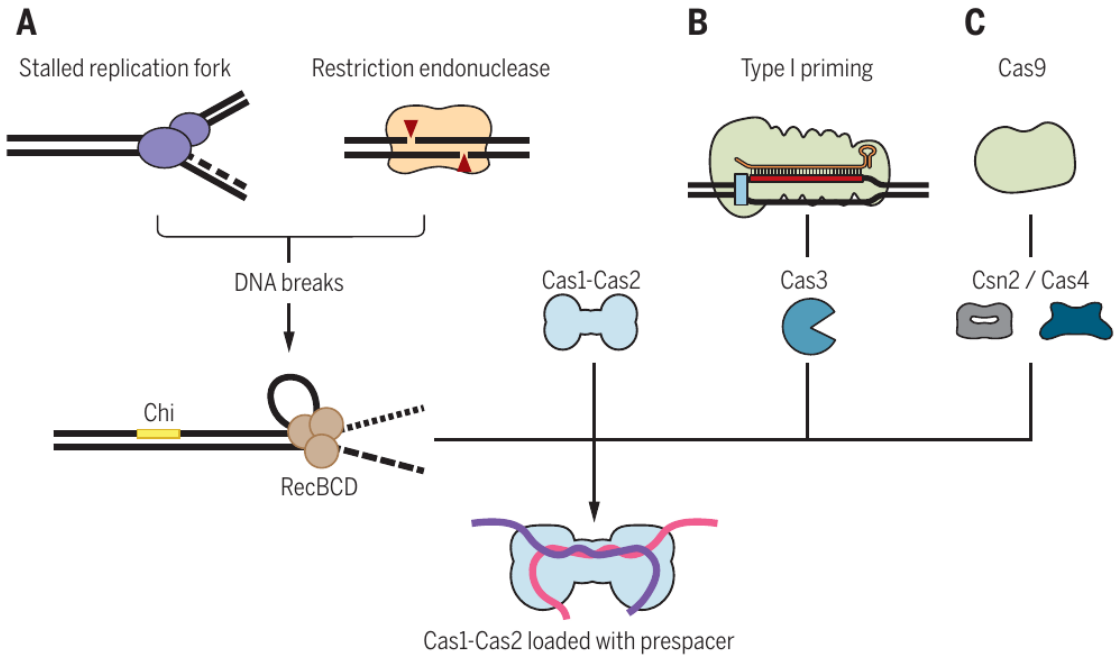

Figure 2-3: Different ways to construct the adaptation complex (image source [47]). **A** naive adaptation. **B** primed adaptation in class 1 systems. **C** primed adaptation in class 2 systems.

sory proteins like Cas4 or Csn2 also may be involved.

Biogenesis of crRNA. Before the effector complex can perform its main function, the RNAs that act as guides for CRISPR interference [52] have to be built. In several stages, the genetic information stored during adaptation phase is recovered from the CRISPR arrays starting from a long precursor RNA molecule (pre-crRNA) followed by separation into several crRNAs each of which contains a single spacer. There are a number of distinct crRNA biogenesis pathways which are specific for different CRISPR systems [27] but careful introduction in crRNA production is out of scope for the current dissertation.

Interference. Interference phase involves destruction of foreign DNA [53] using the crRNA constructed during the biogenesis. The interference mode is performed when the adversarial DNA is caught in the cell. A single interference unit consists of one or several Cas proteins and a number of auxiliary RNAs, one of which, the CRISPR RNA (crRNA) is expressed out of a CRISPR array and should

be complementary to the target inside the adversarial DNA. In Class 2 CRISPR systems, only one protein is responsible for interference. During the interference [54], first, gRNA forms a complex with the effector protein, for example, Cas9. Then, as gRNA-Cas9 complex binds to the target DNA, the complex undergoes conformational changes, unwinds the target DNA, hybridizes the target DNA with the complementary segment of the gRNA, and, in case of success, performs a double-strand break in the target DNA. In case of immunity response, the fragments of the target DNA are recognized by cellular machinery for DNA degradation [53], and in case of gene editing, the double-strand break undergoes reparation and inclusion of the DNA insert [55].

2.3 The molecular mechanism of target recognition

In class 2 CRISPR-Cas systems, interference is performed by a single protein, for example Cas9 or Cas12a (also known as Cpf1 [56]). Effector in class 2 systems is a multidomain protein that has the endonuclease activity used in the interference phase and cuts the foreign DNA [26]. A single protein is the signature of class 2 CRISPR-Cas systems, while in class 1 the interference is performed by a combination of several proteins instead [57].

Before we delve deep into the mechanism of CRISPR target recognition, let us discuss the various RNAs that CRISPR system uses:

gRNA. A guide RNA is a molecule that is used for targeting the adversarial DNA sequence during the interference phase of immune response. It consists of two components, namely, the crRNA and tracrRNA, described below. The term "gRNA" may be used interchangeably [58, 59] with the term "sgRNA" (single guide RNA) but there is distinction in that "gRNA" is usually involved to denote the "natural" combination of crRNA and tracrRNA that occurs during the formation of the effector complex, while "sgRNA" denotes a synthetic combination of crRNA and tracrRNA which exists as a single molecule at all times [60]. However, a single guide RNA can emerge naturally and participate, for example, in regulation of Cas

gene expression [32].

crRNA. A CRISPR RNA [61] is a short molecule that contains a sequence that is complementary to the target DNA region (usually 17-20 base pairs long). It is responsible for targeting the specific region in the DNA.

tracrRNA. A trans-activating CRISPR RNA [62] is a binding scaffold for Cas effector. It induces the conformational change in the effector protein (e.g. Cas9) thus enabling the binding of crRNA. Some class 2 effector proteins do not require tracrRNA, for example, Cas12a [56].

When the gene editing experiment using CRISPR is being designed, the researcher usually controls only the part that is complementary to target sequence in crRNA and the rest is standard and depends on the Cas protein of choice. Structure of the target site is shown in Figure 2-4 for two specific proteins of interest: Cas9 from *S. pyogenes* and Cas12a from *Acidaminococcus* sp.

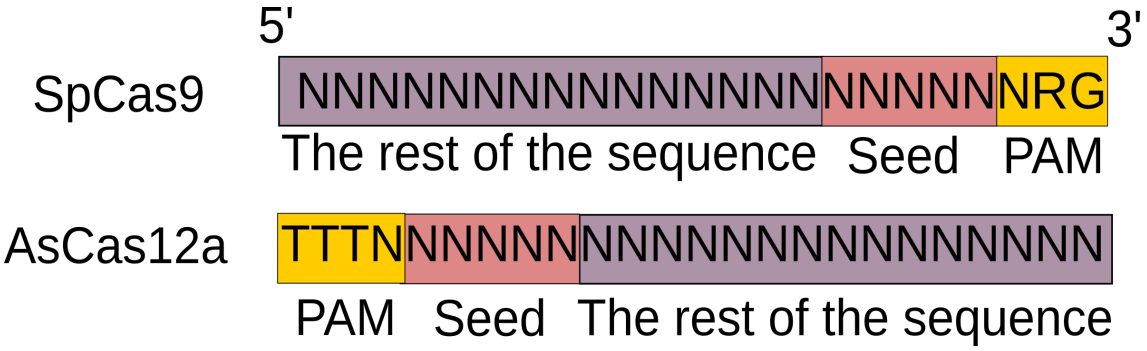

Figure 2-4: Structure of target sites for SpCas9 and AsCas12a.

Protospacer-Adjacent Motif (PAM) is the binding site for an effector complex [63]. Each effector uses a unique PAM. We must note that the PAM is not a sequence, but a set of sequences and different sequences from that set have different binding probabilities [64]. Usually PAMs contain several fixed letters and some variable ones, for example NGG - the Gs refer to guanines and N refers to any letter. PAM may be located on different sides of the target - upstream or downstream. PAM for SpCas9 is 5'-NGG-3' with the target located on 5' end. PAM for AsCas12a is 5'-TTTN-3' with the target located on 3' end.

Seed region is the region of the sequence located adjacently to PAM, usually 5 base pairs long [54]. The mismatches in the seed region can drastically reduce the efficiency of binding and subsequent cleavage. Importance of seed region is known from the mechanics of progressive hybridization of crRNA fragment complementary to the target DNA which linearly progresses to the end of the target through the seed sequence [65]. If there is a mismatch in the seed, the local unwinding of the target double strand DNA does not occur [66]. Mismatches in the rest of the sequence are better tolerated than in the seed region, but still only the perfect match offers the best results for targeting.

Cas9 [67] and Cas12a [68] proteins both consist of six domains:

1. **REC Lobe** - Recognition lobe, consists of two subdomains, REC I and REC II, that bind the DNA and regulate the NWH domain operation [69];
2. **Bridge Helix** - Domain that starts cleavage of the target DNA;
3. **PAM Interacting Domain** - Recognizes the PAM;
4. **NWH** - A nuclease domain of the HNH family, specific to Cas9;
5. **RuvC** - A nuclease domain of the RuvC family, present in both Cas9 and Cas12a;
6. **WED** - A wedge domain, specific for Cas12a.

The way how AsCas12a works is currently considered to be according to the following scheme (Figure 2-5).

First, AsCas12a binds with the target DNA in a random place and moves along it in 1D. When it encounters PAM, it unwinds the DNA locally and pairs target DNA near the PAM and gRNA forming an R-loop (a structure of two DNA strands and one RNA molecule which is hybridized with one of the DNA strands). If the R-loop is formed partially (length is less than 17 bp), the protein comes off the DNA, otherwise it cleaves the DNA starting from the non-target strand. The presence of PAM here is not needed, the DNA cleavage is PAM-independent, PAM only initiates

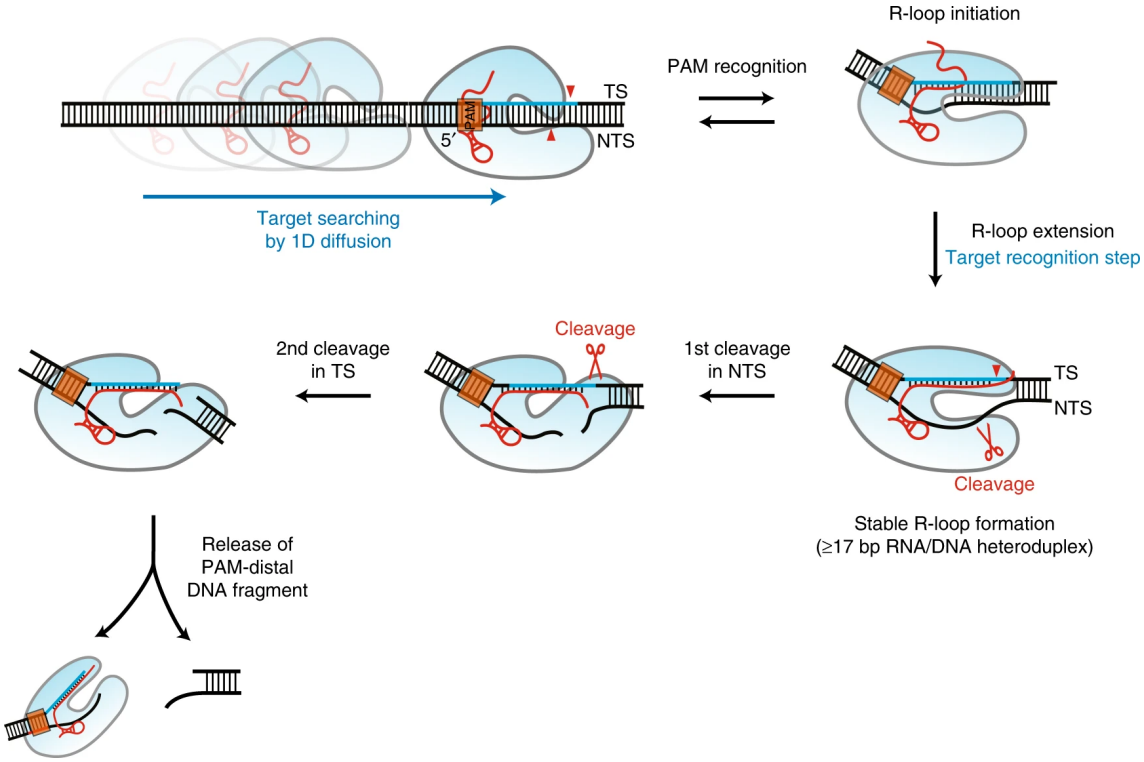

Figure 2-5: Model for AsCas12a mechanism of action (image source [3]).

the formation of the R-loop. After the cleavage, AsCas12a drops the fragment without PAM and protospacers and stays on the other fragment.

While the general mechanism of Cas9 operation is similar to the one described above, the difference between Cas9 and Cas12a lie in what kind of double-strand break they generate – Cas9 performs the break with blunt ends, while Cas12a makes the break with sticky end. Sticky ends promote non-homologous end joining which makes Cas12a more advantageous than Cas9 when the task is a knock-in experiment (insertion of DNA at a specific site) [68]. Also Cas12a requires only crRNA without tracrRNA [56].

To summarize, target recognition in CRISPR/Cas system is driven by formation of R-loop, strand separation, PAM recognition and interaction with the target DNA that is located near the PAM. This process evolved to differentiate between self and non-self DNA to prevent autoimmune action while keeping DNA targeting as effective as possible. Given there is enough Cas-protein in the cell and the target region chromatin is in the accessible state, the main factors of a successful cleavage are PAM recognition and target-gRNA matching.

2.4 Evolution of bioinformatical tools for gRNA activity evaluation

A useful lense to look at the spectrum of Computational Biology tools for gRNA selection, is to view it as an evolutionary process that goes from simple rule-based system towards complex artificial intelligences that provide information beyond cleavage probability.

One can distinguish the following five waves that form this evolution:

Rule-based systems. Within the rule-based framework, cleavage efficiency of gRNA is inferred using a single rule or a set of rules that are relatively easy to compute and self-explanatory. For example, the first rule-based systems, introduced by Gagnon [70] and Wang [71], consider the GC-content of the gRNA as a primary factor influencing cleavage efficiency. Gagnon et al. [70] reported positive correlation of GC-content and indel frequency in zebrafish and Wang et al. [71] reported similar results in human cells. According to them, an efficient gRNA for SpCas9 has guanine near the PAM region and GC-content larger than 50%. Therefore, the algorithm that Wang and Gagnon's groups offer, may be summarized as follows: if GC-content is higher than 0.5 and guanine is present in seed region and/or PAM, then the gRNA is active. This rule provides a natural baseline for gRNA classification by efficiency, but it is not enough for experimental design as GC-content does not correlate well with efficiency – there is a significant positive correlation but also a lot of noise (for example, Figure 2C from [72]). Application of machine learning helps address the problem of accuracy by building constructs that correlate with cleavage efficiency better and have narrow confidence intervals.

Classical Machine Learning systems. In this category, we include tools that employ shallow models (e.g. Logistic Regression, Decision Trees, Support Vector Machines etc) to construct complicated rules, not readable by humans, but still rather easy to interpret. Historically, it started with the usage of Support Vector Machines by Wang et al. [71] and some other classical ML solutions used SVMs

(e.g. geCRISPR [73], sgRNA Scorer 2.0 [74]) as well as gradient boosting [75] and random forests [76].

Black-box Deep Learning systems. After the classical machine learning systems were introduced, the next logical step is building the tools that employ various kinds of artificial neural networks. Such tools gain an increase in accuracy, but forfeit all semblance of explainability (hence, black-box). DeepCRISPR [77] attempted to overcome the problem of black boxness with saliency maps, but they did not use it in their gRNA design tool. DeepHF [78] used Shapley additive explanations [79] for their XGBoost model but not for their CNN and RNN ones (see Figure 2-6) for the same reason, and still did not provide explanation for every single gRNA.

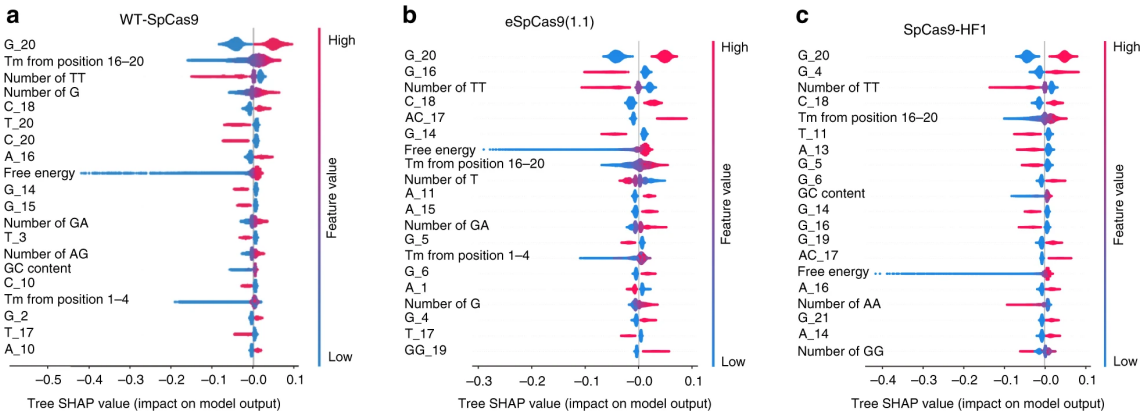

Figure 2-6: Analysis of feature importance associated with gRNA activity by Tree SHAP. a–c Top 20% important features identified by Tree SHAP for WT-SpCas9, eSpCas9(1.1), and SpCas9-HF1, respectively. The nucleotides, as well as their position, were shown on the left. $GG_1 9$ means GG dimer start at position 19. T_m means melting temperature (image source [78]).

Currently, the bulk of the research is headed in this direction (e.g. usage of transformers [80]), which raises a number of concerns related to the probability that we have already reached the limit of model improvement and the reasonable thing to do is to work on gathering better data and on adding new features to the experimental design process.

White-box Deep Learning systems. White-box systems also use deep models, but, unlike the black-box ones, are equipped with a subsystem that highlight

the important parts of the input. Those systems allow for an intelligent choice of gRNA. Unfortunately, currently there is no open public solutions that employ white-box system, except for the result of the current research. SeqCRISPR [81] claimed to do that but there is still no peer reviewed publication, no openly accessible software and since the method is based on word2vec-like [82] embeddings that offer limited interpretability and explainability, it is questionable whether this approach even works.

Uncertainty-aware and Explainable Deep Learning systems. Uncertainty-aware and explainable systems are tools that use deep models, have an explanation routine and also are augmented by a system for estimation of its own prediction variance. Currently the only example is presented in the current dissertation.

The current state of affairs is that there is no toolkit that performs all necessary tasks for gene editing experimental design, but instead the design pipeline is constructed as a combination of several parts, and such combination requires orchestrating by a human. Here is the general outline of the process with examples of popular tools and methods listed within each step:

1. **Selection of the target** – identification of the specific gene or genomic region for editing. People use NCBI Gene [83], Entrez Gene [84], Ensembl [85] and other similar tools;

2. **(optional) Variant calling for the patient or subject** – identification of single nucleotide polymorphisms or larger variations in the gene of interest that the patient has. Software like DeepVariant [86] may be applied but detailed discussion of variant identification is out of scope of this dissertation;

3. **gRNA selection** – Search for the gRNAs that minimize number of off-targets and maximize on-target cleavage probability. One can use the tools discussed in the current section;

4. **Validation and optimization** – Verification of the gRNAs performance *in vitro* using Tracking of Indels by DEcomposition (TIDE) [87], GUIDE-Seq [88]

and other similar methods;

5. **Selection of delivery method** – Choosing how to deliver the components into target cells, for example by lipofection [89], viral vectors [90, 91, 92] or nanoparticles [93, 94, 95];

6. **Experimental execution** – Actual execution of a gene editing experiment with selected gRNA and delivery method. One can use kits that many companies like Thermo Fisher Scientific, Synthego and ODIN Inc. offer;

7. **Analysis and validation of the result** – Testing whether the experiment worked and how well with DNA sequencing (Sanger, Next Generation Sequencing) and PCR-based methods.

The combination of these tools and methods allows researchers to effectively design and execute their experiments. The human element is essential for orchestrating the process and making informed decisions at each step. From the experimental standpoint, it is hard to distinguish between binding and cleavage efficiency for the experimental data do not offer the distinction between the case where the binding happened, but cleavage has not, and the case where binding did not occur. Therefore in this dissertation we use the term "cleavage efficiency" even in the cases where the cleavage is not supposed to happen (dCas9 gene expression modulation experiments). Most deep learning approaches use one-hot encoding of the sequence [14, 77, 78] while classical machine learning approaches may also rely on feature design (e.g. geCRISPR [73] uses thermodynamic properties like melting temperature of RNA, and sgRNA-DNA binding energy, and also structural features like Shannon entropy). However, some deep learning approaches also use sequence embeddings (e.g. [96]) and may incorporate additional non-sequential information like chromatin accessibility (e.g. chromatin-based model of [14]). A thorough study of input features can be found in a review work of [97], in the current work we have used only one-hot encoding for sequence and only sequential features due to the initial focus on studying the sequence feature influence on cleavage efficiency.

In conclusion, experimental verification of CRISPR cleavage efficiency is an important step in the development and application of CRISPR-based technologies. By

combining computational and experimental methods, researchers can develop accurate and reliable predictions of cleavage efficiency, supporting the development of effective and precise genome editing tools.

2.5 In silico detection of a potential off-target event

Regarding the software for off-target analysis, there are two major directions. First one is to improve fast search for slightly mismatched strings (usually no more than four [98], five [99] or six [100] mismatches, and typically the input is given in a form of a string mask [101, 102]) in a large genome. While the brute force string search here is definitely a possibility (although such search is not recommended for genomes of reasonable size), most innovations include either usage of hardware acceleration (e.g. Cas-OffFinder [101]), performant implementations of fast string search with precomputed indices (e.g. FlashFry [102]), or novel string search algorithms (e.g. Crackling [103]).

Machine Learning-based evaluation of off-targets acts as a kind of a filter that sifts through potential off-target sites found by other software in pursuit of a high probability case. This dissertation offers two filters for the output of string search algorithms – inequality-based Anomaly Detection and Uncertainty-Aware cleavage efficiency prediction for pair of strings. The former essentially solves a binary classification problem (0 – no off-target, 1 – off-target) based on the class-wise inequality in the latent space of neural network, while the latter solves a regression problem (label is a real value of cleavage probability from 0 to 1) using an uncertainty-aware modeling approach described in the next section.

Both solutions introduced in this dissertation are designed to be string search agnostic and would work with any kind of previously developed solutions. The models were tested with FlashFry [102] and Cas-OffFinder [101].

2.6 Uncertainty Quantification in Machine Learning applied to Bioinformatics

Main overarching theme of this dissertation is making use of ways to evaluate model prediction error for guidance in experimental design. Such error can emerge from a number of diverse sources. The error of prediction can be described in the following way:

$$E_{total} = E_{bias} + E_{variance} + E_{input\ noise} + E_{label\ noise},$$

where different indices of E denote different sources of error. Note that input noise and label noise may lead to both increased bias and variance, but here it seems to be useful to explicitly introduce them as separate entities.

Bias. Bias is an error that arises from the assumptions made to simplify the problem and fit it into the model family of interest (e.g. linear regression, random forest, gradient boosting,...). High bias leads to underfitting because the model family is not strong enough to capture the correlations between input data and label. For example, Rosenblatt's single layer perceptron is unable to fit the Exclusive Or (xor) problem no matter how the model is trained and will always give biased answers (mathematical proof is given in [104]). An example more relevant to the topic of this dissertation is the study of bias in translational bioinformatics [105] that appears because the chosen family of models, Support Vector Machines, is not strong enough to overcome the label imbalance. The authors had to change the model family by adding Derivative Component Analysis to their SVM pipeline and that model change did help in improvement of predictive performance.

Variance. The model's reaction to small variations in the training data gives rise to another component of error, the variance. High variance leads to overfitting because for some models and some tasks it is much easier to fit the noise in the data than to actually solve the problem. For example, the choice of biomarkers based on transcriptomics data suffers a lot from small differences in gene expression levels thus usage of a stable and robust method is of high importance [106]. What one

would like to achieve is low bias and low variance, but this simplified description does not include noise in data and label, which prevents acquiring the best results. A common depiction of the bias-variance trade-off is shown in Figure 2-7 from [4]:

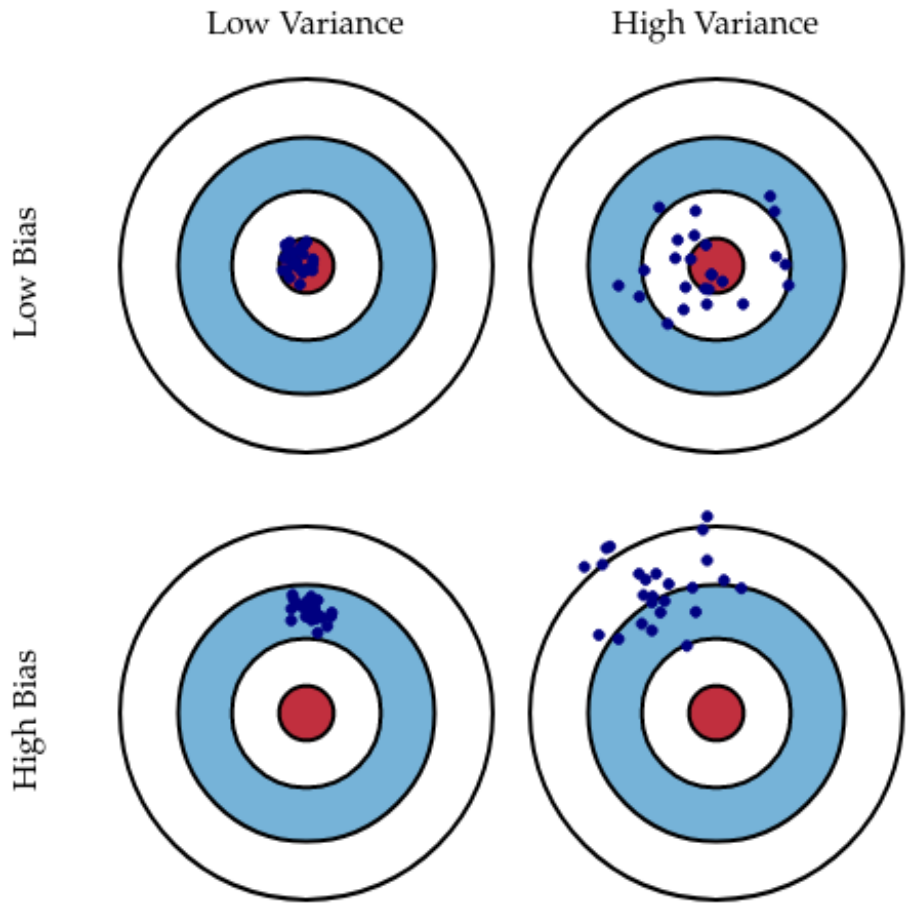

Figure 2-7: An illustration of bias-variance trade-off (image source [4]).

Input noise. An error that emerges because the data is imperfect is called "input noise". There are measurement errors, data collection and preprocessing problems and so on. This part of the noise is not reducible by improvement in the model alone, but rather through better data. A good illustration of input noise is the field of microbiomics which has a lot of systematic errors introduced during the collection of samples, sequencing, binning, genotyping and so on [107].

Label noise. An error that follows from mistakes and discrepancies in data labeling is called "label noise". The labels that were assigned to examples in the

dataset may be assigned incorrectly due to a systematic mistake during data collection, preprocessing or curation, inconsistency of label definition or a real ambiguity in examples that belong to different classes. Case in point is classification of subtypes for class C G protein-coupling receptors. GPCRs are membrane proteins involved in a lot of signaling pathways, a good target for therapeutical interventions. Mislabeling in training set can lead to the assignment of the wrong intervention. A study of label noise in subtyping of GPCRs [108] revealed that most of label noise comes out of expert-made mistakes and lack of correspondence between different conventions regarding assignment of protein family label.

It is crucial to identify those errors during the development of a Machine Learning application if the goal is to build as accurate and robust a model as possible. This is generalized in theory of Uncertainty Quantification (UQ). UQ is a methodology to evaluate the confidence in model prediction which helps to determine whether the model predictions could be trusted and to what extent. In regression, it is done by computing confidence intervals for the predicted label. This is called conformal prediction [109]. Conformal prediction is being currently applied in bioinformatics setting, albeit rarely, for example in drug design [110] for modeling of ATP-binding cassette transporters. An illustration of conformal prediction is given in Figure 2-8. We find it useful to distinguish Uncertainty Quantification in Machine Learning in

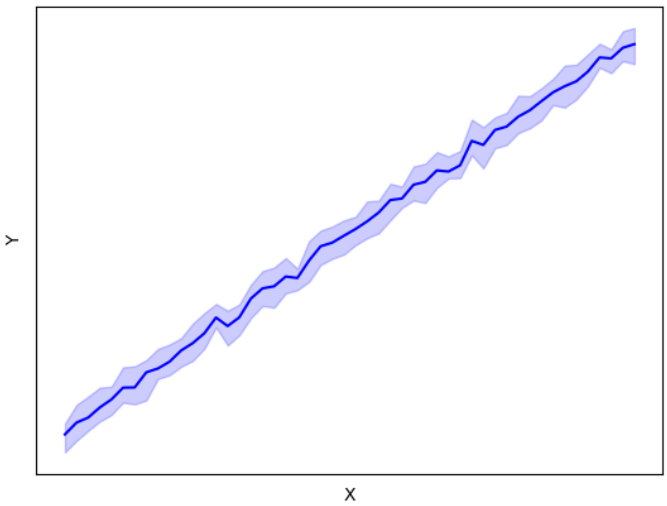

Figure 2-8: An illustration of conformal prediction. Light blue regions show the confidence intervals with $\mu \pm \sigma$ and blue line shows the mean of predictions μ.

two broad categories:

Ensemble Methods. Main idea of ensemble methods is to train several models and use the sample variance of their predictions to compute confidence intervals. For example, HatchEnsemble [111] uses the modified copies of the same seed model (called HatchNets) obtained by application of function-preserving transformations (ones that add new neurons to the network, but keep its output the same) followed by retraining and corruption of parameters with random noise. The uncertainty estimations are given by standard deviation of outputs from a number of HatchNets.

Uncertainty-aware Models. Main idea of uncertainty-aware models is to draw them from model families that are capable to grasp the uncertainty in data by design. To illustrate this let's consider a digital histopathology study [112] where uncertainty-aware models allow for accurate discrimination between lung adenocarcinoma, squamous cell carcinoma and non-cancerous samples. Another example of such models are Bayesian Machine Learning methods [113]. Basic idea of Bayesian Machine Learning is to model the weights as random variables and infer their posterior distributions from the data.

In the research that formed the foundation of the current dissertation, the second approach, Uncertainty-aware Modeling, was used. We have applied Gaussian Processes [114] with the kernel learned [115] by Capsule Networks [116] and used the sample variance from Gaussian Process as a base for confidence intervals. Confidence intervals showed the existence of different off-target classes and allowed the exploration of this diversity. Uncertainty Quantification can be used to detect anomalies and out-of-distribution examples. The inverse is also correct, a good anomaly detection model is an implicit uncertainty quantifier. In the current research, a method to detect anomalies that uses a measure that implicitly reflects the uncertainty in the dataset was developed. This measure reflects on difference between latent space representations of a frequently seen class and a rare one.

2.7 Explainability and Interpretability in Machine Learning applied to Bioinformatics

As Marcinkevics and Vogt put it [117], "interpretability and explainability have escaped universal definition". Both in various literature, such as Gilpin et al. [118], Ribeiro et al. [119] and Linardatos et al. [120], and during the conception of this dissertation's topics, the terms "Interpretable" and "Explainable" Machine Learning have been used interchangeably, nevertheless, interpretability and explainability mean two different, albeit closely related, things. In this dissertation, the following two working definitions are accepted:

- *Interpretability* – existence of a clear understanding what processes are performed by all of the model's internal parts ***regarding the domain where the model is used***;
- *Explainability* – existence of a way to show what kind of inputs lead to what kind of outputs.

In the latter chapters that describe the results of research, we use the former definition when appropriate, but keep in mind, that in the published paper, the language related to explainability and interpretability may differ.

Explainability refers to knowledge of causality between the inputs for the model and its outputs – one should not mix it with causality between the inputs and the labels, which is an entirely different field of research (a thorough review of causality in ML can be found in [121]). In explainability, we do not model the data generation process, but rather we build a simplified version of the model itself, the one that humans can understand, This also should not be mistaken for mechanistic interpretability, which attempts to reverse engineer the computations performed by a black-box model accurately [122] while within the context of explainability we are content with catching local influences of feature on prediction.

Interpretability refers to knowledge of internal processes inside the model and their link to the model's meaning in a greater scale of it's application. For example, having a completely interpretable model of protein folding means knowing that a

certain module performs a computation of secondary structure, while submodules within the secondary structure module perform identification of alpha helices and beta sheets, etc.

A model thus can be neither interpretable nor explainable, or both interpretable and explainable, or either interpretable or explainable – this is determined by model design. Unfortunately, complete interpretability in case of Bioinformatics seems to be currently unabtainable. In various cases of Computer Vision, one can show that, for example, lower layers of a neural network capture simple shapes [123, 124, 125]. The verification of that fact is quite easy, for human visual cortex essentially evolved to solve Computer Vision tasks – one can plot the weights and see for oneself whether the simple shape hypothesis is correct. But humans did not evolve to solve problems of Bioinformatics and visual cortex cannot help with it. Therefore in our research, we opt for interpretability in a rather more narrow sense, let's call it "instrumental interpretability" – it is defined as the former but without the domain part. We can't really know what each part of a model does regarding to the meaning of the task, but at least we can design models with predictable behavior regarding the operations that convert input to output. Following the aforementioned protein folding example, instrumentally interpretable method would have the module intended to perform a known operation on its input, for example mapping the input data to a lower dimensional space. From that standpoint, the models designed for this dissertation have the property of instrumental interpretability – for each module we do know what it does and based on this knowledge we can anticipate the results.

With explainability, the situation is more clear. There are a number of ways to highlight the part of input relevant to the prediction of a label. Some methods are dependent on knowledge of the model's internal structure (e.g. [126, 127, 128] thus blending two former definitions in one, and some are model-agnostic (e.g. [79, 129, 130, 131]). In this dissertation, we use a model-agnostic method, Accumulation of Local Effects [132], which is explained in details in the "Methodology" chapter.

Chapter 3

Thesis objectives

The main goal of this dissertation is to build explainable models for estimation of on-target and off-target CRISPR-Cas cleavage efficiency, identify the sequence determinants of cleavage efficiency and explore the diversity of potential cleavage events. The aims of this dissertation are subdivided into the following groups:

- Construction of classification model for Cas9 off-target events;
- Construction of uncertainty-aware machine learning model for regression of cleavage efficiency for Cas9 and Cas12a proteins;
- Exploration of sequence determinants of cleavage efficiency for several Cas9 orthologs and AsCas12a;
- Characterization of diversity for off-target events in Cas9 and Cas12a cleavage in *Homo sapiens*.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Materials and Methods

This chapter is largely based on papers that constitute the current dissertation – "Measuring internal inequality in capsule networks for supervised anomaly detection" [2] published in Scientific Reports and "Uncertainty-aware and interpretable evaluation of Cas9–gRNA and Cas12a–gRNA specificity for fully matched and partially mismatched targets with Deep Kernel Learning" [1] published in Nucleic Acids Research. The contributions for the first paper are as follows – I (together with my supervisor) have developed the main concept, designed the study and wrote the main manuscript text. I also have performed all computational experiments. The contribution of authors for the second paper are as follows: E.S., S.A.S. and I developed the concepts and designed the study. E.V.K., M.P., A.Y.O., S.A.S. and K.V.S. supervised the research. I have designed the methods and performed all analyses. E.S., S.A.S. and I wrote the manuscript, which was read, edited and approved by all authors.

4.1 Problem setups

CRISPR-Cas off-target event detection. We use the models to estimate the probability of an off-target event from a pair of sequences. One sequence in the pair is sgRNA, and the other one is the target DNA. Each sequence is a string over an alphabet of 4 symbols (A, T, G and C), 20-23 symbols long. sgRNA and target DNA have six or less mismatches, at least one mismatch. The problem is the task

of binary classification, with labels of 0 – no cleavage and 1 – off-target cleavage. In the current dissertation, the problem of off-target event detection is solved by the anomaly detection pipeline described in sections 4.2.3, 4.4 and 4.5 of "Materials and Methods" and the results are presented in chapter 5. The solutions are based on data that are described in "Description of the dataset and preprocessing routine" (subsection 4.2.2) of "Materials and Methods".

CRISPR-Cas cleavage efficiency prediction. We use the GuideHOM family of models to predict cleavage efficiency from sequences. Cleavage efficiency is a real number usually ranging from 0 to 1. It can be slightly greater than 1 or smaller than 0 (depends on the dataset), but for the purpose of prediction, we scale and shift the labels so the maximum is 1 and minimum is 0. The sequences are strings over an alphabet of 4 symbols (A, T, G and C), 20-23 symbols long, as in previous problem setup. The problem is the task of regression which we solve using a neural network-enhanced Gaussian process. We provide the solution that, unlike others, not only gives a point estimate of the cleavage efficiency, but, for the first time, also gives the confidence interval for it. In the current dissertation, the problem of cleavage efficiency prediction is solved by the GuideHOM pipeline described in sections 4.6, 4.7, 4.8, and 4.9 of "Materials and Methods" and the results are presented in chapters 6 and 7. The solutions are based on data that are described in sections "Description of the dataset and preprocessing routine" (subsection 4.2.1) of "Materials and Methods" chapter.

4.2 Description of the datasets and preprocessing routines

4.2.1 Data collection and preprocessing for CRISPR-Cas cleavage efficiency prediction

For on-target Cas9 cleavage efficiency prediction, we used three non-overlapping datasets, geCRISPR, DeepCRISPR and DeepHF. The geCRISPR dataset [73] con-

sists of 4569 experimentally verified gRNAs for Cas9 derived from Homo sapiens, Danio rerio, Mus musculus and Xenopus tropicalis. DeepCRISPR dataset [77] consists of 16492 experimentally verified gRNAs for Cas9 derived from four human cell lines (hela, hl60, hct116, hek239t). DeepHF [78] provides the data for SpCas9 (55604 sequences) and two high fidelity orthologs: SpCas9HF1 (56888 sequences) and eSpCas9 (58617 sequences). For prediction of Cas9 efficiency with mismatched gRNA and target sequences, we used the dataset of Jost et al [5] with 26248 gRNA-target pairs. For on-target Cas12a cleavage efficiency prediction, we used the DeepCpf1 [14] dataset that consist of 20506 experimentally verified gRNAs for Cas12a. For Cas12a off-target prediction, we used the dataset of [133] with 1565 gRNA-target pairs. Before training the neural networks, we used one-hot encoding on every sequence in the on-target datasets, as shown in Figure 4-2A. For the off-target datasets (Cas12a off-target dataset [133], Jost et al [5]), the encoding algorithm was used twice, that is, once on the guide, and the second time, on the target, so that the output is a two-channel image $(2, 4, N)$ where N is the length of the input sequences. For the on-target datasets, the algorithm was applied only once, so that the output is a $(4, N)$ image.

4.2.2 Data collection and preprocessing for CRISPR-Cas9 off-target event detection via anomaly detection

Off-target cleavage in CRISPR/Cas9-based gene editing can lead to various unforeseen consequences. For a design of gene editing experiment it is of high importance to select such gRNAs that minimize the probability of Cas9 performing doublestrand cleavage in a wrong place (off-target effect). To do so using Machine Learning, a dataset of gRNA-target pairs is used. The dataset of CRISPR-Cas9 off-targets taken from the work of Peng et al. [100] consists of 215 low-throughput off-target pairs, 527 high-throughput off-target pairs and a negative subset of 408260 pairs. The low and high throughput pairs are labeled 1 and the negative subset thus 0. Each pair is two strings of "A", "T", "G" and "C" symbols. We convert pairs into images using the following preprocessing routine:

1. One-hot encode the target string (which is 23 nucleotides long) to get I_1;
2. One-hot encode the gRNA string (which is also 23 nucleotides long) to get I_1;
3. Join the encodings to get a tensor of size $(2, 4, 23)$.

We do not use the pairs that have more than 6 mismatches (since the off-target cleavage is considered to be impossible in that case) so the final dataset consists of 615 anomalous pairs (off-targets) and 26038 normal pairs from negative subset.

4.2.3 Benchmarks for Anomaly Detection unrelated to off-target detection

MNIST-like Benchmarks

The studies [134, 135] we base our work on (further described in the next section) are conceptually similar but they offer different ways to measure the performance of the anomaly detection metrics. We first considered an experiment inspired by the work of Piciarelli et al. [134] which is organized following the **model generation procedure for Diverse Outlier setup**:

1. Extract all examples of class i from the training set with N classes, and assign the label $l = 0$;
2. Randomly extract A examples of any other class and assign the label $l = 1$;
3. Train a model to classify the data into two classes.

We apply this procedure to all classes in the datasets and to the fractions of 10%, 1% and 0.1% so we get $4 \times 3 \times 10 = 120$ models to test our results on. This procedure gives us the coherent normal subset and a diverse subset of outliers. Approach based on the work of Li et al. [135] is the reverse one in nature – we use a single class for our anomaly label and all other classes we consider normal, so our normal set is diverse and anomalous set is coherent. We train the models following a **model generation procedure for Diverse Inlier setup**. This gives us another 120 models to test:

1. Extract all examples of class i from the training set with N classes and assign the label $l = 1$;

2. Randomly extract A examples of all other classes and assign the label $l = 0$;

3. Train a model to classify the data into two classes.

We use MNIST [136], FashionMNIST [137], KuzushijiMNIST [138] and CIFAR10 [139] with the diverse outlier and diverse inlier setups to make a comparison of the proposed methods, the previous studies [135, 134] and the baselines. Each dataset except CIFAR10 has 60000 single-channel images of (28,28) size that are separated into 10 classes. CIFAR10 has 60000 images with 3 channels and (32,32) size, also separated into 10 classes.

Malignant Skin Lesion Classification

The HAM10000 [140] dataset contains high-quality photos of 7 skin lesion types three of whom are malignant. The dataset contents are shown on Table 4.1: Inspired

Table 4.1: Detailed information on classes for HAM10000 dataset.

Skin lesion type	# **Images**	**Is malignant?**
Melanoma	1113	Yes
Basal cell carcinoma	514	Yes
Dermatofibroma	115	No
Melanocytic nevus	6705	No
Vascular lesion	142	No
Benign keratosis-like	1099	No
Intraepithelial carcinoma	327	Yes

by the anomaly-based cancer detection pipeline [141], we consider malignant skin lesions anomalies, aberrations of the correct skin cell life-cycle and while benign skin lesions are also a kind of an aberration, we consider them a base for the normal classes in our experiments. From this dataset we derive four experiments:

1. Diverse normal set, diverse anomalous set: all malignant types as an anomaly set, all benign types as a normal set;

2. Diverse normal set, homogeneous anomalous set: melanoma (the most common skin cancer) images as an anomaly set, all benign types as a normal set;

3. Homogeneous normal set, diverse anomalous set: all malignant types as an anomaly set, melanocytic nevus (the most common benign lesion, a birthmark) images as a normal set;

4. Homogeneous normal set, homogeneous anomalous set: melanoma vs melanocytic nevus.

4.2.4 Description of genome analysis pipeline

Input: the raw sequence data, downloaded from the UCSC genome browser and respective annotation. Output: the list of possible gRNAs with computed average cleavage efficiency, standard deviation. The input is downloaded with the following options: Chromosome - Download FASTA, Visible range (while whole chromosome is opened in the browser); Annotation - Download csv. The pipeline proceeds as follows:

1. the genes are extracted from the chromosome sequence according to the annotation using a BioPython-based script;

2. Cas-Offinder [101] is used for each separate gene.fasta to produce a list of potential targets. The mask is NNNNNNNNNNNNNNNNNNNNRG for Cas9, TTTNNNNNNNNNNNNNNNNNNNN for Cas12a. The result is saved in .tsv files, one for each gene;

3. for each Cas-Offinder output, the model is used. For each target sequence, the model computes the average cleavage efficiency and standard deviation (or just the cleavage efficiency in case of [5]). If the model requires two sequences, the input target is duplicated to form $(4, N, 2)$ vector, so the input sequence acts as both gRNA and target;

4. the guides are sorted by cleavage efficiency in descending order. Each result is saved in a table that combines CasOffinder output and model output.

For this analysis we use the best performing Cas9 and Cas12a on-target models. The results are available at a dedicated Zenodo repository, consult the "Code and data availability" of [1] section for details.

4.3 Description of hardware and software used in the studies

For all experiments shown in this dissertation, we have used laptop Alienware 17 R5 with Intel Core i7-8750H, 8 Gb VRAM NVIDIA GeForce GTX 1070 Mobile, 24 Gb GDDR5 RAM. All experiments were performed under Linux environment, Ubuntu 21.10 with kernel 5.13.0-52. The following Table 4.2 includes the versions of all libraries that were used for this dissertation.

Table 4.2: Detailed information on libraries.

Library	**Version**	**Description**	Citation
Pytorch	1.11.0	Deep learning framework	[142]
Torchvision	0.12.1	Computer Vision primitives for Pytorch	[143]
Pytorch Lightning	1.8.6	High-level wrapper for Pytorch	[144]
Einops	0.4.1	Tensor manipulation library	[145]
Alibi	0.9.3	Algorithms for Explainable ML	[146]
GPytorch	1.4.0	Gaussian Processes implementation	[147]
Biopython	1.74.0	Bioinformatical algorithms in Python	[148]
Matplotlib	3.6.3	Scientific visualization	[149]
Numpy	1.21.6	Numerical computations and linear algebra	[150]
Scipy	1.10.1	Statistical computations in Python	[151]
Scikit-Learn	1.2.2	Classical machine learning in Python	[152]
Weblogo	3.6.0	Visualization of logo sequences	[153]

4.4 Supervised anomaly detection baselines

Within the supervised framework, a model for anomaly detection is trained to discriminate normal examples from the anomalous ones. It has a certain advantage in the discrimination ability over the model within the unsupervised framework and in cases where the anomalies are rather homogeneous, it usually performs the best. One of the basic methods in supervised anomaly detection using deep learning is

Negative Learning (NL [154]) – to distinguish between the outliers and the normal data points, one uses the reconstruction error of autoencoder that is trained to reconstruct normal data points perfectly while failing to reconstruct the outliers. Negative learning-based anomaly detection does suffer from inability to reconstruct normal data points though. To overcome this issue, the work of Yamanaka et al. [155] introduces the Autoencoding Binary Classifiers (ABC) which extend the negative learning approach by providing lower and upper boundaries on the loss function with respect to the reconstruction errors. NL and ABC will be used in our work as the baselines. The main idea of negative learning [154] is to permanently damage the reconstructive ability of an autoencoder by forcing it to maximize the reconstruction error on anomalous samples while minimizing it on normal ones. As an autoencoding model, the work of Munawar et al. [154] uses Restricted Boltzmann Machine with a visible layer and a hidden layer. The network is trained using single-step contrastive divergence [156]:

$$\delta w = \sigma[[(vh)_{original} - (vh)_{reconstructed}]],$$

where v is the visible layer, h is the hidden layer, σ is the sign and δw is the gradient of weights. For positive learning stage, $\sigma = 1$, for negative - $\sigma = -1$. During one training pass, the positive learning is done first, on all positive examples, then the negative learning is done on all negative examples. Autoencoding Binary Classifier uses the following loss, constrained in case of anomalous input:

$$L_{ABC}(X, Y) = YMSE(X, \hat{X}) - (1 - Y)\log\Big(1 - e^{-MSE(X,\hat{X})}\Big).$$

The logarithm term caps the loss for the $Y \geq 1$. Additionally, the ABC paper [155] uses multilayer perceptrons instead of RBMs for architecture and gradient descent instead of CD for training algorithm. Capsule networks were already used for anomaly detection in the works of Piciarelli et al. [134] and Li et al [135]. The first paper considers supervised anomaly detection setup while the second one proceeds with an unsupervised formulation. They propose anomaly and normality metrics respectively, that are based on usage of regularizing decoder and the difference between the estimated probabilities of normal and anomalous class. In our paper we

show that those metrics are a direct consequence of internal inequality of "assets" in Capsule Network representations. The probabilities and reconstruction errors are on the end of the pipeline and a lot of information that help distinguish the anomalies from normal data is lost during the process of their computation. We compare our work with both previous studies [135, 134] and a few baselines (NL and ABC). Our working hypothesis implies that the information ignored when the probabilities are computed helps detect anomalies better. We also compute all available anomaly metrics and show that our work provides the best results in most cases. The work of Piciarelli et al. [134] proposes the following *anomaly measure*:

$$A(X, \hat{X}, \hat{Y}_{normal}, \hat{Y}_{anomaly}) = \hat{Y}_{normal} - \hat{Y}_{anomaly} + MSE(X, \hat{X}), \tag{4.1}$$

where X is the input image, $\hat{X}$ is its reconstruction, $\hat{Y}_n, n \in \{normal, anomaly\}$ – the norm of capsule output vectors. It is based on the observation that for an anomaly the difference between probabilities for each class tends to be less drastic than for a normal example. Additionally, this paper [134] proposes filtering the anomaly class from the input to the reconstruction so the reconstruction network is trained to reconstruct only normal images. We include this feature to every experiment with Capsule Network. The work of Li et al. [135] provides two metrics. The first one, given by Eq (4.2), is the largest probability of a class:

$$N_{PP}(\hat{Y}) = \max_{i=1,\ldots,N_S} \hat{Y}_i, \tag{4.2}$$

where $\hat{Y}$ is the vector of norms of the capsule output vectors and N_S is the number of output capsules. Eq (4.2) is introduced under the assumption that for a normal example there would be only one capsule with the norm close to 1 and for an outlier both of capsules would be close to each other. Hence, for normal images the results of Eq (4.2) would be close to 1, but for an outlier they would be close to 0.5. The authors of the previous work [135] normalize MSE by the euclidean norm of inputs in Eq (4.3), because the MSE is dependent on the number of non-background pixel in input and reconstruction and the authors tried to invent a metric that is unaffected

by this issue.

$$N_{RE}(X, \hat{X}) = \frac{MSE(X, \hat{X})}{\sqrt{\sum_{i=1}^{n} X_i^2}}, \tag{4.3}$$

where X is an input image and $\hat{X}$ is the reconstruction computed by the reconstruction subnetwork. We compare the proposed methods with a selected set of previous works [154, 155] because those works provide clear, simple and accurate approaches that are similar enough to ours so the design of a comparison study is pretty straight-forward.

4.5 Capsule Networks as an ensemble of weak learners

The main difference of Capsule Networks from conventional architectures is the output. The output is not probabilities of classes, but rather vectors with information on learned features. This is achieved through combining the output of different small models and summarizing their decisions. There are two types of capsules, primary and secondary, so Capsule Networks are inherently two-layered. Class capsules perform prediction based on information from all primary capsules combined. The output of primary capsule acts as an asset for a class capsule N in a way that it enlarges or shrinks its output vector, giving a large or small probability of an example belonging to class N. Different primary capsules are of different value for class capsule N and the vectors of class capsules act as a kind of summary statistics over the distribution of primary capsule contributions. We hypothesize that the distribution of the "values" in case of a normal example will be different than in case of an anomalous one, so we can detect anomalies based on that difference. The architecture of Capsule Networks proposed by Sabour et al. [157] is shown in Figure 4-1. In case of image classification, capsule networks process the input image in three steps: preprocessing, routing and reconstruction. Preprocessing in our setting consists of a 2D convolutional layer followed by an activation function (ReLU or ELU in our experiments). It is followed by a number of 2D convolutional layers of N_f filters, which produces the output of size (N_f, output width w and output height h). The

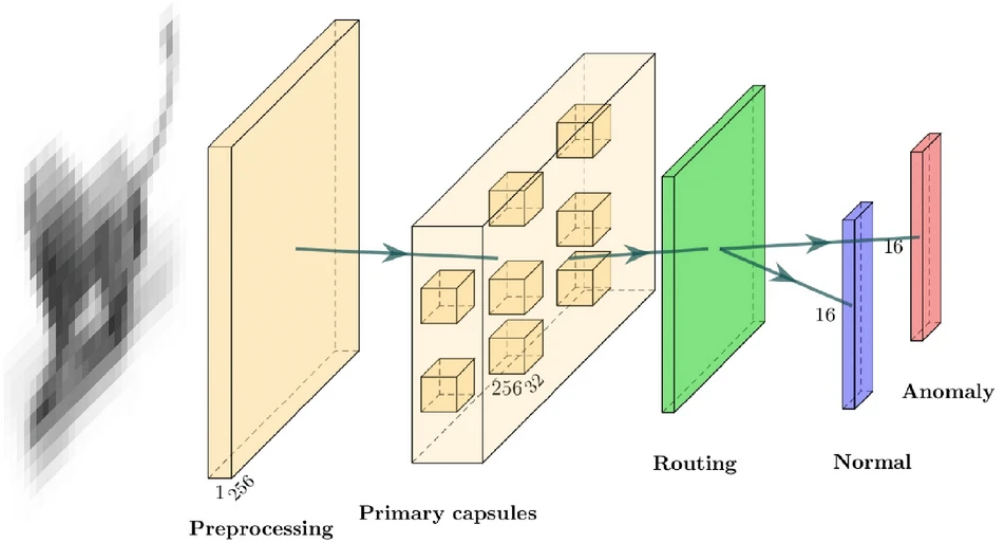

Figure 4-1: Capsule Networks architecture. Primary capsules are formed out of a convolutional layer set. Those layers have the same input and their output is combined for routing. The decoder is not shown. Architecture for MNIST-like experiments is shown here and the changes we made for different cases are described in the respective sections.

outputs of every convolutional layer are joined together and flattened out to form a tensor of size (number of primary capsules $N_P = N_f \times w \times h$, dimension of primary capsule output O_P). Then the squash activation function is applied element-wise separately for each element in batch and for each 2D convolutional layer:

$$S(x) = \frac{\|x\|^2}{1 + \|x\|^2} \frac{x}{\|x\|}, \tag{4.4}$$

where x is the input to the squash activation, e.g. output of the preprocessing stage. The outputs of primary capsules are then fed into N_S secondary or class capsules (in our setting $N_S = 2$) and we enter the stage of routing. The routing is an iterative algorithm and the following steps describe a single iteration. Usually the routing is repeated for 3 iterations. During the routing we first compute the coupling coefficients – tensor c of shape (N_P, N_S, O_S). For each secondary capsule j we compute the prior probability of coupling for all primary capsules and save it into the coupling coefficient table. We will use the following key constructions below:

1. c_{ij} – a slice of tensor c that corresponds to primary capsule i and secondary

capsule j;

2. W_{ij} – the corresponding slice of the weight tensor;

3. u_i – an output of primary capsule i.

Then for each secondary capsule j we compute *unsquashed* secondary capsule output using the product of the weight tensor slice that corresponds to the primary capsule i and secondary capsule j and the outputs of the primary capsules u_i:

$$s_j = \sum_{i=1}^{N_P} c_{ij} \odot (W_{ij} u_i),$$

where $\odot$ denotes elementwise multiplication. The outputs of secondary capsules are computed as in case of primary ones, by the squashing function (see Eq (4.4)) for every secondary capsule j:

$$v_j = S(s_j), \quad j = 1, \ldots, N_S.$$

The "agreement" is computed by scalar multiplication of outputs from secondary capsule j with the product of weights and primary capsule outputs that correspond to primary capsule i and the secondary capsule, and at the end of the routing iteration the routing table is updated. To get the probabilities of classes for an example, we should compute the Euclidean norm of the vectors we get from secondary capsules:

$$P(Y = C_j) = \frac{\exp(\|v_j\|)}{\sum_{k=1}^{N_S} \exp(\|v_k\|)},$$

where C_j is the class label that corresponds to j-th class capsule. We can train the network with any kind of classification objective but following Sabour et al. [157] we use margin loss:

$$L_k = T_k ReLU(m^+ - \hat{Y_k})^2 + \lambda(1 - T_k) ReLU(\hat{Y_k} - m^-)^2,$$

where k is the number of output capsule, $T_k = 1$ if the k-th capsule denotes the class that corresponds to the real label and 0 otherwise, m^+, m^- and λ are hyperparameters, $\hat{Y_k}$ is the norm of the k-th capsule output vector. As a regularization,

like in the original paper [157], the additional reconstruction subnetwork R is used:

$$\hat{X} = R(v_j).$$

So the total loss we optimize is:

$$L_{total} = \sum_{k=1}^{N} L_k + \alpha MSE(X, \hat{X}),$$

where $\alpha > 0$ is a hyperparameter chosen to balance the involvement of reconstruction loss. Most of the learning happens when the weights of the class capsule layer are adjusted via gradient descent during the backward pass. The adjustment is based on the output of the layer during the forward pass. The forward pass is computed iteratively, according to the equations above. The usual way to train capsule networks is not easy: one needs to use a non-standard loss and an additional decoder for regularization, but the most interesting computations happen within the coupling coefficients.

4.6 Deep Kernel Learning for cleavage efficiency estimation

We developed a deep kernel learning model, named GuideHOM (Guide Hit-Or-Miss). GuideHOM employs a Hit-Or-Miss capsule network as an intermediate feature extractor built upon a preprocessing module and a Gaussian Process to estimate the distribution of the cleavage efficiency of RGN programmed with a given gRNA. The estimation is based on the representation produced by the Hit-Or-Miss capsules to predict on-target and off-target cleavage efficiency using gRNA spacer sequence features (see Figure 4-2A for example of a single input encoding). In GuideHOM, we implemented two preprocessing approaches that differed in terms of the prediction strategy and the analyzed datasets. The first approach predicts the on-target cleavage efficiency based on the gRNA spacer sequence only, under the assumption that the target sequence is its exact complement. The second approach aims to estimate the cleavage efficiency based on the mismatching gRNA spacer-target pairs. We

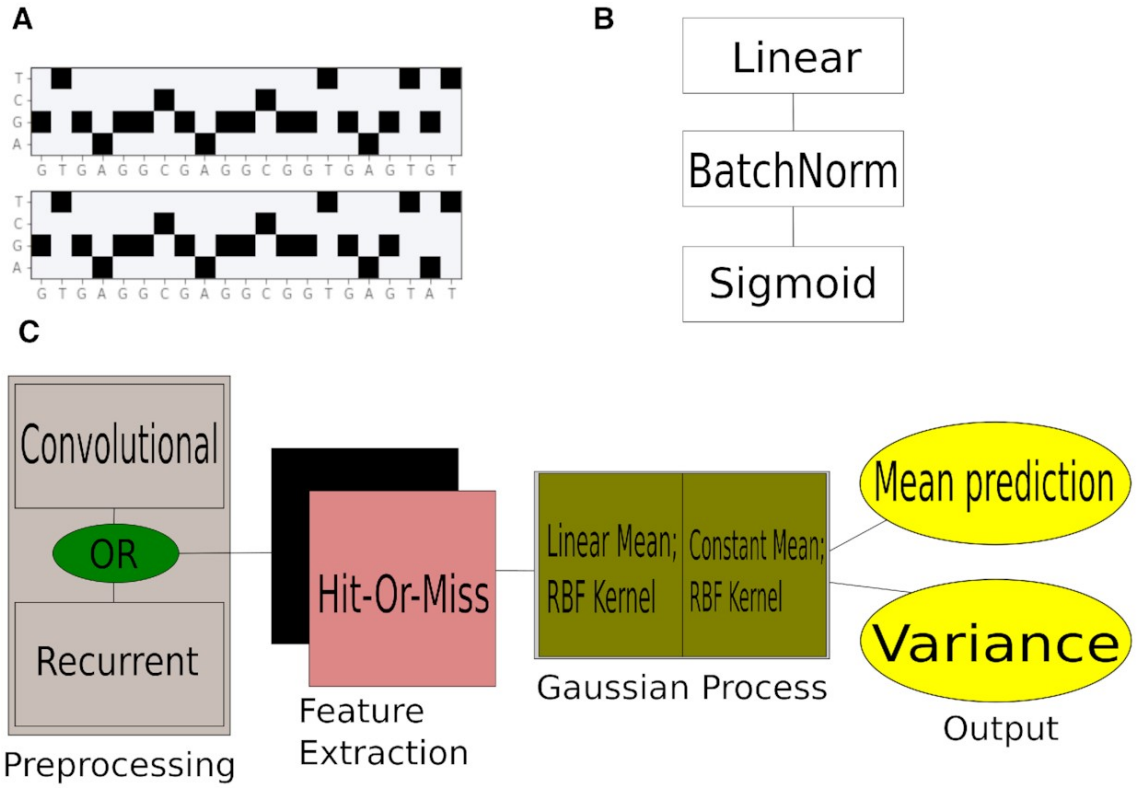

Figure 4-2: The GuideHOM architecture. **(A)** The input: one or two one-hot encoded sequences. To predict on-target cleavage efficiency based only on gRNA, we use the 1D convolutional or recurrent preprocessing modules, and to predict cleavage efficiency based on mismatched gRNA and target sequences, we use 2D convolutional preprocessing. **(B)** The structure of Hit-or-Miss capsule layer; 5 capsule layers are used in the Hit-or-Miss network. **(C)** Schematic illustration of the GuideHOM architecture. See "Materials and Methods" for more details.

assume that all sequences in the genome that match a particular gRNA are cleaved with the same efficiency although this is unlikely to hold precisely, for example, due to differences in chromatin accessibility. However, in this work, we do not use chromatin accessibility or data on other potential contributing factors, relying solely on the gRNA and target sequences. The model is a deep neural network that consists of three modules: (i) preprocessing subnetwork that extracts low-level sequence features using either a 1D or 2D convolutional layer followed by Leaky ReLU (the Rectified Linear Unit, a commonly used activation function) or an LSTM layer (Long Short Term Memory layer); (ii) encoder subnetwork that extracts high-level sequence features from preprocessed inputs using a set of Hit-or-Miss capsules [157], and (iii) a set of Gaussian Processes [158] that estimate cleavage efficiency based on

the extracted features computed by the encoder subnetwork (Figure 4-2C). The sequences are encoded in an one-hot fashion as shown in Figure 4-2A: each nucleotide i is replaced by a vector of 4 components with the i-th component being equal to 1 and every other component set to zero. For gRNA preprocessing, the final, machine-readable input is an array of size $(4, N)$, where N is the length of gRNA spacer; for gRNA-target preprocessing, the final input is an array of size $(2, 4, N)$. To predict on-target cleavage efficiency based only on gRNA, we use the 1D convolutional or recurrent preprocessing modules. To predict cleavage on- and off-target efficiencies based on gRNA and target sequences we use 2D convolutional preprocessing. For a gRNA-target pair, our model estimates the lower and upper bounds of the cleavage efficiency. The recurrent layer is not used for the case with two sequence inputs, in order to minimize unnecessary complexity in the setup. The same architecture and training routine were used for all datasets with minor modifications to accommodate different lengths of the input data (for example, the geCRISPR dataset provides gRNAs with 20 nucleotide spacer lengths only, compared to DeepCRISPR, DeepHF and DeepCpf1 that provide 23nt). The simplest use case of the GuideHOM model is shown in Figure 4-3. As can be seen from Figure 4-3, the choice between gRNA and gRNA-target preprocessing module depends only on the input data in the dataset, whereas the rest of the workflow remains the same: preprocessing, computation of HOM representations, sampling the cleavage efficiency distribution, computation of mean cleavage efficiency and its variance. The general architecture is shown in Figures 4-2B and 4-2C, and the dataset-specific changes are described in Supplementary Table 1 of [1].

4.6.1 Preprocessing subnetwork

The preprocessing subnetwork provides the extraction of low-level sequence features on the level of k-mers. For the experiments in this work, we used three types of preprocessing subnetworks: (i) 1D convolutional, (ii) 2D convolutional and (iii) LSTM-based. An 1D convolutional preprocessing subnetwork consists of a 1d CoordConv layer [159] followed by Leaky ReLU activation. The 2D convolutional layer is the same but with 2d convolutions. We use CoordConv instead of convolutions because

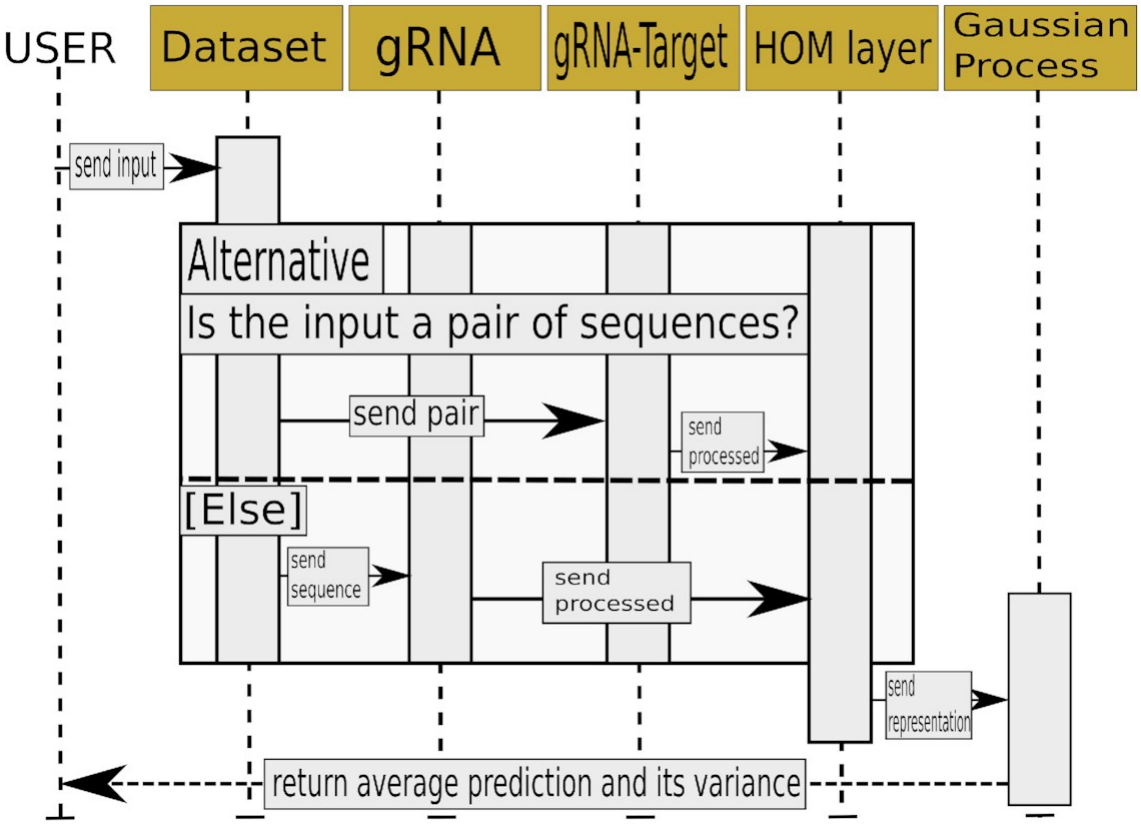

Figure 4-3: The UML (Unified Modeling Language) sequence diagram of a single input use case for the GuideHOM architecture. First, the user supplies the model with an input through the Dataset object. The Dataset object supplies the preselected preprocessing module with one-hot encoded sequence or pair. Either of the preprocessing modules supplies the HOM capsule layer with the preprocessing output. The HOM capsule layer computes coordinates of gRNA/gRNA pairs in the guide space, then, sends the coordinates to the Gaussian Process. The Gaussian Process samples activities from the approximate distribution it has learnt, computes the mean and variance, then, sends the outputs back to the user.

it is shown to work better on one-hot encoded data [159]. The LSTM preprocessing consists of four consecutive LSTM layers.

4.6.2 Encoder subnetwork

The encoder is used to learn patterns based on the sets of k-mers and to present the learned patterns in a concise manner as a matrix of real numbers. The encoder consists of a number of Hit-Or-Miss capsules [116] applied to the preprocessed input in parallel. Each Hit-Or-Miss capsule is a linear layer followed by batch normalization

to accelerate the learning process, and sigmoid activation to constrain the output between 0 and 1. Hit-Or-Miss capsule i encodes the difference between an input example and a center of the space of possible outputs:

$$O_i(X) = C - \textbf{Sigmoid}(\textbf{BatchNorm}_i(\textbf{Linear}_i(X))),$$

where $C = [0.5, 0.5, ...0.5]$. In case of classification, when Hit-or-Miss capsules are optimized with their special Centripetal loss, the perfect "hit" is the vector filled with zeros, and the wrong answers accumulate large "misses". In our case, we did not use the classification (centripetal) loss and classification setup, so the "misses" simply encode the position of the input in the space of all possible inputs, not necessarily the class vectors. Consult the Supplementary Table 1 of [2] for details for the structure and parameters used in the encoder subnetwork.

4.6.3 Gaussian Process

For prediction of the cleavage efficiency, we use a model called Gaussian Process (GP). A Gaussian process is a probability distribution over possible functions that fit a set of points:

$$f(x) = GP(m(x), k(x, x')),$$

where x and x' are the pair of inputs, $f(x)$ is the function we would like to fit, $m(x)$ - mean function and $k(x, x')$ - covariance function, such as:

$$m(x) = \mathbf{E}[f(x)],$$

$$k(x, x') = \mathbf{E}[(f(x)m(x))(f(xj)m(x'))].$$

Mean and covariance functions denote the priors of the distribution over functions. By setting mean and covariance, we choose a set of functions that are used for inference. We fit the Gaussian Process by selecting from a prior distribution only those functions that agree with the observations. This is achieved via optimizing the parameters of the covariance function, the mean function and the encoder neural network using gradient descent. We obtain actual predictions of a value by sampling

the Gaussian Process. Covariance function specifies the covariance between pairs of random variables and mean function specifies the base level of the predicted value. As our loss function we use Evidence Lower Bound [160]:

$$ELBO(q) = \mathbf{E}[\log p(x|z)] - \mathbf{KL}(q(z)||p(z)),$$

where x is the observed cleavage efficiency, z is the latent variable (representation computed by the neural network), $p(x|z)$ is the conditional distribution of cleavage efficiency given the latent variable, $q(z)$ is the variational distribution of the latent variable, $p(z)$ is the prior distribution of the latent variable, **KL** is their Kullback-Leibler divergence. It is a "natural" loss function for deep Gaussian processes that, in our case, can be efficiently computed and optimized via gradient descent using GPytorch primitives. For the gradient descent, we use Adam [161] with starting learning rate $\lambda_0 = 0.01$. We schedule our learning rate to decrease by multiplying it by 0.9 every tenth epoch. The model is trained for 60 epochs. Such step wise reduction helps in convergence towards good local minima of the loss function surface. We also experiment with additional, more traditional loss, Mean Squared Error, so for a number of experiments (in the Supplementary Table 3 of [1] and everywhere else referred as "E+M"), the loss is as follows:

$$Loss_{E+M}(q, y, \hat{y}) = ELBO(q) + \alpha MSE(y, \hat{y}),$$

where y is the optimization target, $\hat{y}$ is the prediction, that is, the mean of a sample from GP and $\alpha = 0.5$. Gaussian Processes can be nested just as linear layers in neural networks. Our network consists of two layers with a linear and constant mean. The output shape of the first layer is 2, and the second process used the previous output as input and outputs the 1d distribution. The scheme is similar to a fully connected neural network, but outputs not a single number but a distribution that can be sampled. After the sampling, we compute the mean prediction and the confidence interval, which is the uncertainty-aware prediction of cleavage efficiency we aim at obtaining.

4.7 Clustering and motif analysis of the guide space

Agglomerative clustering was performed using SciPy library [151], with default settings, max_d of 2 and criterion "distance". To draw the motif logos, the weblogo [153] library was used.

4.8 Performance metrics for regression and quality of confidence intervals

To analyze the performance of the regression models, we use three classical metrics - determination coefficient (r^2), Pearson and Spearman correlation coefficients (PCC and SCC respectively), computed using Scikit-Learn library. To understand whether the confidence intervals the model gives is acceptable, we compute how many real labels for examples from the test set lie between the predicted label and predicted standard deviation:

$$\rho(Y, \hat{Y}, \sigma, M) = \frac{\sum_{i=0}^{N} I(Y_i \in [\hat{Y}_i - M\sigma_i, \hat{Y}_i + M\sigma_i)}{N},$$

where N is the test set size, I is the indicator function which is 1 if the argument is true and 0 otherwise, $M \in [1, 2, 3]$. We compute ρ for one, two and three standard deviations and check whether the values are close to 68%, 95% and 99.7% respectively. We denote those values as ρ_{68}, ρ_{95} and $\rho_{99.7}$.

4.9 Explanation for cleavage efficiency machine learning models

To explain the predictions of the model, the Accumulated Local Effects (ALE, [132]) are computed, Python library alibi [146] is used. ALEs are computed over M randomly generated synthetic gRNAs (or gRNA-target pairs, depends on the model) that are flattened into $(M, 4N)$ or $(M, 4{\times}2N)$ matrix. The model requires $(M, 2, 4, N)$ or $(M, 4, N)$, where N is the length of the input, so an intermediate class to reshape the inputs is used. M in our work is set to 10000. The resulting ALE values have

the shape of $(4N, A, B)$ where A is the number of feature intervals, in our case, 2 for 0 and 1 of one-hot encoded nucleotides, and B is the number of prediction targets, again, 2 in our case, corresponding to mean efficiency and variance. We are interested only in the component of features that corresponds to the values of 1, because we would like to see the importance of a presence of a certain nucleotide on a certain position. The heatmaps are constructed from the ALE explainer class, first and second components of the vector that corresponds to the value 1 of features for mean efficiency and variance respectively. See the Figure9.ipynb, SupplementaryFigure1.ipynb, SupplementaryFigure2.ipynb and reproduce_explanations.py notebooks and scripts from [2]. For logo sequences, constructed out of ALE values, the softmax with additional temperature parameter is calculated:

$$y_i(X, t) = \frac{\exp(\frac{X_i}{t})}{\sum_{j=1}^{N} \exp(\frac{X_j}{t})},$$

where t is temperature parameter, the less temperature is, the more distinct the output probabilities are, X_i are the features that correspond to i-th nucleotide out of four, i and j are the counters. Softmax is applied along the nucleotide axis, the resulting matrix is of $(4, N)$ size, and if we sum the vector along the first axis, we get the vector of N ones. We use weblogo [153] to draw the resulting sequences.

Chapter 5

Inequality in capsule networks for detection of potential off-target events

5.1 Introduction to the project

This chapter is based on one of the papers that constitute the current dissertation – "Measuring internal inequality in capsule networks for supervised anomaly detection" [2] published in Scientific Reports. My contributions in this project are as follows: I (together with my supervisor) have developed the main concept, designed the study and wrote the main manuscript text. I also have performed all computational experiments.

The problem of anticipating rare events is of high interest to the modern technological society. A lot of problems people face, like bank fraud [162], structural defects in materials [163], early development of diseases [141], and manipulation of public opinion in social networks [164], boil down to knowing what a typical behavior for a system is and what is not.

Anomaly detection is the process of examining data to determine where the aberrations lie. Usually, this involves analyzing how well the parts of the system are performing to understand what the normal behavior consists of. Sometimes there is also some degree of knowledge about abnormal behavior. In this paper,

we use the common notion of anomaly in machine learning – an instance of data that is rare and deviates a lot from other, more prevalent ones. Anomaly detection is essential for analysis of almost any complex data. In bioinformatics, one can consider prediction of protein-protein interactions and CRISPR off-target cleavage prediction. In computer vision there are various cases of defect detection. All these tasks require a deep neural network-based solution due to the data complexity.

Anomaly detection then can be supervised or unsupervised [165] depending on whether the examples of atypical behavior are available. Each kind has its benefits and limitations: supervised anomaly detection methods tend to be more accurate with the known anomalies than the unsupervised ones, but also tend to miss the anomalies never observed before [155].

This project is concerned with supervised anomaly detection, which deals with the classification problems of a very abundant normal class and a scarce anomalous class. It is a case of highly imbalanced binary classification. We focus on the problems with relatively complex data such as images or DNA sequences, which are best solved with Deep Learning methods. Given that the anomalous class is usually infrequent (1%-0.01%, mere hundreds of examples), commonly used deep learning methods tend to perform poorly. Supervised anomaly detection via deep neural networks usually employs carefully crafted augmentation [166], complex architectures [167], GAN-based generation [168], and other tricks aimed to expand the number of anomalous examples.

In this project, we present a different approach, based on Capsule Networks with dynamic routing [157]. A capsule network consists of grouped neurons that output vectors encoding parameters of an object or a part of an object. The key difference between Capsule and Convolutional Neural Networks is the output: while Convolutional Networks output a vector of N class probabilities (where N is the number of classes), Capsule Networks output a matrix that consists of N vectors. These vectors are called *capsules* and encode the learned representation of an object given it belongs to a corresponding class. The class probabilities are computed by taking the vector norms. Low-level primary capsules that represent parts of an object feed their output to class capsules that represent the object as a whole. Parts from an

object of a rare class are rarely present in an object of a more common class. If the network detects parts that do not fit into a common class, then the low-level primary capsules that correspond to these parts are triggered and therefore contribute to prediction of the object being an anomaly. A method to detect anomalies with Capsule Networks would benefit from exploiting this "part-whole" relationship expressed in dynamics of primary capsules voting for a class.

Previous works [135, 134] on supervised anomaly detection with capsule networks use the reconstruction ability and class probabilities to separate outliers from inliers, while the methods proposed in this work are based on the evaluation of unequal response of the routing mechanism to normal and aberrant inputs. Class probability is given by the computation of class capsule output via routing. Routing by agreement has an intrinsic property of polarization [169] – convergence on a single route from primary to class capsules. This property gives rise to inequality between a well-predicted and poorly predicted class in case of class imbalance. We can measure such discrepancy using economic inequality metrics, such as Gini [170] and Palma [171] coefficients. Our main contributions can be summarized as follows:

1. We propose a new approach for supervised anomaly detection using capsule networks;

2. We suggest a new application of economic inequality metrics to machine learning which also allows investigating internal mechanisms of capsule networks;

3. We perform a comprehensive review and comparison of different capsule network-based anomaly detection methods on standard benchmarks and real-world data, which confirms state-of-the-art performance of the proposed methods;

4. We use the developed methods to build a new approach towards detection of potential CRISPR-Cas off-target events using supervised anomaly detection.

5.2 Main idea

Both previous studies [134, 135] base their work on the differences between estimated class probabilities and reconstruction subnetwork. We do not use reconstructions in

our method while it could indeed help in a real-life application. Instead, we focus on the estimation of class probabilities. In capsule network setting, the probabilities are formed by softmax of capsule output vector norms. Output vectors provide information beyond class probability, according to the original Capsule Networks paper [157], they capture interpretable properties like thickness of stroke, localization and shape. This information gets lost when we compute the norm.

To gather as much information as possible, we dive deeper into the routing mechanics. Coupling coefficients c, computed from the routing table, contain all the information about the way primary and class capsules would route for a given example. We base our research on the assumption that the couplings on normal and anomalous capsule show different results when encountering an abundant class of examples and a rare one.

To do the classification, we ideally need a summary statistic for the couplings. Let c_j be a part of coupling table c that corresponds to j-th secondary capsule. It is a tensor of size (N_P, O_S) where N_P is the number of primary capsules, O_S is the dimension of class capsule output vector. For each secondary capsule j we first sum the respective couplings along the O_S axis:

$$M_j = \sum_{l=1}^{O_S} c_j^l,$$

where c_j^l is a vector of dimension N_P obtained as a slice of tensor c for secondary capsule j and its output dimension $l \in \overline{1, N_S}$. The distribution of M_j for different cases is shown on the Fig 5-1. This vector can be interpreted as a vector of total contributions of primary capsules to the j-th secondary capsule results. Those contributions due to polarization property of capsule networks would be highly inequal in case the network is well-trained. Fig 5-1 clearly shows the change in the distributions for both normal and anomalous capsules that can be captured by inequality measures discussed below.

To evaluate an internal inequality in capsule networks, we require a short detour to econometrics. Income inequality in economics is measured by a number of statistical criteria [171]. Most popular, the ones we use here, are Gini [170] and

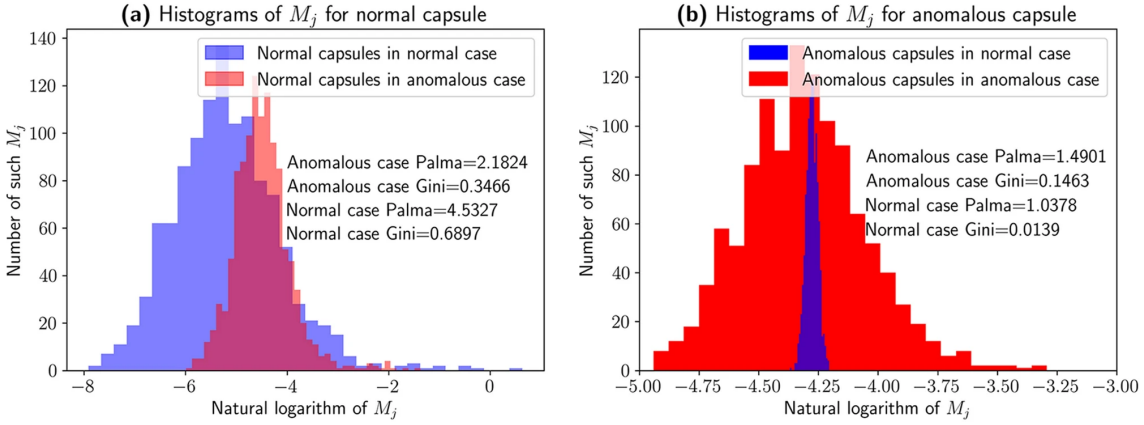

Figure 5-1: An example of coupling coefficients distributions M_j for the data from KuzushijiMNIST. The histograms of M_j for normal and anomalous capsules in normal and anomalous cases are shown with specified Gini and Palma coefficients. Note the difference between the distributions for normal **(a)** and anomalous **(b)** capsules in the normal case and the anomalous one. Normal capsule only mildly captures the difference between the samples. We use the couplings of anomalous capsule to compute Gini and Palma coefficients to capture this more pronounced difference for anomalous capsule. Note that on the plot, the distribution for the logarithm of the summed couplings are shown, but the values of Gini and Palma coefficients given are computed on the summed couplings without the logarithm.

Palma [172] coefficients. Gini coefficient is

$$Gini(Z) = \frac{\sum_{i=1}^{n}(2i - n - 1)Z_i}{n\sum_{i=1}^{n} Z_i}, \tag{5.1}$$

where n is the sample size, i is the number of example in the sample and z_i is the value. Gini coefficient ranges from 0 to 1 and the value for the most inequal case (only one non-zero example) is 1.

More recently, Palma [172] coefficient started to displace Gini as a go-to measure of income inequality:

$$Palma(Z) = \frac{Q_{90}(Z)}{Q_{40}(Z)}, \tag{5.2}$$

where Q_{90} and Q_{40} are the 90-th and 40-th percentiles respectively. Key assumption behind Palma coefficient is that the tails of income distribution contribute to inequality the most and the middle ground remains relatively stable over time. This assumption makes Palma coefficient work rather well in case of yearly assession of economic inequality of countries. We hypothesise that Palma coefficient would work better than Gini because the assumption holds in our case due to polarization.

Now we are equipped to apply income inequality metrics to Capsule Networks. The first proposed criterion is based on Gini coefficient, see Eq (5.1):

$$Gini(M_j) = \frac{\sum_{k=1}^{N_P}(2k - N_P - 1)M_{jk}}{N_P \sum_{k=1}^{N_P} M_{jk}}.$$

The second criterion is based on Palma coefficient and is computed according to Eq (5.2):

$$Palma(M_j) = \frac{Q_{90}(M_j)}{Q_{40}(M_j)}.$$

Coming back to the example from Fig 5-1, we clearly see that both Gini and Palma coefficients allow to capture the difference in distribution for anomalous capsule.

We use both of these criteria as a measure of data point "outlierness" and compute the AUC directly. It is possible to use Logistic Regression or any other classification model (SVM, XGBoost, Random Forest, ...) based on the value of Gini, Palma or both and also consider adding other features derived from the data or reconstruction properties, but we defer it to future work. We use Adam optimizer [161] with default settings and train-test splits defined according to Section 4.2.3 of Chapter 4.

5.3 Performance on CRISPR-Cas off-target event detection

For the CRISPR off-target task we get the best results with Palma (0.9631 ± 0.0125 AUROC, 0.6876 ± 0.0264 average precision) and Gini (0.9666 ± 0.0118 AUROC, 0.6571 ± 0.0318 average precision) coefficients and the worst results with ABC and NL (0.5314 ± 0.0134, 0.5147 ± 0.0483 AUROC respectively, and whopping 0.0271 ± 0.0007, 0.0264 ± 0.0044 average precision respectively). Normalized reconstruction error N_{re} again performs worse than its N_{pp} pair – 0.6756 ± 0.0372 AUROC, 0.2725 ± 0.0328 average precision, and 0.9147 ± 0.0144 AUROC, 0.304 ± 0.0701 average precision respectively, but anomaly score and plain capsules give not the worst, but the average quality in AUROC – 0.6131 ± 0.0471 and 0.7518 ± 0.0404 respectively, while in average precision, plain capsules score below Gini and Palma only (0.4059 ± 0.034). Anomaly score performs in average precision similarly to AUROC – slightly worse

than N_{re} (0.2535 ± 0.0327). The advantage of inequality-based measures over the rest is clearly seen.

5.4 Performance on computer vision benchmarks

5.4.1 MNIST-like Benchmarks

We measure the performance using the AUROC metrics. For each dataset and outlier fraction we compute AUROCs for all classes (10 values), then we report average AUROC and the standard deviation. We denote performance of the capsule network without any additional metrics as "Plain", non-capsule baselines as NL and ABC, anomaly score as A, normality scores as N_{pp} and N_{re}. The results for diverse outliers are shown in Table 5.1.

The proposed methods, either Palma or Gini outperforms other metrics and baselines in 1% and 10% cases for diverse outliers for both AUROC and average precision (shown in Supplementary Table 1 of [2]). For CIFAR10, Palma and Gini also perform the best in 0.1% case. This is probably due to loss of information after computing the norms according to Eqs (4.1) and (4.2). For 1% KMNIST and 1% CIFAR10 case, Palma and Gini respectively come second to N_{pp}. In MNIST and FMNIST 0.1% though (in AUROC, and for KuzijishiMNIST additionally in average precision), Palma and Gini perform way worse than anomaly score and both normality scores. Overall, as the number of anomalous examples grows, the performance of normality measures decreases, performance of anomaly measure increases slightly, and performance of Palma and Gini increases by a large margin.

For diverse inlier settings, Palma and Gini coefficients outperform almost everything for all cases except CIFAR10 0.1% and 1%, KuzijishiMNIST 0.1% in which Gini and Palma coefficients respectively perform the second best to N_{pp} (Table 5.2 and Supplementary Table 2 of [2]). As in the diverse outlier settings, the proposition of the previous study [134] that plain capsule network performs poorly for anomaly detection stands. As in the diverse outlier case, Palma and Gini coefficients are very close to each other.

Table 5.1: AUROCs for diverse outlier setup (fractions 0.1%, 1% and 10%).

	0.1%, mean ± std	**1%, mean ± std**	**10%, mean ± std**	**Dataset**
Palma	**0.6488** ± 0.0598	0.7213 ± 0.0644	**0.8313** ± 0.0473	
Gini	**0.6494** ± 0.0604	**0.7217** ± 0.0646	**0.8322** ± 0.0471	
Plain	0.5000 ± 0.0000	0.5003 ± 0.0009	0.6372 ± 0.0635	
A	0.5813 ± 0.1159	0.5856 ± 0.1207	0.6084 ± 0.0985	
N_{pp}	0.6487 ± 0.0755	**0.7369** ± 0.0644	0.7915 ± 0.0581	CIFAR10
N_{re}	0.5851 ± 0.1085	0.5908 ± 0.1198	0.6159 ± 0.0944	
ABC	0.6407 ± 0.0714	0.6426 ± 0.0772	0.6489 ± 0.0707	
NL	0.5000 ± 0.0811	0.5000 ± 0.0811	0.5000 ± 0.0812	
Palma	0.8387 ± 0.0876	0.9795 ± 0.0123	0.9805 ± 0.0221	
Gini	0.8366 ± 0.0883	**0.9819** ± 0.0131	**0.9984** ± 0.0007	
Plain	0.5767 ± 0.0660	0.8633 ± 0.0584	0.9821 ± 0.0060	
A	**0.9253** ± 0.0522	0.9545 ± 0.0313	0.9541 ± 0.0392	
N_{pp}	0.9144 ± 0.0644	0.9528 ± 0.0297	0.7479 ± 0.0929	MNIST
N_{re}	**0.9618** ± 0.0235	**0.9806** ± 0.0130	**0.9896** ± 0.0087	
ABC	0.8169 ± 0.1073	0.8102 ± 0.1051	0.8293 ± 0.1063	
NL	0.4943 ± 0.1723	0.4947 ± 0.1714	0.4942 ± 0.1716	
Palma	0.7458 ± 0.0489	**0.9109** ± 0.0299	**0.9679** ± 0.0217	
Gini	0.7446 ± 0.0494	0.9096 ± 0.0301	**0.9757** ± 0.0154	
Plain	0.5048 ± 0.0098	0.6642 ± 0.0692	0.9071 ± 0.0362	
A	0.7585 ± 0.0764	0.7881 ± 0.0572	0.7870 ± 0.0764	
N_{pp}	**0.8417** ± 0.0486	**0.9316** ± 0.0247	0.7931 ± 0.0548	KMNIST
N_{re}	**0.8645** ± 0.0547	0.8923 ± 0.0461	0.9066 ± 0.0391	
ABC	0.6025 ± 0.1122	0.6049 ± 0.1127	0.6180 ± 0.1215	
NL	0.5000 ± 0.1078	0.5000 ± 0.1072	0.5000 ± 0.1067	
Palma	0.8388 ± 0.1134	**0.9390** ± 0.0467	**0.9602** ± 0.0340	
Gini	0.8419 ± 0.1119	**0.9392** ± 0.0468	**0.9784** ± 0.0271	
Plain	0.5759 ± 0.1172	0.8027 ± 0.0797	0.9210 ± 0.0554	
A	**0.8914** ± 0.0609	0.9090 ± 0.0570	0.9087 ± 0.0820	
N_{pp}	0.8792 ± 0.0567	0.8834 ± 0.0836	0.7453 ± 0.1117	FMNIST
N_{re}	**0.9153** ± 0.0666	0.9195 ± 0.0715	0.9080 ± 0.0875	
ABC	0.7921 ± 0.1595	0.7921 ± 0.1588	0.7933 ± 0.1611	
NL	0.5000 ± 0.2215	0.5000 ± 0.2218	0.5000 ± 0.2207	

5.4.2 Malignant Skin Lesion Classification

This constitutes the first application of supervised anomaly detection with capsule networks to a real-world non-benchmark dataset. Following footsteps of Quinn et al. [141], we consider that anomaly detection can facilitate the search for actual biological anomalies – malignant skin lesions. The main conceptual difference from this work [141] (apart from using photos instead of transcriptomics data) is that we

Table 5.2: AUROCs for diverse inlier setup (fractions 0.1%, 1% and 10%).

	0.1% mean ± std	**1% mean ± std**	**10% mean ± std**	**Dataset**
Palma	**0.6985** ± 0.0570	0.7980 ± 0.0624	**0.9068** ± 0.0413	
Gini	0.6982 ± 0.0575	**0.7984** ± 0.0622	**0.9084** ± 0.0413	
Plain	0.5000 ± 0.0000	0.5131 ± 0.0157	0.7060 ± 0.0768	
A	0.5111 ± 0.1465	0.5217 ± 0.1388	0.6329 ± 0.1361	
N_{pp}	**0.7013** ± 0.0465	**0.8157** ± 0.0587	0.8341 ± 0.0284	CIFAR10
N_{re}	0.5113 ± 0.1419	0.5223 ± 0.1347	0.6397 ± 0.1250	
ABC	0.5230 ± 0.118	0.5335 ± 0.1163	0.5195 ± 0.1051	
NL	0.5000 ± 0.0812	0.5000 ± 0.0812	0.5000 ± 0.0811	
Palma	**0.9849** ± 0.0080	**0.9981** ± 0.0043	0.9697 ± 0.0035	
Gini	**0.9848** ± 0.0081	**0.9981** ± 0.0012	**0.9997** ± 0.0002	
Plain	0.8039 ± 0.0727	0.9634 ± 0.0138	0.9928 ± 0.0031	
A	0.9508 ± 0.0342	0.9898 ± 0.0080	0.9819 ± 0.0442	
N_{pp}	0.9496 ± 0.0568	0.6210 ± 0.1288	0.3073 ± 0.0997	MNIST
N_{re}	0.9593 ± 0.0248	0.9920 ± 0.0052	**0.9939** ± 0.0111	
ABC	0.5420 ± 0.2011	0.5398 ± 0.2025	0.5598 ± 0.1960	
NL	0.5048 ± 0.1731	0.5028 ± 0.1722	0.5061 ± 0.1723	
Palma	**0.9067** ± 0.0282	**0.9771** ± 0.0125	**0.9925** ± 0.0053	
Gini	0.9066 ± 0.0285	**0.9771** ± 0.0125	**0.9934** ± 0.0056	
Plain	0.6029 ± 0.0322	0.8303 ± 0.0288	0.9558 ± 0.0159	
A	0.7611 ± 0.0875	0.8908 ± 0.0389	0.9501 ± 0.0294	
N_{pp}	**0.9248** ± 0.0268	0.8451 ± 0.0402	0.5218 ± 0.0594	KMNIST
N_{re}	0.8147 ± 0.0670	0.9286 ± 0.0242	0.9784 ± 0.0119	
ABC	0.5000 ± 0.1279	0.5000 ± 0.1279	0.5000 ± 0.1279	
NL	0.5000 ± 0.1071	0.5000 ± 0.1074	0.5000 ± 0.1077	
Palma	**0.9289** ± 0.0424	**0.9617** ± 0.0265	**0.9480** ± 0.0582	
Gini	**0.9284** ± 0.0426	**0.9661** ± 0.0296	**0.9891** ± 0.0136	
Plain	0.6517 ± 0.1410	0.8076 ± 0.1266	0.9376 ± 0.0525	
A	0.7474 ± 0.1705	0.8192 ± 0.1257	0.8527 ± 0.0923	
N_{pp}	0.8762 ± 0.1262	0.7358 ± 0.2231	0.5970 ± 0.2339	FMNIST
N_{re}	0.7374 ± 0.2045	0.7989 ± 0.1598	0.8427 ± 0.1199	
ABC	0.5515 ± 0.2192	0.5499 ± 0.2133	0.5775 ± 0.2011	
NL	0.5000 ± 0.2215	0.5000 ± 0.2208	0.5000 ± 0.2214	

actually use the examples of such anomalies – the detection is not unsupervised.

The results, as Table 5.3 shows, are rather similar to the results on the MNIST-like benchmarks (Tables 5.1 and 5.2). Palma and Gini outperform every other metric by a large margin and provide almost the same performance. For the case B, diverse outliers and homogeneous inliers, Gini outperforms Palma, but not very much. The N_{pp} measure performs close to Palma and Gini, while the rest is far behind.

Analysis of average precision (Supplementary Table 3 of [2]) for this task also

Table 5.3: AUROCs for HAM10000 with the following setups: A – diverse outliers, diverse inliers, B – diverse outliers, homogeneous inliers, C – homogeneous outliers, homogeneous inliers, D – homogeneous outliers, diverse inliers.

	A	**B**	**C**	**D**
Palma	**0.7348** ± 0.0160	**0.7312** ± 0.0137	**0.7880** ± 0.0165	**0.7532** ± 0.0209
Gini	**0.7347** ± 0.0157	**0.7312** ± 0.0137	**0.7883** ± 0.0163	**0.7528** ± 0.0207
Plain	0.5000 ± 0.0000	0.5000 ± 0.0000	0.4996 ± 0.0005	0.5000 ± 0.0000
A	0.5693 ± 0.0120	0.5815 ± 0.0216	0.5846 ± 0.0129	0.5953 ± 0.0247
N_{pp}	0.6950 ± 0.0055	0.7204 ± 0.0120	0.7468 ± 0.0099	0.7406 ± 0.0152
N_{re}	0.5675 ± 0.0122	0.5945 ± 0.0200	0.5848 ± 0.0132	0.6079 ± 0.0234
ABC	0.5456 ± 0.0027	0.5888 ± 0.0300	0.5710 ± 0.0076	0.6066 ± 0.0016
NL	0.5865 ± 0.0030	0.5908 ± 0.0046	0.6173 ± 0.0037	0.6269 ± 0.005

show clear superiority of Gini and Palma over the other metrics, closely mirroring the AUROC results, but the difference here is also more pronounced, because every metric except for Gini and Palma scores only about 2% more on average than the respective proportion of the positive class (anomaly). Such result is close to the one we would expect from a degenerate model that outputs 1 regardless of the input.

Chapter 6

Uncertainty-aware and Explainable Machine Learning for gRNA selection

This chapter is based on one of the papers that constitute the current dissertation – "Uncertainty-aware and interpretable evaluation of Cas9–gRNA and Cas12a–gRNA specificity for fully matched and partially mismatched targets with Deep Kernel Learning" [1] published in Nucleic Acids Research. The contribution of authors are as follows: E.S., S.A.S. and I developed the concepts and designed the study. E.V.K., M.P., A.Y.O., S.A.S. and K.V.S. supervised the research. I have designed the methods and performed all analyses. E.S., S.A.S. and I wrote the manuscript, which was read, edited and approved by all authors.

6.1 GuideHOM provides acceptable confidence intervals and accurate and reliable predictions of on-target cleavage efficiency

In our study, combination of Hit-Or-Miss networks [116] and Deep Gaussian Processes [158] helps to augment the point estimates of cleavage efficiency with the information on prediction uncertainty. To our knowledge, this is the first application

of uncertainty-aware machine learning for CRISPR cleavage prediction efficiency. Using the confidence intervals derived from the model helps to overcome the problem of noisy and biased training data. We attempt to replicate the results of each original study from which we extracted datasets used to train the GuideHOM model. It allows us to systematically compare its performance with the original models.

We use the same data for training and testing as the original works in cases where the actual training and testing sets are available and the same train-test split ratio in other cases. Table 6.1 shows the train-test splits corresponding to each analyzed dataset. The confidence intervals GuideHOM provides tend to be acceptable (see Figure 6-1C for visual explanation): the frequency of the real value Y present within the confidence intervals (with confidence level α) computed on the corresponding test sets was, for all cases, close to the preset confidence level ($\alpha = \sigma, 2\sigma, 3\sigma$). As the loss function, we use ELBO (Evidence Lower Bound [160]) or a sum of ELBO and a half of MSE (Mean Squared Error). For example, the frequencies of the real values in $Y_{mean} \pm \sigma$, $Y_{mean} \pm 2\sigma$, $Y_{mean} \pm 3\sigma$ for a model trained with minimizing ELBO as an optimization objective are 0.7535, 0.9422 and 0.9835, respectively, as estimated on the respective validation set. The full table of confidence intervals for all trained models is given in Supplementary Table 3 of [1].

Table 6.1: Train-test splits for all datasets.

Dataset	Train	Test	Validation	Cell line/species
geCRISPR	0.9 (3258)	0.1 (361)	Separate set (520)	*Homo sapiens, Danio rerio, Mus musculus, Xenopus tropicalis*
DeepHF wt	0.765 (42538)	0.085 (4727)	0.15 (8341)	HEK239T, HeLa
DeepHF SpCas9HF1	0.765 (43520)	0.085 (4836)	0.15 (8534)	HEK239T, HeLa
DeepHF eSpCas9	0.765 (44843)	0.085 (4983)	0.15 (8793)	HEK239T, HeLa
DeepCpf1	1 (20506)	-	3 separate sets	HEK239T, hct16
Jost et al	0.8 (20999)	0.2 (5249)	-	K562, Jurkat
DeepCRISPR	0.8 (13194)	0.2 (3298)	Separate sets	HEK239T, HeLa, hl60, hct16
Cas12a off-target	0.8 (1252)	0.2 (313)	-	HEK239T

According to the learning curves (Figure 6-1A), only about 30% of the DeepHF [78] and Cas9 gRNA-target pair [5] datasets provide sufficient data for the model to

reach results comparable to the original models. The Cas9 gRNA-target pair model reaches lower levels of uncertainty while having less data to train on, which implies a lower noise level in Jost et al [5] dataset (Figure 6-1B) because most of the uncertainty in the output comes from label noise in the dataset [173, 174].

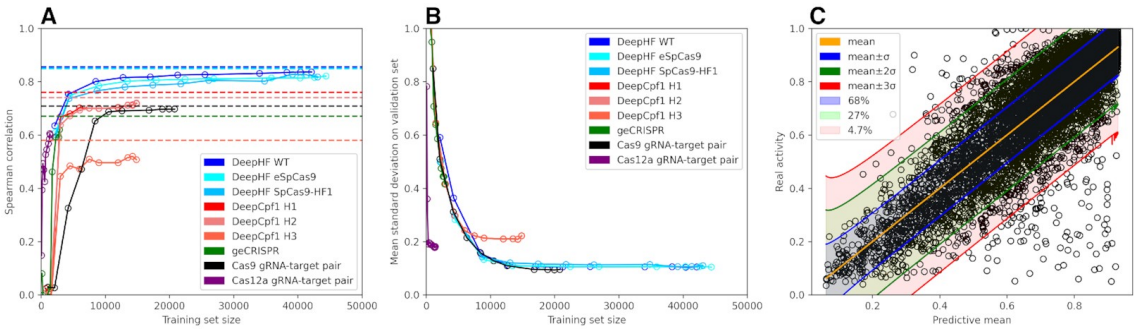

Figure 6-1: Models learning curves on different datasets. **(A)** Learning curves for indicated models/datasets are shown. Dashed lines indicate the performance of previous tools (DeepHF, DeepCpf1 and so on). The line for Cas12a off-target dataset is not shown since only classification models for this dataset are published. **(B)** Reduction of uncertainty dependent on the training set size. The models for DeepHF and DeepCpf1 shown here are CNN-based and were trained to minimize ELBO+MSE. The Cas9 gRNA-target pair model was also trained to minimize ELBO+MSE. **(C)** The output of trained model. The dots denote examples from the validation set. For each input example, the model outputs a sample of predictions, the mean of which is shown as the orange line, and the standard deviation gives the possible range of errors. 68% of all real activities lie in the blue area between the orange and blue lines – for 68% examples, the real cleavage efficiency is no more than one predicted standard deviation larger or smaller than the predicted mean cleavage efficiency. 27% of real activities lie in the green area – only 27% of examples have the real cleavage efficiency larger or smaller than the predicted mean cleavage efficiency for more than one predicted standard deviation. 4.7% of real activities lie in the red area – for 4.7% examples, the real cleavage efficiency is larger or smaller than the predicted mean cleavage efficiency for more than two predicted standard deviations. The rest 0.3% are the ones that have the real cleavage efficiency larger or smaller than the predicted mean cleavage efficiency for more than 3 predicted standard deviations.

The same performance dynamics is observed for all analyzed datasets: after the training sample size grows to about 10,000 gRNAs, the performance converges at a plateau so that the additional improvement is negligible. Uncertainty reduction follows a similar course: the smallest mean standard deviation is achieved near the 10,000 gRNAs mark and remains approximately constant with larger training sets. These observations suggest that for this version of the GuideHOM architecture,

Table 6.2: Performance on benchmark datasets for on-target cleavage efficiency prediction.

Dataset	Model	Metric	Value
DeepHF wildtype	[78], RNN	Hold-out SCC	0.8555
DeepHF wildtype	[78], RNN	10-fold CV SCC	NA
DeepHF wildtype	This study, C E	Hold-out SCC	0.8392
DeepHF wildtype	This study, C E	10-fold CV SCC	0.8066
DeepHF eSpCas9	[78], RNN	Hold-out SCC	0.8491
DeepHF eSpCas9	[78], RNN	10-fold CV SCC	NA
DeepHF eSpCas9	This study, R E	Hold-out SCC	0.8220
DeepHF eSpCas9	This study, R E	10-fold CV SCC	0.6927
DeepHF SpCas9-HF1	[78], RNN	Hold-out SCC	0.8512
DeepHF SpCas9-HF1	[78], RNN	10-fold CV SCC	NA
DeepHF SpCas9-HF1	This study, R E+M	Hold-out SCC	0.8364
DeepHF SpCas9-HF1	This study, R E+M	10-fold CV SCC	0.7900
geCRISPR V520	[73], mono binary	Hold-out PCC	0.6700
geCRISPR V520	[73], mono binary	10-fold CV PCC	0.6800
geCRISPR V520	This study, C E+M	Hold-out PCC	0.6055
geCRISPR T3619	This study, C E+M	10-fold CV PCC	0.5926
DeepCpf1 H1	[14]	Hold-out SCC	0.7600
DeepCpf1 H1	This study, R E	Hold-out SCC	0.7283
DeepCpf1 H2	[14]	Hold-out SCC	0.7400
DeepCpf1 H2	This study, C E+M	Hold-out SCC	0.7184
DeepCpf1 H3	[14]	Hold-out SCC	0.5800
DeepCpf1 H3	This study, R E	Hold-out SCC	0.5478
DeepCpf1 train	[14]	10-fold CV SCC	NA
DeepCpf1 train	This study, C E+M	10-fold CV SCC	0.5165

collecting datasets of more than 10,000 gRNAs yields diminishing returns in terms of the model accuracy, so a better strategy is to focus on the reduction of the label noise. However, the cut-off may have to be re-accessed depending on the type of gRNA library used, i.e., those targeting intronic, exonic regions, high/low expressed genes etc. The performance of the method for on-target cleavage efficiency prediction on benchmark datasets tends to be close to that of the best original point estimate models (Table 6.2; GuideHOM models are denoted as follows: preprocessing module (C for CNN or R for RNN), loss function (E for ELBO or E+M for ELBO+MSE); see the "Deep Kernel Learning for cleavage efficiency estimation" section of "Materials and methods" chapter for more information. Wang et al. [78] and Kim et al. [14] did not perform 10-fold cross validation, which is indicated as "NN" - not applicable

– in Table 6.2.

Table 6.3: Comparison of model predictions of the on-target cleavage efficiency for the AsCas12a subset.

Model	ρ_{68}	ρ_{95}	$\rho_{99.7}$	PCC	SCC	r^2
DeepCpf1 R E:H1	0.69	0.95	1.00	0.74	0.73	0.55
DeepCpf1 C E+M:H2	0.68	0.93	0.99	0.72	0.72	0.52
DeepCpf1 C E:H1	0.67	0.93	0.99	0.73	0.72	0.53
DeepCpf1 C E:H2	0.68	0.93	0.99	0.71	0.71	0.51
DeepCpf1 C E+M:H1	0.67	0.93	0.99	0.72	0.71	0.52
DeepCpf1 R E:H2	0.71	0.95	1.00	0.71	0.71	0.51
DeepCpf1 R E+M:H1	0.70	0.96	1.00	0.72	0.70	0.52
DeepCpf1 R E+M:H2	0.72	0.95	1.00	0.67	0.66	0.45
DeepCpf1 R E:H3	0.27	0.67	0.95	0.51	0.55	0.26
DeepCpf1 C E:H3	0.25	0.63	0.92	0.50	0.53	0.25
DeepCpf1 C E+M:H3	0.24	0.62	0.91	0.49	0.51	0.24
DeepCpf1 R E+M:H3	0.31	0.73	1.00	0.42	0.46	0.18
Cas12a pair E	0.59	0.87	0.96	0.57	0.57	0.32
Cas12a pair E+M	0.59	0.86	0.96	0.56	0.57	0.31

The performance of all trained models is given in Supplementary Table 3 of [1]. Although in the original work on DeepHF datasets, the RNN-based models have been found to be superior to CNN-based ones [78], in our analysis, this distinction was not as pronounced. Some CNN-based GuideHOM models outperform RNN-based ones: for example, for the wildtype, the CNN-based model performs better. From the performance of the best models, it becomes clear that a combination of ELBO and MSE loss functions performs better than ELBO only. Let us consider the following use case: we would like to choose a model for gRNA selection for asCas12a editing experiment in human HEK239T cell line. We aim at maximizing the on-target efficiency but are not interested in minimizing the off-target effect. In Table 6.3, we provide a subset of performance measures for this case.

As it can be seen from Table 6.3, we have a choice between models trained on DeepCpf1 and on Cas12a gRNA-target pair set. All DeepCpf1-based models show acceptable confidence intervals and good r^2, Pearson and Spearman Correlation Coefficients.

For this test case, we are not interested in off-target effects, therefore DeepCpf1 R E is the model of choice (it outperforms all others on H1 and H3 datasets in

Spearman Correlation Coefficient). We also can use all good models as an ensemble, which will result in an improvement of 1-2% in the correlations at the excess of increased uncertainty since we would have to sum the variances to get the correct uncertainty estimation.

Table 7.1 shows the results for the Jost et al. dataset, where the difference between E+M and E only is negligible. Compared with the model [5], changed accordingly to reflect the difference in the input size (we used as input 23 nt sequence, including the gRNA spacer and PAM, whereas in the Jost et al. [5] analysis, 2 upstream and 1 downstream flanking nucleotides were also included, resulting in the input size of 26 nt), the performance improves from 0.617 to 0.625. For the purpose of the comparison, we modify the original code from the supplementary file [5] by taking the sequence parts from second to 24-th nucleotide and changing the input size to (4, 23, 2) instead of (4, 26, 2). A 2D visualization of GuideHOM representations allows for intuitive design for gene editing and modulation of gene expression experiments. The representation computed by the Hit-or-Miss layer can be interpreted as either coordinates of a gRNA in the space of all possible gRNAs or a gRNA/target pair in the space of all possible gRNA/target pairs depending on the model (the former is for RNN and 1D-CNN models, and the latter is for 2D-CNN models used for the Cas9 and Cas12a gRNA-target pair sets). Due to the gradient descent optimization, the sequences in such a space are arranged according to sequences and cleavage efficiencies (similar sequences are expected to occupy a compact subspace of the guide space). The dimensionality of such a guide space is determined by the number and output dimensionality of the HOM capsule layers.

2D visualization of the guide space provides actionable insights into the variety of functional gRNAs available for a gene of interest. Figure 6-2 presents an example of the guide space for LOC440792 gene. The representation of the guide space (Figure 6-2A) as a scatter plot with mean cleavage efficiency denoted as the point color is more visually appealing and intuitive than spreadsheets that are commonly used for the same purpose. On the scatter plot, the most efficient sequences are clustered together with sub clusters formed according to different sequence determinants of cleavage efficiency (Figure 6-2B). An example of a sequence motif associated with

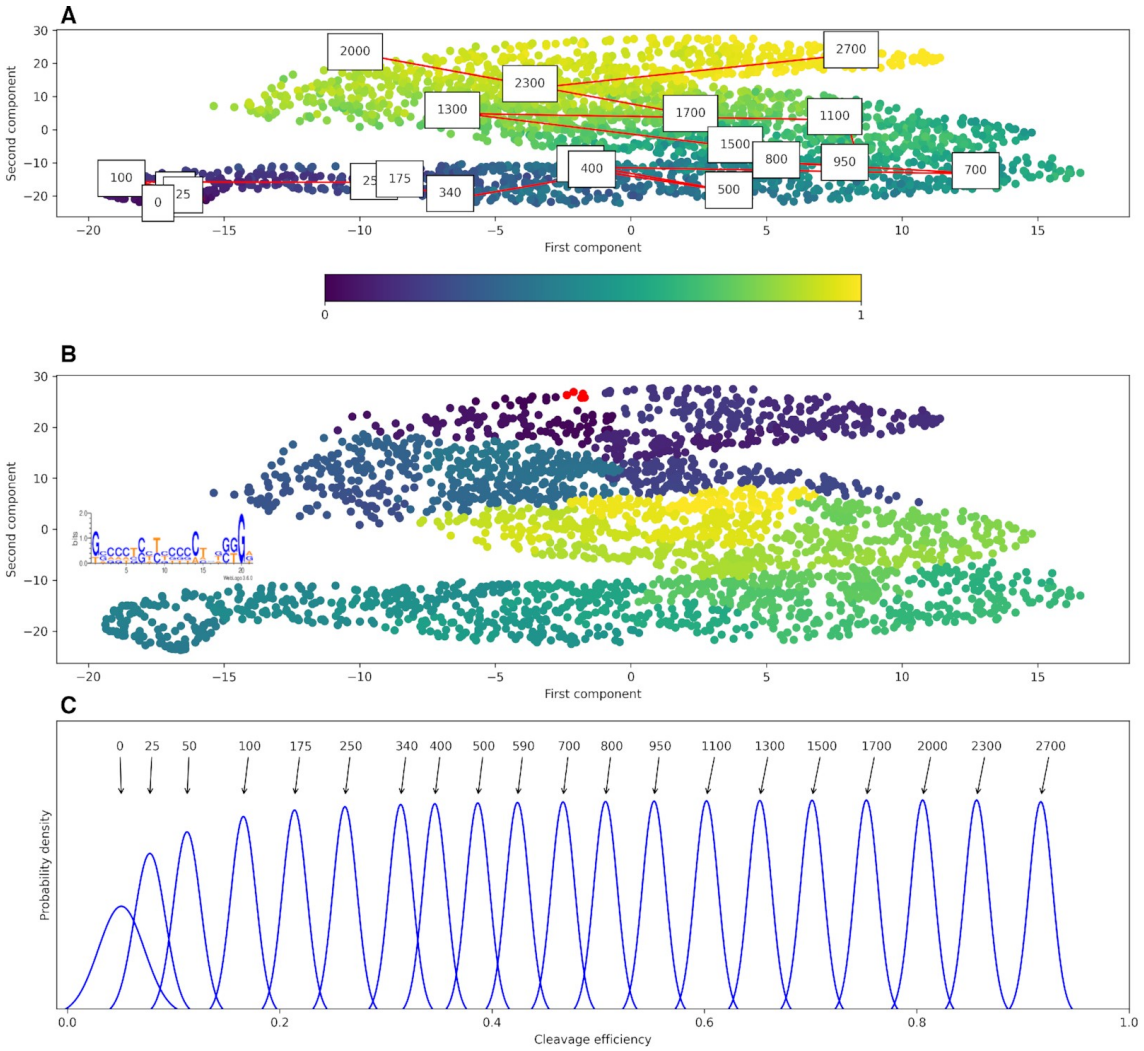

Figure 6-2: 2D visualization of the guide space with NCVis. The active and inactive gRNAs scatter around the plane according to their mean cleavage efficiency **(A)** and cluster into small subgroups based on the sequence determinants of cleavage efficiency **(B)**. The color gradient at **(A)** denotes mean efficiency (yellow – the largest, purple – the smallest). Color of the dots denotes the cluster label at **(B)**. One of the groups is shown as red dots and the sequence logo of that cluster is shown in the inset of **(B)**. A path along the gradient of color (red lines in panel **(A)** gives a functional set of gRNAs sufficient for modulation of gene expression **(C)**. The numbers at the plot **(A)** and at the plot **(C)** denote the same gRNAs. The gRNA sequences are shown in the Supplementary Table 4 of [1]. For this figure, the DeepHF Cas9 C E wildtype model was used.

one such cluster is shown in Figure 6-2B. The sequence determinants are specific for each cluster, which implies no simple linear dependency of cleavage efficiency and/or the cleavage efficiency variance on the sequence motifs. While efficient gRNAs share some common features, such as high GC content, there is no single unifying mo-

tif. Instead, any nucleotide change increases, decreases or, in some cases, does not perceptibly affect the efficiency. Visual arrangement allows for quick search for most efficient gRNA or a functional gRNA sets. Different functional sets of gRNAs can be found by following the color gradient from the most efficient to the least efficient (from purple to yellow dots in Figure 6-2A). An example of this path is shown in Figure 6-2A (numbers in boxes denote the number of gRNAs in the gRNA set, the same numbers correspond to cleavage efficiency distributions in Figure 6-2C). The set of gRNAs from Figure 6-2C found by following the color gradient of the gRNA space visualization, that has the required properties - gRNAs with all range of cleavage efficiency values, can be used to study the metabolic pathway responsible for hyperprolinemia (since the LOC440792 is associated with it, https://www.genecards.org/cgi-bin/carddisp.pl?gene=LOC440792) by performing *in vitro* experiments with each of the gRNAs and measuring the levels of proline. All sequences from this set are presented in the Supplementary Table 4 of [1]. All logo sequences for the identified clusters are presented in Zenodo repository.

The clustering and cleavage efficiency gradients in the 2D visualization of the guide space are apparently agnostic to the method of visualization. We experimented with PCA (Figure 6-3A), UMAP [175] (Figure 6-3B) and NCVis [176] (Figure 6-3C), and the results are qualitatively the same: each visualization method yields partitioning of the gRNAs into motif-dependent subclusters, and the color gradient is sufficient for delineating a functional set of gRNAs.

For all visualizations in Figures 6-2 and 6-3 (except for D and E where we use test set of Cas9 off-target dataset [5]), we use the LOC440792 gene (2781 gRNAs) and the GuideHOM DeepHF wildtype C E+M model. The visualizations are not supervised with respect to either the average cleavage efficiency or the standard deviation, therefore representations encoded in the outputs of capsule network are enough to produce the scattering of gRNAs on the plot by color, which shows once again that the model has learned the representations related to cleavage efficiency.

In addition to reproducing the training routines from previous studies, we conducted a 10-fold cross-validation analysis. Most of the previous studies we based our work on did not perform 10-fold cross validation, so that we cannot compare our

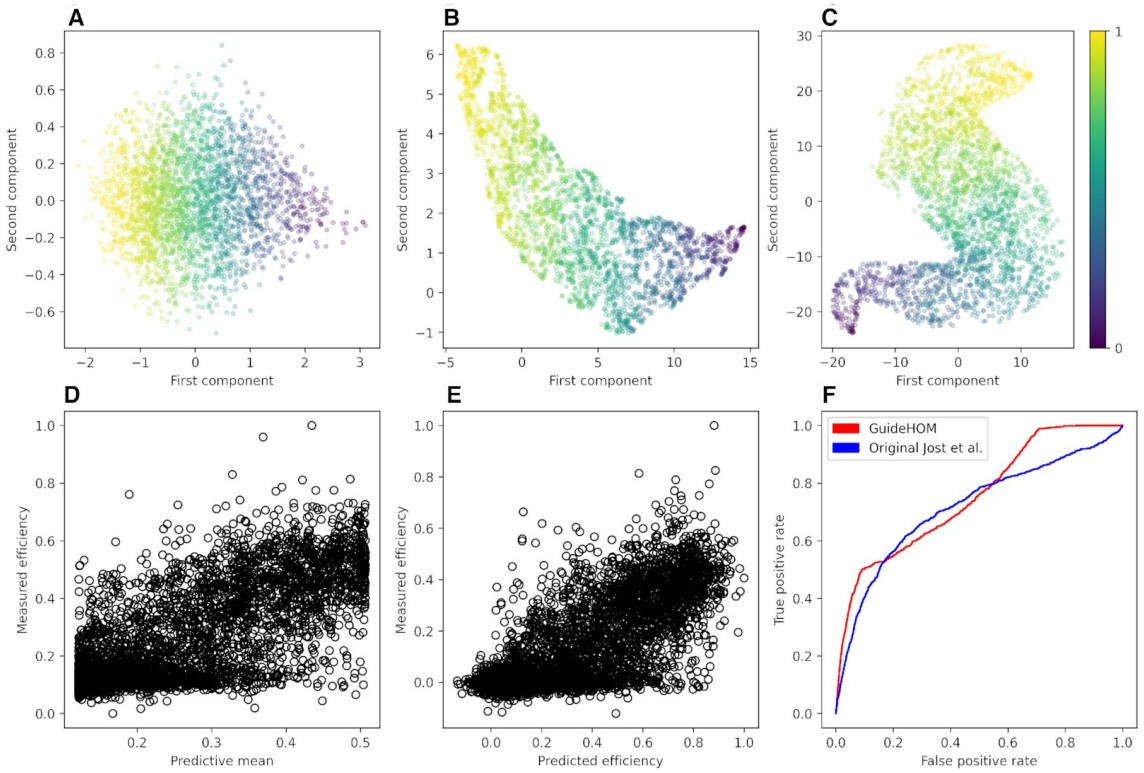

Figure 6-3: 2D visualization of the guide space with different visualization methods: **(A)** PCA, **(B)** UMAP, **(C)** NCVis. All views illustrate the same properties. Predictions vs observed activities for **(D)** GuideHOM and **(E)** Jost et al. models. **(F)** ROC curves of the GuideHOM and the Jost et al. model [5] on the dataset of Peng et al. [100]. For this figure, the DeepHF Cas9 C E wildtype model was used.

performance with that reported in these studies without reproducing them in their entirety, which is outside of the scope of this work. The cross validation results are available in Supplementary table 7 of [1]. Under the 10-fold cross validation scenario, the values of quality metrics tend to be smaller than in the hold-out dataset case by about 0.05 (e.g. DeepHF wildtype CNN ELBO reaches 0.8392 for the hold-out and 0.8066 on average in 10-fold CV, with standard deviation of 0.0216). The largest gap was observed for the Cas12a off-target model, which, in the10-fold CV, on average, does not yield acceptable confidence intervals with respect to p_{68}, p_{95} and $p_{99.7}$. This could be explained by the small size of the dataset, which only includes 1597 gRNA-target pairs, less than half of the next smallest dataset, geCRISPR, with 3619 gRNAs. For the rest of the models, the confidence intervals, on average, remain acceptable. There is also a substantial gap in the performance between the

RNN and CNN-based models for the geCRISPR dataset, with RNN ELBO and RNN ELBO+MSE performing worse than the CNN counterparts (Pearson correlation 0.3882 and 0.4403 versus 0.5946 and 0.5926). The remaining models do not yield significant differences in performance either between RNN and CNN or between ELBO and ELBO+MSE. Overall, the cross validation study shows that the GuideHOM architecture is robust to overfitting provided there is enough data to train it on.

6.2 Explainable machine learning demonstrates the sequential preferences for on-target and off-target cleavage

We use the Accumulated Local Effects (ALE [132]) to explain predictions of trained models. A single ALE value shows the influence of a feature towards the output of the network. In case of one-hot encoded features, it shows the impact of presence and absence of a feature. ALE is a black-box explanation method that requires very few assumptions about the model, so it is perfect for our case, with model that predicts not only the cleavage efficiency, but also its variance. We compute the influences for all nucleotides towards mean efficiency and variance, plot the heatmaps and logo sequences of the resulting matrices. The logo sequences and heatmaps show the preferences for gRNA sequence in on-target cleavage (Figure 6-4) and gRNA-target pair in off-target cleavage (Supplementary Figure 1 for Cas9 and Supplementary Figure 2 for Cas12a from [1]). The region of about 4-5 nt located near the PAM is more important for the prediction than the rest of the sequence. This holds for both Cas9 and Cas12a – for Cas9 it is located on the 3' end, and for Cas12a it is located on 5' end. The mechanics of PAM and target recognition implies the importance of the seed region. The recognition of target starts from PAM and goes towards the end of the target through the seed region. If there is a mismatch in the seed region, the cleavage is highly unlikely. Our model captures the importance of seed region without any additional supervision from the user. We didn't indicate it in

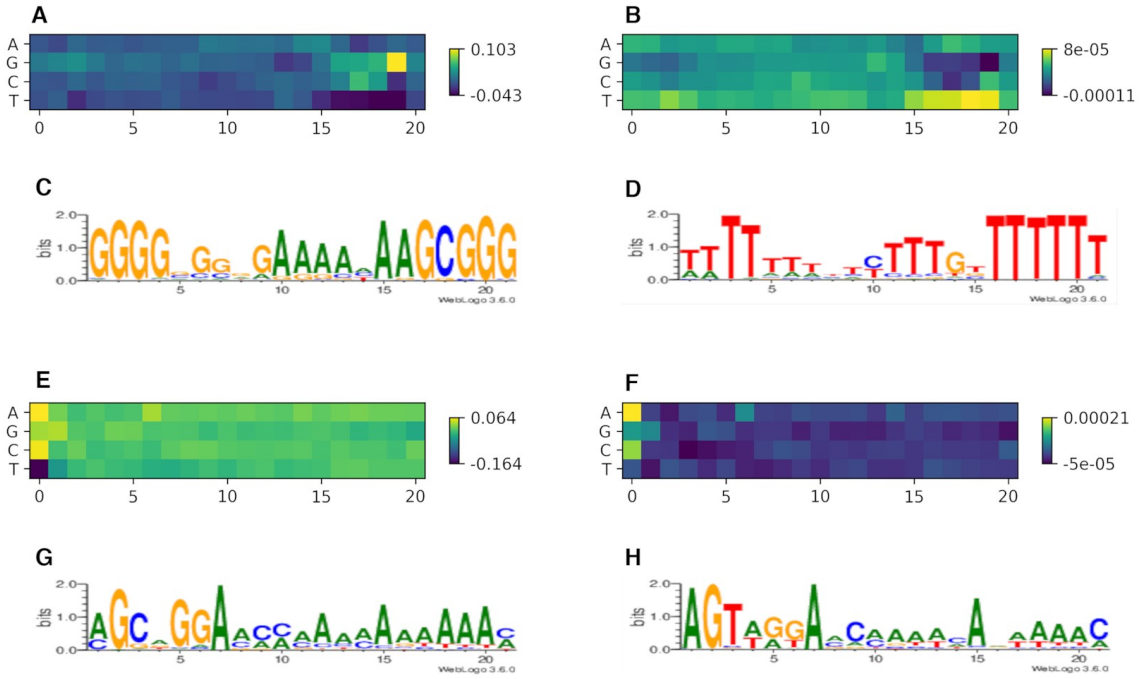

Figure 6-4: Explanations for Cas9 and Cas12a on-target models shows the importance of seed region which is located at the right hand side for Cas9 **(A)** and **(C)**, and at the left hand side for Cas12a **(E)** and **(G)**, for the prediction of cleavage efficiency. The same importance is observed for the variance of predictions **(B)** and **(D)** for Cas9, **(F)** and **(H)** for Cas12a). The models used are DeepHF WT C E and DeepCpf1 C E.

any way (a possible way to indicate would be to give the network a mask in addition to the sequence, which we didn't do), the model had to learn it on its own to use in cleavage efficiency prediction.

Importance heatmaps for efficiency and variance tend to mirror each other, not exactly, but very close. For example, in Cas9 (Figure 6-4A and B), the Gs and Cs in the seed region are very important for efficiency, but for variance, Ts and As are more important. Basically, if a gRNA has Cs and Gs in the seed region, the variance will be low, but the cleavage efficiency will be rather high. However, this holds poorly for Cas12a. In Figure 6-4E and F, in two first letters of the seed, both variance and efficiency matrices value As and Gs, and undervalues Ts. The rest of the seed and the rest of the sequence after the seed have the same overall importance. For mismatching gRNA-target pairs (Supplementary Figure 1 of [1]) in Cas9, the mismatches in PAM are very important, as important as in the seed

region. The lack of importance for Ts in the seed region for Cas12a is also reproduced in Cas12a off-target model despite using different datasets (Supplementary Figure 2 of [1]). The same holds for Gs and Ts in seed region of Cas9 (Supplementary Figure 1 of [1]). It shows that models learn the same features from different datasets of the same Cas effector – those features are not dataset-specific, but Cas effector-specific.

Chapter 7

Uncertainty Quantification highlights the hidden diversity of off-target events

This chapter is based on one of the papers that constitute the current dissertation – "Uncertainty-aware and interpretable evaluation of Cas9–gRNA and Cas12a–gRNA specificity for fully matched and partially mismatched targets with Deep Kernel Learning" [1] published in Nucleic Acids Research. The contribution of authors are as follows: E.S., S.A.S. and I developed the concepts and designed the study. E.V.K., M.P., A.Y.O., S.A.S. and K.V.S. supervised the research. I have designed the methods and performed all analyses. E.S., S.A.S. and I wrote the manuscript, which was read, edited and approved by all authors.

7.1 GuideHOM solves off-target cleavage regression with acceptable confidence intervals

Our approach allows not only for the prediction of the on-target cleavage efficiency based on a single gRNA but also, with minimal changes to the architecture (see "Materials and Methods"), for the prediction of off-target cleavage efficiency from a pair of a gRNA and the target. For the Cas9 gRNA-target pair dataset [5], we

train the model on 80% of the training set and test it on the remaining 20%. We found an r^2 value of 0.625, as compared with 0.617 in the original study, and an acceptable confidence interval (0.7176 on $Y_{mean}+\sigma$, 0.9124 on $Y_{mean}+2\sigma$, 0.9804 on $Y_{mean}+3\sigma$). The prediction plots for the test set are shown in Figure 6-3D and E. The original Cas9 gRNA-target pair model [5] is obtained by slightly modifying the supplementary file in Jupyter Notebook. The sgRNA and genome target sequences are trimmed to remove the first two and one last nucleotides (the flanks) in order to leave only the gRNA and the PAM. The parameter input_shape in the model definition is changed to $(4, 23, 2)$.

Table 7.1: Comparison of the results for the subset from [5] study.

Metric	Jost et al. E	Jost et al. E+M
ρ_{68}	0.7222	0.7125
ρ_{95}	0.9208	0.9155
$\rho_{99.7}$	0.9838	0.9828
PCC	0.7849	0.7805
SCC	0.7003	0.6982
r^2	0.6161	0.6192

The model trained on the Cas9 gRNA-target pair dataset provides Cas9 cleavage efficiency estimates for guide and target pairs with small numbers of mismatches between the guide and the target. This information can be used to solve the off-target classification task using the Peng et al. [100] classification dataset to test our models. We compared the performance of our models with that of the reproduced Jost et al. model and obtain a better AUROC of 0.7528 (the Jost et al. model scored 0.6777). The ROC curves for these models are shown in Figure 6-3F. For the test, we removed from the Peng et al. dataset all pairs that have more than 6 mismatches to reduce the number of negatives. According to the ROC curves, our model tends to produce fewer false positives for low decision thresholds. We next trained a regression model for Cas12a gRNA-target pair dataset. We split the dataset of [133] into 90% for training and 10% for testing. We obtain Spearman Correlation Coefficient of 0.6 and a borderline acceptable confidence interval (0.58 on σ, 0.86 on 2σ, 0.95 on 3σ). The variability and lower correlation are likely due to the small dataset size of only 1565 gRNA-target pairs. We did not use the PAM

information because all sequences in the dataset contained the TTTA PAM but use additional flank region so the overall input length is 23. The limitations of the Cas12a gRNA-target pair dataset are also reflected in the corresponding learning curves (Figure 4-2A and B). The learning curve and mean standard deviations converge towards 0.6 and 0.2, respectively, and are comparable to the results obtained with other small samples, such as the DeepCpf1 set, but the correlation (Spearman Correlation Coefficient) is weaker than that for DeepHF. To our knowledge, this is the first attempt at off-target cleavage efficiency regression for asCas12a and the performance of the model is promising.

7.2 Diversity of CRISPR off-target effect predictions demonstrated by analysis of gRNA-target pairs

We analyze an empirically validated sgRNA library for 2400 genes that are essential for robust cell growth [5]. The consistency and slight but significant superiority of results obtained with our method compared to those in the original study (r^2 value of 0.625 versus 0.617 for the Jost et al. model, as shown in Section 7.1) supports the utility of our approach. As an example of a practical application, we provide the top 5 gRNAs with mean and variance values for each gene in the Homo sapiens reference genome (hg38) chromosome 22. An example of the output for gene SERPIND1 is shown in Table 7.2. We suppose that the gRNAs could be used right away to plan the CRISPR/Cas9 gene editing experiment with the gene SERPIND1.

Table 7.2: Example of top 5 gRNAs for Cas9. The model is DeepHF WT R E.

Start	Sequence	Strand	Mean	Variance
5273	GGATCAGCTAGAGAAAGGAGGGG	+	0.9344	0.0122
12819	CAGCGGCATGAACCCCACCGTGG	-	0.9310	0.0120
10683	TCATGGCAGAAAGAATGGAGAGG	+	0.9230	0.0119
7403	GTGTGTGGACAGATCAGGAGGGG	-	0.9264	0.0119
4807	AGAGACAAAGTTCCCACCAGGGG	-	0.9260	0.0119

Cas-OFFinder [101] was used to make a list of potential targets. A detailed

description of the pipeline used for human genome analysis is presented in "Materials and Methods". To apply our approach for prediction of mismatched gRNA cleavage efficiency and search of sgRNAs with systematically modulated activities, we analyze 1000 random gRNAs which are extracted from the top 10 highly efficient gRNAs (see Supplementary Table 5 of [1] for the 1000 extracted). For each of these gRNAs, all possible off-targets with no more than 6 mismatches are selected from human chromosome 22. The mean cleavage efficiency and cleavage efficiency variance are computed using the GuideHOM model trained on the Cas9 gRNA-target dataset (CNN ELBO) for each identified off-target gRNA-target pair.

The inclusion of uncertainty estimates in the off-target analysis provides for two orthogonal axes that characterize off-target properties: cleavage efficiency and stability of prediction, where cleavage efficiency shows how probable is the cleavage of the DNA strand with this particular target and this particular gRNA, and prediction variance shows how robust the prediction of cleavage efficiency is. Figure 7-1A shows that most of the predicted off-targets (74%) have low cleavage efficiency (less than 0.15) but there is a minority of highly efficient off-targets (26%) that should be avoided for any application that depends on the minimization of off-target effects. Most off-targets (82%) have small variance (less than 0.015, see Figure 7-1B, they differ from the predicted mean cleavage efficiency only by 0.1223 at worst, with the probability of 0.65), so there is a negligible chance that these off-targets are incorrectly predicted. However, a minority of the off-targets with large variance (18%) should be removed from experimental validation because gRNAs with such cleavage efficiency variance on off-targets can be problematic in the experiments (the real cleavage efficiencies of these gRNAs can differ from the predicted mean so much that one can not be sure in the quality of the cleavage efficiency prediction even with the confidence intervals taken into account).

Most off-targets with cleavage efficiency variance less than 0.015 and mean cleavage efficiency less than 0.15 are located in the left bottom corner of the plot on Figure 7-1C. For gene editing, only the gRNAs with the smallest number of off-targets with high prediction variance and the highest on-target cleavage efficiency should be selected. For gene expression modulation, a set of gRNAs that produce

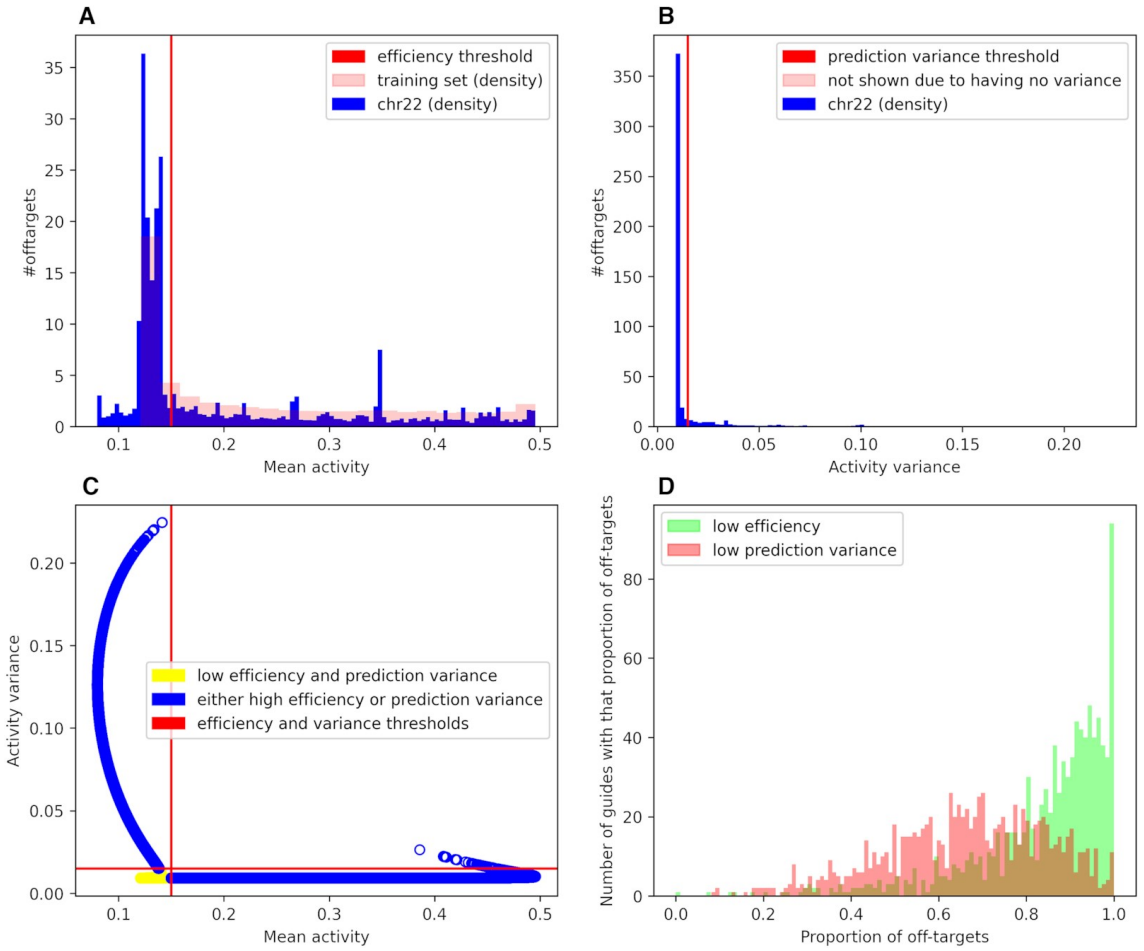

Figure 7-1: Analysis of off-targets in Chromosome 22 shows the existence of different off-target categories: **(A)** distribution of mean off-target cleavage efficiency for real known values extracted from the training set (pink) and for predictions for Chromosome 22 (blue) with cleavage efficiency threshold of 0.15. **(B)** distribution of off-target cleavage efficiency variance for Chromosome 22 with prediction variance threshold of 0.015. **(C)** Mean vs Variance plot for off-targets from gRNAs extracted from top 10 of different Chromosome 22 genes with indicated cleavage efficiency and prediction variance thresholds. **(D)** Distributions for proportions of off-targets with low efficiency (green) and off-targets with low prediction variance (pink) for 1000 of top10 gRNAs.

off-targets with chosen levels of prediction variance can be used. In this case, the low variance of the off-target effect allows the selection of a suitable set of gRNAs with predictable results. For each of 1000 randomly selected highly efficient gRNA, the proportions of the different kinds of off-targets are shown in the Supplementary table 6 of [1]. Figure 7-1D shows that most gRNAs have largely off-targets with both low cleavage efficiency and low prediction variance (the mean fraction of

off-targets with low cleavage efficiency is 0.80, with the standard deviation of 0.13, and the mean fraction of off-targets with low prediction variance was 0.62, with the standard deviation of 0.17). However, for some gRNAs, many off-targets with high cleavage efficiency and high prediction variance were identified (on average, the proportion of both off-targets with high cleavage efficiency and low prediction variance and off-targets with low cleavage efficiency and high prediction variance is 58% with 14% standard deviation, and there is an insignificant amount of the off-targets that both have high cleavage efficiency and high prediction variance, only 23 gRNA out of 1000 have them, there are only 56 such off-targets out of 1994178 total. Such properties would exclude the gRNAs from the candidate pool for an experiment. The proportion of off-targets with low cleavage efficiency and low prediction variance can be used to select gRNA for different types of experiments because the off-targets with low cleavage efficiency and low prediction variance are distributed differently for different gRNAs. Thus, the results show that our approach is useful for the prediction of mismatched gRNA cleavage efficiency and evaluation of systematically attenuated gRNAs that can be used to control gene expression, from tuning biochemical pathways to identifying suppressors for diseases and stress conditions.

Chapter 8

Discussion and conclusions

This dissertation focuses on the application of advanced machine learning techniques, including uncertainty quantification and neural network interpretation, to study CRISPR-Cas behavior. The primary objective of this research is to explore how the knowledge of prediction error and dependencies of the model output on its input can help address questions regarding the behavior of the modeled biological system. The motivation behind this approach is not only to enhance the accuracy of predictive modeling and facilitate better decision-making during experiment design but also to understand the extent to which advanced methods can extract information from a crude data-driven model of a biological process.

Advanced machine learning methods that transcend simple classification and regression setups are becoming increasingly prevalent in biology. These methods have been utilized in various contexts, such as genomics, drug discovery, and protein function prediction. As the size of biological datasets and the complexity of experimental setups increase, there is an increasing need for more sophisticated statistical methods. Consequently, machine learning has emerged as a crucial tool for modern biologists, with each machine learning method providing a trainable mathematical model for a biological system of interest.

However, the cognitive approaches of machine learning specialists and biologists differ significantly, encompassing dissimilarities in setting research goals, formulating hypotheses, and designing studies. Modern machine learning focuses on achieving state-of-the-art performance on benchmark datasets, adhering to stringently defined

tasks and performance metrics. The ultimate objective is to devise a method capable of extracting the most information from unstructured data. An interesting caveat is that models may learn the data noise specific to the training set, rather than a useful signal, and from a purely machine learning perspective, there is no straightforward means of escaping this issue. In contrast, a biologist's perspective prioritizes converging different methods towards a common output, rather than solely pursuing state-of-the-art performance for its own sake. A biologist is more concerned with addressing a wide range of biological questions, rather than merely chasing the highest performance benchmarks.

One notable outcome of this dissertation was the unsupervised rediscovery of seed region importance in CRISPR-Cas interference. This result exhibited conservation across different datasets and was distinct for both Cas effector classes employed in the study. Such rediscoveries serve as valuable means to verify the validity of a machine learning model, and the pursuit of rediscovering known behavior for various tasks can establish a robust meta-approach for building trainable models of biological systems, thereby mitigating potential pitfalls in the process. A promising direction for future research involves identifying potential rediscovery targets when encountering new problems, shifting from explainable to verifiable machine learning.

While state-of-the-art results may be beneficial, the primary interest for a computational biologist in a model lies not only in its ability to perform the intended task but also in its capacity to provide information about the behavior aspect closely related to the intended task, even if it is not present in the training set. Another significant result of this dissertation is the revelation of off-target event diversity. The number of potential off-target sites for a gRNA is known to be insufficient for selecting an optimal gRNA. The introduction of an additional axis, prediction variance, offers a new method for filtering promising candidates.

Designing gene editing experiments with improved on-target efficiency and minimized off-target effects relies on deep and intricate understanding of CRISPR-Cas system behavior. The dissertation aimed to study CRISPR-Cas mechanics using Neural Network Interpretation and Uncertainty Quantification while building tools to facilitate experimental design around this understanding. There are two projects

that constitute this dissertation:

1. Off-target cleavage event recognition based on capsule networks with dynamic routing by agreement with anomaly detection using internal inequality analysis similar to economical studies of income inequality;

2. Deep Kernel Learning-based methods for estimating confidence intervals in gRNA-target cleavage efficiency

The main conclusions of these projects are as follows:

1. Deep Kernel Learning-based methods for gRNA-target cleavage efficiency estimation were introduced. DKL offers advantages over basic deep learning in CRISPR-mediated target cleavage efficiency estimation because it includes explicit modeling of prediction uncertainty which is used to improve gene editing design by supplying the researcher with an additional way to select the best gRNAs for the experiment;

2. Uncertainty quantification with Deep Kernel Learning has helped to highlight a hidden diversity of off-target events in *Homo sapiens*. Four types of off-targets, previously unknown, were characterized and guidelines for using this knowledge for gene experiment design were formed;

3. Using Explainable Machine Learning methods (Accumulated Local Explanations), known behavior of Cas proteins, their dependence on the seed region in the target DNA, was independently rediscovered;

4. Several effector- and cell line-specific models based on publicly available datasets was constructed, including AsCas12a and SpCas9 as well as several high fidelity orthologs (SpCas9 HF-1, eSpCas9), with focus on human cell lines (HeLa, HL60, Hek293t);

5. A set of novel anomaly detection methods presented in this dissertation extended the previous works on supervised anomaly detection with the parallels between economic inequality and inequality in response to rare and familiar examples within the internal mechanisms of capsule networks. This lead to the

development of a novel method to detect off-target events in CRISPR-Cas9 gene editing.

Overall, the methods introduced in this dissertations can aid molecular biologists and bioinformaticians in understanding complex mechanics of CRISPR-Cas systems, detection and analysis of rare events in biological processes, and design of more efficient gene editing experiments.

Bibliography

[1] Bogdan Kirillov, Ekaterina Savitskaya, Maxim Panov, Aleksey Y Ogurtsov, Svetlana A Shabalina, Eugene V Koonin, and Konstantin V Severinov. Uncertainty-aware and interpretable evaluation of cas9–grna and cas12a–grna specificity for fully matched and partially mismatched targets with deep kernel learning. *Nucleic acids research*, 50(2):e11–e11, 2022.

[2] Bogdan Kirillov and Maxim Panov. Measuring internal inequality in capsule networks for supervised anomaly detection. *Scientific Reports*, 12(1):13575, 2022.

[3] Yongmoon Jeon, You Hee Choi, Yunsu Jang, Jihyeon Yu, Jiyoung Goo, Gyejun Lee, You Kyeong Jeong, Seung Hwan Lee, In-San Kim, Jin-Soo Kim, et al. Direct observation of dna target searching and cleavage by crispr-cas12a. *Nature communications*, 9(1):2777, 2018.

[4] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[5] Marco Jost, Daniel A Santos, Reuben A Saunders, Max A Horlbeck, John S Hawkins, Sonia M Scaria, Thomas M Norman, Jeffrey A Hussmann, Christina R Liem, Carol A Gross, et al. Titrating gene expression using libraries of systematically attenuated crispr guide rnas. *Nature biotechnology*, 38(3):355–364, 2020.

[6] Yunkun Liu, Weixin Tao, Shishi Wen, Zhengyuan Li, Anna Yang, Zixin Deng, and Yuhui Sun. In vitro crispr/cas9 system for efficient targeted dna editing. *MBio*, 6(6):e01714–15, 2015.

[7] Yamin Li, Zachary Glass, Mingqian Huang, Zheng-Yi Chen, and Qiaobing Xu. Ex vivo cell-based crispr/cas9 genome editing for therapeutic applications. *Biomaterials*, 234:119711, 2020.

[8] Lukas E Dow, Jonathan Fisher, Kevin P O'rourke, Ashlesha Muley, Edward R Kastenhuber, Geulah Livshits, Darjus F Tschaharganeh, Nicholas D Socci, and Scott W Lowe. Inducible in vivo genome editing with crispr-cas9. *Nature biotechnology*, 33(4):390–394, 2015.

[9] Jiongyu Zhang and Changchun Liu. Crispr-powered dna computing and digital display. *ACS Synthetic Biology*, 10(11):3148–3157, 2021.

[10] Nafisa N Nazipova and Svetlana A Shabalina. Understanding off-target effects through hybridization kinetics and thermodynamics, 2020.

[11] Namita Bisaria, Inga Jarmoskaite, and Daniel Herschlag. Lessons from enzyme kinetics reveal specificity principles for rna-guided nucleases in rna interference and crispr-based genome editing. *Cell systems*, 4(1):21–29, 2017.

[12] Daesik Kim, Kevin Luk, Scot A Wolfe, and Jin-Soo Kim. Evaluating and enhancing target specificity of gene-editing nucleases and deaminases. *Annual review of biochemistry*, 88:191–220, 2019.

[13] Vasileios Konstantakos, Anastasios Nentidis, Anastasia Krithara, and Georgios Paliouras. Crispr–cas9 grna efficiency prediction: an overview of predictive tools and the role of deep learning. *Nucleic Acids Research*, 50(7):3616–3637, 2022.

[14] Hui Kwon Kim, Seonwoo Min, Myungjae Song, Soobin Jung, Jae Woo Choi, Younggwang Kim, Sangeun Lee, Sungroh Yoon, and Hyongbum Henry Kim. Deep learning improves prediction of crispr–cpf1 guide rna activity. *Nature biotechnology*, 36(3):239, 2018.

[15] Kristopher Torp Jensen, Lasse Fløe, Trine Skov Petersen, Jinrong Huang, Fengping Xu, Lars Bolund, Yonglun Luo, and Lin Lin. Chromatin accessibility and guide sequence secondary structure affect crispr-cas9 gene editing efficiency. *FEBS letters*, 591(13):1892–1901, 2017.

[16] Sebald AN Verkuijl and Marianne G Rots. The influence of eukaryotic chromatin state on crispr–cas9 editing efficiencies. *Current opinion in biotechnology*, 55:68–73, 2019.

[17] Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. Explainable machine learning for scientific insights and discoveries. *Ieee Access*, 8:42200–42216, 2020.

[18] HM Dipu Kabir, Abbas Khosravi, Mohammad Anwar Hosen, and Saeid Nahavandi. Neural network-based uncertainty quantification: A survey of methodologies and applications. *IEEE access*, 6:36218–36234, 2018.

[19] Frank Hille, Hagen Richter, Shi Pey Wong, Majda Bratovič, Sarah Ressel, and Emmanuelle Charpentier. The biology of crispr-cas: backward and forward. *Cell*, 172(6):1239–1259, 2018.

[20] Kirill A Datsenko, Ksenia Pougach, Anton Tikhonov, Barry L Wanner, Konstantin Severinov, and Ekaterina Semenova. Molecular memory of prior infections activates the crispr/cas adaptive bacterial immunity system. *Nature communications*, 3(1):945, 2012.

[21] Anaïs Le Rhun, Andrés Escalera-Maurer, Majda Bratovič, and Emmanuelle Charpentier. Crispr-cas in streptococcus pyogenes. *RNA biology*, 16(4):380–389, 2019.

[22] Kira S Makarova, Daniel H Haft, Rodolphe Barrangou, Stan JJ Brouns, Emmanuelle Charpentier, Philippe Horvath, Sylvain Moineau, Francisco JM Mojica, Yuri I Wolf, Alexander F Yakunin, et al. Evolution and classification of the crispr–cas systems. *Nature Reviews Microbiology*, 9(6):467–477, 2011.

[23] Krzysztof Chylinski, Kira S Makarova, Emmanuelle Charpentier, and Eugene V Koonin. Classification and evolution of type ii crispr-cas systems. *Nucleic acids research*, 42(10):6091–6105, 2014.

[24] Eugene V Koonin, Kira S Makarova, and Feng Zhang. Diversity, classification and evolution of crispr-cas systems. *Current opinion in microbiology*, 37:67–78, 2017.

[25] Kira S Makarova, Yuri I Wolf, Omer S Alkhnbashi, Fabrizio Costa, Shiraz A Shah, Sita J Saunders, Rodolphe Barrangou, Stan JJ Brouns, Emmanuelle Charpentier, Daniel H Haft, et al. An updated evolutionary classification of crispr–cas systems. *Nature Reviews Microbiology*, 13(11):722–736, 2015.

[26] Kira S Makarova, Yuri I Wolf, Jaime Iranzo, Sergey A Shmakov, Omer S Alkhnbashi, Stan JJ Brouns, Emmanuelle Charpentier, David Cheng, Daniel H Haft, Philippe Horvath, et al. Evolutionary classification of crispr–cas systems: a burst of class 2 and derived variants. *Nature Reviews Microbiology*, 18(2):67–83, 2020.

[27] Emmanuelle Charpentier, Hagen Richter, John van der Oost, and Malcolm F White. Biogenesis pathways of rna guides in archaeal and bacterial crispr-cas adaptive immunity. *FEMS microbiology reviews*, 39(3):428–441, 2015.

[28] Sergey A Shmakov, Yuri I Wolf, Ekaterina Savitskaya, Konstantin V Severinov, and Eugene V Koonin. Mapping crispr spaceromes reveals vast host-specific viromes of prokaryotes. *Communications biology*, 3(1):321, 2020.

[29] Jake L Weissman, Arlin Stoltzfus, Edze R Westra, and Philip LF Johnson. Avoidance of self during crispr immunization. *Trends in microbiology*, 28(7):543–553, 2020.

[30] Yan Zhang, Jinzhong Lin, Xuhui Tian, Yuan Wang, Ruiliang Zhao, Chenwei Wu, Xiaoning Wang, Pengpeng Zhao, Xiaonan Bi, Zhenxiao Yu, et al. Inactivation of target rna cleavage of a iii-b crispr-cas system induces robust autoimmunity in saccharolobus islandicus. *International Journal of Molecular Sciences*, 23(15):8515, 2022.

[31] Xiaohan Guo, Mariana Sanchez-Londono, José Vicente Gomes-Filho, Rogelio Hernandez-Tamayo, Selina Rust, Leah M Immelmann, Pascal Schäfer, Julia Wiegel, Peter L Graumann, and Lennart Randau. Characterization of the self-targeting type iv crispr interference system in pseudomonas oleovorans. *Nature Microbiology*, 7(11):1870–1878, 2022.

[32] Rachael E Workman, Teja Pammi, Binh TK Nguyen, Leonardo W Graeff, Erika Smith, Suzanne M Sebald, Marie J Stoltzfus, Chad W Euler, and Joshua W Modell. A natural single-guide rna repurposes cas9 to autoregulate crispr-cas expression. *Cell*, 184(3):675–688, 2021.

[33] Adi Stern, Leeat Keren, Omri Wurtzel, Gil Amitai, and Rotem Sorek. Self-targeting by crispr: gene regulation or autoimmunity? *Trends in genetics*, 26(8):335–340, 2010.

[34] Franziska Wimmer and Chase L Beisel. Crispr-cas systems and the paradox of self-targeting spacers. *Frontiers in microbiology*, 10:3078, 2020.

[35] Joseph E Peters, Kira S Makarova, Sergey Shmakov, and Eugene V Koonin. Recruitment of crispr-cas systems by tn7-like transposons. *Proceedings of the National Academy of Sciences*, 114(35):E7358–E7366, 2017.

[36] Detlef D Leipe, Eugene V Koonin, and Lakshminarayanan Aravind. Stand, a class of p-loop ntpases including animal and plant regulators of programmed cell death: multiple, complex domain architectures, unusual phyletic patterns, and evolution by horizontal gene transfer. *Journal of molecular biology*, 343(1):1–28, 2004.

[37] Eugene V Koonin. Crispr: a new principle of genome engineering linked to conceptual shifts in evolutionary biology. *Biology & Philosophy*, 34(1):9, 2019.

[38] Sofia Medvedeva, Ying Liu, Eugene V Koonin, Konstantin Severinov, David Prangishvili, and Mart Krupovic. Virus-borne mini-crispr arrays are involved in interviral conflicts. *Nature communications*, 10(1):5204, 2019.

[39] Eugene V Koonin, Kira S Makarova, and Yuri I Wolf. Evolutionary genomics of defense systems in archaea and bacteria. *Annual review of microbiology*, 71:233–261, 2017.

[40] Guilhem Faure, Sergey A Shmakov, Winston X Yan, David R Cheng, David A Scott, Joseph E Peters, Kira S Makarova, and Eugene V Koonin. Crispr–cas in mobile genetic elements: counter-defence and beyond. *Nature Reviews Microbiology*, 17(8):513–525, 2019.

[41] Ece Topuzlu and C Martin Lawrence. Recognition of a pseudo-symmetric rna tetranucleotide by csx3, a new member of the crispr associated rossmann fold superfamily. *RNA biology*, 13(2):254–257, 2016.

[42] Shiraz A Shah, Omer S Alkhnbashi, Juliane Behler, Wenyuan Han, Qunxin She, Wolfgang R Hess, Roger A Garrett, and Rolf Backofen. Comprehensive search for accessory proteins encoded with archaeal and bacterial type iii crispr-cas gene cassettes reveals 39 new cas gene families. *RNA biology*, 16(4):530–542, 2019.

[43] Sergey A Shmakov, Kira S Makarova, Yuri I Wolf, Konstantin V Severinov, and Eugene V Koonin. Systematic prediction of genes functionally linked to

crispr-cas systems by gene neighborhood analysis. *Proceedings of the National Academy of Sciences*, 115(23):E5307–E5316, 2018.

[44] Aamir Mir, Alireza Edraki, Jooyoung Lee, and Erik J Sontheimer. Type ii-c crispr-cas9 biology, mechanism, and application. *ACS chemical biology*, 13(2):357–365, 2018.

[45] Yuyi Tang and Yan Fu. Class 2 crispr/cas: an expanding biotechnology toolbox for and beyond genome editing. *Cell & bioscience*, 8(1):1–13, 2018.

[46] Sydney Newsom, Hari Priya Parameshwaran, Lindsie Martin, and Rakhi Rajan. The crispr-cas mechanism for adaptive immunity and alternate bacterial functions fuels diverse biotechnologies. *Frontiers in cellular and infection microbiology*, 10:619763, 2021.

[47] Simon A Jackson, Rebecca E McKenzie, Robert D Fagerlund, Sebastian N Kieper, Peter C Fineran, and Stan JJ Brouns. Crispr-cas: adapting to change. *Science*, 356(6333):eaal5056, 2017.

[48] James K Nuñez, Philip J Kranzusch, Jonas Noeske, Addison V Wright, Christopher W Davies, and Jennifer A Doudna. Cas1–cas2 complex formation mediates spacer acquisition during crispr–cas adaptive immunity. *Nature structural & molecular biology*, 21(6):528–534, 2014.

[49] Marin Radovčić, Tom Killelea, Ekaterina Savitskaya, Lukas Wettstein, Edward L Bolt, and Ivana Ivančić-Baće. Crispr–cas adaptation in escherichia coli requires recbcd helicase but not nuclease activity, is independent of homologous recombination, and is antagonized by 5 ssdna exonucleases. *Nucleic acids research*, 46(19):10173–10183, 2018.

[50] Elena Kurilovich, Anna Shiriaeva, Anastasia Metlitskaya, Natalia Morozova, Ivana Ivancic-Bace, Konstantin Severinov, and Ekaterina Savitskaya. Genome maintenance proteins modulate autoimmunity mediated primed adaptation by the escherichia coli type ie crispr-cas system. *Genes*, 10(11):872, 2019.

[51] Anna A Shiriaeva, Konstantin Kuznedelov, Ivan Fedorov, Olga Musharova, Timofey Khvostikov, Yuliya Tsoy, Elena Kurilovich, Gerald R Smith, Ekaterina Semenova, and Konstantin Severinov. Host nucleases generate prespacers for primed adaptation in the e. coli type ie crispr-cas system. *Science Advances*, 8(47):eabn8650, 2022.

[52] Juliane Behler and Wolfgang R Hess. Approaches to study crispr rna biogenesis and the key players involved. *Methods*, 172:12–26, 2020.

[53] Rodolphe Barrangou, Christophe Fremaux, Hélène Deveau, Melissa Richards, Patrick Boyaval, Sylvain Moineau, Dennis A Romero, and Philippe Horvath. Crispr provides acquired resistance against viruses in prokaryotes. *Science*, 315(5819):1709–1712, 2007.

[54] Xuebing Wu, Andrea J Kriz, and Phillip A Sharp. Target specificity of the crispr-cas9 system. *Quantitative biology*, 2:59–70, 2014.

[55] Mengzhu Liu, Weiwei Zhang, Changchang Xin, Jianhang Yin, Yafang Shang, Chen Ai, Jiaxin Li, Fei-Long Meng, and Jiazhi Hu. Global detection of dna repair outcomes induced by crispr–cas9. *Nucleic acids research*, 49(15):8732–8742, 2021.

[56] Bernd Zetsche, Jonathan S Gootenberg, Omar O Abudayyeh, Ian M Slaymaker, Kira S Makarova, Patrick Essletzbichler, Sara E Volz, Julia Joung, John Van Der Oost, Aviv Regev, et al. Cpf1 is a single rna-guided endonuclease of a class 2 crispr-cas system. *Cell*, 163(3):759–771, 2015.

[57] Tina Y Liu and Jennifer A Doudna. Chemistry of class 1 crispr-cas effectors: binding, editing, and regulation. *Journal of Biological Chemistry*, 295(42):14473–14487, 2020.

[58] Samira Kiani, Alejandro Chavez, Marcelle Tuttle, Richard N Hall, Raj Chari, Dmitry Ter-Ovanesyan, Jason Qian, Benjamin W Pruitt, Jacob Beal, Suhani Vora, et al. Cas9 grna engineering for genome editing, activation and repression. *Nature methods*, 12(11):1051–1054, 2015.

[59] Stephanie E Mohr, Yanhui Hu, Benjamin Ewen-Campen, Benjamin E Housden, Raghuvir Viswanatha, and Norbert Perrimon. Crispr guide rna design for research applications. *The FEBS journal*, 283(17):3232–3238, 2016.

[60] Haroon Butt, Ayman Eid, Zahir Ali, Mohamed AM Atia, Morad M Mokhtar, Norhan Hassan, Ciaran M Lee, Gang Bao, and Magdy M Mahfouz. Efficient crispr/cas9-mediated genome editing using a chimeric single-guide rna molecule. *Frontiers in plant science*, 8:1441, 2017.

[61] Tautvydas Karvelis, Giedrius Gasiunas, Algirdas Miksys, Rodolphe Barrangou, Philippe Horvath, and Virginijus Siksnys. crrna and tracrrna guide cas9-mediated dna interference in streptococcus thermophilus. *RNA biology*, 10(5):841–851, 2013.

[62] Elitza Deltcheva, Krzysztof Chylinski, Cynthia M Sharma, Karine Gonzales, Yanjie Chao, Zaid A Pirzada, Maria R Eckert, Jörg Vogel, and Emmanuelle Charpentier. Crispr rna maturation by trans-encoded small rna and host factor rnase iii. *Nature*, 471(7340):602–607, 2011.

[63] Mark D Szczelkun, Maria S Tikhomirova, Tomas Sinkunas, Giedrius Gasiunas, Tautvydas Karvelis, Patrizia Pschera, Virginijus Siksnys, and Ralf Seidel. Direct observation of r-loop formation by single rna-guided cas9 and cascade effector complexes. *Proceedings of the National Academy of Sciences*, 111(27):9798–9803, 2014.

[64] Tautvydas Karvelis, Giedrius Gasiunas, and Virginijus Siksnys. Methods for decoding cas9 protospacer adjacent motif (pam) sequences: a brief overview. *Methods*, 121:3–8, 2017.

[65] Samuel H Sternberg, Sy Redding, Martin Jinek, Eric C Greene, and Jennifer A Doudna. Dna interrogation by the crispr rna-guided endonuclease cas9. *Nature*, 507(7490):62–67, 2014.

[66] Ekaterina Semenova, Matthijs M Jore, Kirill A Datsenko, Anna Semenova, Edze R Westra, Barry Wanner, John Van Der Oost, Stan JJ Brouns, and Konstantin Severinov. Interference by clustered regularly interspaced short palindromic repeat (crispr) rna is governed by a seed sequence. *Proceedings of the National Academy of Sciences*, 108(25):10098–10103, 2011.

[67] Hiroshi Nishimasu, F Ann Ran, Patrick D Hsu, Silvana Konermann, Soraya I Shehata, Naoshi Dohmae, Ryuichiro Ishitani, Feng Zhang, and Osamu Nureki. Crystal structure of cas9 in complex with guide rna and target dna. *Cell*, 156(5):935–949, 2014.

[68] Takashi Yamano, Hiroshi Nishimasu, Bernd Zetsche, Hisato Hirano, Ian M Slaymaker, Yinqing Li, Iana Fedorova, Takanori Nakane, Kira S Makarova, Eugene V Koonin, et al. Crystal structure of cpf1 in complex with guide rna and target dna. *Cell*, 165(4):949–962, 2016.

[69] Giulia Palermo, Janice S Chen, Clarisse G Ricci, Ivan Rivalta, Martin Jinek, Victor S Batista, Jennifer A Doudna, and J Andrew McCammon. Key role of the rec lobe during crispr–cas9 activation by 'sensing','regulating', and 'locking'the catalytic hnh domain. *Quarterly reviews of biophysics*, 51:e9, 2018.

[70] James A Gagnon, Eivind Valen, Summer B Thyme, Peng Huang, Laila Ahkmetova, Andrea Pauli, Tessa G Montague, Steven Zimmerman, Constance Richter, and Alexander F Schier. Efficient mutagenesis by cas9 protein-mediated oligonucleotide insertion and large-scale assessment of single-guide rnas. *PloS one*, 9(5):e98186, 2014.

[71] Tim Wang, Jenny J Wei, David M Sabatini, and Eric S Lander. Genetic screens in human cells using the crispr-cas9 system. *Science*, 343(6166):80–84, 2014.

[72] John G Doench, Ella Hartenian, Daniel B Graham, Zuzana Tothova, Mudra Hegde, Ian Smith, Meagan Sullender, Benjamin L Ebert, Ramnik J Xavier, and David E Root. Rational design of highly active sgrnas for crispr-cas9–mediated gene inactivation. *Nature biotechnology*, 32(12):1262–1267, 2014.

[73] Karambir Kaur, Amit Kumar Gupta, Akanksha Rajput, and Manoj Kumar. ge-crispr-an integrated pipeline for the prediction and analysis of sgrnas genome editing efficiency for crispr/cas system. *Scientific reports*, 6(1):1–12, 2016.

[74] Raj Chari, Nan Cher Yeo, Alejandro Chavez, and George M Church. sgrna scorer 2.0: a species-independent model to predict crispr/cas9 activity. *ACS synthetic biology*, 6(5):902–904, 2017.

[75] Nathan Wong, Weijun Liu, and Xiaowei Wang. Wu-crispr: characteristics of functional guide rnas for the crispr/cas9 system. *Genome biology*, 16(1):1–8, 2015.

[76] Houxiang Zhu and Chun Liang. Crispr-dt: designing grnas for the crispr-cpf1 system with improved target efficiency and specificity. *Bioinformatics*, 35(16):2783–2789, 2019.

[77] Guohui Chuai, Hanhui Ma, Jifang Yan, Ming Chen, Nanfang Hong, Dongyu Xue, Chi Zhou, Chenyu Zhu, Ke Chen, Bin Duan, et al. Deepcrispr: optimized crispr guide rna design by deep learning. *Genome biology*, 19(1):1–18, 2018.

[78] Daqi Wang, Chengdong Zhang, Bei Wang, Bin Li, Qiang Wang, Dong Liu, Hongyan Wang, Yan Zhou, Leming Shi, Feng Lan, et al. Optimized crispr guide rna design for two high-fidelity cas9 variants by deep learning. *Nature communications*, 10(1):1–14, 2019.

[79] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41:647–665, 2014.

[80] Zengrui Guan and Zhenran Jiang. Transformer-based anti-noise models for crispr-cas9 off-target activities prediction. *Briefings in Bioinformatics*, page bbad127, 2023.

[81] Qiao Liu, Di He, and Lei Xie. Identifying context-specific network features for crispr-cas9 targeting efficiency using accurate and interpretable deep neural network. *bioRxiv*, page 505602, 2018.

[82] Kenneth Ward Church. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017.

[83] Garth R Brown, Vichet Hem, Kenneth S Katz, Michael Ovetsky, Craig Wallin, Olga Ermolaeva, Igor Tolstoy, Tatiana Tatusova, Kim D Pruitt, Donna R Maglott, et al. Gene: a gene-centered information resource at ncbi. *Nucleic acids research*, 43(D1):D36–D42, 2015.

[84] Donna Maglott, Jim Ostell, Kim D Pruitt, and Tatiana Tatusova. Entrez gene: gene-centered information at ncbi. *Nucleic acids research*, 39(suppl_1):D52–D57, 2010.

[85] Andrew D Yates, Premanand Achuthan, Wasiu Akanni, James Allen, Jamie Allen, Jorge Alvarez-Jarreta, M Ridwan Amode, Irina M Armean, Andrey G Azov, Ruth Bennett, et al. Ensembl 2020. *Nucleic acids research*, 48(D1):D682–D688, 2020.

[86] Ryan Poplin, Pi-Chuan Chang, David Alexander, Scott Schwartz, Thomas Colthurst, Alexander Ku, Dan Newburger, Jojo Dijamco, Nam Nguyen, Pegah T Afshar, et al. A universal snp and small-indel variant caller using deep neural networks. *Nature biotechnology*, 36(10):983–987, 2018.

[87] Christelle Etard, Swarnima Joshi, Johannes Stegmaier, Ralf Mikut, and Uwe Strähle. Tracking of indels by decomposition is a simple and effective method to assess efficiency of guide rnas in zebrafish. *Zebrafish*, 14(6):586–588, 2017.

[88] Shengdar Q Tsai, Zongli Zheng, Nhu T Nguyen, Matthew Liebers, Ved V Topkar, Vishal Thapar, Nicolas Wyvekens, Cyd Khayter, A John Iafrate, Long P Le, et al. Guide-seq enables genome-wide profiling of off-target cleavage by crispr-cas nucleases. *Nature biotechnology*, 33(2):187–197, 2015.

[89] Maki Hirata, Manita Wittayarat, Zhao Namula, Quynh Anh Le, Qingyi Lin, Koki Takebayashi, Chommanart Thongkittidilok, Fuminori Tanihara, and Takeshige Otoi. Lipofection-mediated introduction of crispr/cas9 system into porcine oocytes and embryos. *Animals*, 11(2):578, 2021.

[90] Paul Boucher, Xiaoxia Cui, and David T Curiel. Adenoviral vectors for in vivo delivery of crispr-cas gene editors. *Journal of Controlled Release*, 327:788–800, 2020.

[91] Wendy Dong and Boris Kantor. Lentiviral vectors for delivery of gene-editing systems based on crispr/cas: current state and perspectives. *Viruses*, 13(7):1288, 2021.

[92] Remi L Gratacap, Tim Regan, Carola E Dehler, Samuel AM Martin, Pierre Boudinot, Bertrand Collet, and Ross D Houston. Efficient crispr/cas9 genome editing in a salmonid fish cell line using a lentivirus delivery system. *BMC biotechnology*, 20:1–9, 2020.

[93] Li Duan, Kan Ouyang, Xiao Xu, Limei Xu, Caining Wen, Xiaoying Zhou, Zhuan Qin, Zhiyi Xu, Wei Sun, and Yujie Liang. Nanoparticle delivery of crispr/cas9 for genome editing. *Frontiers in Genetics*, 12:673286, 2021.

[94] Qi Liu, Kai Zhao, Chun Wang, Zhanzhan Zhang, Chunxiong Zheng, Yu Zhao, Yadan Zheng, Chaoyong Liu, Yingli An, Linqi Shi, et al. Multistage delivery nanoparticle facilitates efficient crispr/dcas9 activation and tumor growth suppression in vivo. *Advanced Science*, 6(1):1801423, 2019.

[95] Achraf Noureddine, Angelea Maestas-Olguin, Edwin A Saada, Annette E LaBauve, Jacob O Agola, Keoni E Baty, Tamara Howard, Jennifer K Sabo, Cindy R Sandoval Espinoza, Jennifer A Doudna, et al. Engineering of monosized lipid-coated mesoporous silica nanoparticles for crispr delivery. *Acta Biomaterialia*, 114:358–368, 2020.

[96] Qiao Liu, Di He, and Lei Xie. Prediction of off-target specificity and cell-specific fitness of crispr-cas system using attention boosted deep learning and network-based gene feature. *PLoS computational biology*, 15(10):e1007480, 2019.

[97] Guishan Zhang, Ye Luo, Xianhua Dai, and Zhiming Dai. Benchmarking deep learning methods for predicting crispr/cas9 sgrna on-and off-target activities. *Briefings in Bioinformatics*, 24(6):bbad333, 2023.

[98] Dominik Modrzejewski, Frank Hartung, Heike Lehnert, Thorben Sprink, Christian Kohl, Jens Keilwagen, and Ralf Wilhelm. Which factors affect the occurrence of off-target effects caused by the use of crispr/cas: a systematic review in plants. *Frontiers in plant science*, 11:574959, 2020.

[99] Xiao-Hui Zhang, Louis Y Tee, Xiao-Gang Wang, Qun-Shan Huang, and Shi-Hua Yang. Off-target effects in crispr/cas9-mediated genome engineering. *Molecular Therapy-Nucleic Acids*, 4:e264, 2015.

[100] Hui Peng, Yi Zheng, Zhixun Zhao, Tao Liu, and Jinyan Li. Recognition of crispr/cas9 off-target sites through ensemble learning of uneven mismatch distributions. *Bioinformatics*, 34(17):i757–i765, 2018.

[101] Sangsu Bae, Jeongbin Park, and Jin-Soo Kim. Cas-offinder: a fast and versatile algorithm that searches for potential off-target sites of cas9 rna-guided endonucleases. *Bioinformatics*, 30(10):1473–1475, 2014.

[102] Aaron McKenna and Jay Shendure. Flashfry: a fast and flexible tool for large-scale crispr target design. *BMC biology*, 16(1):1–6, 2018.

[103] Jacob Bradford, Timothy Chappell, and Dimitri Perrin. Rapid whole-genome identification of high quality crispr guide rnas with the crackling method. *The CRISPR Journal*, 5(3):410–421, 2022.

[104] Marvin L Minsky and Seymour A Papert. Perceptrons: expanded edition, 1988.

[105] Henry Han. Diagnostic biases in translational bioinformatics. *BMC Medical Genomics*, 8:1–17, 2015.

[106] Xiaokang Zhang, Inge Jonassen, and Anders Goksøyr. Machine learning approaches for biomarker discovery using gene expression data. *Bioinformatics [Internet]*, 2021.

[107] Jacob T Nearing, André M Comeau, and Morgan GI Langille. Identifying biases and their potential solutions in human microbiome studies. *Microbiome*, 9(1):1–22, 2021.

[108] Caroline König, Martha I Cárdenas, Jesús Giraldo, René Alquézar, and Alfredo Vellido. Label noise in subtype discrimination of class cg protein-coupled receptors: A systematic approach to the analysis of classification errors. *BMC bioinformatics*, 16(1):1–14, 2015.

[109] Victor Chernozhukov, Kaspar Wüthrich, and Yinchu Zhu. Distributional conformal prediction. *Proceedings of the National Academy of Sciences*, 118(48):e2107794118, 2021.

[110] Jonathan Alvarsson, Staffan Arvidsson McShane, Ulf Norinder, and Ola Spjuth. Predicting with confidence: using conformal prediction in drug discovery. *Journal of Pharmaceutical Sciences*, 110(1):42–49, 2021.

[111] Yufeng Xia, Jun Zhang, Tingsong Jiang, Zhiqiang Gong, Wen Yao, and Ling Feng. Hatchensemble: an efficient and practical uncertainty quantification method for deep neural networks. *Complex & Intelligent Systems*, 7:2855–2869, 2021.

[112] James M Dolezal, Andrew Srisuwananukorn, Dmitry Karpeyev, Siddhi Ramesh, Sara Kochanny, Brittany Cody, Aaron S Mansfield, Sagar Rakshit, Radhika Bansal, Melanie C Bois, et al. Uncertainty-informed deep learning models enable high-confidence predictions for digital histopathology. *Nature communications*, 13(1):6572, 2022.

[113] Junyu Xuan, Jie Lu, and Guangquan Zhang. A survey on bayesian nonparametric learning. *ACM Computing Surveys (CSUR)*, 52(1):1–36, 2019.

[114] Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for regression. *Advances in Neural Information Processing Systems*, 1996.

[115] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[116] Adrien Deliege, Anthony Cioppa, and Marc Van Droogenbroeck. Hitnet: a neural network with capsules embedded in a hit-or-miss layer, extended with hybrid data augmentation and ghost capsules. *arXiv preprint arXiv:1806.06519*, 2018.

[117] Ričards Marcinkevičs and Julia E Vogt. Interpretability and explainability: A machine learning zoo mini-tour. *arXiv preprint arXiv:2012.01805*, 2020.

[118] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[119] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[120] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

[121] Jean Kaddour, Aengus Lynch, Qi Liu, Matt J Kusner, and Ricardo Silva. Causal machine learning: A survey and open problems. *arXiv preprint arXiv:2206.15475*, 2022.

[122] Arthur Conmy, Augustine N Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*, 2023.

[123] Zhuoyao Zhong, Lianwen Jin, and Zecheng Xie. High performance offline handwritten chinese character recognition using googlenet and directional feature maps. In *2015 13th international conference on document analysis and recognition (ICDAR)*, pages 846–850. IEEE, 2015.

[124] Mohammad Mustafa Taye. Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. *Computation*, 11(3):52, 2023.

[125] Quanshi Zhang, Xin Wang, Ruiming Cao, Ying Nian Wu, Feng Shi, and Song-Chun Zhu. Extraction of an explanatory graph to interpret a cnn. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3863–3877, 2020.

[126] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[127] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[128] Anna Palczewska, Jan Palczewski, Richard Marchese Robinson, and Daniel Neagu. Interpreting random forest classification models using a feature contribution method. *Integration of reusable systems*, pages 193–218, 2014.

[129] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[130] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Nothing else matters: Model-agnostic explanations by identifying prediction invariance. *arXiv preprint arXiv:1611.05817*, 2016.

[131] Xingyu Zhao, Wei Huang, Xiaowei Huang, Valentin Robu, and David Flynn. Baylime: Bayesian local interpretable model-agnostic explanations. In *Uncertainty in artificial intelligence*, pages 887–896. PMLR, 2021.

[132] Daniel W Apley and Jingyu Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(4):1059–1086, 2020.

[133] Hui K Kim, Myungjae Song, Jinu Lee, A Vipin Menon, Soobin Jung, Young-Mook Kang, Jae W Choi, Euijeon Woo, Hyun C Koh, Jin-Wu Nam, et al. In vivo high-throughput profiling of crispr–cpf1 activity. *Nature methods*, 14(2):153–159, 2017.

[134] Claudio Piciarelli, Pankaj Mishra, and Gian Luca Foresti. Image anomaly detection with capsule networks and imbalanced datasets. In *International Conference on Image Analysis and Processing*, pages 257–267. Springer, 2019.

[135] Xiaoyan Li, Iluju Kiringa, Tet Yeap, Xiaodan Zhu, and Yifeng Li. Exploring deep anomaly detection methods based on capsule net. In *Canadian Conference on Artificial Intelligence*, pages 375–387. Springer, 2020.

[136] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[137] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[138] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018.

[139] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

[140] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5:180161, 2018.

[141] Thomas Patrick Quinn, Thin Nguyen, Sam Charles Lee, and Svetha Venkatesh. Cancer as a tissue anomaly: classifying tumor transcriptomes based only on healthy data. *Frontiers in genetics*, 10:599, 2019.

[142] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[143] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

[144] William A Falcon. Pytorch lightning. *GitHub*, 3, 2019.

[145] Alex Rogozhnikov. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2021.

[146] Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi: Algorithms for monitoring and explaining machine learning models. *URL https://github. com/SeldonIO/alibi*, 2020.

[147] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *arXiv preprint arXiv:1809.11165*, 2018.

[148] Peter JA Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422, 2009.

[149] Paul Barrett, John Hunter, J Todd Miller, J-C Hsu, and Perry Greenfield. matplotlib–a portable python plotting package. In *Astronomical data analysis software and systems XIV*, volume 347, page 91, 2005.

[150] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[151] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[152] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[153] Gavin E Crooks, Gary Hon, John-Marc Chandonia, and Steven E Brenner. Weblogo: a sequence logo generator. *Genome research*, 14(6):1188–1190, 2004.

[154] Asim Munawar, Phongtharin Vinayavekhin, and Giovanni De Magistris. Limiting the reconstruction capability of generative neural network using negative learning. In *IEEE 27th International Workshop on Machine Learning for Signal Processing*, pages 1–6, 2017.

[155] Yuki Yamanaka, Tomoharu Iwata, Hiroshi Takahashi, Masanori Yamada, and Sekitoshi Kanai. Autoencoding binary classifiers for supervised anomaly detection. In *Pacific Rim International Conference on Artificial Intelligence*, pages 647–659. Springer, 2019.

[156] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

[157] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017.

[158] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995.

[159] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*, 2018.

[160] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[161] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[162] Wei Wei, Jinjiu Li, Longbing Cao, Yuming Ou, and Jiahang Chen. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4):449–475, 2013.

[163] Diego Carrera, Fabio Manganini, Giacomo Boracchi, and Ettore Lanzarone. Defect detection in sem images of nanofibrous materials. *IEEE Transactions on Industrial Informatics*, 13(2):551–561, 2016.

[164] David Savage, Xiuzhen Zhang, Xinghuo Yu, Pauline Chou, and Qingmai Wang. Anomaly detection in online social networks. *Social Networks*, 39:62–70, 2014.

[165] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[166] Huiwen Wang, Jie Gu, and Shanshan Wang. An effective intrusion detection framework based on svm with feature augmentation. *Knowledge-Based Systems*, 136:130–139, 2017.

[167] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018.

[168] Swee Kiat Lim, Yi Loo, Ngoc-Trung Tran, Ngai-Man Cheung, Gemma Roig, and Yuval Elovici. Doping: Generative data augmentation for unsupervised anomaly detection with gan. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1122–1127. IEEE, 2018.

[169] Inyoung Paik, Taeyeong Kwak, and Injung Kim. Capsule networks need an improved routing algorithm. In *Asian Conference on Machine Learning*, pages 489–502. PMLR, 2019.

[170] Robert Dorfman. A formula for the gini coefficient. *The review of economics and statistics*, pages 146–149, 1979.

[171] Fernando G De Maio. Income inequality measures. *Journal of Epidemiology & Community Health*, 61(10):849–852, 2007.

[172] Alex Cobham, Lukas Schlögl, and Andy Sumner. Inequality and the tails: the palma proposition and ratio. *Global Policy*, 7(1):25–36, 2016.

[173] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[174] Yingjun Deng, Alessandro Di Bucchianico, and Mykola Pechenizkiy. Controlling the accuracy and uncertainty trade-off in rul prediction with a surrogate wiener propagation model. *Reliability Engineering & System Safety*, 196:106727, 2020.

[175] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[176] Aleksandr Artemenkov and Maxim Panov. Ncvis: Noise contrastive approach for scalable visualization. In *Proceedings of The Web Conference 2020*, pages 2941–2947, 2020.