Predicting, Engineering and Interpreting Gene Regulatory Sequences and Proteins with Deep Learning

Johannes Staffan Anders Linder

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Georg Seelig, Chair

Sreeram Kannan

Sara Mostafavi

Program Authorized to Offer Degree:
Paul G. Allen School of Computer Science and Engineering

University of Washington

Abstract

Predicting, Engineering and Interpreting Gene Regulatory Sequences and Proteins with Deep Learning

Johannes Staffan Anders Linder

Chair of the Supervisory Committee:
Associate Professor Georg Seelig
Electrical and Computer Engineering
Computer Science and Engineering

The vast majority of the 3.1 billion base-pairs in the (haploid) human genome do not code for a particular protein, yet mutations in these non-coding regions can have a profound impact on phenotype and be deleterious. The reason is that within these regions – enhancers, promoters, introns and untranslated regions (UTRs) – reside a cis-regulatory code which governs gene expression and is sensitive to disruption. Ongoing efforts of mapping the relationship between genetic variants and disease phenotype are limited by data and the lack of generalizability. Furthermore, engineering *de novo* gene-regulatory sequences and proteins according to target specifications, which would aid the development of vaccines, medical therapeutics, molecular sensing devices and more, is hampered by the lack of methods that can reliably generate large sets of diverse and optimized candidate designs for high-throughput screening.

This dissertation presents an approach combining Massively Parallel Reporter Assays (MPRAs) with Deep Learning to obtain a sequence-predictive model of Alternative Polyadenylation (APA), a regulatory process occurring mainly in the 3' UTR of pre-mRNA. The trained neural network predicts 3'-end cleavage at base-pair resolution and can accurately prioritize human variants. By developing methods to visualize features learned in higher-order network layers, we extract a cis-regulatory APA code that aligns well with established biology.

Next, the dissertation presents a family of methods that were developed to design *de novo* biological sequences based on the response of a differentiable fitness predictor. These methods, which are based on *activation maximization*, can be used to efficiently generate millions of diverse, optimized sequence designs on the basis of a deep generative model. Finally, we present a feature attribution method for interpreting neural network predictions. The method, which learns input masks that either reconstruct or destroy the prediction, implements a masking operator based on probabilistic sampling that is shown to be particularly well-suited for interpreting biological sequence models. The developed design- and interpretation methods are demonstrated on several DNA-, RNA- and protein function predictors and outperform state-of-the-art methods for multiple target applications.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

3' UTR: Untranslated (non-coding) region at the end of an mRNA transcript.

APA: Alternative Polyadenylation, the alternative choice of Polyadenylation site.

POLY-A SITE: Polyadenylation Site, a site in the transcript where Polyadenylation is initiated.

PAS: Polyadenylation Signal, the core sequence element that defines a Poly-A site.

PROXIMAL ISOFORM: The shorter isoform, produced by polyadenylation at the first Poly-A Site.

DISTAL ISOFORM: The longer isoform, produced by polyadenylation at other sites.

POSITION SPECIFIC SCORING MATRIX: PSSM – A matrix where each column corresponds to nucleotide or amino acid probabilities at each position within a sequence.

ORACLE MODEL: A predictor model that we typically explicitly optimize or design sequences for.

VALIDATION MODEL: An independent model that we have not explicitly optimized sequences for.

ACKNOWLEDGMENTS

I want to thank my advisor and mentor Prof. Georg Seelig for all the advice and help he has provided throughout my years at the University of Washington and in Seelig Lab. I would also like to thank my doctoral committee members: Prof. Jay Shendure, Prof. Sreeram Kannan, Prof. Sara Mostafavi, Prof. Sewoong Oh and Prof. Jennifer Listgarten. Finally, I would like to thank the members of Seelig Lab for the great collaborations we have had throughout the years (Alberto Carignano, Alex Baryshev, Alexander Rosenberg, Alyssa La Fleur, Angela Yu, Anna Kuchina, Arjun Khakar, Ban Wang, Ben Groves, Chandler Petersen, Charlie Roco, Christopher Yin, Erin Wilson, Matthew Hirano, Max Darnell, Nick Bogard, Paul Sample, Randolph Lopez, Sam Koplik, Sebastian Castillo Hair, Sergii Pochekailov, Sifang Chen, Sumit Mukherjee, Sunny Rao, Yuan-Jyue Chen, Yue Zhang). I want to give a special thanks to Nick Bogard, who taught me a lot about biology and was a great collaborator to work with.

Chapter 1

INTRODUCTION

Humans have about 20,000 protein-coding genes, the expression levels of which govern a cell's function and phenotype [92]. Gene expression is a tightly regulated process where the genetic code – the human genome – not only codes for specific proteins but also contains a vast amount of non-coding 'instructions' for how expression of each protein should be modulated. These instructions come in various forms: At the level of DNA, transcription is activated or silenced by Transcription Factors (TFs) and regulation at the chromatin level [145, 156]. After transcription, the mRNA molecule itself is controlled by cis-regulatory, non-coding, regions within the transcript [112]: the 5' and 3' untranslated regions (UTRs) influence translation efficiency and stability [5, 57], and mechanisms such as Alternative Splicing (AS) [15] and Alternative Polyadenylation (APA) [154] can create differentially regulated isoforms by conditionally excluding parts of these regulatory regions. How sequence variation in these non-coding regions alters expression is for the large part not well-understood.

Genetic variants that interfere with these cis-regulatory mechanisms have been implicated in disease [39, 147, 51]. However, identifying which variants are pathogenic can be quite challenging; hundreds of thousands of nucleotides differ between the genomes of two individuals [92], which means a large population sample is necessary to statistically associate loci with a particular phenotype. Consequently, rare and *de novo* variants become difficult, if not impossible, to characterize with genome-wide association studies (GWAS) [84, 152]. Improving variant interpretation remains an open, yet important, problem if we want to further our understanding of the link between genetic variation and disease.

Beyond variant prediction, engineering gene-regulatory sequences and proteins with improved biological function is a defining goal of synthetic biology. Rational sequence design is

expected to accelerate the development of molecular therapeutics, vaccines, nanotechnology and other applications [123, 157, 93, 27, 90]. Developing efficient methods that can scale to generate millions of candidate designs with controllable fitness, confidence and diversity in order to support these applications is an ongoing effort in the field.

Finally, to tackle these challenges, researchers are increasingly relying on models based on deep learning that relate nucleic acid or protein sequences to function [44, 175, 1, 173, 114, 68, 28]. Neural Networks (NNs) can learn complex relationships from large datasets, but interpreting the functional rules encoded in these sequence-predictive networks remains challenging. Yet, this is crucial if we want to associate variant predictions with regulatory biology or understand the determinants governing the function of a protein or a non-coding sequence in a gene [78, 172, 77, 171, 140, 8]. Model interpretation for sequence-predictive neural networks is thus another important open problem within the research community.

This dissertation presents work conducted by me and colleagues in Seelig Lab in all three of the problem areas introduced above. First, in the context of gene-regulatory modeling, I will describe an approach where we measured a large collection of random sequence variants for Alternative Polyadenylation, an RNA-regulatory mechanism, and combined these data with deep learning for predicting polyadenylation at nucleotide resolution. Next, in the context of sequence design, we developed a family of gradient-based design methods which could be used to generate diverse distributions of sequences with optimized functional fitness. Finally, we developed an interpretation method for sequence-predictive neural networks which explain input patterns by learning input masks that either preserve or destroy the original model prediction. Our approach, which adopts a temperature-based masking operator where sequences are converted to nucleotide distributions with high sample entropy for unimportant features, is particularly well-suited for biological sequences.

1.1 Learning Alternative Polyadenylation from Millions of Sequence Variants

Alternative Polydenylation (APA) is a regulatory process that happens as a gene is transcribed into mRNA. Signaling elements referred to as Polydenylation Signals (PASes) located

within the transcript compete for cleavage and polyadenylation – an event where the 3'-end processing machinery cleaves the molecule at the chosen PAS and appends a tail of adenine bases – resulting in distinct RNA isoforms [54, 154]. The regulatory effect of APA depends on where the PASes are located in the transcript: The most common form is 3' UTR APA, where the competing signals reside in the 3' untranslated region (UTR) and hence produce RNA isoforms with 3' UTRs of distinct lengths [43]. These isoforms may have widely different characteristics in terms of RNA stability, as regulatory elements such as miRNA binding sites and structural elements could have been cleaved off for the shorter isoforms [154].

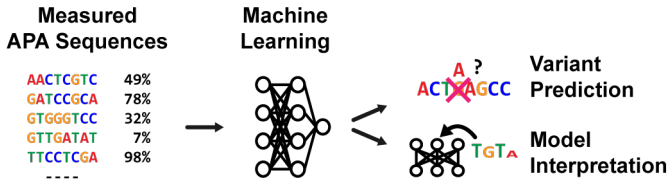

Figure 1.1: Using a trained machine learning model to predict the impact of human APA variants and interpret cis-regulatory rules.

APA is itself governed by a regulatory code, where the 'strength' of a PAS (its preference for being selected) is determined by a central hexamer motif (CSE), most commonly AATAAA, and cis-regulatory sequences surrounding the CSE [154]. These cis-acting elements include motifs for RNA binding proteins (RBPs) such as CPEB1, SRSF-family proteins, ELAV-like proteins, positioning of core processing motifs such as CFI and CsTF, as well as structural determinants like stem loops [43, 162]. Even though these regulators are well-characterized individually, prior to our work we lacked a model that quantitatively combined this information to provide APA isoform predictions. Moreover, several genetic variants that disrupt APA have been implicated in disease [39, 13, 161]. Since experimentally screening every possible variant is infeasible, and genome-wide association studies (GWAS) are of limited use for rare variants [84], an accurate predictive model is crucial if we want to identify and characterize pathogenic variants at genome scale.

In Chapter 3, we describe an approach based on massively parallel reporter assays

(MPRAs) to experimentally measure millions of randomized 3' UTR sequences with targeted variation to the proximal polyadenylation signal of a plasmid reporter. This sequencing-based MPRA provided rich measurements in the form of relative cleavage abundances at each sequence position, which allowed us to train a neural network – APARENT – for predicting cleavage and polyadenylation at base-pair resolution given only the primary sequence as input. We used this model to predict the effect of variants on human APA sites (Figure 1.1) and showed that APARENT significantly outperform other neural networks trained on native genomic data in characterizing cryptic APA variants outside the CSE. Furthermore, by extending sampling-based techniques for visualizing convolutional filters of the first layer [1], and by adopting optimization methods from computer vision [139], we could extract a comprehensive cis-regulatory code of APA learned in higher-order layers of the network. While we here specifically investigated APA, the combined approach of MPRAs and deep neural networks, as well as the visualization methods developed here, is likely to be widely useful for the study of many other gene-regulatory phenomena.

This work was a collaboration between me and colleagues of Seelig Lab, and was later published by Dr. Nicholas Bogard and I as co-first authors [17].

1.2 Engineering Gene-Regulatory Sequences and Proteins

Most sequence design methods are guided by predictive models that relate sequence to biological fitness. These models are increasingly being based on deep learning [44, 175]. Concurrently, a wide selection of molecular design methods are being developed [168]. For example, Rocklin et al. (2017) combined rule-based design with a massively parallel stability assay to explore the space of stably folded miniproteins [128]. Biswas et al., (2018) and Sample et al. (2019) used genetic algorithms to optimize green fluorescent proteins and 5' UTR sequences respectively [14, 130]. Recently, deep generative models such as variational autoencoders (VAEs), autoregressive networks and generative adversarial networks (GANs) have been used to design proteins [34], *de novo* antibodies [127, 3], optimized promoter sequences [158] and enzymes [125]. Furthermore, methods based on directed evolution have been developed

to condition such models for a biological property [59, 18].

Clearly, the design methods being developed are highly diverse and it is not obvious which method is suitable for what design problem, or even if one method is universally appropriate. Here, the design problem is defined as *the task of designing improved sequences for a biological function given an initial set of sequences* $\{\boldsymbol{x}\}_{i=1}^{N}$ *and associated fitness measurements* $\{f(\boldsymbol{x})\}_{i=1}^{N}$ (Figure 1.2). In its most basic form, we study the following optimization problem:

$$\max_{\boldsymbol{x}} \quad f(\boldsymbol{x}) \tag{1.1}$$

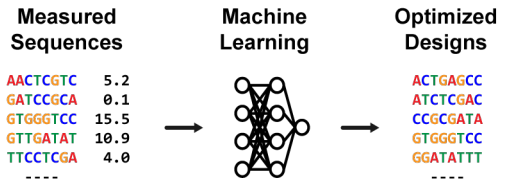

Figure 1.2: Designing new improved biological sequences based on a functional model.

Due to the abundance of differentiable sequence-predictive fitness models $\mathcal{P}(\boldsymbol{x})$ that approximate $f(\boldsymbol{x})$, this work investigates the utility of gradient-based optimization methods for sequence design. Gradient-based optimization has previously been used for molecular design [86, 80]. The method, commonly referred to as *activation maximization*, uses the gradient of the neural network output to make incremental changes to the input pattern. While simple, activation maximization cannot be directly applied to discrete sequences and several re-parameterizations and relaxations have been proposed to rectify this [80, 61]. While showing early promise, it is largely unknown whether gradient-based design is prone to getting stuck in local minima, whether they can generate controllably diverse designs, or how to efficiently generate many (millions) of candidate designs. Finally, the accuracy of the predictor may diminish as sequences drift from the training data. It is unclear how we would enforce constraints such as a "minimum level of confidence" using activation maximization.

In Chapter 4, this dissertation presents an improved gradient-based sequence design

method that outperforms current activation maximization algorithms as well as other commonly used global optimization meta-heuristics. The improved method builds on the following two key ideas: Sequences are optimized on the basis of discrete samples drawn from a distribution, and gradients are backpropagated using straight-through (ST) approximation [31] or the Gumbel distribution [70]. Bottlenecks arising from skewed sampling parameters are removed by normalization and the introduction of an adaptive, global entropy parameter. Part of this work was published by me and colleagues of Seelig Lab in [17] and an updated version of the method was later deposited on a pre-print server [100].

Next, in Chapter 5, we present an extension of activation maximization into a class of generative neural networks which, by considering two independent sequence samples in each forward pass, can optimize both fitness and diversity by backpropagation. Also, by incorporating models that estimate data likelihood, these networks can maintain sequence confidence during end-to-end differentiable training. We refer to these activation-maximizing generative models as Deep Exploration Networks (DENs), and we show that they can efficiently generate millions of diverse sequences that are optimized for various biological fitness predictors, often producing more diverse designs than other estimation-of-distribution algorithms (EDAs). This work was published in [96].

1.3 Interpreting Sequence-based Biological Neural Networks

As more and more sequence-predictive models are based on neural networks, it becomes increasingly important to develop accurate interpretation methods for these architectures. Due to the difficulty in extracting the learned rules from a neural network, we often instead interpret individual input patterns using *feature attribution* methods which treat the network as a black box (Figure 1.3). Many existing attribution methods use local approximation to estimate feature importance [139, 170, 146, 150, 138, 103, 126]. However, such approaches suffer three potential pitfalls: first, they assume independence among input features, while sets of coupled features often impose non-linear effects on predictions. Second, many of these methods struggle with saturated activations within the network. Third, most methods are

not optimized for interpreting discrete input patterns such as biological sequences.

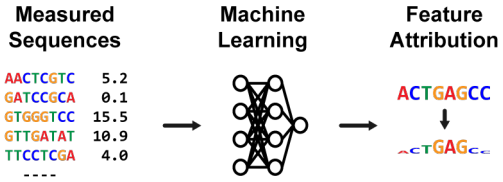

Figure 1.3: Discovering salient nucleotides or amino acids based on a functional model.

In Chapter 6, we develop an alternate approach to feature attribution for biological sequences based on previous work in *mask-based interpretation* [49, 48, 38, 26, 169, 24, 36]. A deep generative model is trained to mask the input patterns such that it either includes or excludes the smallest set of features in order to reconstruct or destroy the prediction. Our version of this approach – *Scrambler Networks* – is made suitable for discrete sequence patterns using a temperature-based masking operator: Given an input sequence, the Scrambler outputs a Position-Specific Scoring Matrix (PSSM). Discrete samples drawn from this PSSM are sent to the predictor and the Scrambler is trained by backpropagation to reconstruct or distort the prediction. A sequence position is 'masked' by raising the entropy of the underlying PSSM distribution. Scramblers can efficiently discover multiple salient feature sets within a single input pattern using a mask dropout layer. Furthermore, in addition to finding reconstructive features, we can alternatively optimize the Scrambler to find feature sets that maximize or minimize the prediction, allowing us to separate repressive and enhancing motifs.

In this dissertation, Scramblers are shown to produce meaningful feature attributions for several visual- and sequence-predictive tasks. Due to its use of a generative model, Scramblers are much more computationally efficient than per-example interpretation methods and they often learn to produce higher-quality attributions than competing methods. The majority of the content in Chapter 6 was presented by me, Alyssa La Fleur and several other collaborators at the MLCB 2020 conference [98] and was later deposited on a pre-print server [97].

Chapter 2
BACKGROUND AND RELATED WORK

This chapter introduces the relevant background and related computational methods necessary to understand the context of this work. The first section provides a brief review of the mechanism and regulation of Alternative Polyadenylation (APA), which is the major gene-regulatory phenomenon studied in Chapter 3 and for which we developed an accurate predictive model based on deep learning [17]. The next section briefly introduces the concept of Massively Parallel Reporter Assays – the experimental method relied upon to collect large volumes of measured, synthetic APA events for training. This is followed by a review of earlier work in using machine learning models based on deep learning for predicting gene-regulatory sequences. We will then go over various methods and generative models that have been used for molecular sequence design prior to the design methods presented in Chapter 4 and 5 [100, 96]. Finally, we will review earlier work in *mask*-based interpretation methods for neural networks, which is the foundation that the attribution framework developed in Chapter 6 is based on [97].

2.1 Alternative Polyadenylation

2.1.1 The Mechanism of Polyadenylation

Polyadenylation is a modification of recently transcribed, premature, RNA where the transcript is cleaved at some position – polyadenylation mainly occurs in the 3' unstranslated region (3' UTR) but intronic or exonic polyadenylation is also possible – and a tail consisting solely of adenine-bases ('A') is attached. This adenine-tail is often called the *polyA-tail*. Most protein-coding genes in eukaryotic cells are polyadenylated [43]. Polyadenylation plays an important role in certain cellular processes, including mRNA-export from the nucleus and

protection from degradation [58, 33]. Also, as detailed below, it can change the characteristics and regulation of mRNA depending on where the transcript is polyadenylated. The overall sequence signal recognized by the 3'-end processing machinery is called the polyadenylation signal (PAS). A 3' UTR PAS is depicted in Figure 2.1 below.

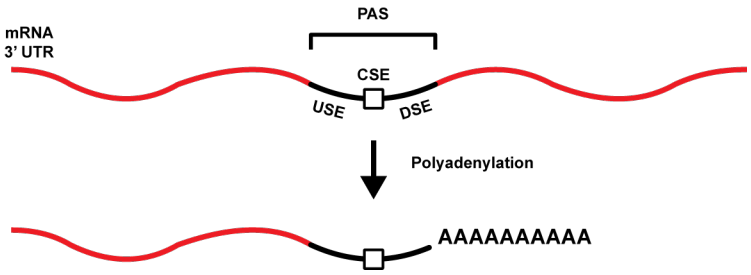

Figure 2.1: Illustration of a 3' UTR Polyadenylation Signal.

The basic mechanism of polyadenylation is summarized from Colgan & Manley's extensive article (Genes & Development, 1997) on the process [32]: The first stage of polyadenylation, cleavage, starts with the enzyme *Cleavage and Polyadenylation Specificity Factor* (CPSF) binding to a specific hexamer at the premature transcript, which we here refer to as the central sequence element (CSE). The canonical form of the CSE is AAUAAA, although more than 10 sequence variants are known [10]. The protein *Cleavage Stimulation Factor* (CSTF) binds to a GU-rich sequence motif downstream of CPSF (downstream sequence elements; DSE). Similarly, the protein *Cleavage Factor I* (CFI) binds to UGUA-motifs upstream of CPSF (upstream sequence elements; USE) [19]. The assembled polyadenylation machinery cleaves the transcript between the CSE and the DSE (approximately 15-30 nt downstream of CSE). As cleavage occurs, Polyadenylation Polymerase (PAP) starts building the polyA-tail by appending adenine-bases up to a maximum length of approximately 250 nt.

2.1.2 The Extent and Regulation of Alternative Polyadenylation

Research conducted in the last two decades has discovered that most genes in eukaryotic cells have more than one cleavage- and polyadenylation site and thus are alternatively polyadeny-

lated (APA). By analyzing large expressed sequence tag (EST) databases containing information on expressed transcripts, researchers in 2005 showed that about 54% of human genes have multiple polyadenylation sites [153]. In more recent years, studies conducted using RNA-sequencing techniques to analyze gene expression have increased this estimate to approximately 70% [137, 42]. Four categories are often used to group different types of APA events [43]. They are:

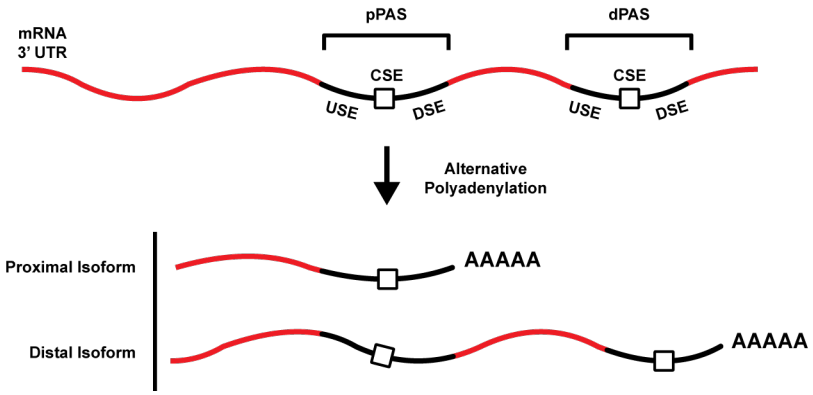

Figure 2.2: Tandem 3' UTR APA resulting in either a proximal and distal isoform.

1. **Tandem 3' UTR APA**: The alternative polyA-sites are both located in the 3' UTR. This means alternate cleavage will only affect the length of the 3' UTR, which may or may not affect which regulatory elements like microRNA are kept in the UTR.

2. **Alternative terminal exon APA**: Cleavage at the proximal PAS removes part of the last protein-coding exon.

3. **Intronic APA**: Cleavage at a proximal intronic PAS removes one or more downstream protein-coding exons.

4. **Internal exon APA**: The proximal PAS is located within one of the non-final exons and removes multiple downstream coding exons.

The categories are ordered by relative frequency. The first category, which only affects the 3' UTR, is the most prevalent form [43]. A 3' UTR APA event is illustrated in Figure 2.2.

The regulation of APA has been studied extensively in the last few years and various mechanisms have been found to influence the choice of polyA-site [54, 154]. For example, a study on embryonic development of mice found strong correlation between long 3' UTRs and low expression levels of polyadenylation trans-factors (CPSF, CSTF etc.) [72]. When the expression levels of these proteins increased, 3' UTRs generally decreased in length. Other studies suggest a correlation between the separation of consecutive polyA-sites and choosing the proximal PAS [120]. Since the proximal PAS is visited first by RNA Polymerase, it is believed to have a better chance of being chosen if the distal site is far away.

Numerous RNA-binding proteins (RBPs) have also been shown to regulate the choice of APA site. For example, some RBPs act repressive on proximal PAS selection by binding close to the CSE and as a result most likely physically blocking polyadenylation. One such protein is Polyadenylation Binding Protein Nuclear 1 (PABPN1), which has been found to contribute to human genetic disorders when its binding motif near APA sites is altered by mutation [71]. Another example is ELAV from Drosophila melanogaster. Human analogs to this protein have been shown to act repressively on proximal PAS selection [65]. In contrast, activating or promoting properties by proteins for choosing a proximal site have also been discovered. For example, Cytoplasmic Polyadenylation Element Binding Protein 1 (CPEB1) binds near proximal polyA-sites and recruits 3'-end processing machinery in order to increase polyadenylation efficiency [9]. Finally, secondary structures such as the formation of hairpin loops in the DSE have recently been found to influence the cleavage position within a PAS [162].

2.1.3 Alternative Polyadenylation and Disease

While most cis-regulatory elements are well-characterized individually, prior to the work presented in Chapter 3 on training a predictive model of APA, there were very few (if any) models in place for quantitatively scoring the net strength of a candidate PAS given

only its sequence as input. Being able to accurately score sequences is of particular importance in the context interpreting genetic variants and assessing their impact on APA, as many polyadenylation-disrupting variants have been implicated in disease [39, 13, 161, 147]. Most well-characterized variants interfere with APA by altering the CSE. However, cryptic pathogenic variants found in the DSE are also prevalent [109, 163].

2.2 Massively Parallel Reporter Assays

The experimental data collection method employed in our study of APA was based on Massively Parallel Reporter Assays (MPRAs). Before MPRAs, experiments that were intended to measure the effect or function of a set of biological sequences had to be performed in low-throughput, usually by measuring each sequence separately per experiment. Patwardhan et al. (2009) introduced one of the first MPRAs by successfully measuring the activity level of every single nucleotide variant of a promoter in a single experiment [118]. Since then, many MPRA-style experiments have been done to measure the impact of genetic variants on gene expression [110, 135, 142, 160, 46, 108].

Many MPRAs use molecular counts obtained from DNA- and RNA sequencing as a proxy for the true experimental read-out they want to measure. To that end, they rely on next-generation sequencing technologies to collect these count measurements in high-throughput. For example, the STARR-Seq protocol measures DNA- and RNA counts of plasmid reporters in order to approximate enhancer activity by RNA expression fold change [7]. There are even a number of MPRAs for quantifying Alternative Polyadenylation by 3'-end sequencing of RNA transcripts [136, 69, 42, 106]. These protocols use a poly-T primer to attach to the start of the polyA-tail and perform reverse transcription, after which the double-stranded products are sequenced and the resulting RNA isoforms can be counted.

MPRAs based on RNA-Seq have also been developed for large-scale measurements of randomized mini-gene reporters, where the goal is to study a gene-regulatory mechanism in isolation by targeted variation introduced in the regulatory region of interest. For example, Rosenberg et al. (2015) measured hundreds of thousands of splicing variants by randomizing

regions around the splice junctions of a plasmid reporter and measuring isoform abundance by sequencing [129]. More recently, RNA-Seq in combination with polysome profiling has been used to measure 5' UTR translational efficiency in high-throughput [130].

2.3 Sequence-Predictive Machine Learning Models in Biology

Deep Neural Networks are becoming increasingly popular for biological sequence prediction problems. In recent years, these models have been applied with overwhelming success to a diverse set of biological domains, such as predicting Transcription Factor (TF) binding [1, 44, 114, 8], chromatin modification and accessibility [173], RNA processing [6, 28, 68, 95], regulation of translation [130] and even protein structure prediction [133, 167]. Part of what makes neural networks popular is their ability to learn complex relationships from large datasets without requiring much tuning. This increased flexibility often results in more accurate predictions than other machine learning models.

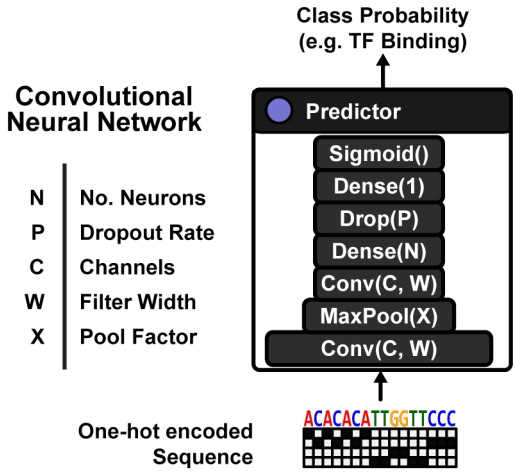

Figure 2.3: A typical Convolutional Neural Network architecture.

A number of different deep learning architectures have been employed to predict gene-regulatory function from native high-throughput sequence data. For example, Zhou et al. (2015) trained a 'classic' convolutional neural network (CNN; example architecture given in

Figure 2.3) consisting of convolutional layers, pooling layers and dense fully connected layers to predict TF binding, DNA accessibility and chromatin modifications from ChIP-Seq and DNAse Hypersensitivity data [173]. Another network architecture based on dilated residual blocks [63] trained on ChIP-Seq data could predict TF peaks at nucleotide-resolution [8], providing finer-granularity predictions and interpretations than earlier models. See Figure 2.4 for a typical residual network architecture with dilated convolutions. This type of network has also been used to learn splicing from native RNA-Seq data and was shown to improve variant prediction compared to previous models [68]. Neural network models trained on native 3'-sequencing data have even been employed for APA prediction [91, 6, 95]. Some of these studies report anecdotal evidence of accurate APA variant prediction, yet rigorous benchmarks of variant prediction performance remains to be done.

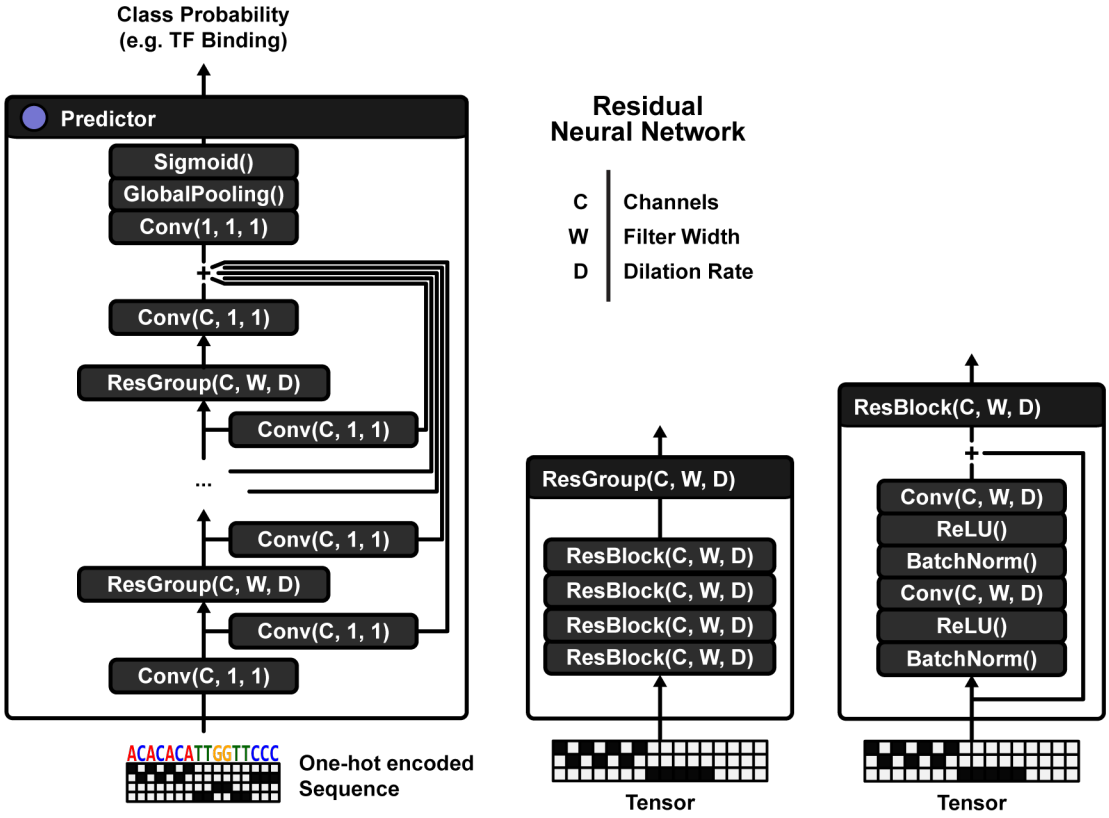

Figure 2.4: A Residual Neural Network architecture with dilated convolutions.

Lately, a number of models have been trained on synthetic MPRA datasets to learn about gene-regulatory functions in isolation (as compared to native data where there could be confounding effects from variation in other regulatory elements). For example, convolutional neural networks have been trained on gene enhancer MPRA data to predict transcriptional activity [114]. Another model was used to learn 5' UTR translational efficiency from ribosome loading MPRA data [130].

2.4 Sequence Designs Methods and Generative Sequence Models

This section gives an overview of current sequence design methods; it describes their basic theory, applications they have been demonstrated on and the broader context into which they fit. These methods are classified into four classes: (1) Methods based on discrete search, (2) methods based on activation maximization, (3) forward-sampling of generative models and (4) iterated forward-sampling of generative models. The first two classes describe model-free per-sequence optimization algorithms. The latter two describe parametric generative models.

2.4.1 Methods Based on Discrete Search

A wide selection of discrete search algorithms and meta-heuristics have been applied to computational sequence design. We will assume access to a sequence-predictive fitness model $\mathcal{P}(\boldsymbol{x}) \in \mathbb{R}$, and the task is to maximize the predicted fitness $\mathcal{P}(\boldsymbol{x}) \in \mathbb{R}$ by tuning the input sequence $\boldsymbol{x}$:

$$\max_{\boldsymbol{x}} \ \mathcal{P}(\boldsymbol{x}) \tag{2.1}$$

A simple randomized search algorithm was used in [130] to design maximally translationally efficient 5' UTR sequences. Given an intial start sequence $\boldsymbol{x}$, at each iteration the algorithm mutates either 1 or 2 randomly chosen nucleotides, resulting in a new candidate sequence $\boldsymbol{x}'$. $\boldsymbol{x}'$ is only accepted if $\mathcal{P}(\boldsymbol{x}') > \mathcal{P}(\boldsymbol{x})$. Because of the greedy selection criteria, this method can not overcome any local minimum wider than 2 nucleotide substitutions.

To cross such a minimum, we would have to simultaneously make 3 mutations. Of course, increasing the number of simultaneous mutations vastly expands the search space.

Many different heuristics have been proposed to improve the performance of discrete search methods. Common to all heuristics, the goal is to minimize the risk of getting stuck in local minima while keeping the search space reasonably small. For example, design methods based on Genetic Algorithms [41] use the same basic mutation- and selection criteria as above. The difference is that such methods maintain a larger set of sequences which are optimized simultaneously. Genetic algorithms heuristically re-combine these sequences during optimization in order to provide sequence changes which are likely to raise fitness while altering many nucleotides at once (thus increasing the chance of jumping out of the current minimum). More complex heuristics such as Particle Swarms and Ant Colonies have been applied extensively to the design of DNA strand displacement sequences [164, 67, 115]. While these methods use different mechanisms for re-combination and selection compared to genetic algorithms, they too rely on maintaining a diverse pool of candidate sequences.

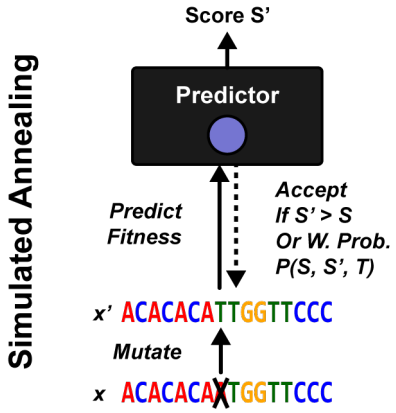

Figure 2.5: Simulated Annealing heuristic for optimizing a DNA sequence.

Another well-known meta heuristic for sequence design is Simulated Annealing (Figure 2.5) [83, 62]. In Simulated Annealing, we generally make a small number of simultaneous mutations at each iteration (keeping the search graph small). However, mutations are accepted

even if they result in lower fitness with probability $P(\boldsymbol{x}', \boldsymbol{x}, T)$, where T is a temperature parameter that is decrementally lowered as optimization progresses. $P(\boldsymbol{x}', \boldsymbol{x}, T)$ is usually chosen to be the Metropolis acceptance criterion [111]:

$$P(\boldsymbol{x}', \boldsymbol{x}, T) = e^{-(\mathcal{P}(\boldsymbol{x}) - \mathcal{P}(\boldsymbol{x}'))/T}$$

2.4.2 Methods Based on Activation Maximization

As an alternative to discrete nucleotide-swapping heuristics, gradient-based methods (or *activation maximization*) directly optimize sequences $\boldsymbol{x}$ for maximal fitness by gradient ascent through a neural network fitness predictor $\mathcal{P}$. Assuming $\mathcal{P}$ is differentiable, we can compute the gradient $\nabla_{\boldsymbol{x}}\mathcal{P}(\boldsymbol{x})$ with respect to the input and optimize $\boldsymbol{x}$ by updating the variable in the direction of the fitness gradient [139]:

$$\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \eta \cdot \nabla_{\boldsymbol{x}}\mathcal{P}(\boldsymbol{x}) \tag{2.2}$$

However, sequences are usually represented as one-hot coded patterns ($\boldsymbol{x} \in \{0, 1\}^{N \times M}$, where N is the sequence length and M the number of channels), and discrete variables cannot be directly optimized by gradient ascent. Several different reparameterizations of $\boldsymbol{x}$ have been proposed to bypass this issue. In one of the earliest implementations, Lanchantin et al. (2016) represented the sequence as an unstructured, real-valued pattern ($\boldsymbol{x} \in \mathbb{R}^{N \times M}$) but imposed an L2-penalty on $\boldsymbol{x}$ in order to keep it from growing too large and causing predictor pathologies [86]:

$$\max_{\boldsymbol{x}} \;\; \mathcal{P}(\boldsymbol{x}) - \frac{1}{2}\lambda \cdot \|\boldsymbol{x}\|_2^2 \tag{2.3}$$

Lanchantin et al. used this method to visualize maximally strong transcription factor binding motifs learned from ChiP-Seq data. Killoran et al. (2017) later introduced a softmax reparameterization, turning $\boldsymbol{x}$ into a continuous relaxation $\sigma(\boldsymbol{l})$ (Figure 2.6) [80]:

$$\sigma(\boldsymbol{l})_{ij} = \frac{e^{\boldsymbol{l}_{ij}}}{\sum_{k=1}^{4} e^{\boldsymbol{l}_{ik}}} \tag{2.4}$$

Here $\boldsymbol{l}_{ij} \in \mathbb{R}$ are differentiable *nucleotide logits*. The gradient of $\sigma(\boldsymbol{l})$ with respect to $\boldsymbol{l}$ is defined as:

$$\frac{\partial \sigma(\boldsymbol{l})_{ij}}{\partial \boldsymbol{l}_{ik}} = \sigma(\boldsymbol{l})_{ik} \cdot (\mathbb{1}_{(j=k)} - \sigma(\boldsymbol{l})_{ij}) \tag{2.5}$$

Given Equation 2.4 and 2.5, we can maximize $\mathcal{P}(\sigma(\boldsymbol{l}))$ with respect to the logits $\boldsymbol{l}$ by gradient ascent:

$$\max_{\boldsymbol{l}} \ \mathcal{P}(\sigma(\boldsymbol{l})) \tag{2.6}$$

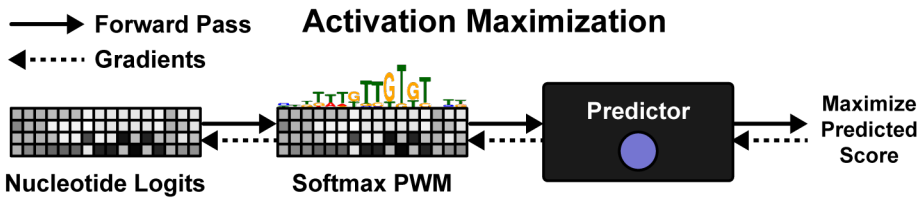

Figure 2.6: Softmax-relaxed Activation Maximization for optimizing a DNA sequence.

Gradient-based optimization has a clear advantage in comparison to the discrete search methods described in the previous section: Activation maximization makes stepwise local improvements to all nucleotides at once based on a gradient direction, rather than heuristically making a small number of changes and evaluating the effect afterwards. As a consequence, gradient-based methods do not necessarily require more iterations for longer sequences or for design problems where we need to make many simultaneous nucleotide substitutions to cross boundaries of minima.

The method has also been used to indirectly optimize sequences with respect to a fitness predictor by traversing a pre-trained generative model and iteratively updating its latent input [80]. In this setting, it is still a per-sequence optimization method that has to be re-initialized and executed again for each sequence to design. However, now the optimization is performed on the latent variable $\boldsymbol{z}$, which is used as input to a pre-trained (differentiable) generator $\mathcal{G}$ to sample sequence $\boldsymbol{x}$, i.e. $\boldsymbol{x} = \mathcal{G}(\boldsymbol{z})$. A similar method was developed by [61].

However, instead of training the manifold model $\mathcal{G}$ and the fitness predictor $\mathcal{P}$ separately, they jointly trained them. In this instance, $\mathcal{G}$ was a variational autoencoder (VAE), which allowed the predictor $\mathcal{P}$ to be fit directly on the latent space rather than on decoded sequence patterns as in [80].

2.4.3 Forward-Sampling of Generative Models

Motivated by the efficiency of parametric generative models for other types of high-dimensional data, a number of generative neural networks from the computer vision and language domains have been adapted to molecular sequence design. What is common to all these models is that they are unsupervised and distribution-capturing; they learn to encode the same input distribution as the training set. However, they cannot be directly conditioned to generate maximally fit sequences. To utilize these models for sequence optimization, a large candidate set of sequences $\{\boldsymbol{x}_{\text{new}}\}$ has to be sampled from the generative model $\mathcal{G}$ and ranked based on $\mathcal{P}$ (outlined in Algorithm 1).

Algorithm 1 Obtain optimized sequence set $\{\boldsymbol{x}_{\text{optimized}}\}$, given a generator $\mathcal{G}$, training data $\{\boldsymbol{x}_{\text{data}}\}$, a predictor $\mathcal{P}$ and a target number s of optimized sequences.

$\mathcal{G} \leftarrow \text{Train}(\{\boldsymbol{x}_{\text{data}}\})$

$\{\boldsymbol{x}_{\text{new}}\} \leftarrow \text{Generate}(\mathcal{G})$

$\{\boldsymbol{x}_{\text{optimized}}\} \leftarrow \text{Select-Best}(\mathcal{P}, \{\boldsymbol{x}_{\text{new}}\}, s)$

For example, Autoregressive Neural Networks have been trained on immune repertoire data to generate *de novo* antibodies [127]. An autoregressive network $\mathcal{R}$ is trained to autoencode the input pattern $\boldsymbol{x}$, but uses internal masking to make the prediction of $\boldsymbol{x}_{j+1}$ only depend on $\boldsymbol{x}_1$ to $\boldsymbol{x}_j$ [53]. For sequences where $\boldsymbol{x} \in \{0,1\}^{N \times M}$, the autoregressive network $\mathcal{R}$ outputs a matrix of logits $\mathcal{R}(\boldsymbol{x}) \in \mathbb{R}^{L \times M}$ which, after applying softmax (Equation 2.4), encodes a probability distribution of the reconstructed sequence. The network is trained to minimize a reconstruction loss with respect to training sequences. After training, new

patterns $\boldsymbol{x}$ are generated by sampling the first nucleotide from the predicted distribution $\boldsymbol{\sigma}(\mathcal{R}(\boldsymbol{x})_1)$, followed by sampling of $\boldsymbol{x}_{j+1}$ from $\boldsymbol{\sigma}(\mathcal{R}(\boldsymbol{x})_{j+1})$, where $\boldsymbol{x}_1$, ..., $\boldsymbol{x}_j$ have been filled in with the previously sampled nucleotides.

Similarly, Variational Autoencoders (VAEs) have been used to generate new stably folding protein sequences [34]. A VAE consists of two networks, an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. Given a one hot-coded input sequence $\boldsymbol{x} \in \{0,1\}^{L \times M}$, the encoder $\mathcal{E}$ outputs two vectors - a mean and a variance $\boldsymbol{\mu}, \boldsymbol{\epsilon} \in \mathbb{R}^d$. The decoder $\mathcal{D}$ reconstructs $\boldsymbol{x}$ by outputting a softmax relaxation $\boldsymbol{\sigma}(\mathcal{D}(\boldsymbol{z}))$ given a latent vector $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\epsilon})$ as input. The VAE is trained end-to-end to minimize a cross-entropy reconstruction loss and a KL-divergence loss to force $\boldsymbol{z}$ to follow $\mathcal{N}(0,1)^D$ [82]. New sequence patterns can be generated from the decoder $\mathcal{D}$ by randomly sampling latent vectors $\boldsymbol{z} \sim \mathcal{N}(0,1)^D$ and decoding them into sequences $\boldsymbol{x} = \boldsymbol{\sigma}(\mathcal{D}(\boldsymbol{z}))$. Beyond regular VAEs, conditional VAEs have been demonstrated on the task of designing novel protein sequences by providing an encoding of a target fold as input alongside the latent seed [56].

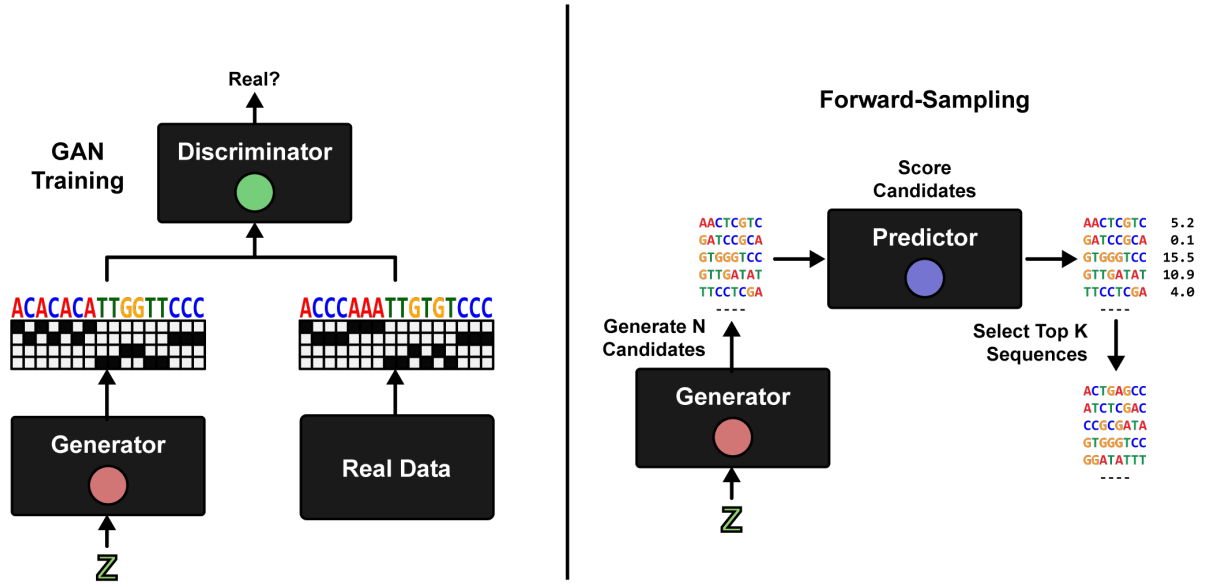

Figure 2.7: Training and Forward-Sampling of a Sequence-GAN.

Finally, Generative Adversarial Networks (GANs) have previously been used to design new, functionally equivalent enzymes [125]. GANs have also been used to design promoter sequences in E. Coli [158]. A GAN $\mathcal{G}$ is trained to minimize some cost $\mathcal{C}(\mathcal{D}(\text{data}), \mathcal{D}(\mathcal{G}(z)))$ such that an adversarial discriminator $\mathcal{D}$ can not distinguish between the real data and the distribution generated by $\mathcal{G}$ [55]. After training, $\mathcal{G}$ can be used to directly sample new sequence patterns $\boldsymbol{x} = \mathcal{G}(\boldsymbol{z})$ by randomly sampling new seeds $\boldsymbol{z} \in \mathcal{N}(0, 1)$. An illustration of the training- and sampling procedure of a GAN for DNA design is shown in Figure 2.7.

These generative models are distribution-capturing and thus support diverse sequence generation, as they naturally encode the same amount of entropy as was found in the training data. However, they do not take advantage of the fitness measurements associated with each sequence during training. Consequently, finding new high-fitness sequences can be very inefficient and may require a large number of samples. For example, in [59], the initial GAN trained on protein sequences from UniProt has a small probability of sampling patterns with antimicrobial fitness.

2.4.4 Iterated Forward-Sampling of Generative Models

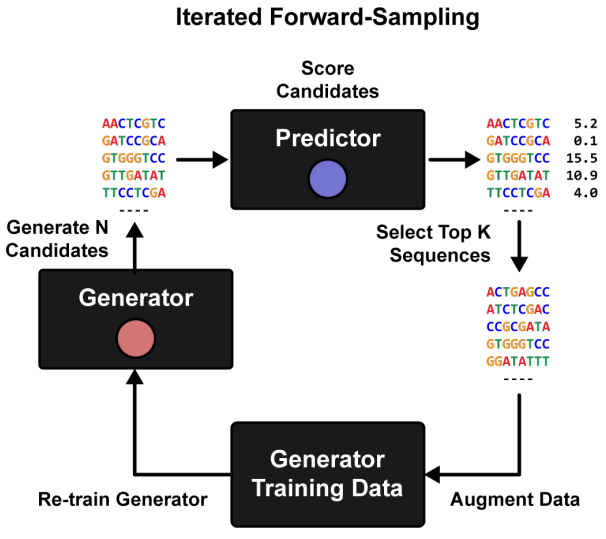

Figure 2.8: Iterated Forward-Sampling of a Generative Model (In-silico Directed Evolution).

Distribution-capturing generative models have also been combined with in-silico directed evolution in order to iteratively condition the model on high-fitness sequences. The high-level method, which is outlined in Algorithm 2, begins by training the generative model $\mathcal{G}$ on an initial data set. The trained model is then used to sample a set of new sequences, which are ranked and filtered by the fitness predictor $\mathcal{P}$. This optimized set of sequences is augmented with the original training data, and the generative model is re-trained on the combined dataset. This cycle is repeated until convergence (Figure 2.8).

This type of algorithm was first used to optimize a GAN for peptide sequences with antimicrobial properties [59] (Feedback-GANs). The original algorithm used a fixed predicted fitness threshold to filter the newly generated sequences at each training cycle, and the augmented training set was kept at a fixed size, throwing out the oldest sequences first.

Algorithm 2 Optimize generator $\mathcal{G}$ for predictor $\mathcal{P}$, given initial training data $\{\boldsymbol{x}_{\text{data}}\}$ of size N and a mixing factor λ of new sequences to include per round.

while $\mathcal{G}$ not converged **do**

$\mathcal{G} \leftarrow \text{Train}(\{\boldsymbol{x}_{\text{data}}\})$

$\{\boldsymbol{x}_{\text{new}}\} \leftarrow \text{Generate}(\mathcal{G})$

$\{\boldsymbol{x}_{\text{optimized}}\} \leftarrow \text{Select-Best}(\mathcal{P}, \{\boldsymbol{x}_{\text{new}}\}, \lambda \cdot N)$

$\{\boldsymbol{x}_{\text{data}}\} \leftarrow \{\boldsymbol{x}_{\text{data}}\}_{i=1}^{(1-\lambda)\cdot N} \cup \{\boldsymbol{x}_{\text{optimized}}\}_{i=1}^{\lambda \cdot N}$

end while

Brookes et al. (2019) later proposed an improved design method, Conditioning by Adaptive Sampling (CbAS) [18]. The method extends directed evolution of generative models by explicitly maintaining the marginal likelihood $p_{\mathcal{D}}(\boldsymbol{x})$ of generated sequences $\boldsymbol{x}$ with respect to the original dataset $\mathcal{D}_{\text{orig}}$, restricting $\mathcal{G}$ from drifting towards low-confidence regions in design space. Given a generative model capable of estimating $p_{\mathcal{D}}(\boldsymbol{x})$, CbAS initially trains the generator $\mathcal{G}$ on the original data $\mathcal{D}_{\text{orig}}$ and samples a new dataset $\mathcal{D}_{\text{new}} = \{\boldsymbol{x}_{new}\}$ using the same selection mechanism as in Algorithm 2. However, instead of treating each sampled sequence equally, the CbAS method re-weighs each sample $\boldsymbol{x}$ based on the likelihood ratio

$p_{\mathcal{D}_{\text{orig}}}(\boldsymbol{x})/p_{\mathcal{D}_{\text{new}}}(\boldsymbol{x})$ and on a confidence measure of $\mathcal{P}(\boldsymbol{x})$. Intuitively, if a newly generated sequence pattern is unlikely to be from the original dataset $\mathcal{D}_{\text{orig}}$, or if the predictor is not confident in that regime of design space, the sample is down-weighted.

Directed evolution solves the issues with sampling inefficiency highlighted in the previous section; after each cycle, the input distribution captured by the generative model will shift slightly towards a basin with high fitness score. However, it is not entirely clear how well methods based on directed evolution maintain sequence diversity in the generator.

2.5 Mask-based Interpretation of Neural Networks

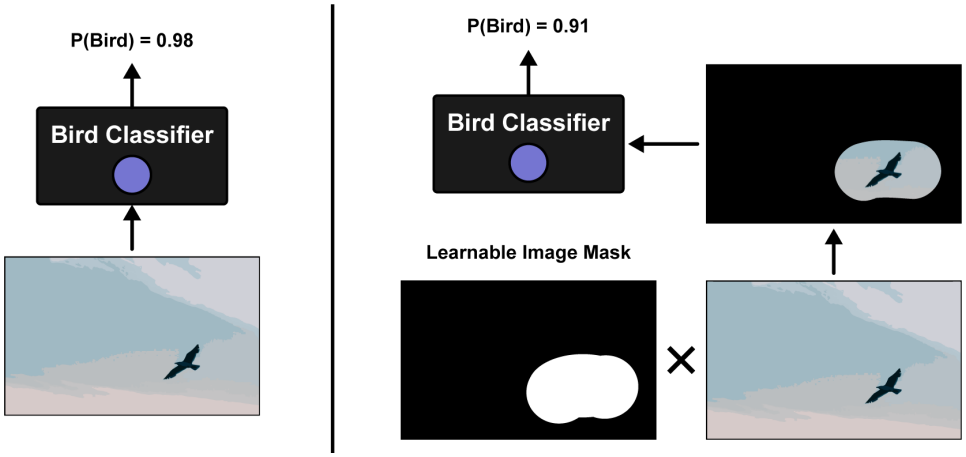

Figure 2.9: A learnable mask is used to preserve only the salient part of an input image.

Given a pre-trained, differentiable predictor $\mathcal{P}$ and an input pattern $\boldsymbol{x}$, *feature attribution* is the problem of crafting a set of importance scores $\boldsymbol{s}$ whose magnitudes reflect the contribution of each input feature on the prediction $\mathcal{P}(\boldsymbol{x})$. Many existing attribution methods for neural networks rely on local approximation to estimate feature importance [151], either basing their approximation on the gradients $\partial\mathcal{P}(\boldsymbol{x})/\partial\boldsymbol{x}_i$ [139, 170, 146, 150, 138, 103] or they train a local linear model [126]. A related family of methods, gradient-based activation maximization, has been used for both biological design and visualization [86, 132]. Nonetheless, these methods are not suitable for attributing the importance of an *existing* input pattern. Instead, they

find *de novo* sequence patterns that exhibit some important property learned by the predictor (e.g. maximizing input examples).

An entirely different approach to feature attributions was proposed simultaneously by Fong et al., (2017) and Dabkowski et al. (2017) [49, 38], where the objective is learn an input mask which either includes or excludes the smallest set of features such that the original prediction made by the neural network is either maximally reconstructed or maximally destroyed (Figure 2.9). Dabkowski et al. coined these two dual salient feature sets the Smallest Sufficient Region (SSR) and Smallest Destroying Region (SDR) respectively. By definition, a mask that fulfills its optimization objective overcomes issues inherent to local perturbation methods such as saturation artifacts. Masking methods were first developed in the context of computer vision [49, 48, 38], where the input images are masked by either fading or blurring. Similarly, feature selection methods based on masking for discrete input variables learn to retain a small set of features by either zeroing out unimportant dimensions, replacing them with a mean value or replacing them with counterfactual generated values [26, 169, 22, 21, 24].

Chapter 3

PREDICTING ALTERNATIVE POLYADENYLATION

Alternative Polyadenylation (APA) in the 3' UTR is a major driver of human transcriptome diversity by generating multiple RNA isoforms with distinct 3' UTR lengths. Even though individual regulators governing APA is well-characterized, we lack an accurate quantitative model for the cis-regulatory APA code. This hinders our ability to engineer polyadenylation signals for synthetic biology or to predict the impact of genetic variants. This chapter introduces a neural network for predicting APA from DNA sequence alone. The model (APARENT, APA REgression NeT) was trained on isoform expression data from over 3 million APA reporters. APARENT's predictions were highly accurate when tasked with inferring APA in synthetic and human 3' UTRs. This work was published in [17] in 2019 with myself and Dr. Nicholas Bogard as co-first authors. The majority of content presented here is from that publication.

3.1 An MPRA for 3' UTR APA

A set of minigene libraries totaling > 3 million unique 3' UTRs – each harboring a synthetic polyadenylation signal (PAS) – was constructed and assayed in a human cell-line (Figure 3.1). The entire experimental workflow was done by Dr. Nicholas Bogard of Seelig Lab.

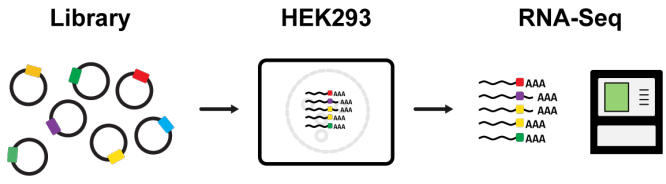

Figure 3.1: Minigene reporters are cloned from degenerate oligos and transiently transfected in human cell culture where they are expressed, alternatively polyadenylated and sequenced.

Each 3' UTR library expressed multiple PASs in a unique context—10 libraries were derived from seven different human 3'UTRs but with USE and DSE regions replaced by randomized sequence; another 2 libraries were fully degenerate, with either a canonical (AWTAAA) or doped (95% A, 2% G/C, 3% T) CSE (Figure 3.2). In addition to a randomized library, upstream of a distal bGH PAS, a reporter library was cloned with 1,085 PASs from the human genome. The 1,085 PASs came from a total of 817 genes and constitute the complete set of all PASs in the ClinVar database with at least one annotated variant. To quantify isoform expression, the libraries were transiently transfected into HEK293 cells and the expressed RNA was extracted and sequenced. In total, isoform expression data was collected for $3,372,030$ unique reporters with an average of 39 reads per reporter for random libraries and over $4,000$ reads per native human APA reporter. In addition, we were able to map exact cleavage-position counts for all libraries except Alien2 (due to poor read quality). 53% of reads mapped to proximal isoforms and 28% mapped to distal isoforms. The remaining reads (19%) mapped to de novo PASs in the degenerate regions.

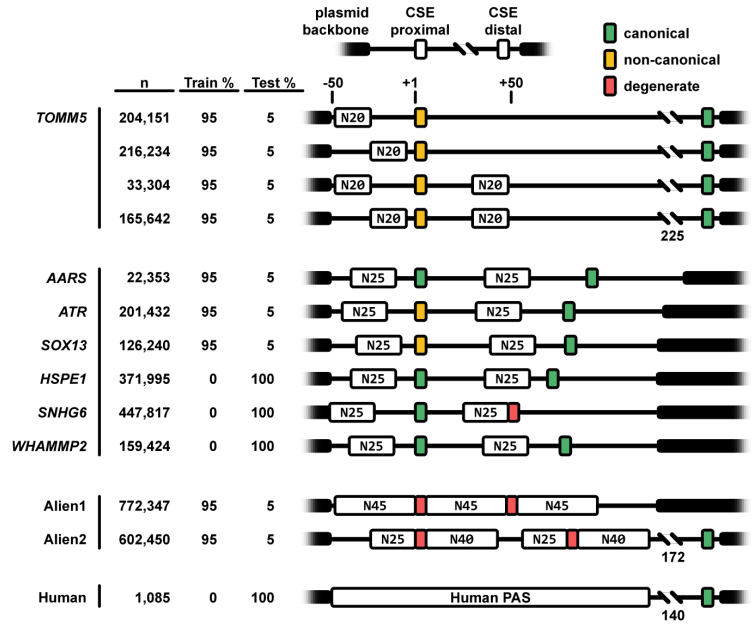

Figure 3.2: Multiple libraries that vary in structure and 3'UTR context (human is italicized; CSE sequence is denoted in legend; thick black bar, plasmid; thin, native sequence).

3.2 A Neural Network for Predicting 3' Cleavage and Polyadenylation

We then trained a deep neural network (DNN), APARENT, on this large synthetic dataset to predict the proximal APA isoform proportion of each variant UTR given its sequence as input. Assuming that competing PASs are independently and identically regulated when they do not physically overlap, the DNN will learn a general model of APA from the minigene libraries even though only the proximal PAS was randomized. We used 95% of the data from 9 out of 13 libraries for training (∼2.4 million variants), 2% for validation (∼50,000 variants) and 3% for testing (∼80,000). Four libraries (including the 1,085 human reference PASs) were held out entirely for independent testing. The best-performing model architecture consisted of two convolutional layers (70 filters of width 8 in layer 1 and 110 filters of width 6 in layer 2) interlaced with a maxpooling layer ($2x$ pool size), a fully connected layer (80 hidden units, 0.2 dropout rate), and a logistic regression output node (Figure 3.3, left).

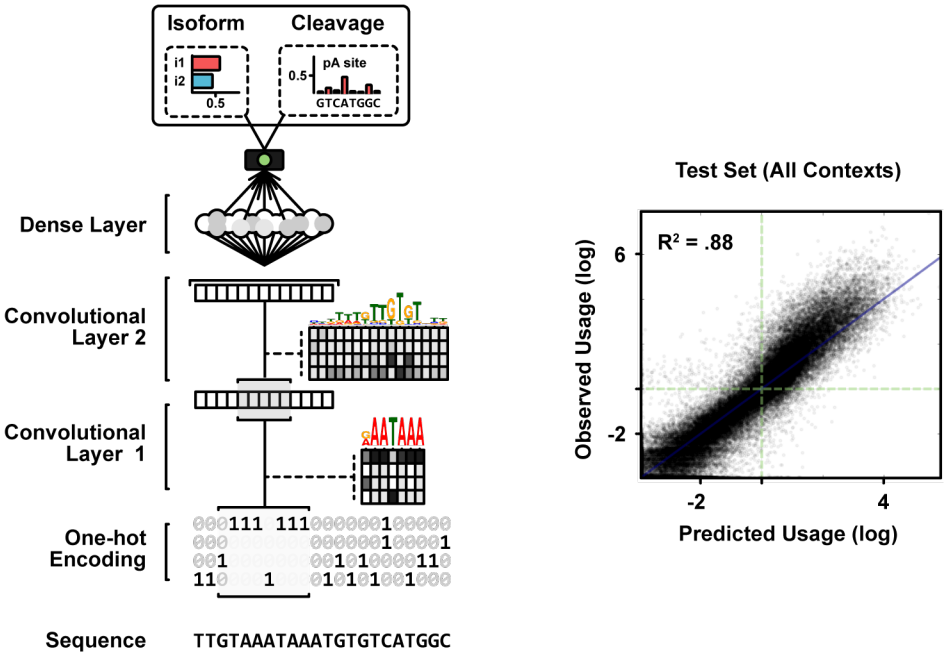

Figure 3.3: Left: APARENT takes a 1-Hot-coded PAS sequence as input to predict % proximal isoform. Right: Predicted vs. observed proximal isoform log odds of the test set.

Given the training data $\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$, where $\boldsymbol{x} \in \{0,1\}^{185\times 4}$ is a one-hot encoded representation of the proximal (degenerate) polyadenylation signal and $y \in [0,1]$ is the measured proximal isoform proportion, we trained APARENT to minimize the mean KL-divergence between targets $y^{(i)}$ and the predicted isoform proportions $\hat{y}^{(i)} = \mathcal{P}(\boldsymbol{x}^{(i)})$ (Equation 3.1; $\mathcal{P}$ is the neural network). We trained APARENT by mini-batch gradient descent until the validation error started to increase.

$$\mathcal{L}_{\text{train}}(\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}) = \frac{1}{N}\sum_{i=1}^{N} y^{(i)} \cdot \log\left(\frac{y^{(i)}}{\hat{y}^{(i)}}\right) + (1 - y^{(i)}) \cdot \log\left(\frac{1-y^{(i)}}{1-\hat{y}^{(i)}}\right) \tag{3.1}$$

To evaluate APARENT, we tested its ability to infer the proportion (log odds) of proximal isoform expression (isoform use). The DNN could accurately predict the isoform use of the combined test set ($R^2 = 0.88$; Figure 3.3, right). Next, we asked whether APARENT could generalize to entirely new UTR contexts. First, we predicted proximal isoform use of the three held-out libraries (HSPE1, SNHG6, and WHAMMP2). The predictions had a mean correlation and error comparable to the trained-on libraries (mean library $R^2 = 0.58$, Figure 3.4, right). Second, we evaluated APARENT's performance on the fully held-out library of $> 1,000$ reference human PASs. APARENT performed remarkably well ($R^2 = 0.69$, Figure 2D), confirming that the model generalizes from completely random to naturally occurring human 3' UTRs.

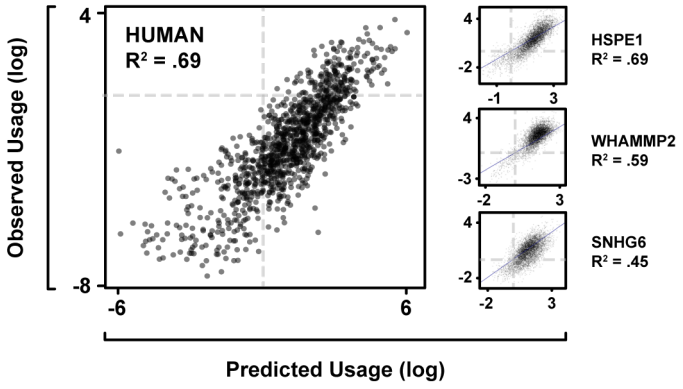

Figure 3.4: Predicted vs. observed proximal use on held-out random libraries and on the held-out native human PAS library.

Since RNA-seq provides information about the precise cut position, we investigated whether APARENT could be trained to predict the probability of cleavage occurring at any given nucleotide position. To directly learn the cleavage distribution for each PAS, we used a DNN almost identical to the isoform-based model but with a final network layer that outputs a categorical probability distribution of cleavage at any position across the sequence (Figure 3.5, left; a 186-way softmax output layer is used). To train this network, we minimized the mean KL-divergence between measured cleavage distributions $\boldsymbol{y}^{(i)} \in [0, 1]^{186}$ and predicted distributions $\hat{\boldsymbol{y}}^{(i)} = \mathcal{P}(\boldsymbol{x}^i)$ (Equation 3.2). The last position in the target vectors ($y_{186}^{(i)}$) corresponded to the total isoform proportion (cumulative cleavage) of the distal PAS. However, since exact cleavage positions were missing for a subset of MPRA sub-libraries, we instead minimized the original isoform proportion loss for those sequences.

$$\mathcal{L}_{\text{train}}(\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{186} y_k^{(i)} \cdot \log \left(\frac{y_k^{(i)}}{\hat{y}_k^{(i)}} \right) \tag{3.2}$$

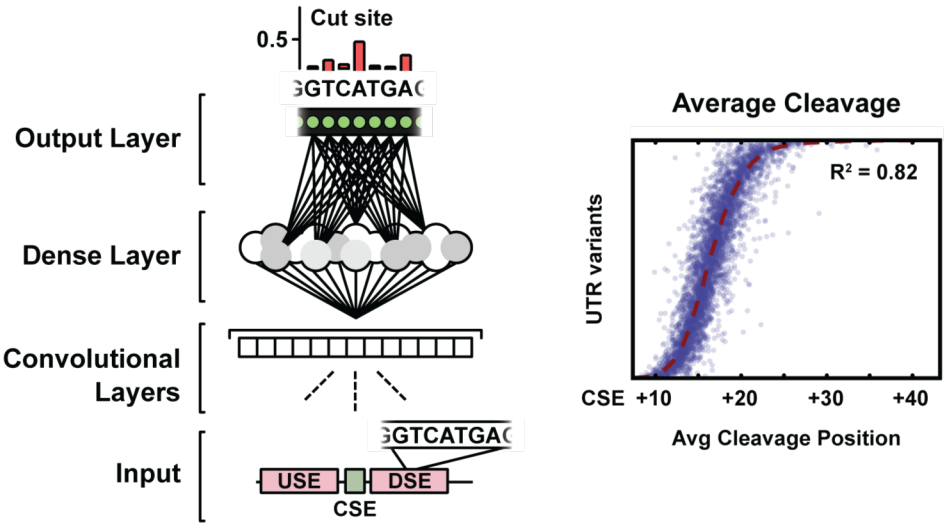

Figure 3.5: Left: APARENT's output layer predicts the probability of cleavage at each nucleotide. Right: Average predicted cut position on the Alien1 test set (x-axis). The y-axis sorts each sequence according to the observed cut position.

We evaluated this version of APARENT in two ways. First, we compared the predicted mean cut position with the observed mean position of every test set sequence (Figure 3.5, right). The two quantities correlated well ($R^2 = 0.82$ for library Alien1, $R^2 = 0.55$ for the held-out WHAMMP2 library). Second, we compared the area under the proximal region of the predicted cleavage curve, which corresponds to isoform abundance, against the previously predicted proximal proportions from the isoform-based network. The two predictions showed strong agreement ($R^2 = 0.95$).

3.3 Visualizing Learned cis-Regulatory Biology

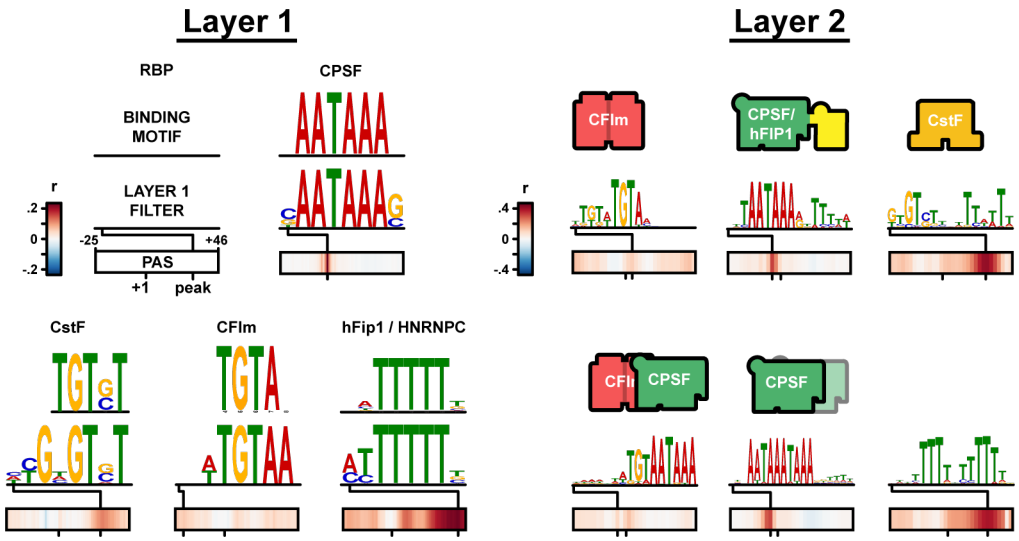

Figure 3.6: RBP motif logos and per-position effects (Pearson's r between filter activation and proximal use) learned in the first and second convolutional layers. Layer 2 filters are shown with their proposed effector interactions.

To visualize sequence determinants of PAS preference learned in the first convolutional layer, we generated consensus sequence logos which represented each filter's maximal activation [1] (Figure 3.6, left). Specifically, for filter k, the 5,000 input subsequences $\boldsymbol{s}_{j:j+8}^{(i)}$ of the test set that resulted in maximal filter activation $\max_{1 \leq i \leq N, 1 \leq j \leq 178} a_{k,j}^{(i)}$ were stacked into a position weight matrix (PWM) and used to generate a sequence logo for each filter. By purposefully

biasing the selection of sequences to be maximally activating examples, the consensus logos effectively visualized the canonical form of the motif learned by each filter. The position-specific effect that the k:th filter had on PAS usage was quantified by measuring the pearson correlation between activations $\{a_{k,j}^{(i)}\}_{i=1}^{N}$ and isoform use $\{y^{(i)}\}_{i=1}^{N}$ at each position j across the PAS [37]. We cross-referenced the PWMs with published binding data, as well as the Compendium of RNA Binding Protein motifs [124], using the Tomtom comparison tool [60], and we surveyed the top-scoring results for APA mediators.

We identified motifs matching known binding sites for all components of the core polyA machinery, including the CstF [23, 149] and CFIm25 binding motifs [107, 174], and the motif recognized by hFip1 – a subunit of CPSF [76, 106]. Importantly, the position-specific enhancing or suppressing effect assigned to each filter by APARENT is consistent with the known preferred binding position of its matched RBP.

Next, we generalized the method of [1, 37] to visualize features learned in the second layer. By estimating PWMs and position-dependent effects from sequences that give rise to maximal filter activations in the second layer (using the same method of accumulating maximally activating samples from the test set), we captured longer motifs or combinations of short motifs (Figure 3.6, right). Combinatorial effects of cis-elements have been suggested in earlier bioinformatics studies of PASs [29]. Filters that capture known motif interactions align well with the current understanding of APA regulation. For example, APARENT identifies the core hexamer AATAAA in combination with a downstream polyT motif to upregulate selection compared to the average effect of AATAAA, reflecting known interactions with hFip1 [76]. We also found novel motif interactions with strong net effect sizes. One of these filters is sensitive to a "multi-CSE" consisting of two overlapping canonical hexamers forming the 10-mer AAWAAAWAAA, estimated to substantially increase proximal selection compared to a single AATAAA.

Similarly, we could extend the above analysis to visualize determinants of cleavage sites. For the cleavage-based APARENT network, the model now outputs 186 cleavage probabilities rather than one single isoform probability. As such, each layer 1 filter activation at every

position was recorded and correlated with the magnitude of cleavage at every other position, resulting in set of two-dimensional plots that, together with their consensus sequence logos (which again were generated by accumulating maximally activating subsequences into a PWM), describe the regulatory impact of each filter motif on every cleavage site as a function of position (Figure 3.7).

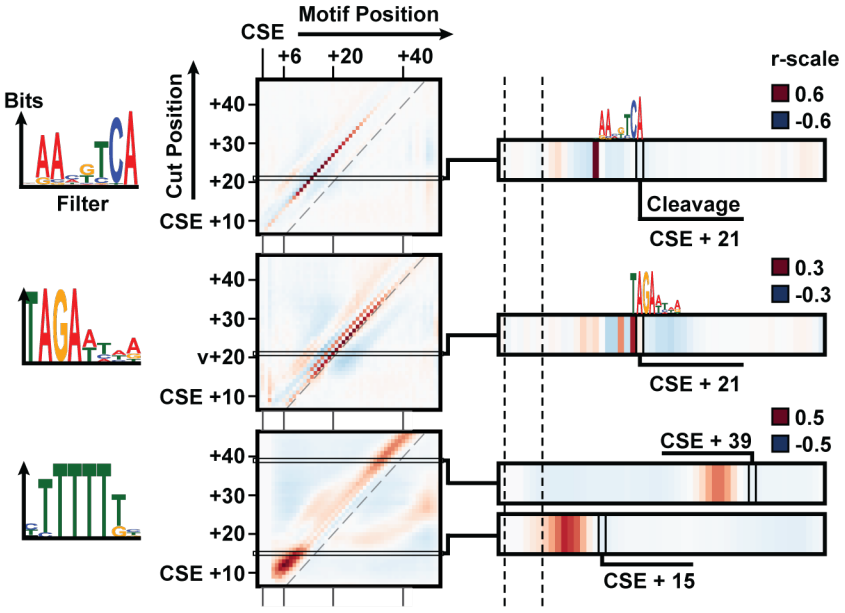

Figure 3.7: Selection of layer 1 filters validating known and newly discovered cleavage determinants. Each plot measures correlation between a filter activating at a certain position and cleavage occurring at some other position.

Consistent with earlier studies, the DNN filter heatmap characterizes the CA dinucleotide as the most favorable substrate for cleavage [42] (Figure 3.7, top). However, TA and GA are also identified as functional, albeit weaker, cleavage sites [25, 94] (Figure 3.7, middle). Beyond cut site dinucleotides, polyT motifs were found to be highly effective cleavage site regulators; polyT preceding a cut site enhances its usage (Figure 3.7, bottom).

When visualizing higher-layer features, we cannot resort to simply sampling the data (as we did for visualizing motifs in the first and second layer), because the very long length

of the total receptive field of dense layer neurons (186 nt) makes it unlikely that we would find maximally activating examples by chance. Instead, we developed a sequence optimization method based on gradient ascent (or *activation maximization*, AM) for visualization of higher-layer features in a DeepDream-style procedure whereby dense neurons are maximized by gradient ascent [116] (Figure 3.8, left). This optimization method will be covered in great detail in Chapter 4. In summary, we randomly initialize a matrix of nucleotide logits $\boldsymbol{l} \in \mathbb{R}^{185\times 4}$, which represents the PWM (Position Weight Matrix) of a sequence distribution, that we optimize according to Equation 2.1-4.1 (see Section 2.4.2) in order to maximize the activation $\boldsymbol{a}(\delta(\boldsymbol{l}))_j$ of a particular neuron j in the network. Here δ is a link function that transforms $\boldsymbol{l}$ into a suitable representation (in Section 2.4.2 this link function was the softmax σ, but in Chapter 4 we will introduce a more appropriate stochastic sampler that was used here). To visualize the canonical sequence of each neuron in the dense layer, we randomly initialized and optimized 50 patterns $\boldsymbol{l}$ by AM. The consensus sequences of these optimized patterns were collapsed into a PWM, resulting in a sequence logo for each neuron.

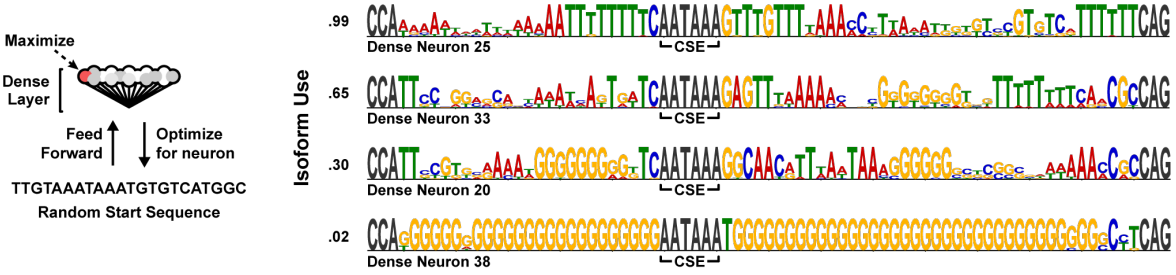

Figure 3.8: Gradient-based optimization performed on 50 random start sequences which maximize the activation of a particular dense neuron. The generated sequences are stacked into a PWM for each neuron (average predicted isoform use annotated).

As shown in Figure 3.8 (right), these sequence logos combine functional motifs identified in lower layers and describe global sequence determinants of PAS strength. The neurons specialize on different sequence subspaces, where G- and GC-rich sites are much less favored compared to sequences containing various conserved T- and GT-rich motifs.

3.4 Predicting Variants of APA in the Human Genome

Nucleotide variants near PASs in human 3' UTRs have been implicated in genetic disease, including IPEX syndrome, Thrombophilia, and Alpha and Beta Thalassaemia. Most disease-implicated APA variants identified so far act by disrupting the CSE hexamer [13, 52, 64]. While a number of cryptic pathogenic variants have been identified in the USE and DSE, it is largely unknown how many high-impact non-CSE mutations exist in disease-implicated PASs. Using APARENT, we set out to characterize the frequency and complexity of cryptic APA variants across the human genome. We curated and synthesized $> 1,000$ ClinVar variants – SNVs and InDels linked to medical phenotypes [88] – occurring within 50 nt upstream and 100 nt downstream of an annotated PAS, as well as $> 10,000$ variants from saturation mutagenesis of PASs in disease-implicated UTRs (ACMG, ClinVar, HGMD, Figure 3.9) [73, 89, 87, 148].

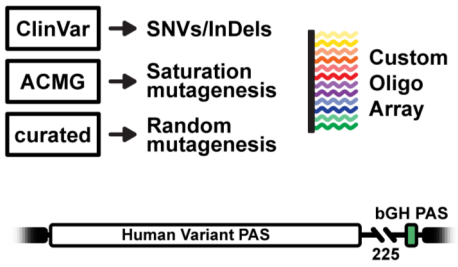

Figure 3.9: MPRA for measuring APA variants from ClinVar, HGMD, and ACMG genes.

The curated wildtype- and variant PASs were synthesized in a custom oligo array with 50 nt of USE sequence and 107 nt of DSE sequence from the original UTR context. To robustly identify variants after sequencing, we included 20 nt random barcodes 70 nt upstream of the CSE. To average out barcode effects, we synthesized 5 replicates per sequence with unique barcodes. The PASs were inserted into 3' UTR reporters upstream of a distal PAS, expressed in HEK293 cells on plasmids, and their APA profiles were measured by 3' sequencing. The array was experimentally replicated twice with identical barcodes. The

measured APA estimates agreed well among replicates with distinct barcodes, with almost perfect correlation between sequences with identical barcodes measured in the two separate experimental replicates ($R^2 = 0.98$). The average read count for each PAS (pooled across barcode replicates) was more than $4,000$.

Next, we estimated the log odds ratio (log fold change) $\text{LOR}(y_{\text{wt}}, y_{\text{var}})$ of each variant's isoform abundance y_{var} with respect to the wildtype abundance y_{wt} (Equation 3.3) and evaluated to what extent APARENT's predicted log-fold changes $\text{LOR}(\hat{y}_{\text{wt}}, \hat{y}_{\text{var}})$ agreed.

$$\text{LOR}(y_{\text{wt}}, y_{\text{var}}) = \log\left(\frac{y_{\text{var}}/(1-y_{\text{var}})}{y_{\text{wt}}/(1-y_{\text{wt}})}\right) \tag{3.3}$$

The model predictions agreed well with the measured fold changes of variant isoforms and could accurately classify the direction of change (Figure 3.10; $R^2 = 0.64$, correctly predicted direction = 90.3% for CSE and non-CSE variants, $R^2 = 0.51$, correctly predicted direction = 88.2% for non-CSE variants only). Of the $12,348$ measured variants, 757 non-CSE variants resulted in at least a 2-fold change in isoform abundance. The occurrence rate of such variants was about 3.7% in the USE and 8.7% in the DSE.

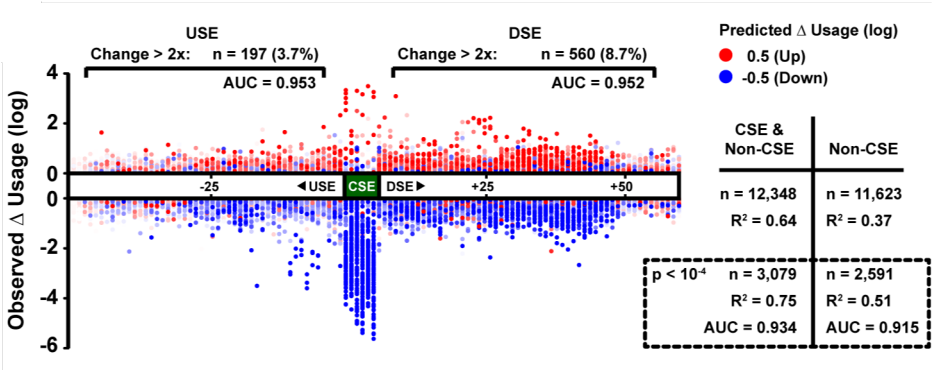

Figure 3.10: Measured isoform fold changes of all variants (y-axis). The x-axis denotes PAS position. The color indicates predicted fold change (blue/red = -/+ log odds ratio).

We then studied the composition of known pathogenic PASs through experimental and computational saturation mutagenesis (Figure 3.11). The TP53 1175A>C mutation transforms the canonical CSE AATAAA into the weaker AATACA and has been implicated in

basal cell carcinoma [147]. We identified two important regulatory regions in the DSE: the CstF binding site TGT[C/G]T and an HNRNPH binding site. We also find an SNV in the DSE that results in a 9-fold isoform change, as it interferes with CstF binding and cut site secondary structure. The PAS contains two additional annotated ClinVar variants, a benign variant 1422G¿C further downstream in the DSE and a VUS 1160T>G in the USE, both of which are predicted and measured to have negligible effects. Given that basal cell carcinoma is caused by APA misregulation in the TP53 UTR, our results suggest there are many more mutations that could potentially lead to disease (6.0% of USE and DSE variants are estimated to have at least a 2-fold change). APARENT's predictions agree well with the measured fold changes in TP53 ($R^2 = 0.86$ for all variants).

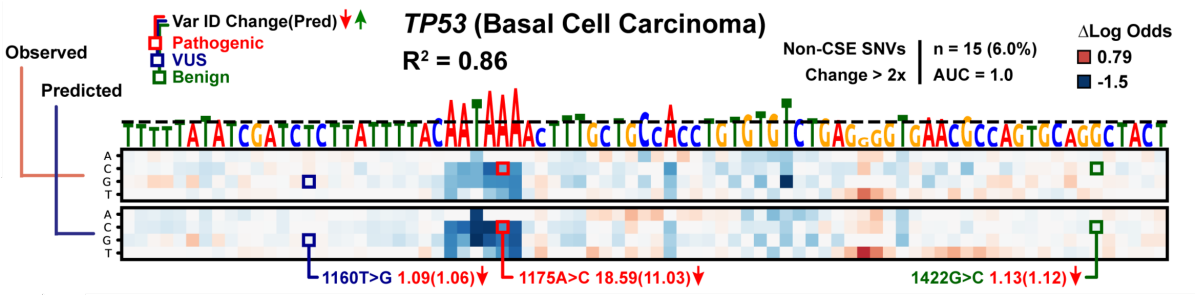

Figure 3.11: Saturation mutagenesis of the TP53 PAS. The heatmaps visualize measured and predicted isoform fold changes and are annotated with ClinVar variants.

Finally, we evaluated APARENT's ability to classify disruptive APA variants in comparison to related neural network models which have been trained solely on native transcriptomic data. We first compared to an LSTM-based network, DeeReCT-APA, which predicts relative isoform abundances for two or more competing polyadenylation signals [95]. This model was trained on mouse 3' sequencing data. We also compared APARENT to DeepPASTA, an ensemble of CNNs which had been trained on 3' sequencing data from multiple human tissue to predict relative isoform use [6]. We used the mean prediction from all DeepPASTA models for the comparison. There was also a single-input PAS predictor model from the DeepPASTA paper that we evaluated as well. Finally, we tested another APA model that had been trained

on an MPRA of 3' UTR PAS activity, PolyApredictor. This model, a CNN, was trained to predict RNA-to-DNA expression fold changes [141]. To score a particular APA variant using the DeeReCT-APA or DeepPASTA models, we predicted the wildtype- and variant proximal isoform abundances when competing with a fixed distal signal (a distal PAS from the training set), and used these predictions to values isoform odds ratio predictions. We could not use the (fixed) distal PAS from our dataset, since the models' input windows extended beyond the plasmid reporter 3' UTR. For PolyApredictor, the RNA/DNA fold change predictions were used to directly approximate the isoform odds ratios.

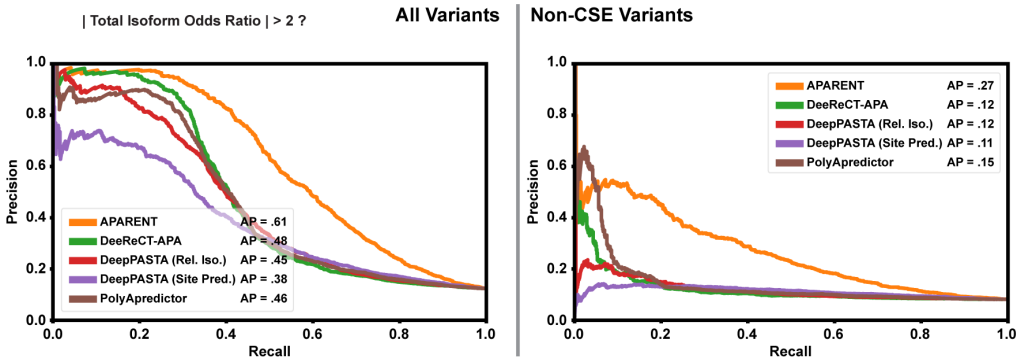

Figure 3.12: Precision-recall curves for classifying disruptive APA variants. Left: All variants. Right: Non-CSE variants only.

To label our dataset, we classified any variant as 'disruptive' if the absolute value of its isoform odds ratio with respect to the wildtype abundance was larger than 2: Disruptive = 1 if $\left|\frac{\hat{y}_{\text{var}}/(1-\hat{y}_{\text{var}})}{\hat{y}_{\text{wt}}/(1-\hat{y}_{\text{wt}})}\right| > 2$ else 0. Each model was then tasked with classifying variants as disruptive or not based on the absolute value of their predicted isoform odds ratios. When considering all variants in the dataset, the precision-recall curves indicate that all methods are fairly accurate, with APARENT having a moderately better Average Precision (AP) (Figure 3.12, left; AP = 0.61). However, many of the disruptive variants occur across the CSE hexamer, which are quite trivial to classify. When we consider only variants outside the CSE, the predictive power decreases for all models (Figure 3.12, right). For these more complex variants, APARENT is significantly more accurate than other models (AP = 0.27).

3.5 Improved Variant Prediction with Residual Learning

We recently trained an improved APA predictor on a re-processed version of our random APA MPRA data. Specifically, the original version of the MPRA dataset consisted of 185 nt long sequences, starting 50 nt upstream of the proximal CSE. However, for some of the sub-libraries, an additional random barcode was located from 70 to 50 nt upstream of the CSE. In the re-processed version of the data, we include 20 nt of additional sequence upstream of the CSE to capture these barcodes. Second, for some of the sub-libraries, the exact cleavage distributions were not estimated from the RNA-Seq data. Instead, these sub-libraries only included total proximal-to-distal isoform proportions. We later re-mapped the RNA-Seq reads to these sub-libraries and augmented the data with the missing cleavage distributions.

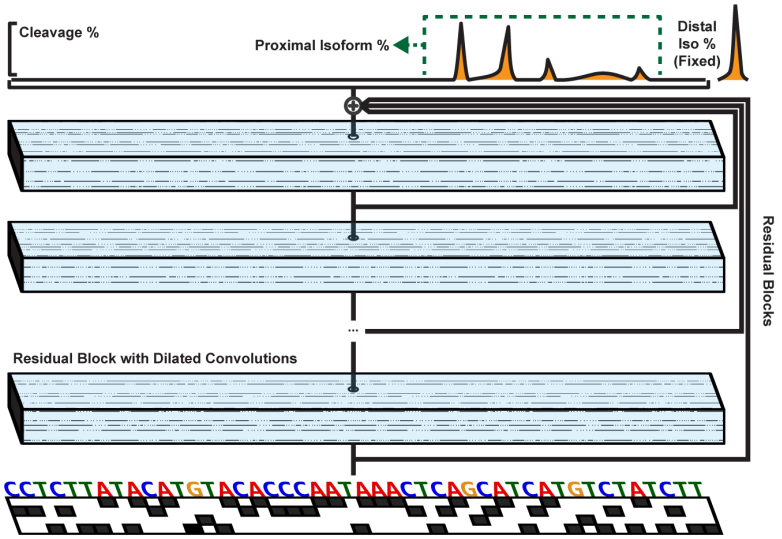

Figure 3.13: Residual neural network architecture. A one-hot encoded representation of the proximal PAS is transformed into a 3' cleavage distribution.

The network, which is illustrated in Figure 3.13 and is referred to as APARENT-ResNet, is based on residual blocks of dilated convolutions [63]. Architecturally, the model is similar to the SpliceAI network [68] as well as other residual network models [8]. Let $\mathcal{P}$ be the APARENT-ResNet model. As input, $\mathcal{P}$ receives a one-hot encoded sequence $\boldsymbol{x} \in \{0,1\}^{205\times 4}$,

which represents the proximal PAS, and a one-hot encoded indicator variable $\boldsymbol{l} \in \{0,1\}^{13}$ which indicates the source 3' UTR context from the MPRA. Internally, $\mathcal{P}$ consists of 7 **Residual Groups**, and each residual group is made up of 4 **Residual Blocks**. A residual block consists of two batch-normalized, ReLU-activated one-dimensional convolutional layers with a specific filter dilation rate. For our particular network, the 7 residual groups use the following sequence of dilation rates: 1, 2, 4, 8, 4, 2, 1. Between each residual group, there is a skip connection to the output layer, which mathematically performs an unweighted element-wise addition. Only $\boldsymbol{x}$ is passed through the series of residual blocks, producing in the end a single-channel vector of un-normalized cleavage scores $\boldsymbol{s}(\boldsymbol{x}) \in \mathbb{R}^{206}$ (Note that $\boldsymbol{s}$ has one position more than $\boldsymbol{x}$; this extra position represents the total isoform score of the distal signal). The library indicator variable $\boldsymbol{l}$ is multiplied with a position-specific weight vector $\boldsymbol{w}_j \in \mathbb{R}^{13}$ and linearly combined with $\boldsymbol{s}(\boldsymbol{x})$, producing new scores $\hat{s}(\boldsymbol{x}, \boldsymbol{l})_j = s(\boldsymbol{x})_j + \sum_{k=1}^{13} w_{jk} \cdot l_k$ which have effectively been scaled with a library-specific intercept. Finally, $\mathcal{P}$ produces a normalized 206-way categorical cleavage distribution $\hat{\boldsymbol{y}} \in [0,1]^{206}$ by applying the softmax transform (Equation 3.4).

$$\hat{y}_j = \mathcal{P}(\boldsymbol{x}, \boldsymbol{l})_j = \frac{e^{\hat{s}(\boldsymbol{x},\boldsymbol{l})_j}}{\sum_{k=1}^{206} e^{\hat{s}(\boldsymbol{x},\boldsymbol{l})_k}} \tag{3.4}$$

All residual blocks in APARENT-ResNet have 32 channels and all convolution filters are 3 positions wide. Note that there is no explicit sigmoid output representing the total proximal isoform proportion. Rather, the proximal isoform proportion is computed as the sum of cleavage probability mass 10–40 nt downstream of the start of the proximal CSE (which is located at position 70): $\hat{y}_{\text{iso}} = \sum_{j=80}^{110} \hat{y}_j$. We trained APARENT-ResNet with mini-batch SGD using the Adam optimizer in Keras [30, 81] with default parameters and batch size = 64. We trained the network for 5 full epochs, after which training was halted by early stopping. We minimized both the isoform proportion loss and cleavage distribution loss of Equation 3.1 and 3.2 (the predicted isoform proportions were computed by aggregating a subset of cleavage softmax outputs).

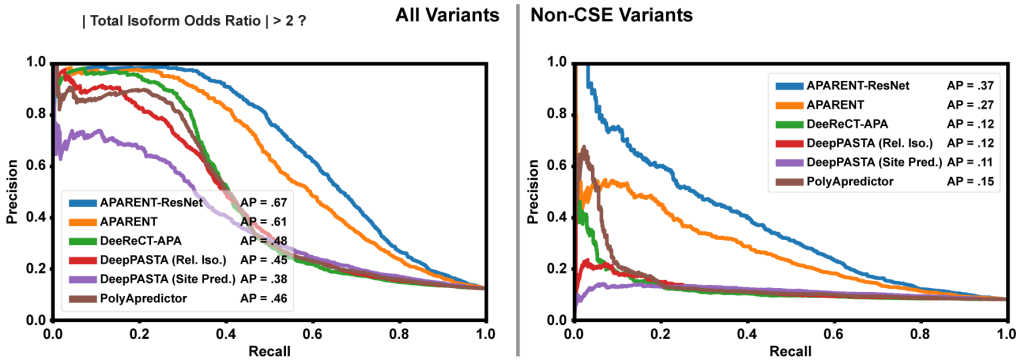

Figure 3.14: Precision-recall curves for classifying disruptive APA variants, including APARENT-ResNet in the comparison. Left: All variants. Right: Non-CSE variants only.

In Figure 3.14, we again test how well each model classifies 'disruptive' variants based on their predicted isoform log odds ratios, now including APARENT-ResNet in the benchmark. When classifying all variants, APARENT-ResNet had moderately better performance than APARENT (AP = 0.67 compared to 0.61). The performance gap increased further when considering only non-CSE variants (AP = 0.37 compared to 0.27).

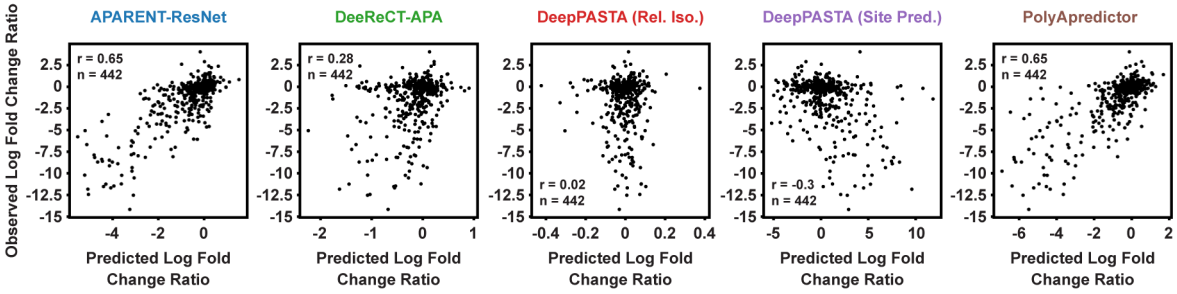

Figure 3.15: Comparison of predicted vs. measured total RNA/DNA log fold change ratios on the data from [141].

As external validation, we compared the models on how well their predictions correlated with the RNA/DNA log fold changes measured on the 3' UTR MPRA from [141]. We filtered this data to include only sequences that were in the test set of the PolyApredictor model. We also removed sequences that contained either a stretch of at least 10 consecutive

A's, or sequences containing the subsequence 'AGA' at position 41, as their measurements seemed to be influenced by artifacts. We also excluded the original APARENT model from further analysis, as APARENT-ResNet was clearly better on our own data from the previous benchmark.

We extracted the subset of sequences that were part of a scanning mutagenesis experiment, and matched each variant PAS with their wildtype sequence in order to calculate the RNA/DNA fold change ratios due to each variant. In figure 3.15, we compare each model's predicted log odds ratios (or fold changes) to the measurements. Here, APARENT-ResNet and PolyApredictor had identical correlations (spearman $r = 0.65$, $n = 442$), which was significantly higher than other models.

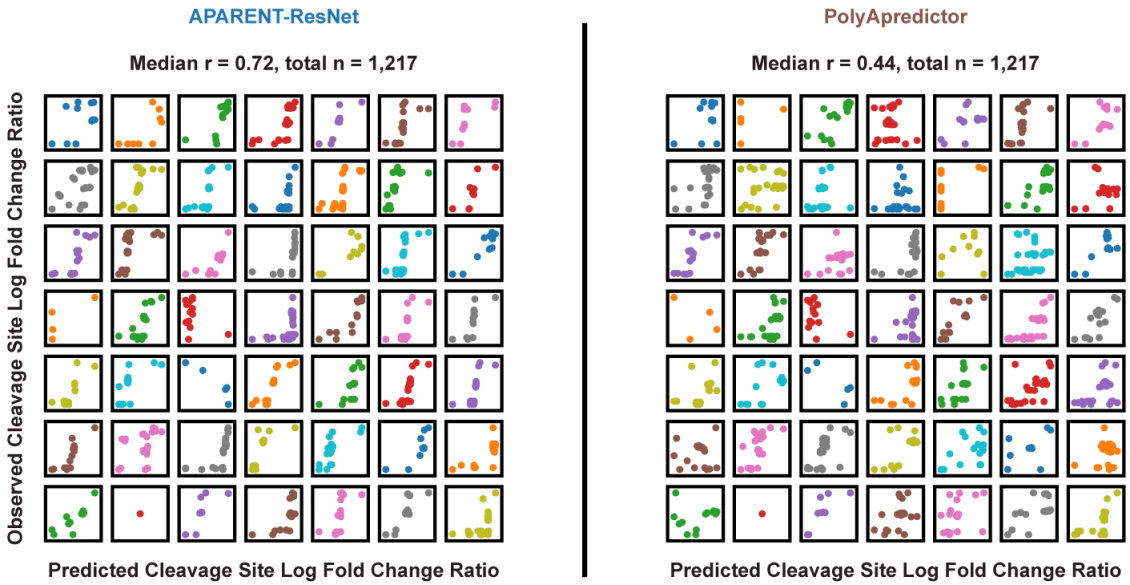

Figure 3.16: Comparison of predicted vs. measured cleavage site RNA/DNA log fold change ratios. Each plot corresponds to mutations of a single gene (UTR).

Finally, we compared APARENT-ResNet and PolyApredictor on the task of predicting RNA/DNA log fold change ratios due to variants at each wildtype cleavage site of every native UTR (Figure 3.16). APARENT-ResNet had considerably more accurate cleavage site predictions (median spearman $r = 0.72$, total $n = 1,217$) than PolyApredictor (me-

dian spearman $r = 0.44$). This is an interesting result given that their total RNA/DNA fold change correlations (from Figure 3.15) were identical. A plausible explanation is that APARENT-ResNet is better at inferring regulation of polyadenylation (hence the improved cleavage site predictions), whereas PolyApredictor is better at predicting regulatory effects of transcript stability (thus increasing its performance on total expression prediction).

Chapter 4

ENGINEERING DNA- AND PROTEIN SEQUENCES BY ACTIVATION MAXIMIZATION

This chapter introduces a method for biological sequence design based on gradient ascent, or *Activation Maximization* (AM). Part of this work was published in [17] with myself and Dr. Nicholas Bogard as co-first authors. An updated version of the method has been deposited to a pre-print server [100] where I was the first author. The majority of content presented here is from the pre-print paper [100].

4.1 Improved Activation Maximization for Sequence Design

While the gradient-based design method presented in Equation 2.2-2.5 (Section 2.4.2) is elegant, there are two issues that significantly limit its utility. First, the gradient in Equation 2.5 becomes vanishingly small for large values of $\boldsymbol{l}_{ij}$ (when $\sigma(\boldsymbol{l})_{ik} \approx 0$ or $\sigma(\boldsymbol{l})_{ij} \approx 1$), potentially halting convergence. Second, sequence-predictive neural networks have only been trained on discrete one-hot coded patterns and the predictive power of $\mathcal{P}$ may be poor on a continuous relaxation such as $\sigma(\boldsymbol{l})$, in particular at high entropies.

Following advances in gradient estimators for discretized neurons [12, 35], we first developed a version of the gradient ascent design method replacing the softmax transform σ of [80] (Equation 2.4) with a discrete, stochastic sampler δ (Equation 4.1). This version of the algorithm, which we called *Stochastic Sequence Backpropagation* (or SeqProp), was published in [17]. The main idea was to remove potential pathologies that result from passing continuous sequence relaxations as input to the predictor. Instead, SeqProp used one-hot-encoded sequences that were sampled from the (trainable) nucleotide softmax distributions.

$$\delta(\boldsymbol{l})_{ij} = \mathbb{1}_{(Z_i=j)} \tag{4.1}$$

In Equation 4.1, $Z_i \sim \sigma(\boldsymbol{l})_i$ is a randomly drawn nucleotide at the i:th position from the (softmax) probability distribution $\sigma(\boldsymbol{l})_i$. The nucleotide logits $\boldsymbol{l}_{ij}$ are thus used as parameters to N categorical distributions, from which we sample nucleotides $\{Z_i\}_{i=1}^{N}$ and construct a discrete, one-hot coded pattern $\delta(\boldsymbol{l}) \in \{0,1\}^{N \times M}$. While $\delta(\boldsymbol{l})$ is not directly differentiable, $\boldsymbol{l}$ can be updated based on the estimate of $\nabla_{\boldsymbol{l}} \mathcal{P}(\delta(\boldsymbol{l}))$ using straight-through (ST) approximation. Rather than using the original ST estimator of [12], SeqProp adopted an estimator with theoretically better properties from [31] where the gradient of $\delta(\boldsymbol{l})_{ij}$ was replaced by that of the softmax $\sigma(\boldsymbol{l})_{ij}$:

$$\frac{\partial \delta(\boldsymbol{l})_{ij}}{\partial \boldsymbol{l}_{ik}} = \frac{\partial \sigma(\boldsymbol{l})_{ij}}{\partial \boldsymbol{l}_{ik}} = \sigma(\boldsymbol{l})_{ik} \cdot (\mathbb{1}_{(j=k)} - \sigma(\boldsymbol{l})_{ij}) \tag{4.2}$$

Using discrete samples as input to $\mathcal{P}$ removes any pathologies from continuous input relaxations. But, as demonstrated in the next section, the convergence of SeqProp remained almost as slow as the softmax method. In work following the original SeqProp publication, we developed an improved version referred to as *Fast SeqProp*. This method was recently deposited to a pre-print server [100]. Here, we hypothesized that the main bottleneck of SeqProp originated from overly large and disproportionally scaled logit parameters $\boldsymbol{l}$. Fast Seqprop mitigated this problem by normalizing the logits across positions, i.e. by inserting a normalization layer between the trainable logits $\boldsymbol{l}$ and the sampling layer $\delta(\boldsymbol{l})$ (Figure 4.1).

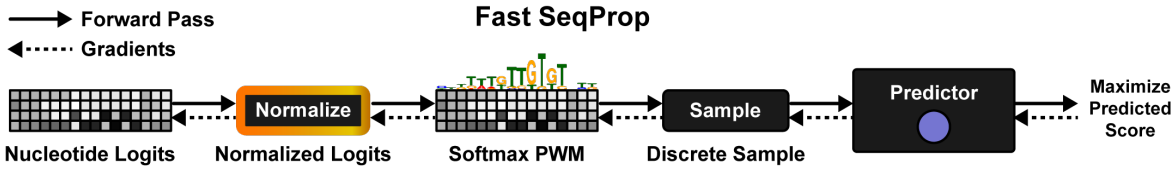

Figure 4.1: The Fast SeqProp pipeline. A normalization layer is prepended to a softmax layer, which is used as parameters to a sampling layer.

The Fast SeqProp method works as follows: For DNA sequence design, where the number of one-hot channels M is small ($M = 4$), we use a normalization scheme commonly referred to as *instance*-normalization. In this scheme, the nucleotide logits of each channel are nor-

malized independently across positions. Let $\bar{\mu}_j = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{l}_{ij}$ and $\bar{\varepsilon}_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{l}_{ij} - \bar{\mu}_j)^2}$ be the sample mean and deviation of logits for nucleotide j across all positions i. For each step of gradient ascent, we compute the normalized logits $\boldsymbol{l}_{ij}^{(\text{norm})}$ as:

$$\boldsymbol{l}_{ij}^{(\text{norm})} = \frac{\boldsymbol{l}_{ij} - \bar{\mu}_j}{\bar{\varepsilon}_j} \tag{4.3}$$

Since logits with zero mean and unit variance have limited expressiveness when used as parameters to a probability distribution, we associate each channel j with a global scaling parameter γ_j and offset β_j. Having an independent offset β_j per channel is particularly well-suited for DNA, as nucleotides are often associated with a global preferential bias. The scaled, re-centered logits are calculated as:

$$\boldsymbol{l}_{ij}^{(\text{scaled})} = \boldsymbol{l}_{ij}^{(\text{norm})} * \gamma_j + \beta_j \tag{4.4}$$

For protein sequence design, the number of one-hot channels M is considerably larger ($M = 20$), resulting in fewer samples per channel and noisier normalization statistics. Here we found that *layer*-normalization was more stable: We compute a global mean $\bar{\mu} = \frac{1}{N \cdot M}\sum_{i=1}^{N}\sum_{j=1}^{M} \boldsymbol{l}_{ij}$ and deviation $\bar{\varepsilon} = \sqrt{\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}(\boldsymbol{l}_{ij} - \bar{\mu}_j)^2}$, and use a single scale γ and offset β for all M channels.

Given the normalized and scaled logits $\boldsymbol{l}^{(\text{scaled})}$ as parameters for the nucleotide sampler δ defined in Equation 4.1, we maximize $\mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))$ with respect to $\boldsymbol{l}_{ij}$, γ_j and β_j (or γ and β in the context of proteins). Using the softmax ST estimator from Equation 4.2, we arrive at the following gradients:

$$\frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \boldsymbol{l}_{ij}} = \sum_{k=1}^{M} \frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \delta(\boldsymbol{l}^{(\text{scaled})})_{ik}} \cdot \frac{\partial \sigma(\boldsymbol{l}^{(\text{scaled})})_{ik}}{\partial \boldsymbol{l}_{ij}^{(\text{scaled})}} \cdot \gamma_j \cdot \frac{\partial \boldsymbol{l}_{ij}^{(\text{norm})}}{\partial \boldsymbol{l}_{ij}} \tag{4.5}$$

$$\frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \gamma_j} = \sum_{i=1}^{N}\sum_{k=1}^{M} \frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \delta(\boldsymbol{l}^{(\text{scaled})})_{ik}} \cdot \frac{\partial \sigma(\boldsymbol{l}^{(\text{scaled})})_{ik}}{\partial \boldsymbol{l}_{ij}^{(\text{scaled})}} \cdot \boldsymbol{l}_{ij}^{(\text{norm})} \tag{4.6}$$

$$\frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \beta j} = \sum_{i=1}^{N}\sum_{k=1}^{M} \frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \delta(\boldsymbol{l}^{(\text{scaled})})_{ik}} \cdot \frac{\partial \sigma(\boldsymbol{l}^{(\text{scaled})})_{ik}}{\partial \boldsymbol{l}_{ij}^{(\text{scaled})}} \tag{4.7}$$

The normalization removes logit drift by keeping the values proportionally scaled and centered at zero ($\mathrm{E}[l_{ij}^{(\text{norm})}] = 0$, $\mathrm{Var}[l_{ij}^{(\text{norm})}] = 1$). Furthermore, the scaling parameter γ_j

adaptively adjusts the sampling entropy to control global versus local optimization. This can be deduced by studying the components of $\frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \gamma_j}$ in Equation 4.6:

1. $\frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \delta(\boldsymbol{l}^{(\text{scaled})})_{ik}}$ is positive for nucleotides which increase fitness and negative otherwise.

2. $\frac{\partial \sigma(\boldsymbol{l}^{(\text{scaled})})_{ik}}{\partial \boldsymbol{l}_{ij}^{(\text{scaled})}}$ is positive when $j = k$ and negative otherwise.

3. $\boldsymbol{l}_{ik}^{(\text{norm})}$ is positive only when we are likely to sample the corresponding nucleotide.

Here, the product of the first two terms, $\frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \delta(\boldsymbol{l}^{(\text{scaled})})_{ik}} \cdot \frac{\partial \sigma(\boldsymbol{l}^{(\text{scaled})})_{ik}}{\partial \boldsymbol{l}_{ij}^{(\text{scaled})}}$, is positive if $j = k$ and nucleotide j raises fitness or if $j \neq k$ and nucleotide k lowers fitness. Hence, the gradient for γ_j increases when our confidence $\boldsymbol{l}_{ij}^{(\text{norm})}$ in nucleotide j at position i is *consistent* with its impact on fitness, such that $\text{sign}\left(\sum_{k=1}^{M} \frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \delta(\boldsymbol{l}^{(\text{scaled})})_{ik}} \cdot \frac{\partial \sigma(\boldsymbol{l}^{(\text{scaled})})_{ik}}{\partial \boldsymbol{l}_{ij}^{(\text{scaled})}}\right) = \text{sign}\left(\boldsymbol{l}_{ij}^{(\text{norm})}\right)$. Conversely, *inconsistent* nucleotides decrement the gradient. At the start of optimization, γ_j is small, leading to high PWM entropy and large jumps in sequence design space. As optimization progresses and we start sampling consistent nucleotides (they are likely to be sampled and they improve fitness), the entropy gradient $\frac{\partial \mathcal{P}(\delta(\boldsymbol{l}^{(\text{scaled})}))}{\partial \gamma_j}$ turns positive and γ_j grows larger. Larger γ_j lowers the entropy and leads to more localized optimization. However, if we sample sufficiently many inconsistent nucleotides (likely nucleotides that decrease fitness or unlikely nucleotides that improve fitness), the gradient of γ_j may turn negative, again raising entropy and promoting global exploration.

Note that, in the context of protein design where we have a single scale γ and offset β, the gradients of Equation 4.6 and 4.7 are additively pooled across all M channels. The argued benefits of instance-normalization thus holds true for layer-normalization as well.

4.2 Maximizing Nucleic Acid Sequence-Predictive Neural Networks

Fast SeqProp was first evaluated on the task of maximizing the classification or regression scores of five DNA- or RNA-level deep neural network predictors: (1) *DragoNN*, a model trained on ChIP-seq data to predict Transcription Factor (TF) binding (in this case binding

of SPI1), (2) *DeepSEA* [173], which predicts multiple TF binding probabilities and chromatin modifications (we use it here to maximize the probability of CTCF binding in the cell type Dnd41), (3) *APARENT* [17], which predicts alternative polyadenylation isoform abundance given an input polyadenylation signal, (4) MPRA-DragoNN [114], a neural network trained to predict transcriptional activity of short enhancer sequences and, finally, (5) *Optimus 5'* [130], which predicts ribosomal load (translation efficiency) of 5' UTR sequences.

We compared Fast SeqProp to the previous versions of the algorithm, namely: (1) *PWM* – the original method with continuous softmax-relaxed inputs [80] and (2) *SeqProp* – the categorical sampling method described in [17] using the (non-normalized) softmax straight-through gradient estimator. We also tested a logit-normalized version of the softmax-relaxed method, *Fast PWM*, in order to disentangle the individual contributions of the normalization scheme and the sampling scheme with respect to the baseline PWM method. We define the optimization loss (or the 'train' loss) as:

$$\mathcal{L}_{\text{train}}(\boldsymbol{l}) = -\mathcal{P}(\boldsymbol{x}(\boldsymbol{l})) \tag{4.8}$$

For PWM and Fast PWM, $\boldsymbol{x}(\boldsymbol{l}) = \sigma(\boldsymbol{l})$. For SeqProp and Fast SeqProp, $\boldsymbol{x}(\boldsymbol{l}) = \delta(\boldsymbol{l})$. The actual loss (or 'test' loss) is evaluated on the basis of discrete sequence samples drawn from the nucleotide sampler $\delta(\boldsymbol{l})$, regardless of design method:

$$\mathcal{L}_{\text{test}}(\{\boldsymbol{l}^{(k)}\}_{k=1}^{K}) = -\frac{1}{K}\frac{1}{S}\sum_{k=1}^{K}\sum_{s=1}^{S}\mathcal{P}(\delta(\boldsymbol{l}^{(k)})^{(s)}) \tag{4.9}$$

Here S refers to the number of samples drawn from each softmax sequence $\sigma(\boldsymbol{l}^{(k)})$ at every weight update t, and K is the number of independent optimization runs. In all experiments, we set $K = 10$ and $S = 10$. Unless otherwise stated, the optimization trajectories plotted in this section are based on the test loss of Equation 4.9.

Figure 4.2 shows the result of using the design methods to generate maximally scored sequences for each of the five DNA-based predictors. Fast SeqProp converges to 95% - 99% of its minimum test loss within $2,000$ logit updates, and reaches 50% of the minimum loss after only 200 updates for all predictors except MPRA-DragoNN and Optimus 5'. In contrast,

PWM and SeqProp do not converge within 20,000 updates. Fast SeqProp converges to up to 3-fold better optima than all other compared methods. In fact, Fast SeqProp reaches the same or better optima in 200 updates than the competing methods reach in 20,000 updates for DragoNN, MPRA-DragoNN and DeepSEA, marking a 100x speedup. For Optimus 5' and APARENT, the speedup is 20x-50x.

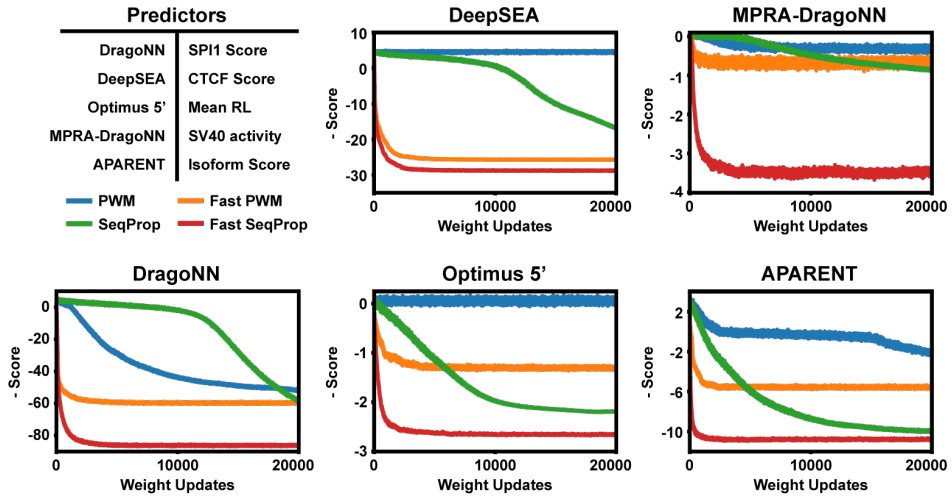

Figure 4.2: Maximizing the predictors DragoNN (SPI1), DeepSEA (CTCF Dnd41), MPRA-DragoNN (SV40), Optimus 5' and APARENT.

Figure 4.3 compares example sequence optimizations of the PWM and Fast SeqProp methods. As can be seen, even after 20,000 updates, the PWM method has not converged for most of the tested predictors. In contrast, Fast SeqProp contains plenty of *cis*-regulatory motifs in the converged sequences generated by Fast SeqProp.

For example, when maximizing APARENT, Fast SeqProp generates multiple CFIm binding motifs, dual CSE hexamers, and multiple cut sites with CstF binding sites. These are all regulatory motifs known to enhance cleavage and polyadenylation by stimulating recruitment of the polyadenylation machinery [54, 137, 43, 154]. For DeepSEA, Fast SeqProp generates four CTCF binding sites. For MPRA-DragoNN, we identify both CRE- and a CTCF binding sites embedded within a GC-rich context, which aligns well with what we might expect to

find in a strong enhancer [79, 45]. Finally, for Optimus 5', Fast SeqProp generates a T-rich sequence with multiple in-frame (IF) uAUGs. These determinants were previously found to improve ribosome loading [130]. See the Supplementary Information (Figure S2) for additional visualizations comparing the PWM and Fast SeqProp methods during different stages of optimization.

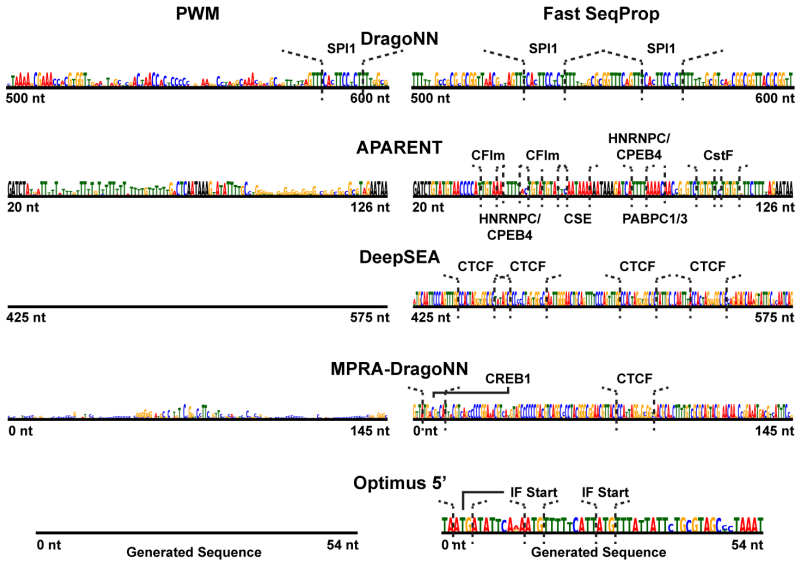

Figure 4.3: Example softmax sequences generated by the PWM and Fast SeqProp methods after 20,000 updates of gradient ascent with default optimizer parameters (Adam).

4.3 Pathologies of Softmax Relaxation

Some predictors, having been trained only on discrete one-hot coded patterns, were hypothesized to have poor predictive power on continuous-valued softmax sequence relaxations (input to PWM). To test this, we measured both the training loss $\mathcal{L}_{\text{train}}$ (which is based on the softmax sequence input $\sigma(\boldsymbol{l})$) and test loss $\mathcal{L}_{\text{test}}$ (which is based on discrete samples $\delta(\boldsymbol{l})$) when maximizing MPRA-DragoNN.

Indeed, maximizing MPRA-DragoNN with PWM leads to a considerably overestimated predictor score on the softmax input (Figure 4.4, left), as the training loss is more than 6-fold lower than the test loss. Using Fast SeqProp, on the other hand, the training and test losses

are identical (Figure 4.4, right). While the training loss is 2x higher than the training loss of PWM, the test loss is more than 3x lower.

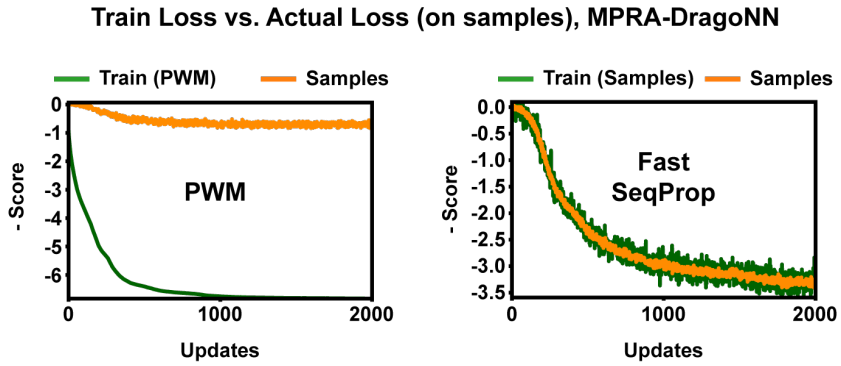

Figure 4.4: Maximizing MPRA-DragoNN with PWM (left) or Fast SeqProp (right).

4.4 Comparison to Evolutionary Algorithms and Simulated Annealing

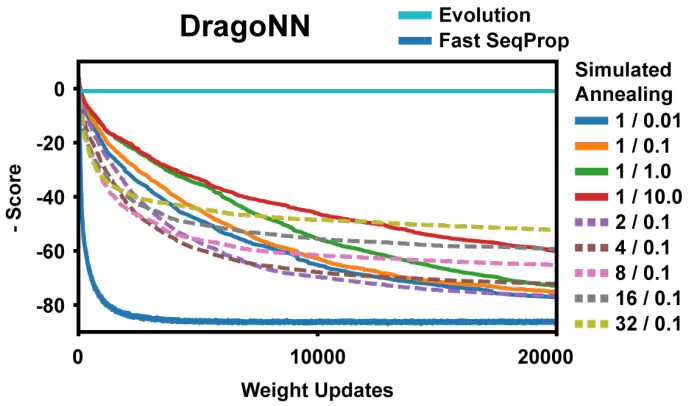

Figure 4.5: Maximizing DragoNN SPI1. Simulated Annealing is tested at several parameter configurations (number of substitutions per step / initial temperature).

Next, Fast SeqProp was compared to the discrete nucleotide-swapping search algorithm 'Evolved' as well as the global optimization heuristic Simulated Annealing ('SA') described in Section 2.4.1. The methods were benchmarked on the DragoNN maximization task. The results in Figure 4.5 show that Fast SeqProp significantly outperforms SA; the best fitness

score that SA reached in $20,000$ iterations was reached by Fast SeqProp in less than $1,000$ iterations, marking a 20x speed-up. Interestingly, increasing the number of substitutions at each step of SA initially increases convergence, but results in worse optima. Furthermore, the pairwise nucleotide-swapping search, Evolved, never converges, suggesting that DragoNN maximization is a challenging optimization problem.

4.5 Comparing Different Gradient Estimators

We next compared the performance of Fast SeqProp to a version of the method using the Gumbel distribution for sampling and backpropagation instead of the Softmax ST estimator [70] (temperature $\tau = 0.1$). While the Gumbel variant of the design method reached the same optima as Fast SeqProp on the DragoNN task, it converged slower (Figure 4.6, left). Importantly, same as PWM and SeqProp, the Gumbel design method benefited substantially from logit normalization ('Gumbel-IN' in the figure). We also compared Fast SeqProp to a version using the original ST estimator $\frac{\partial \delta(\boldsymbol{l})_{ij}}{\partial \boldsymbol{l}_{ij}} = 1$. As shown in Figure 4.6 (right), the Softmax ST estimator reaches much better optima. Sampling multiple sequences $\{\delta(\boldsymbol{l})^{(s)}\}_{s=1}^{S}$ at each update and walking down the average gradient $\frac{1}{S}\sum_{s=1}^{S} \nabla_{\boldsymbol{l}} \mathcal{P}(\delta(\boldsymbol{l})^{(s)})$ slightly speeds up convergence, but does not improve optima.

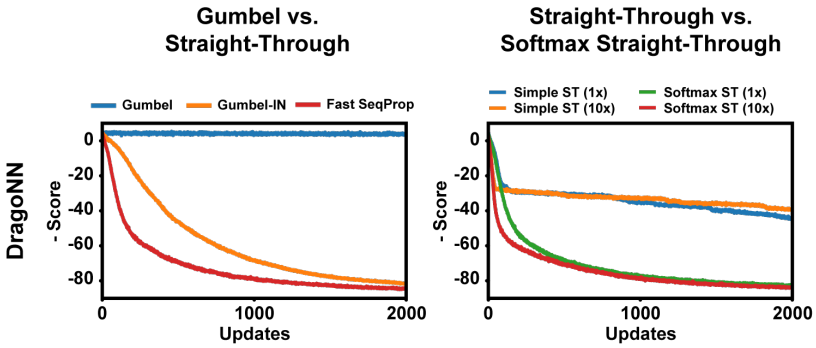

Figure 4.6: Comparing Fast SeqProp to a version of the same method using the Gumbel distribution (left) or standard straight-through approximation (right).

4.6 Regularized Activation Maximization using Variational Autoencoders

Finding regulatory logic in the sequences produced by activation maximization in Figure 4.3 is a good indication that we generate patterns with biological meaning. However, there is the potential issue of overfitting to the predictor during sequence optimization, as the model may lose its accuracy when drifting out of the training data distribution to maximize predicted fitness. Likelihood models such as variational autoencoders (VAEs) [82] have previously been used to prevent drift to low-confidence regions of design space [61, 18]. By training VAEs on sample from the same data as the predictor, we can use them to estimate and maintain the likelihood of candidate designs. Here, because VAEs are end-to-end differentiable, we can integrate them directly into the optimization objective as a regularizer (Figure 4.7, left).

In summary, we extend the original optimization objective (Eq. 2.1) by passing the sampled one-hot pattern $\delta(\boldsymbol{l})$ to the VAE and penalize the pattern based on its VAE-estimated log-likelihood, $\log p_{\text{VAE}}(\delta(\boldsymbol{l}))$, using ST approximation for backpropagation. The log-likelihood $\log p_{\text{VAE}}(\delta(\boldsymbol{l}))$ is approximated in the forward pass as follows: The one-hot pattern $\delta(\boldsymbol{l})$ is used as input to the VAE encoder $\mathcal{E}$, producing a mean vector $\mu(\boldsymbol{l}) \in \mathbb{R}^d$ and a log-variance vector $\epsilon(\boldsymbol{l}) \in \mathbb{R}^d$. We sample S latent vectors $\boldsymbol{v}^{(s)} \sim \mathcal{N}(\mu(\boldsymbol{l}), \epsilon(\boldsymbol{l}))$. Each vector $\boldsymbol{v}^{(s)}$ is decoded by the VAE decoder $\mathcal{D}$ into a reconstructed matrix of nucleotide logits $\boldsymbol{r}^{(s)} = \mathcal{D}(\boldsymbol{v}^{(s)})$ and passed through a softmax layer σ to obtain S reconstructed softmax-relaxed PWMs $\sigma(\boldsymbol{r}^{(s)})$. Next, we use importance-weighted inference to estimate the log likelihood $\log p_{\text{VAE}}(\delta(\boldsymbol{l}))$ [117, 82, 16]:

$$\log p_{\text{VAE}}(\delta(\boldsymbol{l})) \approx \log \left[\frac{1}{S} \cdot \sum_{s=1}^{S} p_s(\delta(\boldsymbol{l})) \right] = \log \left[\frac{1}{S} \cdot \sum_{s=1}^{S} p_{\mathcal{D}}(\delta(\boldsymbol{l}) | \boldsymbol{v}^{(s)}) \cdot \frac{p_{\mathcal{N}(0,1)}(\boldsymbol{v}^{(s)})}{p_{\mathcal{N}(\mu(\boldsymbol{l}),\epsilon(\boldsymbol{l}))}(\boldsymbol{v}^{(s)})} \right] \quad (4.10)$$

The summands may be very small at the start of optimization. To maintain numerical stability, we calculate Equation 4.10 in log space using the log-sum-exp trick:

$$\log p_{\text{VAE}}(\delta(\boldsymbol{l})) \approx \max_{s=1}^{S} \left[\log p_s(\delta(\boldsymbol{l})) \right] + \log \left[\sum_{s=1}^{S} e^{\log p_s(\delta(\boldsymbol{l})) - \max_{t=1}^{S}(\log p_t(\delta(\boldsymbol{l})))} \right] - \log S \quad (4.11)$$

We compute the importance-weighted probability $p_s(\delta(\boldsymbol{l}))$ of each sample s in log space:

$$\begin{aligned}\log p_s(\delta(\boldsymbol{l})) &= \log\left[p_{\mathcal{D}}(\delta(\boldsymbol{l})|\boldsymbol{v}^{(s)}) \cdot \frac{p_{\mathcal{N}(0,1)}(\boldsymbol{v}^{(s)})}{p_{\mathcal{N}(\mu(\boldsymbol{l}),\epsilon(\boldsymbol{l}))}(\boldsymbol{v}^{(s)})}\right] \\ &= \log p_{\mathcal{D}}(\delta(\boldsymbol{l})|\boldsymbol{v}^{(s)}) + \log p_{\mathcal{N}(0,1)}(\boldsymbol{v}^{(s)}) - \log p_{\mathcal{N}(\mu(\boldsymbol{l}),\epsilon(\boldsymbol{l}))}(\boldsymbol{v}^{(s)})\end{aligned} \tag{4.12}$$

Here, $\log p_{\mathcal{D}}(\delta(\boldsymbol{l})|\boldsymbol{v}^{(s)})$ is the decoder reconstruction probability of PWM $\sigma(\boldsymbol{r}^{(s)})$:

$$\log p_{\mathcal{D}}(\delta(\boldsymbol{l})|\boldsymbol{v}^{(s)}) = \sum_{i=1}^{N}\sum_{j=1}^{M}\left[\delta(\boldsymbol{l})_{ij} \times \log \sigma(\boldsymbol{r}^{(s)})_{ij}\right] \tag{4.13}$$

$\log p_{\mathcal{N}(0,1)}(\boldsymbol{v}^{(s)})$ is the density of latent sample $\boldsymbol{v}^{(s)}$ under the standard normal distribution:

$$\log p_{\mathcal{N}(0,1)}(\boldsymbol{v}^{(s)}) = -\frac{1}{2} \cdot \sum_{d=1}^{D}\left[(\boldsymbol{v}_d^{(s)})^2 + \log(2\pi)\right] \tag{4.14}$$

Finally, $\log p_{\mathcal{N}(\mu(\boldsymbol{l}),\epsilon(\boldsymbol{l}))}(\boldsymbol{v}^{(s)})$ is the density of $\boldsymbol{v}^{(s)}$ under the importance sampling distribution:

$$\log p_{\mathcal{N}(\mu(\boldsymbol{l}),\epsilon(\boldsymbol{l}))}(\boldsymbol{v}^{(s)}) = -\frac{1}{2} \cdot \sum_{d=1}^{D}\left[\left(\frac{\boldsymbol{v}^{(s)} - \mu(\boldsymbol{l})}{\epsilon(\boldsymbol{l})}\right)_d^2 + \log(2\pi) + \log \epsilon(\boldsymbol{l})_d\right] \tag{4.15}$$

Note that all quantities in Equations 4.11-4.15 are differentiable with respect to $\delta(\boldsymbol{l})$ using the reparameterization trick of [82]. Given this differentiable estimate of $\log p_{\text{VAE}}(\delta(\boldsymbol{l}))$ above, we can use this quantity to form a cost function $\mathcal{C}_{\text{Likelihood}}\big[\log p_{\text{VAE}}(\delta(\boldsymbol{l}))\big]$ that we optimize with respect to the logits $\boldsymbol{l}$, allowing control of the likelihood ratio of generated sequences during training. Specifically, using a logarithmic margin loss, we bound the likelihood ratio with respect to the mean likelihood of the training data p_{ref} by a factor $10^{-\rho}$ (assuming $\log_{10}$-base):

$$\mathcal{C}_{\text{Likelihood}}\big[\log p_{\text{VAE}}(\delta(\boldsymbol{l}))\big] = \max\big[\log p_{\text{ref}} - \log p_{\text{VAE}}(\delta(\boldsymbol{l})) - \rho, 0\big] \tag{4.16}$$

Integrating $\mathcal{C}_{\text{Likelihood}}\big[\log p_{\text{VAE}}(\delta(\boldsymbol{l}))\big]$ into the objective of Equation 2.1, we end up with the cost function in Equation 4.17. We refer to this design method as Fast SeqProp-VAE.

$$\min_{\boldsymbol{l}} -\mathcal{P}(\delta(\boldsymbol{l})) + \lambda \cdot \max\big[\log_{10} p_{\text{ref}} - \log_{10} p_{\text{VAE}}(\delta(\boldsymbol{l})) - \rho, 0\big] \tag{4.17}$$

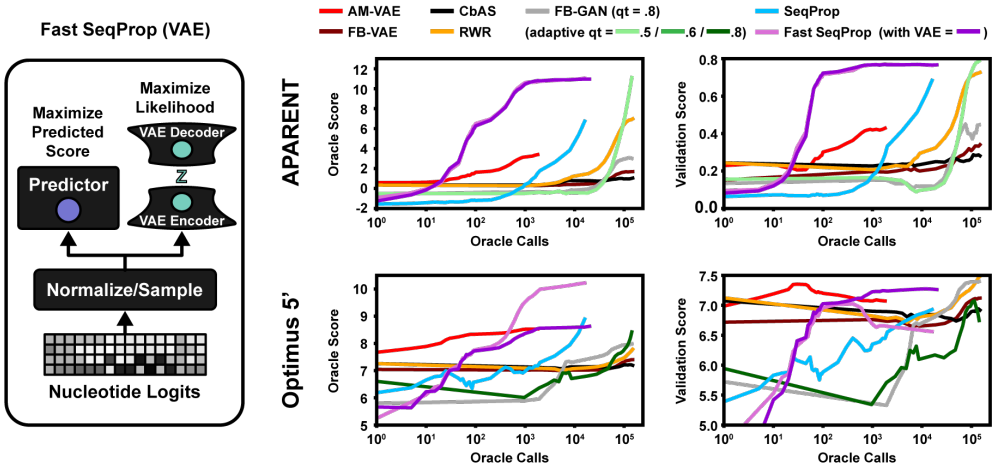

Figure 4.7: Left: VAE-regularized Fast SeqProp. A variational autoencoder (VAE) is used to control the estimated likelihood of designed sequences during gradient ascent optimization. Right: Predictor fitness score and validation model score trajectories as a function of the cumulative number of predictor calls made during the sequence design phase. Shown are the median scores across 10 samples per design method, for three repeats.

We tasked the VAE-regularized Fast SeqProp method with designing maximally strong polyadenylation signals (using APARENT as the predictor) and maximally translationally efficient 5' UTRs (using Optimus 5'). For each task, we trained a β-VAE and W-GAN on a sample of $5,000$ high-fitness sequences. We then used the methods CbAS [18], FB-GAN [59], AM-VAE [80], RWR [119] and FB-VAE (VAE-version of FB-GAN) for comparisons. During optimization, we measured the fitness scores of both the predictor and an independent validation model that we did not directly optimize for, allowing us to estimate sequence fitness in an unbiased way. Specifically, when designing polyadenylation signals based on APARENT, we validated the designs using DeeReCT-APA [95], an LSTM trained on 3'-sequencing data of mouse cells. To validate Optimus 5' designs, we used a newer version of the model that had been trained on additional MPRA data, making it more robust particularly on outlier sequences such as long homopolymers [130]. On a practical note, we found it quite difficult to train a VAE on the APARENT and Optimus 5' datasets, and the convergence of CbAS, RWR and FB-GAN appeared sensitive to quantile threshold settings, which we

believe stem from the considerable data heterogeneity and variability.

For the APA design task, Fast SeqProp reaches higher predictor fitness scores and validation model scores with significantly fewer calls to the predictor than any other design method (Figure 4.7, right). Interestingly, Fast SeqProp reaches identical validation scores (DeeReCT-APA) with or without VAE-regularization, which suggests that overfitting is not an issue for the APARENT predictor. For the 5' UTR design task, AM-VAE initially reaches higher predicted fitness and validation scores, but after 50-100 iterations the method starts to overfit and produces designs with lower validation score. Fast SeqProp without VAE-regularization experiences the same phenomenon; after approximately 100 iterations the fitness score rapidly increases while the validation score decreases. However, Fast SeqProp with VAE-regularization successfully restricts the designs from overfitting and produces stable monotonic validation score trajectories.

4.7 Regularized Activation Maximization using Uncertainty Estimation

To further improve the robustness of sequence designs, we can use predictor oracles capable of estimating uncertainty in their fitness predictions as additional regularizers. To demonstrate this idea, we train the same kind of oracles as was used by Brookes et al. [18] to estimate uncertainty in the fitness predictions [85], and use these models to replicate their Green Fluorescent Protein (GFP) design task in the context of activation maximization. Specifically, for Fast SeqProp, we use the (differentiable) survival function of the normal distribution in order to maximize the probability that the predicted fitness of designed sequences is larger than a particular quantile q of the training data.

Assume that the oracle model predicts the mean $\mu\big[\delta(\boldsymbol{l})\big]$ and standard deviation $\epsilon\big[\delta(\boldsymbol{l})\big]$ of fitness scores for the designed (sampled) pattern $\delta(\boldsymbol{l})$. We then use the (differentiable) survival function of the normal distribution to maximize the probability $p_{\mu[\delta(\boldsymbol{l})],\epsilon[\delta(\boldsymbol{l})]}(\mathbb{Y} > q)$ that the predicted fitness of sequence $\delta(\boldsymbol{l})$ is larger than quantile q of the training data (Here, we use $q = 0.95$). We also impose the VAE-regularization from the previous section in the cost function. The complete cost function is given in Equation 4.18 below. We refer to this

version of activation maximization as Fast SeqProp-PI-VAE (Figure 4.8, left).

$$\min_{\boldsymbol{l}} -\log_{10} p_{\mu[\delta(\boldsymbol{l})],\epsilon[\delta(\boldsymbol{l})]}(\mathbb{Y} > q) + \lambda \cdot \max\left[\log p_{\text{ref}} - \log_{10} p_{\text{VAE}}(\delta(\boldsymbol{l})) - \rho, 0\right] \tag{4.18}$$

We compared this version of Fast SeqProp to the methods AM-VAE, CbAS, FB-VAE and RWR on the GFP design task of Brookes et al. [18] (replicating their test suite). The predictor consisted of an ensemble of 5 models (each consisting of a single fully connected hidden layer). For validation, we used the same GP regression model. The results showed that methods AM-VAE, CbAS, FB-VAE and RWR initially start with significantly higher oracle- and validation scores than Fast SeqProp (Figure 4.8, right). The reason is that these methods start by sampling sequences from the pre-trained VAE, which already has high validation scores. Throughout the optimization, these methods marginally increase their validation scores. However, after $\sim 1,000$ predictor calls, Fast SeqProp quickly converges to designs with higher validation scores than all other methods.

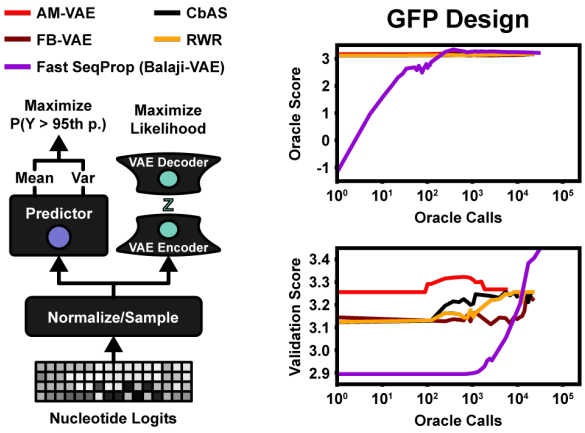

Figure 4.8: Left: Using both VAE-regularization and a probabilistic oracle ensemble to estimate uncertainty. Right: Median oracle fitness scores and validation scores across three repeats for the GFP design task.

4.8 Experimental Validation of Designed APA Sequences

To experimentally validate designed polyadenylation signals, we synthesized, assembled, and expressed the engineered PASs in the APA assay from Section 3.1. For comparison, we used

data from the human wild-type reporter library (Section 3.4) as a reference for the native range of alternative isoform use. These results were published in [17] and used the original SeqProp design method (without the parameter normalization introduced in [100]).

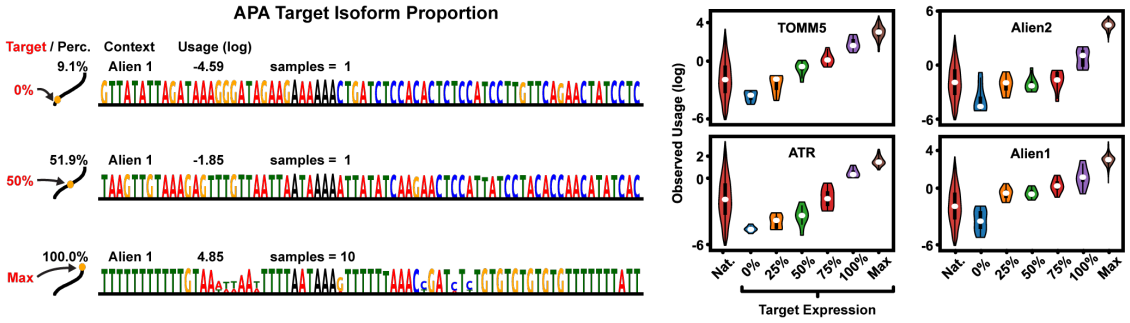

Figure 4.9: Left: Example PWMs that have been optimized using SeqProp for target APA isoform abundances. Right: Measured isoform abundances (log) of designed sequences.

By redefining the fitness cost as the KL-divergence $\mathrm{KL}(\mathcal{P}(\delta(\boldsymbol{l}))||t)$ between the predicted isoform abundance and a target t, we used SeqProp to optimize PASs for a range of target isoform abundances: 0%, 25%, 50%, 75%, and 100%. We generated 10 sequences per target expression for 6 different APARENT UTR contexts (Figure 4.9, left; see Chapter 3 for a description of UTR contexts). We also optimized 20–50 PWMs per UTR context for maximal proximal preference using the classical SeqProp 'Max'-fitness objective. We sampled and synthesized up to 10 sequences from each PWM for a total of 1,200 'Max' PASs. The predicted isoform use of these generated sequences was highly correlated with the experimental measurements (Figure 4.9, right; $R^2 = 0.90$). The 'Max' sequences had a mean usage higher than any native PAS measured in the array, and the strongest sequence had a usage of 99.7% against a strong distal PAS (the bottom sequence shown in Figure 4.9, left).

4.9 Protein Structure Optimization

Multiple deep learning models have recently been developed for predicting tertiary protein structure [2, 133, 167]. Here, we compared Fast SeqProp to other design methods on the task

of designing *de novo* protein sequences which conform to a target residue contact map as predicted by trRosetta [167]. The predictor takes three inputs: A one-hot coded sequence, a PSSM constructed from a multiple-sequence alignment (MSA) and a direct-coupling analysis (DCA) map. For our design task, we pass the optimizable one-hot pattern $\boldsymbol{x}$ to the first two inputs and an all-zeros tensor as the DCA feature map. Given the predicted distance distribution $\boldsymbol{D}^P \in [0,1]^{N\times N\times 37}$ and angle distributions $\boldsymbol{\theta}^P, \boldsymbol{\omega}^P \in [0,1]^{N\times N\times 24}$, $\boldsymbol{\phi}^P \in [0,1]^{N\times N\times 12}$, we minimize the mean KL-divergence against target distributions $\boldsymbol{D}^T$, $\boldsymbol{\theta}^T$, $\boldsymbol{\omega}^T$ and $\boldsymbol{\phi}^T$:

$$\begin{aligned} \min_{\boldsymbol{l}} \quad & \mathrm{KL}(\boldsymbol{D}^P||\boldsymbol{D}^T) + \mathrm{KL}(\boldsymbol{\theta}^P||\boldsymbol{\theta}^T) + \mathrm{KL}(\boldsymbol{\omega}^P||\boldsymbol{\omega}^T) + \mathrm{KL}(\boldsymbol{\phi}^P||\boldsymbol{\phi}^T) \\ \text{where} \quad & \mathrm{KL}(\boldsymbol{X}||\boldsymbol{Y}) = \frac{1}{N^2} \cdot \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K} \boldsymbol{Y}_{ijk} \cdot \log\left(\frac{\boldsymbol{Y}_{ijk}}{\boldsymbol{X}_{ijk}}\right) \end{aligned} \tag{4.19}$$

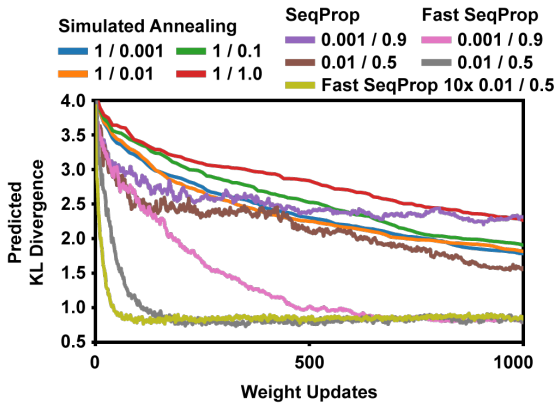

Figure 4.10: Designing sequences which conform to the target predicted structure of a Sensor Histidine Kinase. Simulated Annealing was tested at multiple initial temperatures. SeqProp and Fast SeqProp were tested at several combinations of learning rate and momentum.

The methods were tasked with designing sequences for the target structure of the protein Sensor Histidine Kinase. The optimization trajectories (Figure 4.10) indicate that Fast SeqProp converges faster and reaches better minima; after 200 iterations, Fast SeqProp reached 4x lower KL-divergence than all other methods, and much of the target structure was visible (Figure 4.11). While the choice of learning rate changed the rate of convergence,

it did not alter the minima found by Fast SeqProp. Additionally, by sampling multiple sequences $\{\delta(\boldsymbol{l})^{(s)}\}_{s=1}^{S}$ at once and walking down the average gradient $\frac{1}{S}\sum_{s=1}^{S}\nabla_{\boldsymbol{l}}\mathcal{P}(\delta(\boldsymbol{l})^{(s)})$, we can improve the rate of convergence by making the gradients less noisy (e.g. $S = 10$ for 'Fast SeqProp 10x' in Figure 4.10). Importantly, this scales significantly better than linear in execution time, since multiple samples can be differentiated in parallel on a GPU.

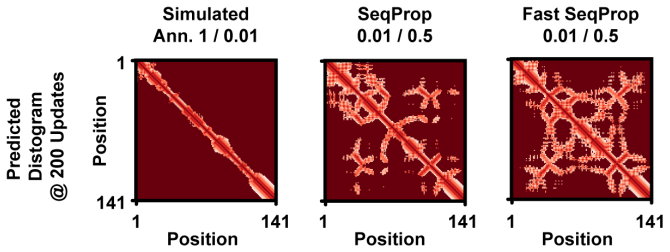

Figure 4.11: Predicted residue distance distributions after 200 iterations. Simulated annealing was run with 1 mutation per step at an initial temperature of 0.01. SeqProp and Fast SeqProp were run at 0.01 learning rate and 0.5 momentum.

Chapter 5

DIVERSE SEQUENCE DESIGN WITH ACTIVATION-MAXIMIZING GENERATIVE NETWORKS

In the previous chapter, we presented an improved gradient-based design method that demonstrated a significant speedup to previous sequence design algorithms. Still, the method is fundamentally limited for large-scale design problems. The reason is that the gradient ascent optimization must be re-run for every sequence we want to design, which incurs considerable computational cost if we want to generate millions or billions of candidate sequences. Furthermore, independent sequence optimization has no means of controlling the diversity of optimized sequences. To address these limitations, we developed Deep Exploration Networks (DENs), an activation-maximizing generative neural network capable of optimizing sequences for a differentiable fitness predictor while maintaining sequence diversity. This work was presented at the MLCB 2019 conference [99] and was later published in [96] where I was the first author. The majority of content presented in this chapter is from that paper.

5.1 Deep Exploration Networks

The DEN architecture, as illustrated in Figure 5.1, is based around a generative neural network $\mathcal{G}$ and a differentiable fitness predictor $\mathcal{P}$. Given a D-dimensional seed vector $\boldsymbol{z} \in \mathbb{R}^D$ as input, the generator $\mathcal{G}$ outputs a nucleotide logit matrix $\boldsymbol{l}(\boldsymbol{z}) = \mathcal{G}(\boldsymbol{z})$. An approximate one-hot-coded pattern $\boldsymbol{x}(\boldsymbol{z})$ is obtained by transforming $\boldsymbol{l}(\boldsymbol{z})$ through the formula:

$$\boldsymbol{x}(\boldsymbol{z}) = \delta\big(\boldsymbol{l}(\boldsymbol{z})\big) * \mathbf{M} + \mathbf{T} \tag{5.1}$$

Here, δ is the discrete nucleotide sampler from Equation 4.1, which transforms the real-valued nucleotide scores generated by $\mathcal{G}$ into a one-hot-coded representation. Mask matrix $\mathbf{M}$ zeroes out fixed sequence positions and template matrix $\mathbf{T}$ encodes fixed nucleotides. We

propagate gradients through this representation using either the Softmax Straight-Through estimator from Equation 4.2 [12, 35, 31] or the Gumbel distribution [70]. The predictor output $\mathcal{P}(\boldsymbol{x}(\boldsymbol{z}))$ is used to define a fitness cost $\mathcal{C}_{\text{Fitness}}\left[\mathcal{P}(\boldsymbol{x}(\boldsymbol{z}))\right]$, and the overall goal is to optimize $\mathcal{G}$ such that the generated sequences minimize this cost. Only $\mathcal{G}$ is optimized, having pre-trained $\mathcal{P}$ to accurately predict the target biological function.

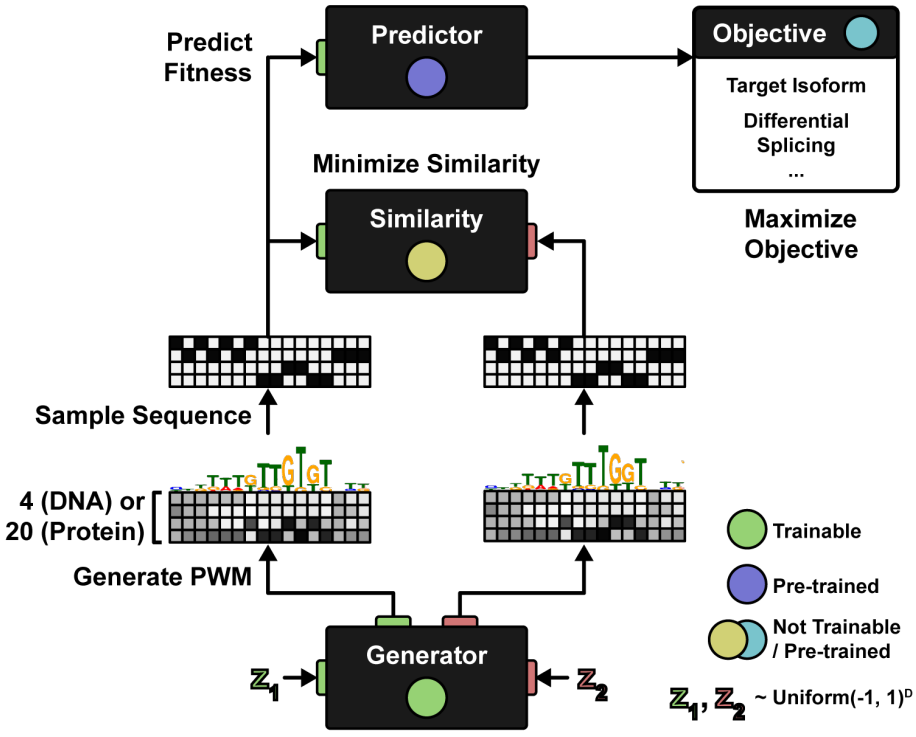

Figure 5.1: The Deep Exploration Network Architecture. Two generator samples are considered simultaneously, enabling similarity comparisons in addition to evaluating fitness.

By strictly minimizing $\mathcal{C}_{\text{Fitness}}\left[\mathcal{P}(\boldsymbol{x}(\boldsymbol{z}))\right]$, the generator will likely only learn to produce one single pattern, regardless of $\boldsymbol{z}$, since it is trivially optimal to always output the pattern located at the bottom of the local fitness cost minimum. There may however exist much better minima. Additionally, as the predictor may be inaccurate for certain sequences, the generator should ideally learn to sample many diverse patterns with maximal fitness scores. The distinguishing feature of a DEN is to force the generator to map randomly sampled

vectors from the D-dimensional uniform distribution $U(-1,1)^D$ to many different sequences with maximal fitness score. This is achieved by making the generator compete with itself; $\mathcal{G}$ is executed twice at each step of the optimization for a pair of sampled seeds $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)} \sim U(-1,1)^D$. The generator is penalized based on both the fitness cost $\mathcal{C}_{\text{Fitness}}\big[\mathcal{P}(\boldsymbol{x}(\boldsymbol{z}^{(1)}))\big]$ and a diversity cost $\mathcal{C}_{\text{Diversity}}\big[\boldsymbol{x}(\boldsymbol{z}^{(1)}), \boldsymbol{x}(\boldsymbol{z}^{(2)})\big]$ evaluated on the generated patterns $\boldsymbol{x}(\boldsymbol{z}^{(1)}), \boldsymbol{x}(\boldsymbol{z}^{(2)})$:

$$\min_{\mathcal{G}} \mathcal{C}_{\text{Fitness}}\big[\mathcal{P}(\boldsymbol{x}(\boldsymbol{z}^{(1)}))\big] + \lambda \cdot \mathcal{C}_{\text{Diversity}}\big[\boldsymbol{x}(\boldsymbol{z}^{(1)}), \boldsymbol{x}(\boldsymbol{z}^{(2)})\big] \tag{5.2}$$

This monte-carlo optimization differs from a classical GAN [55], which is trained to minimize a cost $\mathcal{C}\big[\mathcal{D}(\mathbf{Data}), \mathcal{D}(\mathcal{G}(\boldsymbol{z}))\big]$ such that an adversarial discriminator $\mathcal{D}$ cannot distinguish between the real data and the distribution generated by $\mathcal{G}$. Also note that, in contrast to [80] where optimization is done on a single input seed of a pre-trained GAN, $\min_{\boldsymbol{z}} \mathcal{C}_{\text{Fitness}}\big[\mathcal{P}(\mathcal{G}(\boldsymbol{z}))\big]$, here $\mathcal{G}$ itself is optimized for all seeds. As training progresses, the generator will become injective over different seeds $\boldsymbol{z} \sim U(-1,1)^D$. Consequently, we can sample diverse high-fitness sequences by drawing samples from $U(-1,1)^D$ and transforming them through $\mathcal{G}$.

Several different losses are investigated for the diversity cost $\mathcal{C}_{\text{Diversity}}\big[\boldsymbol{x}(\boldsymbol{z}^{(1)}), \boldsymbol{x}(\boldsymbol{z}^{(2)})\big]$. First, a cosine distance is used to directly penalize one-hot patterns based on the fraction of identical nucleotides (Figure 5.2, left). To support translational invariance, patterns are penalized by the worst possible offset σ, ranging from $\sigma = -\sigma_{max}$ to σ_{max}. The penalty is defined as a margin loss, allowing patterns to be identical up to a margin ϵ without incurring any cost. Given two masked one-hot coded patterns $\boldsymbol{x}^{(1)} = \mathbf{f}(\mathcal{G}(\boldsymbol{z}^{(1)})) * \mathbf{M}$ and $\boldsymbol{x}^{(2)} = \mathbf{f}(\mathcal{G}(\boldsymbol{z}^{(2)})) * \mathbf{M}$, the sequence similarity cost is defined as:

$$\mathcal{C}_{\text{S}}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) = \max\Bigg[-\epsilon + \frac{1}{N} \max_{\sigma} \sum_{i=1}^{N-|\sigma|} \sum_{j=1}^{M} \boldsymbol{x}^{(1)}_{i+\max(\sigma,0),j} \cdot \boldsymbol{x}^{(2)}_{i+\max(-\sigma,0),j}, 0\Bigg] \tag{5.3}$$

Here, N refers to the sequence length and M the number channels. Theoretically, we should set $\sigma_{max} = N - 1$ to test all possible offsets, however practically we found that $\sigma_{max} = 1$ was sufficient.

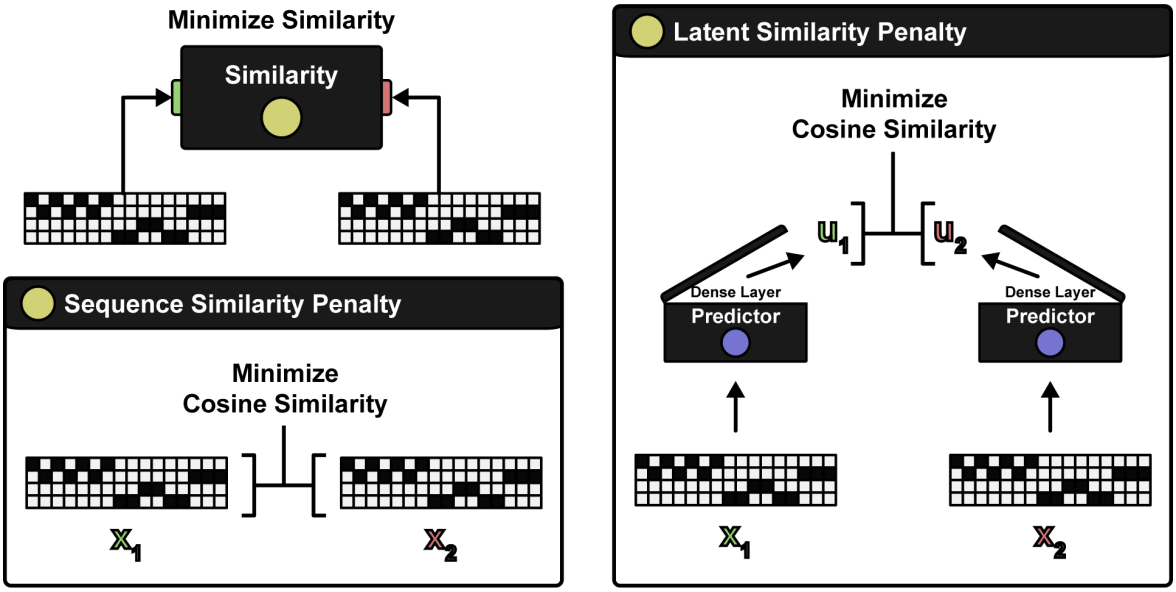

Figure 5.2: The sequence similarity penalty is computed as the cosine distance between two one-hot samples. The latent similarity penalty is computed as the cosine distance between latent (dense) vectors.

A latent diversity cost was also evaluated, which is defined on a pair of latent feature vectors $\boldsymbol{u}^{(1)}$ and $\boldsymbol{u}^{(2)}$ obtained from the generated sequence patterns $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ by applying some (differentiable) feature transform $\boldsymbol{u}^{(1)} = \mathcal{M}(\boldsymbol{x}^{(1)})$ and $\boldsymbol{u}^{(2)} = \mathcal{M}(\boldsymbol{x}^{(2)})$. In theory, $\mathcal{M}$ can be any encoder model, e.g. a variational autoencoder (VAE). Here, we let the first fully connected layer of $\mathcal{P}$ be the latent feature space for $\boldsymbol{u}^{(1)}$ and $\boldsymbol{u}^{(2)}$ (Figure 5.2, right). The latent similarity cost is defined as the cosine similarity margin loss between $\boldsymbol{u}^{(1)}$ and $\boldsymbol{u}^{(2)}$:

$$\mathcal{C}_{\mathrm{L}}(\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}) = \max\left[\frac{\sum_{i=1}^{d} \boldsymbol{u}_i^{(1)} \cdot \boldsymbol{u}_i^{(2)}}{\sqrt{\sum_{i=1}^{d} \left(\boldsymbol{u}_i^{(1)}\right)^2} \cdot \sqrt{\sum_{i=1}^{d} \left(\boldsymbol{u}_i^{(2)}\right)^2}} - \epsilon, 0\right] \tag{5.4}$$

5.2 Engineering Alternative Polyadenylation

DENs were first demonstrated in the context of Alternative Polyadenylation (APA). Same as in the previous chapter, we used the neural network APARENT as the fitness predictor [17] and considered the task of designing polyadenylation signals with maximal APA iso-

form selection. Thus, we defined the fitness cost as the negative of the predicted isoform score, $\mathcal{C}_{\text{Fitness}}\big[\mathcal{P}(\boldsymbol{x}(\boldsymbol{z}))\big] = -\mathcal{P}(\boldsymbol{x}(\boldsymbol{z}))$. The generator $\mathcal{G}$ followed a DC-GAN architecture [122] consisting of one fully connected layer, three strided de-convolutional layers (each with an upsampling rate of 2) and three convolutional layers.

In Figure 5.3, two independent DENs were trained, each tasked with generating PASs with maximal predicted APA. In one of the two instances, the diversity cost coefficient λ in Equation 5.2 was lowered to a small value (0.25), and in the other instance it was increased (5.0). With a low coefficient, the generator learned to sample few, low-diversity sequences, all of similar isoform log odds (Figure Figure 5.3, left; mean isoform log odds = 6.06, 99.5% sequence duplication rate at $100,000$ sampled sequences). With an increased coefficient, the generator became highly diverse and the mean predicted isoform odds increased almost 20-fold (Figure 5.3, right; mean isoform log odds = 8.91, 0% duplication rate). These results indicate that exploration during training drastically improves the final fitness of the generator by traversing local minima of the fitness cost landscape.

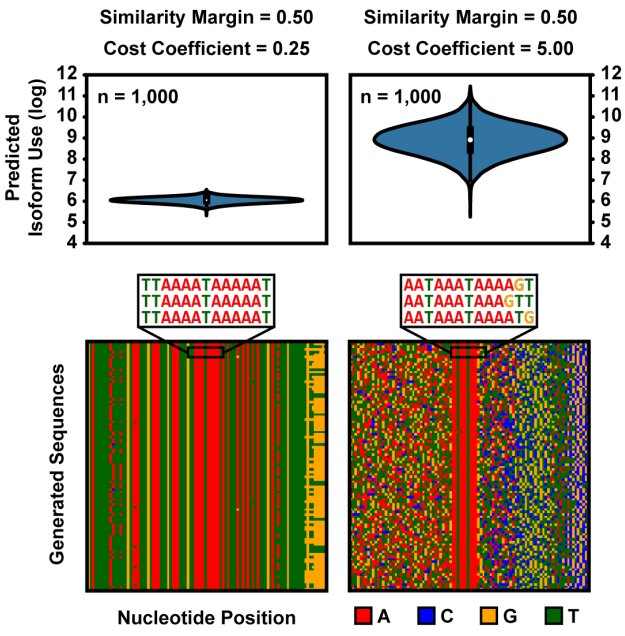

Figure 5.3: The DEN was trained with a low diversity cost coefficient (left) and a high coefficient (right). (Top) Predicted isoform proportion. (Bottom) Sequence pixel grid.

To find the optimal similarity penalty (for this design problem), we re-trained the Max-isoform DEN under different similarity margins (the fraction by which we allow sequences to be similar), measuring the 50th and 99th percentile of fitness scores and sequence edit distances of each converged generator (Figure 4.4). Both fitness and diversity increased up to a sequence dissimilarity of 40%. Beyond that point, the generated fitness scores monotonically decreased. These results are quite interesting, as they suggest that enforcing diversity results in better fitness optima up to a certain point, presumably by helping the generator overcome local minima and explore new parts of design space. At too large magnitudes, however, the diversity cost conversely restricts the generator from finding good minima.

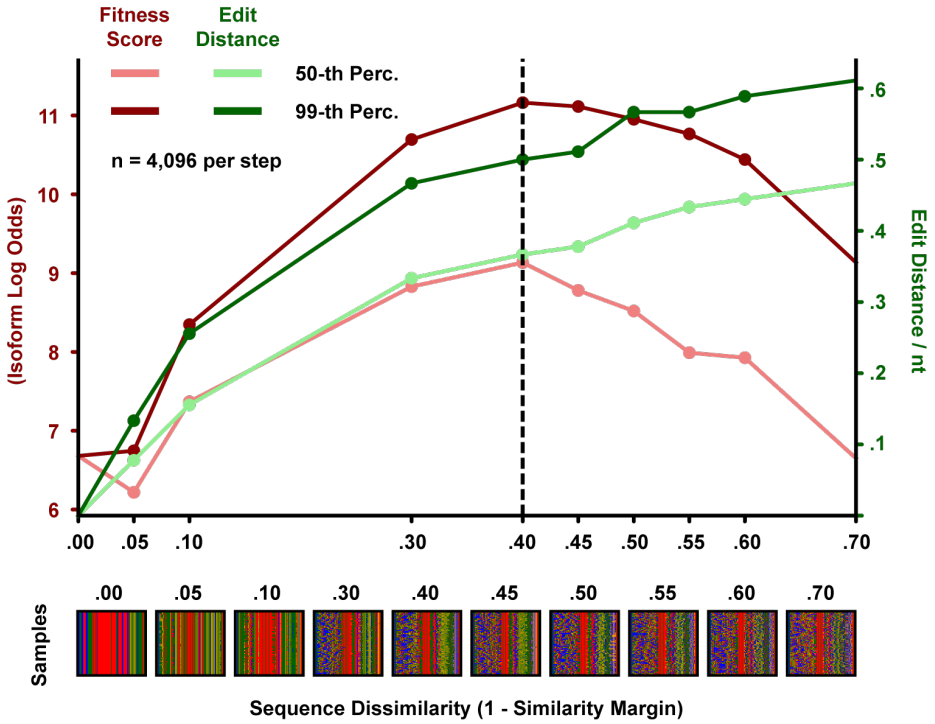

Figure 5.4: The DEN was retrained for different allowable sequence similarity margins. Plotted are the 50th/99th percentile of predicted fitness scores and pairwise edit distances.

Next, we characterized DEN-generated PASs experimentally and compared their fitness to sequences generated by the activation maximization method SeqProp. To that end, we

synthesized APA reporters with two adjacent PASs (Figure 5.5): Each reporter contained a Max-isoform PAS sampled from a DEN with 40% sequence similarity margin, as well as one of the strongest SeqProp-optimized signals from [17] (see Chapter 4.8). To discount any bias from aspecific proximal preference, we experimentally assayed both signal orientations where the DEN-generated PAS was either the proximal signal or the distal signal. The reporters were cloned onto plasmids and delivered to HEK293 cells. We quantified the expressed RNA isoform levels using a qPCR assay, measuring the Ct values of total and distal RNA respectively. Using Ct differences to estimate odds ratio lower bounds, we found that the DEN-generated sequences were on average 11.6-fold more preferred (odds ratio) than the SeqProp-generated sequences. To put this in perspective, the strongest SeqProp sequence had usage odds of 127:1 (99.22%) relative to a distal bGH PAS separated by 200 nt. The DEN-sequences would have usage odds of 1481:1 (99.93%) relative to the same signal.

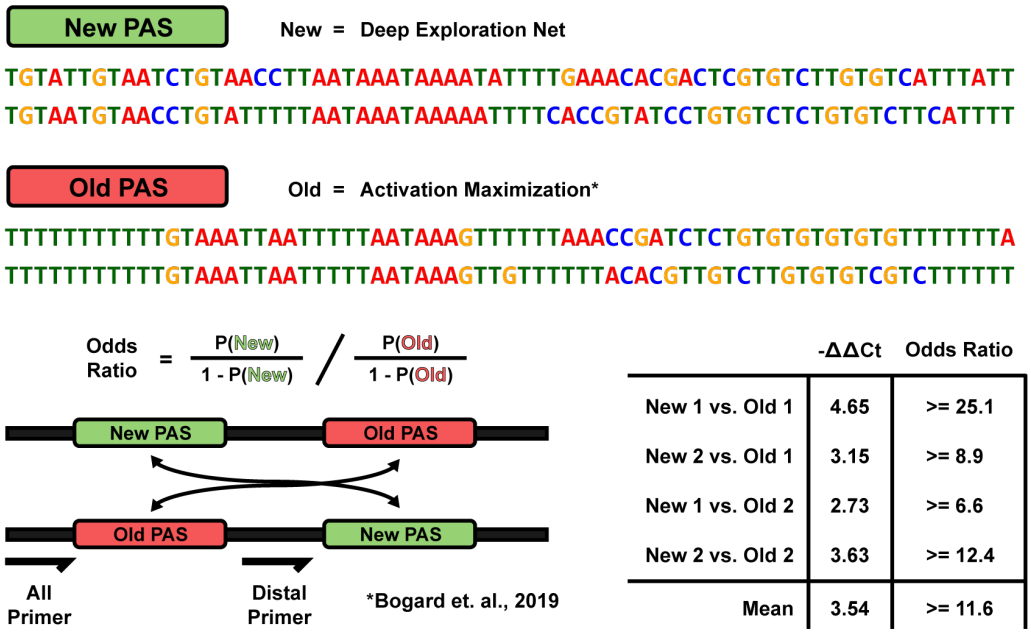

Figure 5.5: Experimental validation of two DEN-generated polyadenylation signals, using an APA 3' UTR reporter with isoform estimation by qPCR.

5.3 A Comparison of Generative Models for APA Sequence Design

We carried out extensive benchmark comparisons between DENs and competing methods at the task of designing PASs with maximal APARENT isoform score. DENs were compared to 6 other design methods: (1) Generative adversarial networks (GANs) trained either on randomly sampled sequences or on the subset of the data with the highest measured fitness score [158], (2) Activation-maximization of GANs (AM-GAN) [80], (3) Feedback-GANs (FB-GANs) [59], (4) Optimizing a single softmax-relaxed PWM by gradient ascent and sampling sequences from it (the baseline method), (5) Optimizing several PWMs by gradient ascent, and (6) Simulated Annealing with the Metropolis acceptance criterion [83, 111].

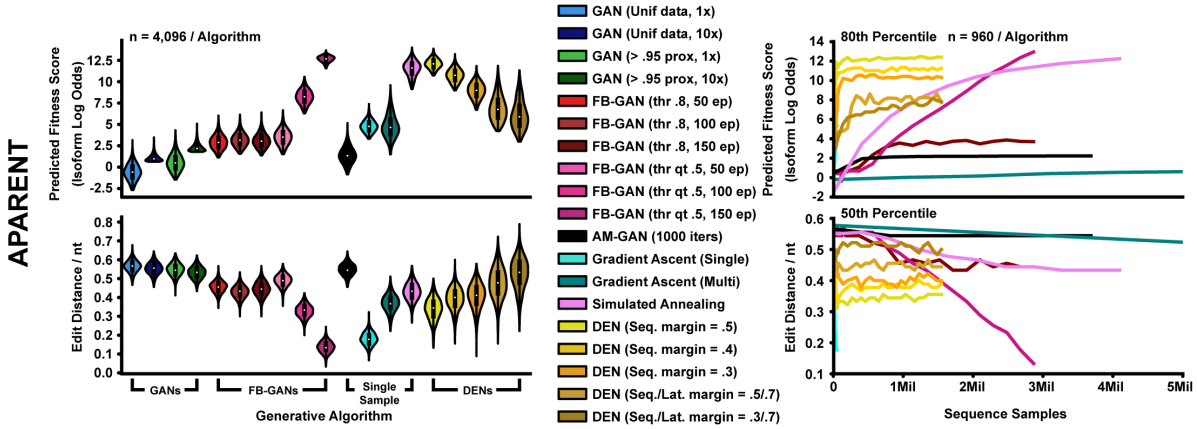

Figure 5.6: Benchmark comparison of 6 design methods for designing sequences with maximal APA isoform abundance. Left: Predicted fitness scores and pairwise edit distances. Right: Fitness scores and edit distances as a function of the number of sequences queried.

We trained one GAN version on a random subset of 10,000 APA sequences from the APARENT MPRA (see Chapter 3.1). We trained another GAN version on a high-fitness subset of APA sequences. Specifically, we selected sequences with a minimum read count of 50 and a minimum isoform abundance of 0.95 (resulting in approximately $50,000$ sequences), and randomly sampled $10,000$ sequences from this subset. For FB-GAN, we tried using both a fixed feedback threshold of 0.8 (in isoform proportion) as well as an adaptive threshold that

was adjusted at each iteration to the 50th percentile of the predicted isoform proportions for the current FB-GAN training set. Five different DEN configurations were trained for a range of different diversity cost magnitudes and similarity margins (three DENs with 50%, 40% and 30% sequences similarity margins, as well as two DENs with 70% and 50% latent similarity margins). Fitness comparisons were made relative to the median score of the baseline method (PWM Gradient Ascent), where the median score of random sequences was treated as the baseline value (i.e. +0%). In Figure 5.6 and 5.7, we compare the design methods on fitness and diversity as a function of the total number of iterations (sequence budget) needed to design a target number of sequences. For per-sequence optimization methods (PWM Gradient Ascent, AM-GAN, Simulated Annealing), the sequence budget is computed as the number of iterations spent on designing a single sequence (number of predictor calls) multiplied by the total number of sequences designed. For GAN, FB-GAN and DEN, the sequence budget is computed as the number of generator forward passes.

The methods DEN, FB-GAN (adaptive threshold) and Simulated Annealing generated sequences with fitness score medians that were approximately +115% compared to baseline (Figure 5.6). However, sequence diversity was significantly lower for FB-GAN (15% median normalized edit distance) compared to DEN (35%) and Simulated Annealing (45%). DEN-generated diversity could be increased by changing the allowable similarity margin, at the cost of diminished fitness scores. While sequence diversity increased for FB-GAN with fixed threshold (45% median edit distance), its median fitness score dropped below comparably diverse DEN models (-23% of the baseline median). Similarly, fitness scores were low for GAN and Activation-maximization of GAN (-61.5% and -46.1% below baseline respectively). Measured as the total number of sequences sampled during optimization, DEN converges to near-optimal fitness scores in fewer iterations than all other methods.

5.4 A Comparison of Generative Models for Gene Enhancer Design

The same benchmark comparison was replicated for the task of designing gene enhancers with maximal transcriptional activity according to the predictor MPRA-DragoNN [114].

Here, the results were more pronounced (Figure 5.7): Fitness score medians were +600% and +660% compared to baseline for DEN and Simulated Annealing respectively, while FB-GAN had a median score of +200%. DEN had approximately 3% lower median edit distance compared to baseline, but it was 17% higher compared to FB-GAN. Overall, DEN and Simulated Annealing generated sequences of comparable fitness, but Simulated Annealing was more diverse (between 7% and 10% increased edit distance). In fact, Simulated Annealing is a meta-heuristic known to find near-global minima given enough iterations, and the method optimally finds diverse basins by starting from random sequences. But Simulated Annealing is fundamentally more computationally expensive when generating many samples; the method must be re-initialized and optimized for every sequence to make. By extrapolating the optimization trajectories of Figure 5.7, we find that DENs can generate approximately 100,000 sequences in fewer iterations (total sequence budget) than Simulated Annealing requires for only 1,000 of comparable fitness.

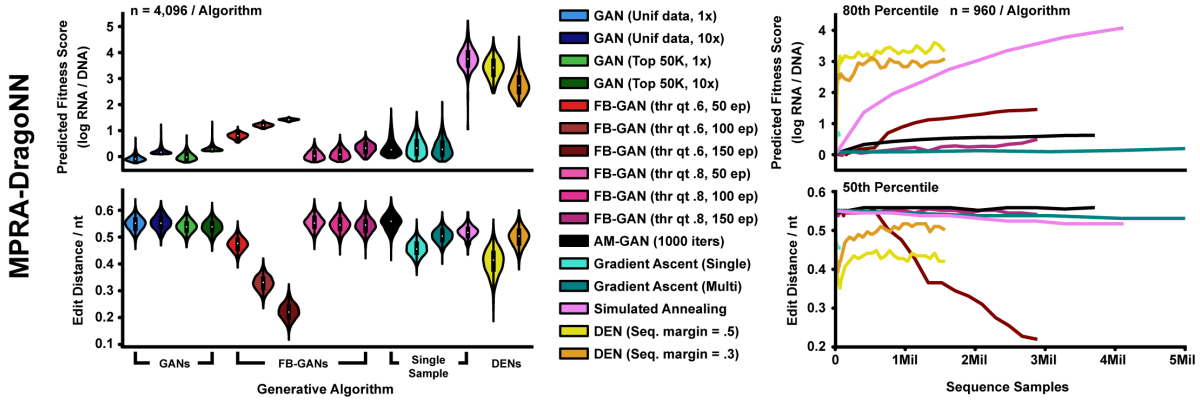

Figure 5.7: Replicate benchmark of 6 design methods for designing gene enhancers sequences with maximal predicted transcriptional activity.

5.5 Engineering Green Fluorescent Proteins while Maintaining Confidence

The DEN formulation described in previous sections maximizes fitness and diversity without regard to how confident the predictor (or any other model) is in the generated sequences. However, assuming the predictor loses its predictive power as the generated sequences drift

from the measured training data in design space, it becomes necessary to maintain the marginal likelihood of sequences with respect to the data. This problem is particularly evident in protein design, where functional sequences are thought to reside in a manifold much smaller than the space of all possible sequences, and where measured training data only span this manifold. A related design method based on in-silico directed evolution, Conditioning by Adaptive Sampling (CbAS) [18], controls the likelihood by adaptively sampling and retraining a variational autoencoder (VAE). Here, VAEs are directly integrated in the DEN cost function (Figure 5.8), enabling direct control of the approximate likelihood ratio of generated sequences with respect to a reference likelihood estimated on the training data. This allows for tuning how confident the DEN should be in the generated sequences during backpropagation. The expected log likelihood is estimated by importance sampling [117, 82, 16], and ST gradients are used to optimize the generator for the VAE likelihood penalty.

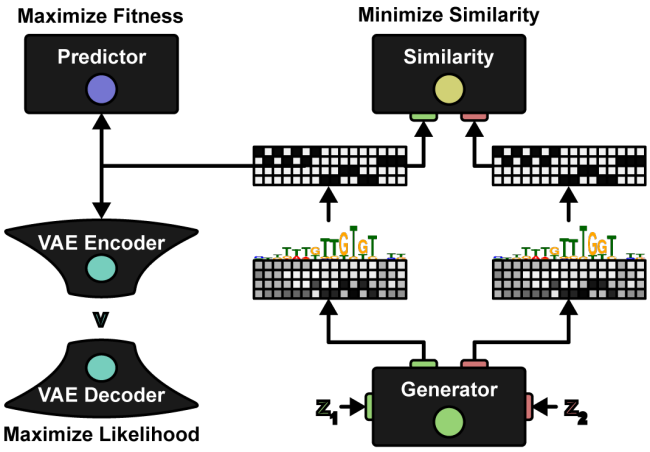

Figure 5.8: The integration of a VAE in the DEN framework.

Given the differentiable estimate of $\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))$ derived in Section 4.6, we could theoretically use the cost function from Equation 4.16. However, there is a problem with this cost: It skews the distribution $\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))$ when taken over many seeds $\boldsymbol{z}$. This is because every value $\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))$ crossing the allowable margin will be penalized by the cost

function, which means that the expected log likelihood $\mathbb{E}_z[\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))]$ will be shifted far beyond the margin $\log p_{\text{ref}} - \rho$ to accommodate the worst-case samples. Rather, what we want is for $\mathbb{E}_z[\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))]$ to be centered at $\log p_{\text{ref}} - \rho$. To achieve this, we estimate $\mathbb{E}_z[\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))]$ during each forward pass of backpropagation as an empirical mean over mini-batches of generator seeds. Specifically, if we optimize the DEN on a batch of L seeds $\boldsymbol{z}[l] \sim U(-1, 1)^D$ (the brackets denote batch index), then the L-sized batch is divided into H-sized mini-batches, and we compute $V = L/H$ independent estimates of $\mathbb{E}_z[\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))]$:

$$\mathbb{E}_z[\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))]^{(v)} \approx \frac{1}{H} \sum_{h=1}^{H} \log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}[v \cdot H + h])) \tag{5.5}$$

We now have an L/H-sized batch of estimates $\left\{\mathbb{E}_z[\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))]^{(v)}\right\}_{v=1}^{L/H}$, which are broadcasted back to the original batch size L and used with each respective seed $\boldsymbol{z}[l]$ of the batch to compute the expected likelihood ratio cost function:

$$\mathcal{C}_{\text{Likelihood}}\left[\mathbb{E}_z[\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))]\right] = \max\left[\log p_{\text{ref}} - \mathbb{E}_z[\log p_{\text{VAE}}(\boldsymbol{x}(\boldsymbol{z}))] - \rho, 0\right] \tag{5.6}$$

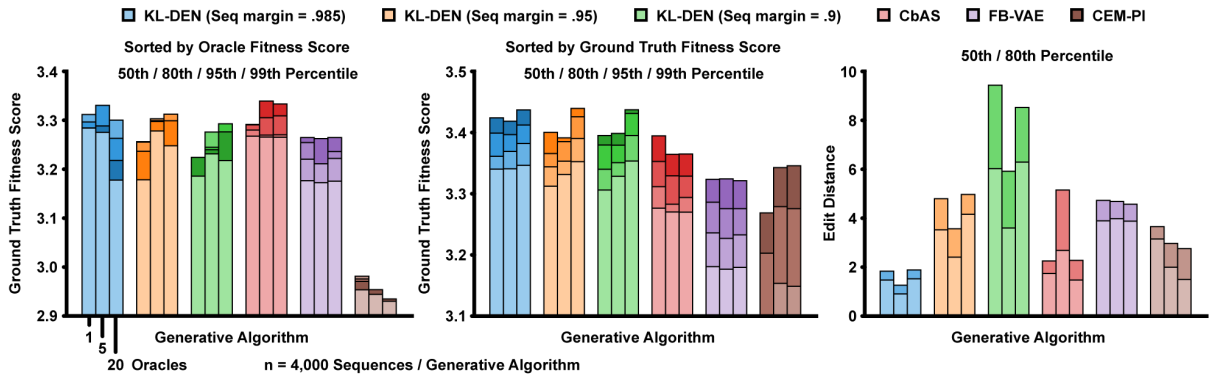

Figure 5.9: GFP design benchmark, measuring "ground truth" scores sorted by oracle scores (left) or by ground truth scores (middle), and pairwise edit distances (right).

The Likelihood-bounded DEN was benchmarked on the task of GFP sequences for maximal brightness, using variant data from [131]. Replicating the analysis of [18], three increasingly large predictor ensembles were evaluated [85]. The generator architecture was

also altered by appending an LSTM-layer, as it increased convergence speed. The DEN was tasked with maximizing the probability of the predicted brightness being above the 95th percentile of the training data (Using the fitness cost from Equation 4.18 of Section 4.7), while bounding sequences to be at least 1/10th as likely as the training data. Three DEN versions were tested: with either 98.5%, 95% and 90% sequence similarity penalty margins (the allowable fraction of identical residues). Performance was evaluated using an independent Gaussian Process regression model (considered the ground truth). The DENs were compared against CbAS, FB-VAE (a VAE-based version of FB-GAN) [59] and CEM-PI (Probability of Improvement) [143]. While the DEN models were less consistent than CbAS (Figure 5.9, left), they generated higher overall ground truth scores (Figure 5.9, middle). Furthermore, the DEN with 90% similarity margin generated more diverse sequences (Figure 5.9, right). Finally, we compared against a regular DEN with no likelihood regularization penalty (Figure 5.10). Although the predicted oracle scores increased rapidly, the corresponding ground truth scores never improved from the minimum value.

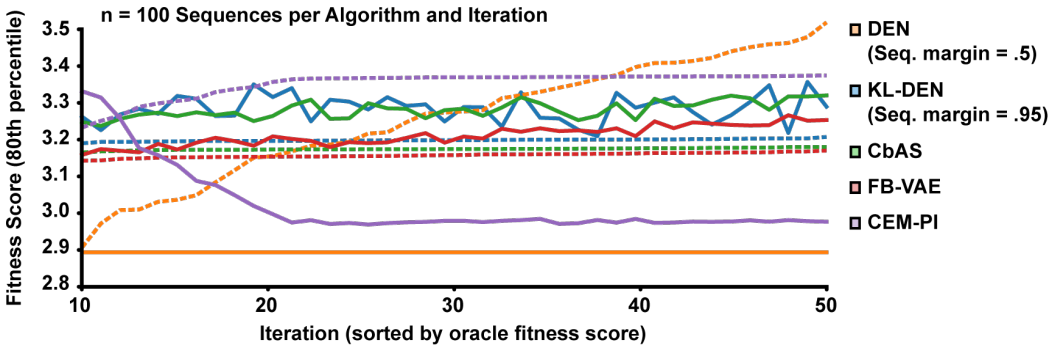

Figure 5.10: Oracle (dashed line) and ground truth (solid line) fitness scores as a function of training epoch. The values (x-axis) are sorted on oracle scores.

Chapter 6

INTERPRETING NEURAL NETWORKS FOR DNA- AND PROTEIN SEQUENCES

Sequence-based neural networks can learn complex relationships from large-scale datasets in order to accurately predict gene-regulatory phenomena and protein function, but model interpretation remains challenging due to the internal complexity of such networks. Instead, one may attempt to directly *attribute* a prediction to the most salient input features. While there are many attribution methods available, few are optimized for sequence-predictive neural networks where the input patterns are discrete. Furthermore, sequence regulation commonly include non-linear interactions, requiring importance to be assessed over groups of features. Most methods, however, are based on local approximation and analyze each feature in isolation. Here we develop an alternate approach to sequence attribution where a deep generative model learns to either preserve or destroy the most important input positions with masks. Our version of this method – *Scrambler Neural Networks* – learns to generate a Position-Specific Scoring Matrix (*PSSM*) where the entropy of each nucleotide or residue is proportional to its importance. This work was presented at the MLCB 2020 conference [98] and was later deposited to a pre-print server [97] where I was the first author. The second author, Alyssa La Fleur, performed part of the analysis and contributed significantly to the paper. The majority of content presented here is from that paper.

6.1 Interpreting Sequence-based Networks with Deep Generative Masking

The Scrambler operation is illustrated in Figure 6.1. Given a differentiable pre-trained predictor $\mathcal{P}$ and a one-hot encoded input pattern $\boldsymbol{x} \in \{0,1\}^{N \times M}$ (representing an N-length sequence), we let a trainable network $\mathcal{S}$ called the *Scrambler* generate a set of real-valued

importance scores $\mathcal{S}(\boldsymbol{x}) \in (0, \infty]^N$. These scores are used in Equation 6.1 below to produce a probability distribution which interpolates between the input pattern $\boldsymbol{x}$ and a non-informative background distribution $\tilde{\boldsymbol{b}} \in \mathbb{R}^{N \times M}$.

$$\hat{\boldsymbol{x}}_{\mathcal{S}} = \sigma\big(\log \tilde{\boldsymbol{b}} + \boldsymbol{x} \times \dot{\mathcal{S}}(\boldsymbol{x})\big) \tag{6.1}$$

Here, σ denotes the softmax $\sigma(\boldsymbol{l})_{ij} = \frac{e^{l_{ij}}}{\sum_{k=1}^{4} e^{l_{ik}}}$ and $\dot{\mathcal{S}}(\boldsymbol{x}) \in (0, \infty]^{N \times M}$ represent the importance scores $\mathcal{S}(\boldsymbol{x})$ which have been broadcasted at position i to all channels j. The output $\hat{\boldsymbol{x}}_{\mathcal{S}}$ becomes a parameterization of a probability distribution of the input. Specifically, $\hat{\boldsymbol{x}}_{\mathcal{S}}$ is a set of N categorical softmax-nodes, or a PSSM. The role of $\tilde{\boldsymbol{b}}$ is to keep samples from this PSSM in-distribution and along the manifold of valid patterns, and is here taken as the mean input pattern across the training set. When $\mathcal{S}(\boldsymbol{x})_i$ is close to 0, $\hat{\boldsymbol{x}}_{\mathcal{S},i}$ becomes $\tilde{\boldsymbol{b}}_i$ (the background distribution) and when $\mathcal{S}(\boldsymbol{x})_i$ is close to ∞, $\hat{\boldsymbol{x}}_{\mathcal{S},i}$ becomes $\boldsymbol{x}_i$ (the original input). $\mathcal{S}(\boldsymbol{x})_i$ thus defines the inverse temperature of the feature sampling distribution at position i in the PSSM.

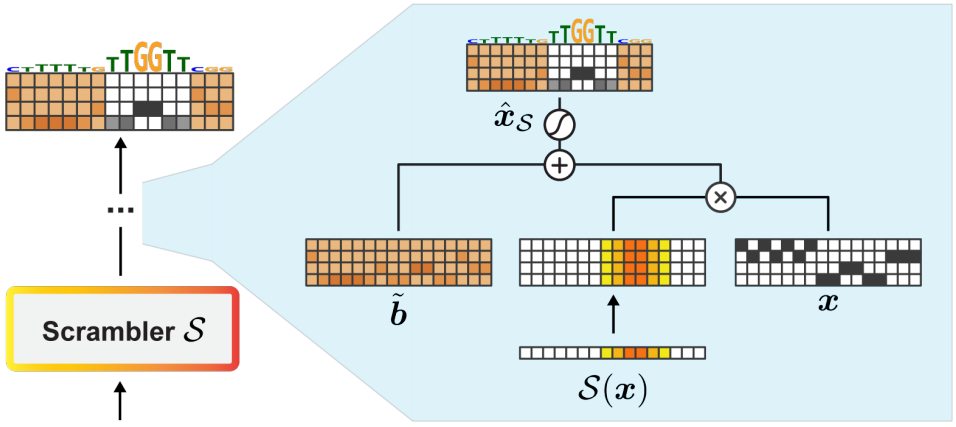

Figure 6.1: The temperature-based masking operation.

Discrete, categorical samples $\{\boldsymbol{x}_{\mathcal{S}}^{(k)}\}_{k=1}^{K}$ drawn from $\hat{\boldsymbol{x}}_{\mathcal{S}}$ are passed to the predictor $\mathcal{P}$ and gradients are backpropagated to $\mathcal{S}$ using either Softmax Straight-Through estimation [31] or the Gumbel distribution [70]. By comparing the predictions $\mathcal{P}(\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)})$ of the scrambled input

samples to the original prediction $\mathcal{P}(\boldsymbol{x})$, we train the Scrambler $\mathcal{S}$ to minimize a predictive reconstruction error subject to a conservation penalty which enforces high entropy. We refer to this formulation as the *Inclusion*-Scrambler, as it must learn to include features to reconstruct the original prediction while maintaining high entropy of $\hat{\boldsymbol{x}}_{\mathcal{S}}$ (Figure 6.2, left). We define the reconstruction error as the KL-divergence $\mathrm{KL}\big[\mathcal{P}(\boldsymbol{x}_{\mathcal{S}}^{(k)})||\mathcal{P}(\boldsymbol{x})\big]$ between scrambled and original predictions. To minimize the conservation of $\hat{\boldsymbol{x}}_{\mathcal{S}}$ while still keeping samples in-distribution, we optimize the KL-divergence $\mathrm{KL}\big[\tilde{\boldsymbol{b}}||\hat{\boldsymbol{x}}_{\mathcal{S}}\big]$ between $\hat{\boldsymbol{x}}_{\mathcal{S}}$ and the background distribution $\tilde{\boldsymbol{b}}$. We control the expected entropy by fitting $\mathrm{KL}\big[\tilde{\boldsymbol{b}}||\hat{\boldsymbol{x}}_{\mathcal{S}}\big]$ to a target conservation value t_{bits} rather than minimizing or maximizing it unbounded. The full training cost for the Inclusion-Scrambler is given in Equation 6.2.

$$\min_{\mathcal{S}} \left(\frac{1}{K} \sum_{k=1}^{K} \mathrm{KL}\big[\mathcal{P}(\boldsymbol{x}_{\mathcal{S}}^{(k)})||\mathcal{P}(\boldsymbol{x})\big] \right) + \lambda \cdot \left(t_{\text{bits}} - \frac{1}{N} \cdot \mathrm{KL}\big[\tilde{\boldsymbol{b}}||\hat{\boldsymbol{x}}_{\mathcal{S}}\big] \right)^2 \tag{6.2}$$

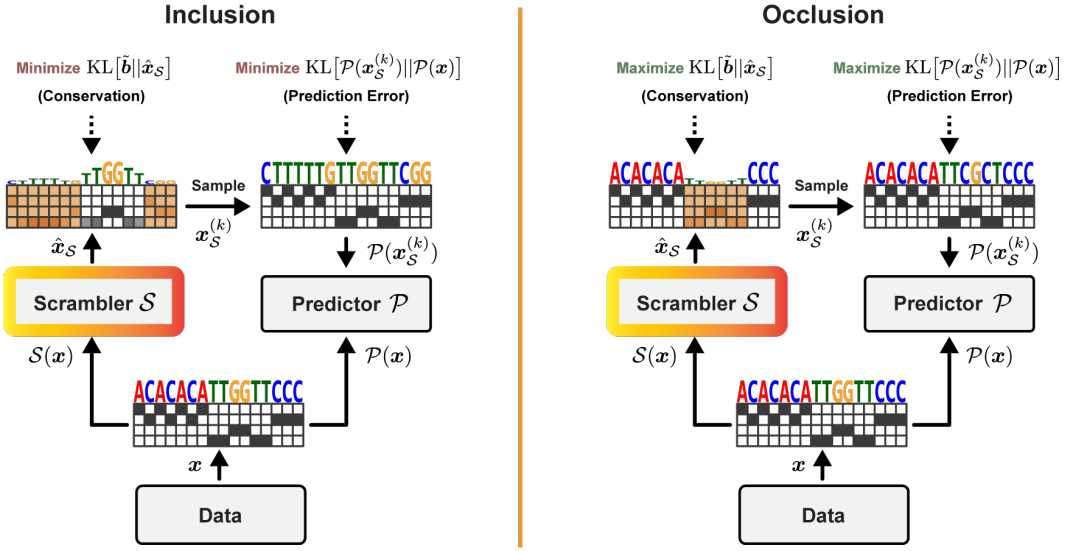

Figure 6.2: Inclusion: Maximize the entropy of the PSSM predicted by the Scrambler and minimize the prediction error of samples drawn from it. Occlusion: Minimize PSSM entropy and maximize sample prediction error.

Alternatively, we can train $\mathcal{S}$ to find the smallest set of features in $\boldsymbol{x}$ to randomize (i.e.,

maximizing the conservation of $\hat{\boldsymbol{x}}_{\mathcal{S}}$) to maximally perturb $\mathcal{P}(\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)})$ from $\mathcal{P}(\boldsymbol{x})$. We refer to this inverse formulation as the *Occlusion-Scrambler* (Figure 6.2, right). The scrambling operation for the Occlusion-model is defined in Equation 6.3 below. Here, the expression $\boldsymbol{x} \times \dot{\mathcal{S}}(\boldsymbol{x})$ has been replaced with $\boldsymbol{x} / \dot{\mathcal{S}}(\boldsymbol{x})$. Similarly, the cost function is redefined to instead *maximize* the predictive reconstruction error (Equation 6.4).

$$\hat{\boldsymbol{x}}_{\mathcal{S}} = \sigma\big(\log \tilde{\boldsymbol{b}} + \boldsymbol{x} / \dot{\mathcal{S}}(\boldsymbol{x})\big) \tag{6.3}$$

$$\min_{\mathcal{S}} -\left(\frac{1}{K}\sum_{k=1}^{K} \mathrm{KL}\big[\mathcal{P}(\boldsymbol{x}_{\mathcal{S}}^{(k)}) || \mathcal{P}(\boldsymbol{x})\big]\right) + \lambda \cdot \left(t_{\text{bits}} - \frac{1}{N} \cdot \mathrm{KL}\big[\tilde{\boldsymbol{b}} || \hat{\boldsymbol{x}}_{\mathcal{S}}\big]\right)^2 \tag{6.4}$$

In all experiments, the Scrambler $\mathcal{S}$ consisted of a residual network with 4-20 blocks of dilated convolutions and filter sizes between 3 and 8 [63], and Scramblers were trained on input examples separate from those used in the benchmark tests. By training $\mathcal{S}$ on a large representative data set, we learn a parametric model of feature importance whose predictions generalize to new examples, preventing overfitting to spurious predictor signals. The baseline method used for comparison, *Perturb*, exchanges the categorical value of one letter or pixel at a time and estimates the absolute value in predicted change as the importance score. Comparisons are made against Perturb (baseline), Gradient Saliency [139], Guided Backprop [146], Integrated Gradients [150], DeepLIFT [138] (using RevealCancel for MNIST and Rescale from DeepExplain for the remaining tasks due to predictor compatibility issues) [4], SHAP DeepExplainer [103] and the extremal preservation/perturbation methods of [49, 48, 38, 22, 21]. For comparison, we also use a version of the Scrambler with a masking operator similar to L2X and INVASE (referred to as 'Zero Scrambler'): $\min_{\mathcal{M}} \mathrm{KL}\big[\mathcal{P}(\boldsymbol{x} \times \mathcal{M}^{(\mathrm{G})}(\boldsymbol{x})) || \mathcal{P}(\boldsymbol{x})\big] + \lambda \cdot \big(t_{\text{area}} - 1/N \cdot \sum_{i=1}^{N} \mathcal{M}^{(\mathrm{S})}(\boldsymbol{x})_i\big)^2$, where $\mathcal{M}$ is a binary 0/1 mask generator. The continuous-valued mask $\mathcal{M}^{(\mathrm{S})}(\boldsymbol{x})$ and binarized mask $\mathcal{M}^{(\mathrm{G})}(\boldsymbol{x})$ are obtained by applying sigmoid activations and Gumbel sampling on the output vector of $\mathcal{M}$ respectively. Internally, the network $\mathcal{M}$ is identical to the Scrambler.

6.2 The cis-Regulatory Code of Alternative Polyadenylation

We first used Scramblers to interpret regulatory sequence elements in the 3' untranslated region (UTR) of pre-mRNA. Specifically, using APARENT [17] (see Chapter 3) we trained an Inclusion-Scrambler to reconstruct isoform predictions for the original APARENT training data. As we anticipated important polyadenylation features like RNA binding protein (RBP) motifs to consist of short subsequences, we regularized the Scrambler by fixing the final convolutional layer of the network to a Gaussian filter (reminiscent of a technique proposed by [48]) to encourage the selection of contiguous features for masking. By training three independent Scramblers at separate target conservations ($t_{\text{bits}} = 0.25$, 0.5 and 1.0) We found that they learned to recognize known regulatory binding factors associated with alternative polyadenylation (Figure 6.3), such as CFIm, CstF, HNRNPL, ENOX1, PABPN1, HuR, and RBM4 [54, 137, 43, 154].

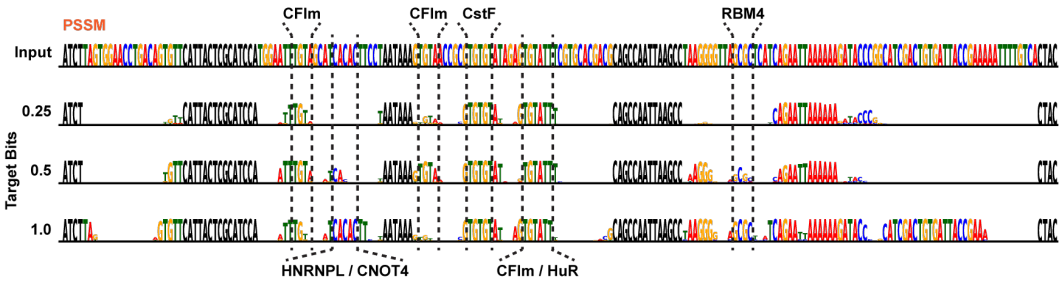

Figure 6.3: Example attribution of a strong polyadenylation signal sequence from the APARENT test set, using Inclusion-Scramblers trained with increasing conservation.

To compare the Scrambler to other attribution methods we performed two different benchmarks: first, based on the importance scores of each method, we replaced all but the top $X\%$ most important nucleotides with random letters sampled from the background distribution and measured the KL-divergence between the predictions of the original and randomized sequences. The second benchmark inversely tested how well these top-ranked nucleotides perturbed the prediction when replaced with random samples, leaving the rest of the sequence

intact. The results showed that the Inclusion- and Occlusion-Scramblers were superior to the other tested methods on each benchmark, as the measured KL-divergences were consistently the lowest and highest, respectively (Figure 6.4).

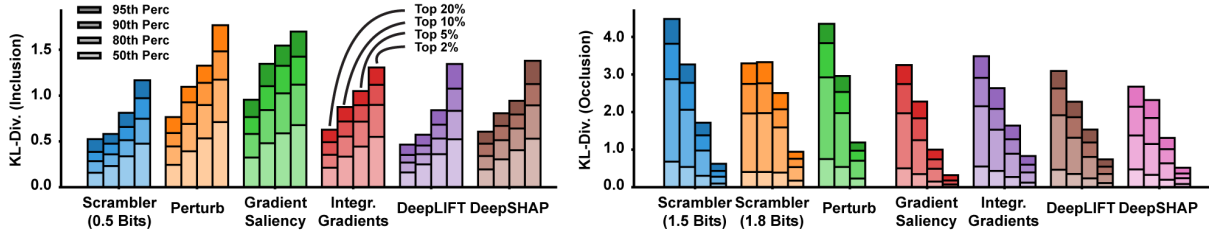

Figure 6.4: Left: Replacing all but the 20%, 10%, 5% and 2% most important positions per sequence with random nucleotides. Right: Inversely replacing these highest-scoring positions with random samples. Measuring prediction KL-divergence against the original sequences.

Figure 6.5 displays example attributions of a relatively weak polyadenylation signal for each of the tested methods. For the Scrambler, we show both the importance scores $\mathcal{S}(\boldsymbol{x})$ as well as the scrambled PSSM $\hat{\boldsymbol{x}}_{\mathcal{S}}$.

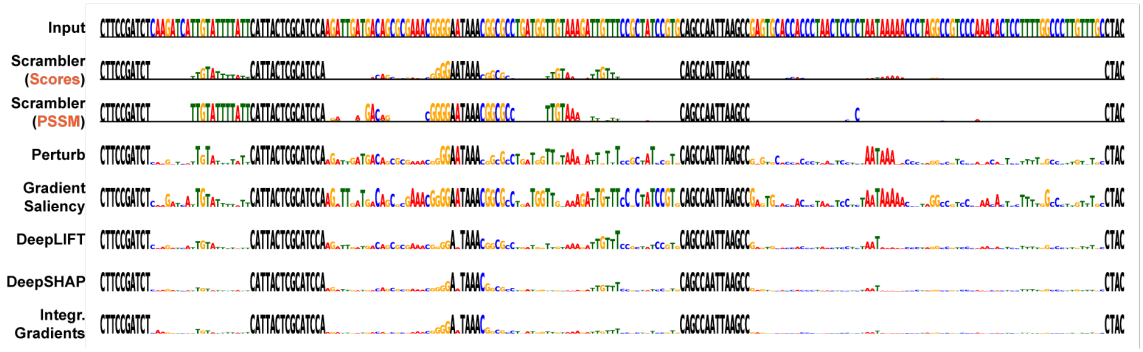

Figure 6.5: Example attribution of a polyadenylation signal, comparing different methods.

When removing the fixed Gaussian filter from the Scrambler, the network learned occlusion patterns which resulted in more extreme predicted perturbations. However, when inspecting the generated patterns, they clearly exploited the importance of scattered T's in favor of selecting biologically relevant motifs (Figure 6.6).

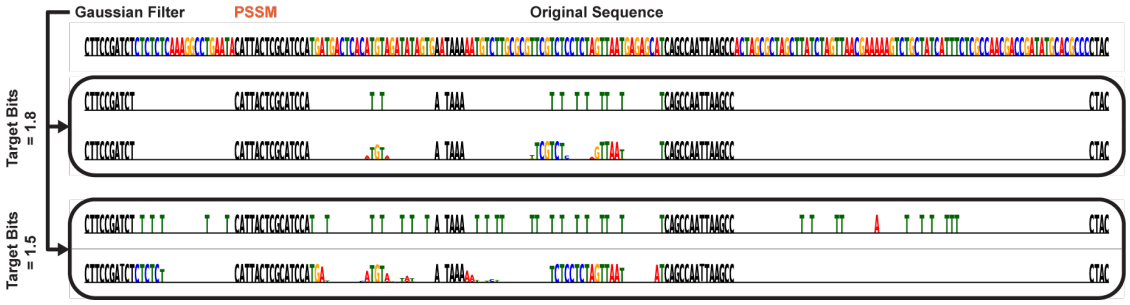

Figure 6.6: Example attribution using two different Occlusion-Scramblers ($t_{\text{bits}} = 1.8$ and 1.5 respectively), with and without the Gaussian filter.

In addition to overall reconstructive features, with interpretation methods like SHAP or DeepLIFT the *sign* of each feature's importance score describes whether the feature is positively or negatively influencing the prediction. Similarly, we can alter the Scrambler training cost to find the smallest set of features that either maximize or minimize a prediction, rather than reconstructing it. Specifically, to find positive APA features, we trained the Scrambler to minimize Equation 6.5. To find negative features, we minimized Equation 6.6.

$$\min_{\mathcal{S}} -\left(\frac{1}{K}\sum_{k=1}^{K}\log \mathcal{P}(\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)})\right) + \lambda \cdot \left(t_{\text{bits}} - \frac{1}{N}\cdot \text{KL}\left[\tilde{\boldsymbol{b}}||\hat{\boldsymbol{x}}_{\mathcal{S}}\right]\right)^2 \tag{6.5}$$

$$\min_{\mathcal{S}} -\left(\frac{1}{K}\sum_{k=1}^{K}\log \left(1 - \mathcal{P}(\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)})\right)\right) + \lambda \cdot \left(t_{\text{bits}} - \frac{1}{N}\cdot \text{KL}\left[\tilde{\boldsymbol{b}}||\hat{\boldsymbol{x}}_{\mathcal{S}}\right]\right)^2 \tag{6.6}$$

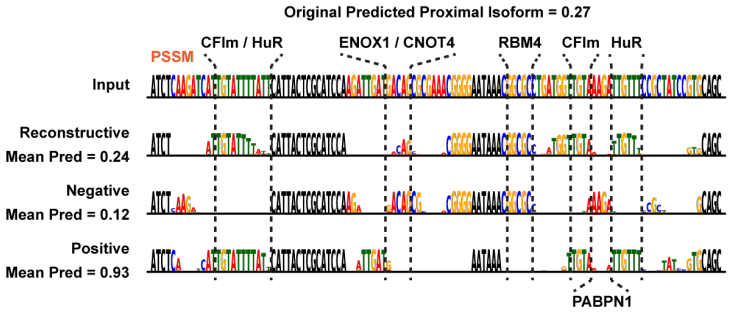

Figure 6.7: Example of Inclusion-Scramblers trained to reconstruct, maximise or minimize predictions, thus finding overall important, enhancing or repressing motifs, respectively.

With this approach, we partitioned cis-regulatory elements within polyadenylation signals into enhancing (e.g., CFIm, HuR, Cstf) and repressive motifs (e.g., ENOX1, RBM4, PABPN1), in agreement with the known function for these motifs and associated RBPs (Figure 6.7).

6.3 The Rules of Translation Efficiency in the 5' UTR

The translation efficiency of mRNA is controlled by complex regulatory logic in its 5' UTR. For example, in-frame (IF) start and stop codons exhibit NAND logic by creating an IF upstream open reading frame (uORF), which represses translation. Sequences with multiple IF starts and stops can further complicate translational logic by creating NAND-OR hybrid functions with overlapping IF uORFs.

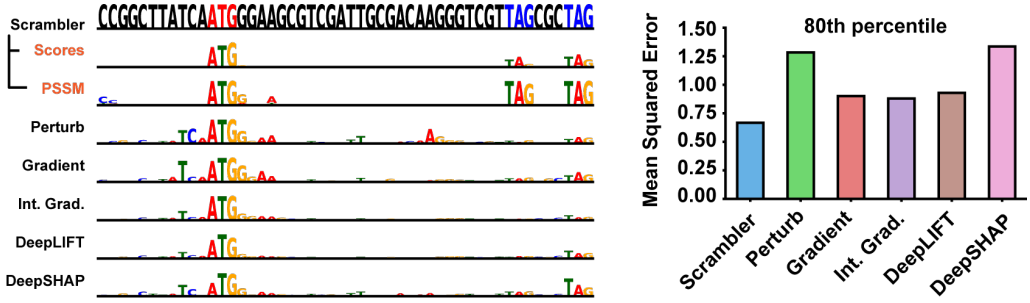

Figure 6.8: Left: Example attribution in a 5' UTR with multiple IF stop codons. Right: Predictive reconstruction in a test set of synthetic IF uORF 5' UTRs, when only keeping the 6 most important nucleotides.

We next trained Inclusion-Scramblers to reconstruct the mean ribosome Load (MRL – a proxy for translation efficiency) of synthetic 5' UTRs as predicted by the CNN Optimus 5-Prime [130]. To test how well attribution methods detect regulatory involving multiple components, we generated a synthetic 5' UTR dataset with one IF start and two IF stops, creating two overlapping IF uORFs in each sequence. We then measured to what extent the most important nucleotides of each method reconstructed the CNN predictions (identical to the benchmark of the previous section). An example 5' UTR from the test set with multiple

IF uORFs is shown in Figure 6.8 (left); here, the multiple stops effectively hide one another from many local attribution methods. These troublesome artifacts are also reflected in the benchmark comparison (Figure 6.8, right): the Scrambler has a considerably better (lower) reconstruction error compared to other methods.

The combinatorial nature of out-of-frame start codons (OOF uAUGs) and IF uORFs complicates variant interpretation in human 5' UTRs [20, 5, 159]. In Figure 6.9, we interpret a variant, *rs1160558441*, which creates an OOF uAUG that Optimus 5-Prime predicts to be functionally silent. The OOF uAUG is created within an IF uORF which hides its effect, as either element alone is sufficient to repress translation. Therefore, the variant sequence is not interpretable by Perturbation as the uAUGs are not found. A Scrambler trained with a low entropy penalty ($t_{\text{bits}} = 0.125$, $\lambda = 1$) detects both possible interpretations. With a higher penalty ($\lambda = 10$), the Scrambler must find a smaller set of salient features and marks only the OOF uAUG as important, which is sufficient to explain the repression. However, the Scrambler identifies the subsequence 'ATGA,' but the trailing adenine is not actually important for the explanation. The problem is that the number of salient features is highly variable in any given 5' UTR, and so a Scrambler trained for a target conservation t_{bits} may 'over-interpret' some examples.

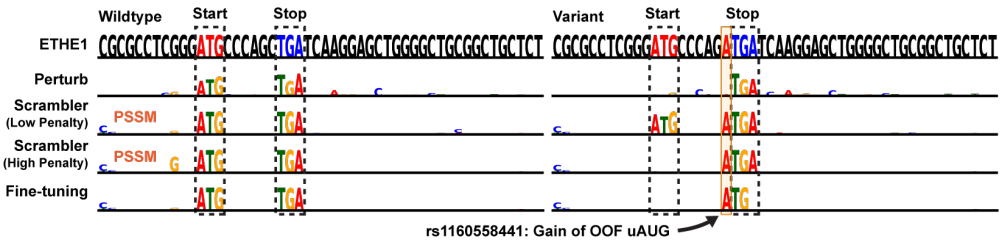

Figure 6.9: Interpreting a functionally silent mutation (*rs1160558441*) in the 5' UTR of the *ETHE1* gene, where an OOF uAUG is created in an IF uORF.

To overcome issues with over-interpretation for datasets with a highly variable number of important features, we apply a per-example fine-tuning step: starting with $\mathcal{S}(\boldsymbol{x})$, we optimize new scores $\boldsymbol{s}$ by gradient descent which are specific to $\boldsymbol{x}$ and remove excessive features of

$\mathcal{S}(\boldsymbol{x})$ and maximize the entropy of the PSSM unbounded (Equation 6.7-6.8). Here, the scrambling equation is reformulated with the expression $\max[\dot{\mathcal{S}}(\boldsymbol{x}) - \boldsymbol{s}, \boldsymbol{0}]$; this forces the resulting fine-tuned importance scores to select a subset of the features identified by the original scores $\mathcal{S}(\boldsymbol{x})$, i.e. the new scores cannot find a new explanation for $\boldsymbol{x}$.

$$\min_{\boldsymbol{s}} \left(\frac{1}{K} \sum_{k=1}^{K} \mathrm{KL}\big[\mathcal{P}(\boldsymbol{x}_{\boldsymbol{s}}^{(k)}) || \mathcal{P}(\boldsymbol{x})\big] \right) + \lambda \cdot \frac{1}{N} \cdot \mathrm{KL}\big[\tilde{\boldsymbol{b}} || \hat{\boldsymbol{x}}_{\boldsymbol{s}}\big] \tag{6.7}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{s}} = \sigma\big(\log \tilde{\boldsymbol{b}} + \boldsymbol{x} \times \max\big[\dot{\mathcal{S}}(\boldsymbol{x}) - \boldsymbol{s}, \boldsymbol{0}\big] \big) \tag{6.8}$$

As can be seen for the variant example (Figure 6.9), fine-tuning removes the trailing 'A'. Importantly, when compared on the '1 Start / 2 Stop' benchmark, per-example fine-tuning applied to the pre-trained Scrambler scores produces more robust attribution results compared to running per-example optimization of importance scores without a Scrambler (Optimus 5-Prime MSE after fine-tuning: 0.098; MSE after independent optimization: 0.161). This is likely because the latter approach can become stuck in local minima and produce poorly generalizable interpretations (essentially 'overfitting' to spurious predictor signals).

Sequences with multiple IF uORFs are examples of patterns with redundant salient feature sets. Scramblers can find multiple salient feature sets such as these with the use of dropout masks, which disallow specific sequence positions in the retained feature set. Specifically, we let $\boldsymbol{D} \in \{0, 1\}^N$ be the dropout mask and apply them to the importance scores $\mathcal{S}(\boldsymbol{x})$ by element-wise multiplication:

$$\hat{\boldsymbol{x}}_{\mathcal{S}} = \sigma\big(\log \tilde{\boldsymbol{b}} + \boldsymbol{x} \times \dot{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{D}) \times \dot{\boldsymbol{D}} \big) \tag{6.9}$$

Importantly, the Scrambler network $\mathcal{S}$ also receives $\boldsymbol{D}$ as additional input, allowing the network to learn to output alternate scores conditioned on which positions were dropped or enforced. During training, we use random samples of $\boldsymbol{D}$. After training on randomized dropout patterns, we can provide hand-crafted patterns at inference time to detect alternate feature sets of new input examples. With the added dropout mechanism, we obtain a model capable of dynamically proposing different IF uORFs as attribution solutions (Figure 6.10).

Without dropout and a low entropy penalty ($t_{\text{bits}} = 0.125$, $\lambda = 1$), the Scrambler marks both IF stops (i.e. both IF uORFs). However, with a higher penalty ($\lambda = 10$), the Scrambler finds a smaller interpretation with only one IF stop. When using dropout patterns to exclude either of the IF stops, the Scrambler dynamically finds the alternate IF uORF.

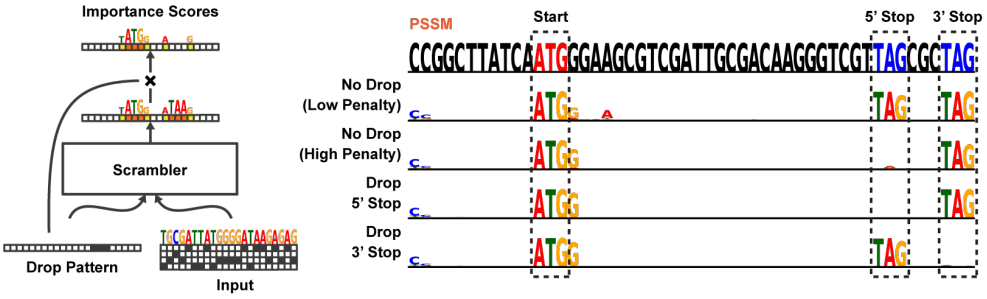

Figure 6.10: Finding alternate IF uORFs by separately dropping each of the stops.

6.4 The Determinants of Protein-Protein Interactions

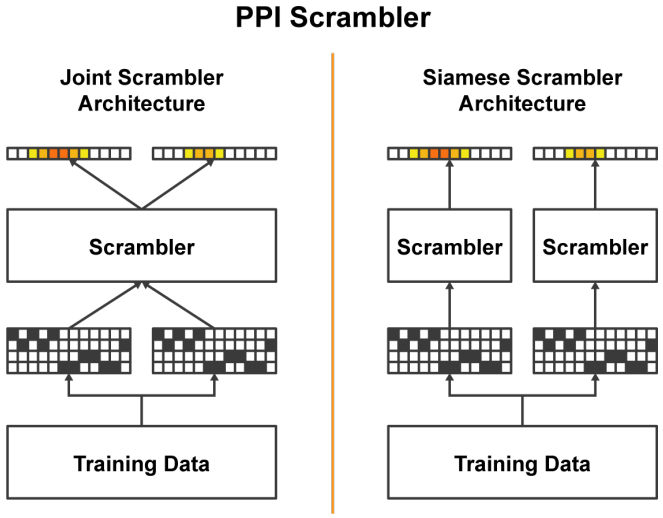

Figure 6.11: The *joint* and *siamese* Scrambler architectures.

Finally, we applied Scramblers to interpret predicted interactions for pairs of proteins. This can be challenging, as proteins are defined along a narrow manifold of stably folded se-

quences. Any interpretation method must ensure that masked or perturbed sequences stay in-distribution for predictions to remain accurate. We focused on a set of rationally designed coiled-coil heterodimers, where binding specificity is induced by hydrogen bond networks (HBNets) at the dimer interface [105, 27]. Using the designed heterodimers ($\sim 180,000$) as positive training data and a negative set consisting of randomly paired monomers, we trained a recurrent neural network (RNN) to predict dimer binding (AUC = 0.96 on held-out test data). With a test set of 480 designed heterodimers, we asked how well attribution methods could recapitulate HBNet positions.

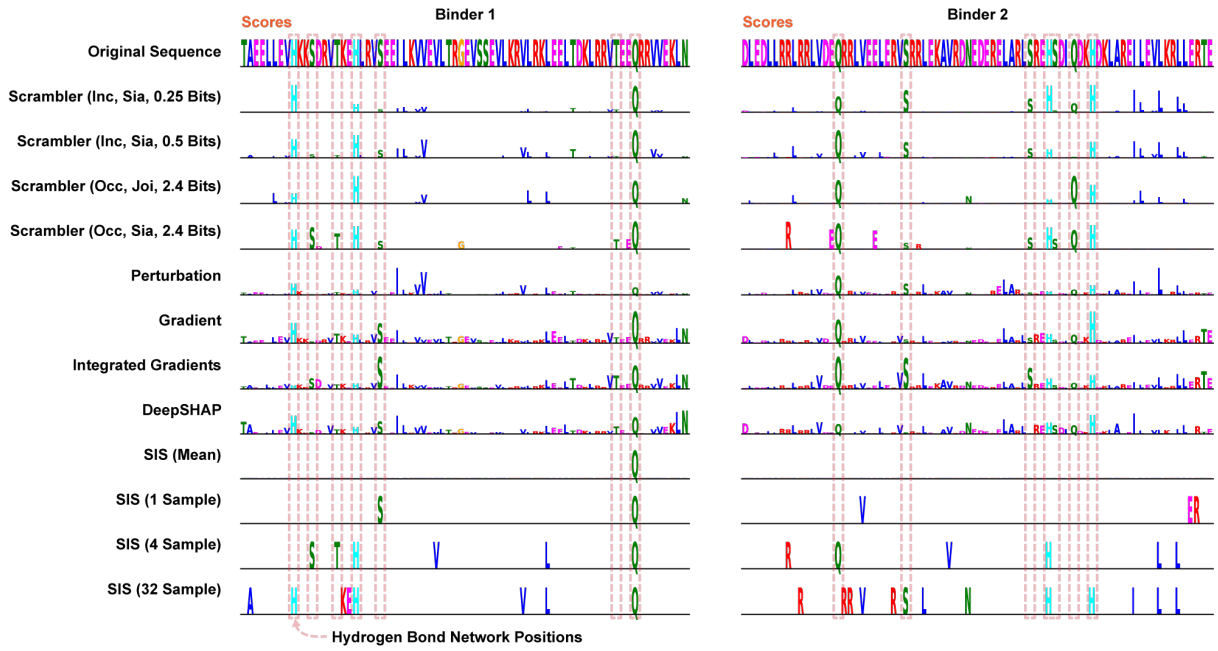

Figure 6.12: Example attributions of a designed heterodimer binder pair, for a selection of all the benchmarked methods.

For the DNA attribution tasks of previous sections, the predictors used only one input sequence. Here, however, each dimerization prediction involves two inputs. Consequently, there are multiple ways in which to define the Scrambler architecture. We first tested a *joint* Occlusion-Scrambler (Figure 6.11, left), which saw both input sequences. This architecture learned to identify a subset of HBNet positions and hydrophobic residues at the interface

necessary for binding to its cognate partner (Figure 6.12). We also tested a *siamese* architecture, which saw only one input sequence at a time and hence must learn features which are good determinants of binding specificity independent of the binding partner (Figure 6.11, right). This architecture learned to identify nearly all HBNets (Figure 6.12).

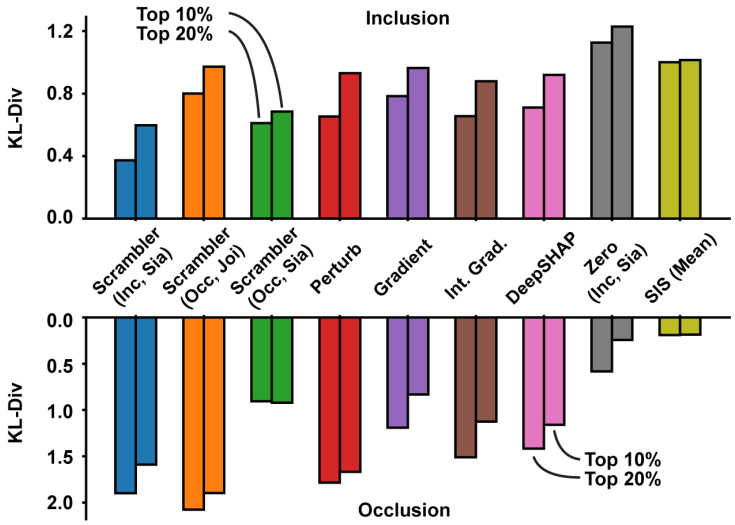

Figure 6.13: Keeping the top $X\%$ residues according to the importance scores of each attribution method and replacing the rest with random amino acids (top), or replacing the top $X\%$ with random amino acids (bottom), measuring prediction KL-divergence.

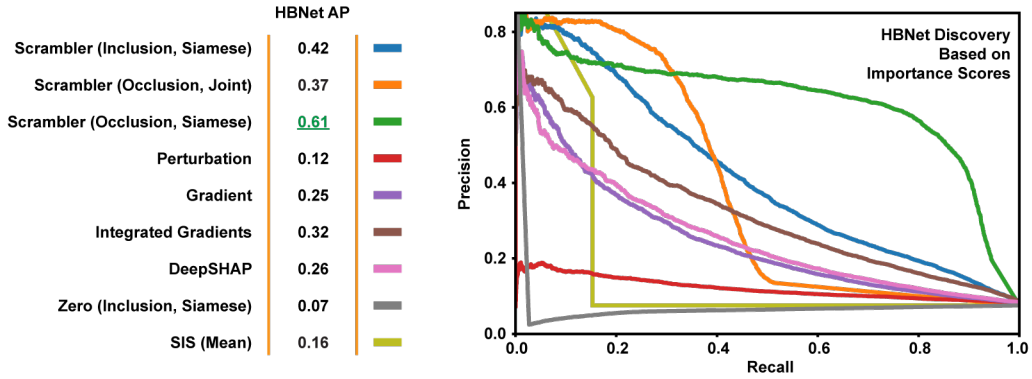

Figure 6.14: Precision-recall curves for discovering HBNet positions based on the importance scores of each benchmarked method.

We then compared these Scrambler architectures to other attribution methods based on how much the positions with largest absolute-valued score either preserved or perturbed the RNN predictions (Figure 6.13; $t_{\text{bits}} = 0.25$, Inclusion; $t_{\text{bits}} = 2.4$, Occlusion). The (siamese) Inclusion-Scrambler and (joint) Occlusion-Scrambler had the best (lowest and highest) median KL-divergence for these tasks.

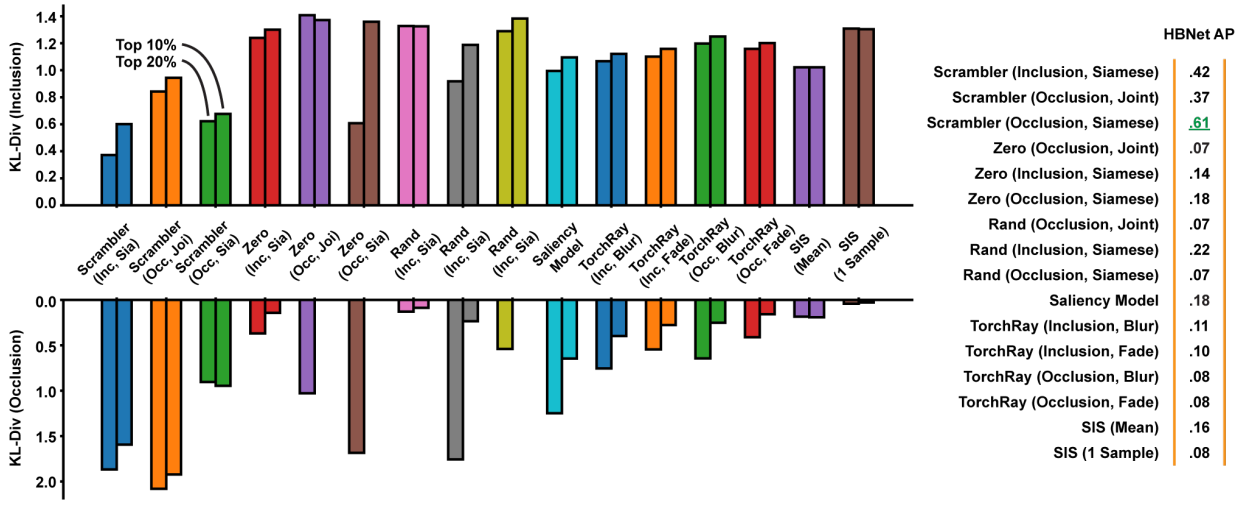

Figure 6.15: Benchmark comparison of Scrambler-like methods. Left: KL-divergence benchmark based on the predictor RNN. Right: HBNet Discovery Average Precision.

Next, we compared the methods by their ability to discover HBNet positions based on their importance scores (the 'ground truth' binding rules). While the joint Scrambler discovered significantly more HBNet positions than other methods – likely because a generative model trained on many examples learns more generalizable attributions – the siamese Scrambler was superior with an average precision of 0.61 (Figure 6.14). These results support the idea that the siamese architecture learned discriminative features for both interacting and non-interacting pairs – HBNet residues. The fact that the joint Scrambler had better precision than per-example attribution methods suggest that using a generative model results in more generalizable features by overcoming spurious RNN signals. We performed additional comparisons to other attribution methods in Figure 6.15. We found that the temperature-

based masking operator of Equation 6.1 consistently outperformed other masks and that generative interpretations were better than per-example masking approaches.

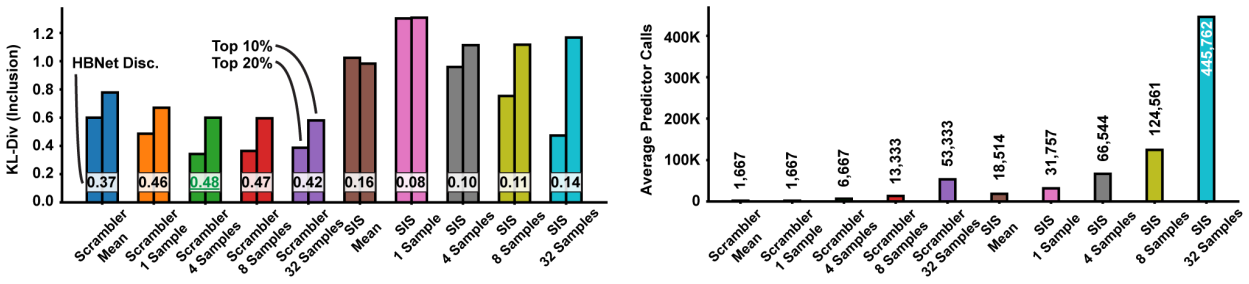

Figure 6.16: Supplemental comparison between Scramblers and Sufficient Input Subsets (SIS) with 'hot-deck' sampled masking. Left: KL-divergence benchmark based on the predictor RNN, HBNet Discovery Precisions annotated on top of the bar chart. Right: Average number of predictor queries used to interpret a single input pattern.

Finally, we tested the Scrambler against different versions of the 'hot-deck' SIS method [22]: For each iteration of SIS, we sampled a number of background sequences and used these to mask de-selected features. The mean sample prediction was used as the function value. Similarly, we varied the number of Gumbel patterns sampled from the Scrambler PSSM during training (K in Equation 6.2). The Scrambler operated well with few (≥ 4) samples and consistently outperformed SIS with 32 samples, both in predictive reconstruction and ground-truth comparisons (Figure 6.16, left). Additionally, the total number of predictor queries required to interpret the entire dataset with comparable quality was $\sim$ 70 times lower than SIS (Figure 6.16, right). Interestingly, using a simple masking scheme such as mean features resulted in bad interpretations for the dimer RNN predictor.

Chapter 7

FUTURE DIRECTIONS

In Chapter 3, we presented an approach based on neural networks and MPRAs for learning to accurately predict a gene-regulatory phenomenon (Alternative Polyadenylation) and classify disruptive genetic variants. In Chapter 4 and 5, we developed efficient, scalable sequence design methods based on activation maximization that supports, rapid, data-driven biomolecular engineering. Finally, in Chapter 6, we developed an interpretation method suitable for sequence-based neural networks that enables attributing DNA-, RNA- and protein function predictions to features in the primary sequence.

In this chapter, we elaborate on extensions and applications to variant prediction and molecular sequence design that could potentially improve upon the state-of-the-art in these areas. First, we discuss augmenting variant prediction pipelines with deep learning models that are capable of providing uncertainty estimates in addition to point estimates of their predictions. Second, we briefly explore ideas of combining uncertainty estimation models with generative networks such as Deep Exploration Networks to support efficient, amortized, batched Bayesian Optimization. Third, we discuss potential use cases and application areas for sequence design methods. Finally, we propose an architecture and report initial results for designing *de novo* protein binders using activation-maximizing generative networks.

7.1 Variant Prediction with Uncertainty Estimation

Frameworks based on deep learning for inferring the impact of genetic variants often rely on neural networks that provide point-estimate predictions [1, 173, 17, 28, 68]. For example, if the predictor $\mathcal{P}$ returns a probability prediction $\mathcal{P}(\boldsymbol{x})$ in the range $[0, 1]$, there are two common proxies for measuring the impact of variant sequence $\boldsymbol{x}_{\text{var}}$ with respect to wildtype

sequence $\boldsymbol{x}_{\text{wt}}$: (1) We either calculate the change in probability, $\Delta p = \mathcal{P}(\boldsymbol{x}_{\text{var}}) - \mathcal{P}(\boldsymbol{x}_{\text{wt}})$ or (2) we calculate the log odds ratio (fold change), $\text{LOR} = \frac{\mathcal{P}(\boldsymbol{x}_{\text{var}})/(1-\mathcal{P}(\boldsymbol{x}_{\text{var}}))}{\mathcal{P}(\boldsymbol{x}_{\text{wt}})/(1-\mathcal{P}(\boldsymbol{x}_{\text{wt}}))}$. These point estimates provide no information about how *confident* $\mathcal{P}$ is in the prediction.

Bayesian Neural Networks (BNNs) were in fact used early on for estimating uncertainty in the context of variant prediction [166, 165], but have since been abandoned in favor of very deep neural networks. BNNs are also known to excessively regularize the model, which is often not desired as it can lead to less accurate predictions. Recently, however, researchers have started applying modern uncertainty estimation methods, for example based on ensembles of neural networks, to variant prediction [66]. These methods are compatible with high-capacity predictors such as residual neural networks and can likely be extended to many different gene-regulatory functions – splicing, polyadenylation, translation and stability – without losing predictive power.

These methods provide not only a point-estimate (the mean) prediction $\mu_{\mathcal{P}}(\boldsymbol{x})$, but also an uncertainty estimate $\sigma_{\mathcal{P}}(\boldsymbol{x})$ for a given input pattern $\boldsymbol{x}$. Given these entities, we can reframe the variant prediction problem to provide confidence intervals of the estimated effect sizes. This can increase the trust we place in neural networks for clinical settings and can also increase our precision in variant prioritization, as we could ignore predictions with low confidence. However, there are several different techniques for uncertainty estimation in deep learning, such as adaptive basis regression that uses Bayesian Regression for their final layer [144], ensembles of neural networks [85] or random weight dropout (DropConnect) [113]. It is not immediately clear which method would yield the best confidence scores while still maintaining predictive power, or for that matter if a single method is universally applicable to all gene-regulatory prediction problems. Hence, systematic evaluation of these methods will be necessary to determine their usefulness and in which settings they are appropriate.

7.2 Bayesian Optimization and Generative Models

In Section 4.7 and Section 5.5 we used predictor models capable of estimating uncertainty in their predictions in order to generate more confident sequence designs. Specifically, given

mean- and uncertainty predictions $\mu_{\mathcal{P}}(\boldsymbol{x})$ and $\sigma_{\mathcal{P}}(\boldsymbol{x})$, we could use activation maximization (or DENs) to maximize $\mu_{\mathcal{P}}(\boldsymbol{x})$ while minimizing $\sigma_{\mathcal{P}}(\boldsymbol{x})$ according to the Probability-of-Improvement objective [134].

However, we could also use these types of predictor models to construct efficient design methods based on Bayesian Optimization (BO). In BO, the goal is still to generate designs which maximize the fitness objective, but now we are allowed to conduct multiple rounds of experiments in order to collect new measurements. This paradigm of *Active Learning* sounds like a great fit for synthetic molecular biology. However, most (tractable) BO methods assume that we only collect data for one new candidate design in each round of experiments. But modern experimental protocols in genomics and proteomics support high-throughput screening of hundreds of thousands, if not millions, of designs in parallel.

Researchers have recently started to close this gap, for example by developing methods that rely on large-scale Thompson Sampling to support approximate batched BO [11], but this research area is still largely unexplored. We hypothesize that we can train a deep generative model to learn to sample sequences under an approximate BO objective in order to provide efficient, amortized design. Specifically, we might be able to combine DENs, which are tailored for generating a large batch of diverse sequence designs, with a BO acquisition function such as Thompson Sampling or Expected Improvement under the Constant Liar approximation [50]. For example, if we combine DENs with a predictor model based on DropConnect and we deterministically tie every generator seed to a specific DropConnect dropout configuration, we would theoretically learn a parametric generative model of Thompson Sampling.

7.3 Vaccine-, Antimicrobial peptide- and Antibody Design

Machine Learning is beginning to be applied to the design of novel drug molecules, vaccine development and other important proteins [168]. For example, Amimeur et al. (2020) used a Generative Adversarial Network (GAN) trained on a naive llama antibody repertoire to learn the manifold of functional antibody proteins, after which new antibodies could be

sampled from the GAN and enriched for favorable properties using filtering [3]. Similarly, Riesselman et al. (2019) used an Autoregressive network to learn the manifold of single-domain antibody peptides (nanobodies) [127]. It is worth exploring whether DENs, which provide a more direct optimization and control of diversity, can be helpful for antibody design. Many of the predictors for the most important antibody properties, such as MHC Class I and II binding, are already based on deep learning [104, 75], so they are easily integrated in the DEN pipeline. Furthermore, while designing naive antibodies with improved properties is useful, the ultimate task is to design antibodies for a specific target. Using DENs, we could utilize protein structure predictors such as AlphaFold [133] or trRosetta [167] to start approaching this difficult task as a direct optimization problem.

Beyond antibody design, there have been recent ML-guided efforts to design coronavirus vaccines [74, 101]. We could similarly apply DENs to this design task, since the core predictive models required, such as predicted display of epitopes by MHC molecules, are based on deep learning. Another important therapeutic innovation is the development of antimicrobial peptides which are effective against antibiotic-resistant bacteria [40]. A DEN approach could be used to design diverse peptides directly optimized for antimicrobial activity. Finally, DEN-generated antibody, antimicrobial and vaccine molecules can be validated with low- or high-throughput experiments using for example cell death, growth or cellular immunology assays.

7.4 Heterodimer Binder Design with Contrastive Generative Networks

In the design tasks covered so far, fitness has always been defined as a local metric evaluated independently on each designed sequence. However, many design problems pose global optimization objectives that involve multiple, if not all, designed sequences. An important such problem is that of protein heterodimer binder design. Here, the goal is to construct a large set of protein binder pairs, which strongly bind to their cognate partner but have minimal off-target interactions with other binders. Large orthogonal sets of binders would dramatically accelerate synthetic circuit building inside cells. The current state-of-the-art design

methods do not, however, globally minimize off-target effects during the initial candidate design step [27]. Here we propose to take advantage of DENs to do just this.

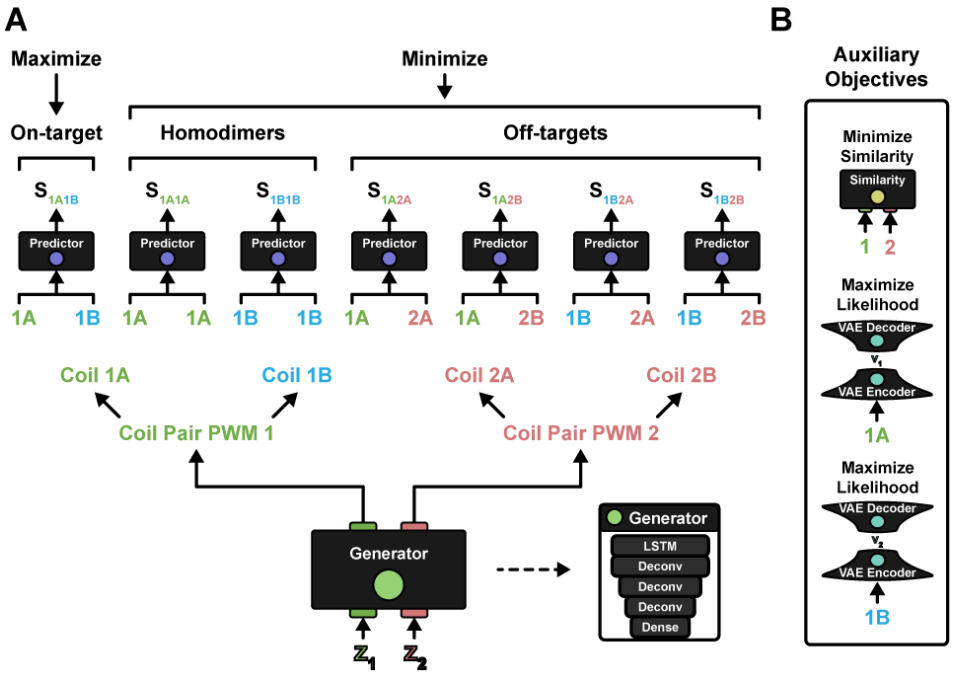

Figure 7.1: DEN architecture for designing orthogonal heterodimer protein binders (A) and auxiliary cost functions for maintaining diversity and confidence (B).

$$
\begin{aligned}
\min_{\mathcal{G}} \quad & -\lambda_{\text{On}} \cdot \mathcal{P}\left[\boldsymbol{a}(\boldsymbol{z}^{(1)}), \boldsymbol{b}(\boldsymbol{z}^{(1)})\right] \\
& +\lambda_{\text{H}} \cdot \left(\mathcal{P}\left[\boldsymbol{a}(\boldsymbol{z}^{(1)}), \boldsymbol{a}(\boldsymbol{z}^{(1)})\right] + \mathcal{P}\left[\boldsymbol{b}(\boldsymbol{z}^{(1)}), \boldsymbol{b}(\boldsymbol{z}^{(1)})\right]\right) \\
& +\lambda_{\text{Off}} \cdot \left(\mathcal{P}\left[\boldsymbol{a}(\boldsymbol{z}^{(1)}), \boldsymbol{a}(\boldsymbol{z}^{(2)})\right] + \mathcal{P}\left[\boldsymbol{a}(\boldsymbol{z}^{(1)}), \boldsymbol{b}(\boldsymbol{z}^{(2)})\right] + \mathcal{P}\left[\boldsymbol{b}(\boldsymbol{z}^{(1)}), \boldsymbol{a}(\boldsymbol{z}^{(2)})\right] + \mathcal{P}\left[\boldsymbol{b}(\boldsymbol{z}^{(1)}), \boldsymbol{b}(\boldsymbol{z}^{(2)})\right]\right) \\
& +\lambda_{\text{D}} \cdot \mathcal{C}_{\text{Diversity}}\left[\boldsymbol{x}(\boldsymbol{z}^{(1)}), \boldsymbol{x}(\boldsymbol{z}^{(2)})\right] \\
& +\lambda_{\text{L}} \cdot \mathcal{C}_{\text{Likelihood}}\left[\mathbb{E}_{z}[\log p_{\text{VAE}}(\boldsymbol{a}(\boldsymbol{z}))]\right] + \lambda_{\text{L}} \cdot \mathcal{C}_{\text{Likelihood}}\left[\mathbb{E}_{z}[\log p_{\text{VAE}}(\boldsymbol{b}(\boldsymbol{z}))]\right]
\end{aligned}
\tag{7.1}
$$

The architecture is illustrated in Figure 7.1. At each step, the generator produces two sequences $\boldsymbol{x}(\boldsymbol{z}^{(1)})$ and $\boldsymbol{x}(\boldsymbol{z}^{(2)})$ given two seeds $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)} \sim U(-1, 1)^D$ as input. Each sequence

actually encodes a pair of binders (the first half of the sequence is binder one, etc.). Let $\boldsymbol{a}(\boldsymbol{z}^{(1)})$ refer to the first binder of seed $\boldsymbol{z}^{(1)}$ and let $\boldsymbol{b}(\boldsymbol{z}^{(1)})$ refer to the second binder. Given a differentiable binding affinity predictor $\mathcal{P}$, we can optimize the DEN to maximize the on-target binding while minimizing off-target effects (Equation 7.1).

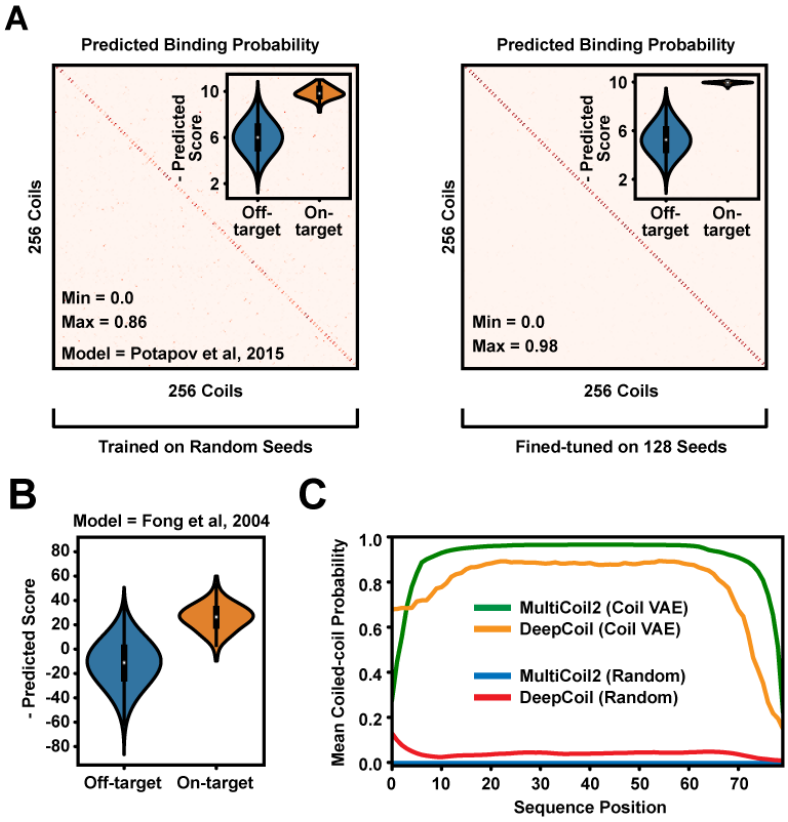

Figure 7.2: Computational design of 128 binder pairs. (A) Predicted off- and on-target binding probabilities using the SVM model of Potapov et al. Left: Trained on randomly sampled pairs of seeds. Right: Further fine-tuning on a fixed set of seeds. (B) Independent validation of binding specificity using the model of Fong et al. (C) Predicted average probability that the designed sequences fold into stable coils, using either an HMM model (Multicoil2) or a deep neural network (DeepCoil).

As optimization progresses, the DEN will learn to generate binder pairs which, in expectation over all latent seeds, bind specifically to their cognate partner. To fine-tune the DEN for a smaller, finite set of binders, we can briefly stop training, sample a fixed set of

seeds (one for each pair we wish to design), and resume optimization only on those seeds. A VAE is used to enforce that binders respect some appropriate manifold, for example coil-like proteins. As in Section 4.6, the VAE likelihood is estimated by importance sampling and gradients are backpropagated using a ST estimator. Using this approach, we designed a set of 128 bZIP-like coiled heterodimer binders that were 80 residues long, using the predictor of [121] for optimization and the one from [47] for validation. The results show that we can design binders which are predicted to target their partners with high specificity (Figure 7.2A-B). The binders are also predicted to form stable coils by two independent models [155, 102] (Figure 7.2C).

BIBLIOGRAPHY

[1] B. Alipanahi, A. Delong, M.T. Weirauch, and B.J. Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

[2] M. AlQuraishi. End-to-end differentiable learning of protein structure. *Cell systems*, 8(4):292–301, 2019.

[3] T. Amimeur, J.M. Shaver, R.R. Ketchem, J.A. Taylor, R.H. Clark, J. Smith, D. Van Citters, C.C. Siska, P. Smidt, M. Sprague, and B.A. Kerwin. Designing feature-controlled humanoid antibody discovery libraries using generative adversarial networks. *bioRxiv*, 2020.

[4] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv*, 2017.

[5] P.R. Araujo, K. Yoon, D. Ko, A.D. Smith, M. Qiao, U. Suresh, S.C. Burns, and L.O. Penalva. Before it gets started: regulating translation at the 5' utr. *Comparative and functional genomics*, 2012.

[6] A. Arefeen, X. Xiao, and T. Jiang. Deeppasta: deep neural network based polyadenylation site analysis. *Bioinformatics*, 35(22):4577–4585, 2019.

[7] C.D. Arnold, D. Gerlach, C. Stelzer, Ł.M. Boryń, M. Rath, and A. Stark. Genome-wide quantitative enhancer activity maps identified by starr-seq. *Science*, 339(6123):1074–1077, 2013.

[8] Ž. Avsec, M. Weilert, A. Shrikumar, S. Krueger, A. Alexandari, K. Dalal, R. Fropf, C. McAnany, J. Gagneur, A. Kundaje, and J. Zeitlinger. Base-resolution models of transcription-factor binding reveal soft motif syntax. *Nature Genetics pp.1-13*, pages 1–13, 2021.

[9] F.A. Bava, C. Eliscovich, P.G. Ferreira, B. Miñana, C. Ben-Dov, R. Guigó, J. Valcárcel, and R. Méndez. Cpeb1 coordinates alternative 3'-utr formation with translational regulation. *Nature*, 495(7439):121–125, 2013.

[10] E. Beaudoing, S. Freier, J. R. Wyatt, J. M. Claverie, and D. Gautheret. Patterns of variant polyadenylation signal usage in human genes. *Genome research*, 10(7):1001–1010, 2000.

[11] D. Belanger, S. Vora, Z. Mariet, R. Deshpande, D. Dohan, C. Angermueller, K. Murphy, O. Chapelle, and L. Colwell. Biological sequences design using batched bayesian optimization. 2019.

[12] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv*, 2013.

[13] C.L. Bennett, M.E. Brunkow, F. Ramsdell, K.C. O'Briant, Q. Zhu, R.L. Fuleihan, A.O. Shigeoka, H.D. Ochs, and P.F. Chance. A rare polyadenylation signal mutation of the foxp3 gene (aauaaa -¿ aaugaa) leads to the ipex syndrome. *Immunogenetics*, 53(6):435–439, 2001.

[14] S. Biswas, G. Kuznetsov, P.J. Ogden, N.J. Conway, R.P. Adams, and G.M. Church. Toward machine-guided design of proteins. *bioRxiv*, 2018.

[15] D.L. Black. Mechanisms of alternative pre-messenger rna splicing. *Annual review of biochemistry*, 72(1):291–336, 2003.

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

[17] N. Bogard, J. Linder, A.B. Rosenberg, and G. Seelig. A deep neural network for predicting and engineering alternative polyadenylation. *Cell*, 178(1):91–106, 2019.

[18] D. Brookes, H. Park, and J. Listgarten. Conditioning by adaptive sampling for robust design. *In International Conference on Machine Learning*, pages 773–782, 2019 May.

[19] K. M. Brown and G. M. Gilmartin. A mechanism for the regulation of pre-mrna 3' processing by human cleavage factor i m. *Molecular cell*, 12(6):1467–1476, 2003.

[20] S.E. Calvo, D.J. Pagliarini, and V.K. Mootha. Upstream open reading frames cause widespread reduction of protein expression and are polymorphic among humans. *Proceedings of the National Academy of Sciences*, 106(18):7507–7512, 2009.

[21] B. Carter, M. Bileschi, J. Smith, T. Sanderson, D. Bryant, D. Belanger, and L.J. Colwell. Critiquing protein family classification models using sufficient input subsets. *Journal of Computational Biology*, 27:1219–1231, 2020.

[22] B. Carter, J. Mueller, S. Jain, and D. Gifford. What made you do this? understanding black-box decisions with sufficient input subsets. *In The 22nd International Conference on Artificial Intelligence and Statistics*, page 567, 2019 April.

[23] J.M.P. Cañadillas and G. Varani. Recognition of gu-rich polyadenylation regulatory elements by human cstf-64 protein. *The EMBO journal*, 22(11):2821–2830, 2003.

[24] C.H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud. Explaining image classifiers by counterfactual generation. *arXiv*, 2018.

[25] F. Chen, C.C. MacDonald, and J. Wilusf. Cleavage site determinants min the mammalian polydenylation signal. *Nucleic acids research*, 23(14):2614–2620, 1995.

[26] J. Chen, L. Song, M.J. Wainwright, and M.I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. *arXiv*, 2018.

[27] Z. Chen, S.E. Boyken, M. Jia, F. Busch, D. Flores-Solis, M.J. Bick, P. Lu, Z.L. VanAernum, A. Sahasrabuddhe, R.A. Langan, and S. Bermeo. Programmable design of orthogonal protein heterodimers. *Nature*, 565(7737):106–111, 2019.

[28] J. Cheng, T.Y.D. Nguyen, K.J. Cygan, M.H. Çelik, W.G. Fairbrother, and J. Gagneur. Mmsplice: modular modeling improves the predictions of genetic variant effects on splicing. *Genome biology*, 20(1):1–15, 2019.

[29] Y. Cheng, R.M. Miura, and B. Tian. Prediction of mrna polyadenylation sites by support vector machine. *Bioinformatics*, 22(19):2320–2325, 2006.

[30] F. Chollet. Keras: The python deep learning library. *Astrophysics Source Code Library*, page 1806, 2018.

[31] J. Chung, S. Ahn, and Y. Bengio. Hierarchical multiscale recurrent neural networks. *arXiv*, 2016.

[32] D. F. Colgan and J. L. Manley. Mechanism and regulation of mrna polyadenylation. *Genes & development*, 11(21):2755–2766, 1997.

[33] J. M. Coller, N. K. Gray, and M. P. Wickens. mrna stabilization by poly (a) binding protein is independent of poly (a) and requires translation. *Genes & development*, 12(20):3226–3235, 1998.

[34] Z. Costello and H.G. Martin. How to hallucinate functional proteins. *arXiv*, 2019.

[35] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv*, 2016.

[36] I. Covert, S. Lundberg, and S.I. Lee. Feature removal is a unifying principle for model explanation methods. *arXiv*, 2020.

[37] J.T. Cuperus, B. Groves, A. Kuchina, A.B. Rosenberg, N. Jojic, S. Fields, and G. Seelig. Deep learning of the regulatory grammar of yeast 5' untranslated regions from 500,000 random sequences. *Genome research*, 27(12):2015–2024, 2017.

[38] P. Dabkowski and Y. Gal. Real time image saliency for black box classifiers. *In Advances in Neural Information Processing Systems*, pages 6967–6976, 2017.

[39] S. Danckwardt, M.W. Hentze, and A.E. Kulozik. 3' end mrna processing: molecular mechanisms and implications for health and disease. *The EMBO journal*, 27(3):482–498, 2008.

[40] P. Das, T. Sercu, K. Wadhawan, I. Padhi, S. Gehrmann, F. Cipcigan, V. Chenthamarakshan, H. Strobelt, C.D. Santos, P.Y. Chen, and Y.Y. Yang. Accelerating antimicrobial discovery with controllable deep generative models and molecular dynamics. *arXiv*, 2020.

[41] R.J. Deaton, R.C. Murphy, M.H. Garzon, D.R. Franceschetti, and S.E. Stevens Jr. Good encodings for dna-based solutions to combinatorial problems. *In DNA Based Computers*, pages 247–258, 1996 June.

[42] A. Derti, P. Garrett-Engele, K.D. MacIsaac, R.C. Stevens, S. Sriram, R. Chen, C.A. Rohl, J.M. Johnson, and T. Babak. A quantitative atlas of polyadenylation in five mammals. *Genome research*, 22(6):1173–1183, 2012.

[43] R. Elkon, A. P. Ugalde, and R. Agami. Alternative cleavage and polyadenylation: extent, regulation and function. *Nature Reviews Genetics*, 14(7):496–506, 2013.

[44] G. Eraslan, Ž. Avsec, J. Gagneur, and F.J. Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.

[45] J. Ernst, A. Melnikov, X. Zhang, L. Wang, P. Rogov, T.S. Mikkelsen, and M. Kellis. Genome-scale high-resolution mapping of activating and repressive nucleotides in regulatory regions. *Nature biotechnology*, 34(11):1180–1190, 2016.

[46] G.M. Findlay, E.A. Boyle, R.J. Hause, J.C. Klein, and J. Shendure. Saturation editing of genomic regions by multiplex homology-directed repair. *Nature*, 513(7516):120–123, 2014.

[47] J.H. Fong, A.E. Keating, and M. Singh. Predicting specificity in bzip coiled-coil protein interactions. *Genome biology*, 5(2):1–10, 2004.

[48] R. Fong, M. Patrick, and A. Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 2950–2958, 2019.

[49] R.C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

[50] P.I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[51] M.A. Garcia-Blanco, A.P. Baraniak, and E.L. Lasda. Alternative splicing in disease and therapy. *Nature biotechnology*, 22(5):535–546, 2004.

[52] N.H. Gehring, U. Frede, G. Neu-Yilik, P. Hundsdoerfer, B. Vetter, M.W. Hentze, and A.E. Kulozik. Increased efficiency of mrna 3' end formation: a new genetic mechanism contributing to hereditary thrombophilia. *Nature genetics*, 28(4):389–392, 2001.

[53] M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. *In International Conference on Machine Learning*, pages 881–889, 2015 June.

[54] D.C. Di Giammartino, K. Nishida, and J.L. Manley. Mechanisms and consequences of alternative polyadenylation. *Molecular cell*, 43(6):853–866, 2011.

[55] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv*, 2014.

[56] J.G. Greener, L. Moffat, and D.T. Jones. Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific reports*, 8(1):1–12, 2018.

[57] J. Guhaniyogi and G. Brewer. Regulation of mrna stability in mammalian cells. *Gene*, 265(1-2):11–23, 2001.

[58] J. Guhaniyogi and G. Brewer. Regulation of mrna stability in mammalian cells. *Gene*, 265(1):11–23, 2001.

[59] A. Gupta and J. Zou. Feedback gan for dna optimizes protein functions. *Nature Machine Intelligence*, 1(2):105–111, 2019.

[60] S. Gupta, J.A. Stamatoyannopoulos, T.L. Bailey, and W.S. Noble. Quantifying similarity between motifs. *Genome biology*, 8(2):1–9, 2007.

[61] R. Gómez-Bombarelli, J.N. Wei, D. Duvenaud, J.M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T.D. Hirzel, R.P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[62] G.F. Hao, W.F. Xu, S.G. Yang, and G.F. Yang. Multiple simulated annealing-molecular dynamics (msa-md) for conformational space search of peptide and miniprotein. *Scientific reports*, 5(1):1–10, 2015.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[64] D.R. Higgs, S.E.Y. Goodbourn, J. Lamb, J.B. Clegg, D.J. Weatherall, and N.J. Proudfoot. alpha-thalassaemia caused by a polyadenylation signal mutation. *Nature*, 306(5941):398–400, 1983.

[65] V. Hilgers, S. B. Lemke, and M. Levine. Elav mediates 3' utr extension in the drosophila nervous system. *Genes & Development*, 26(20):2259–2264, 2012.

[66] S. Höllerer, L. Papaxanthos, A.C. Gumpinger, K. Fischer, C. Beisel, K. Borgwardt, Y. Benenson, and M. Jeschek. Large-scale dna-based phenotypic recording and deep learning enable highly accurate sequence-function mapping. *Nature communications*, 11(1):1–15, 2020.

[67] Z. Ibrahim, N.K. Khalid, K.S. Lim, S. Buyamin, and J.A.A. Mukred. A binary vector evaluated particle swarm optimization based method for dna sequence design problem. *In 2011 IEEE Student Conference on Research and Development*, pages 160–164, 2011 December.

[68] K. Jaganathan, S.K. Panagiotopoulou, J.F. McRae, S.F. Darbandi, D. Knowles, Y.I. Li, J.A. Kosmicki, J. Arbelaez, W. Cui, G.B. Schwartz, and E.D. Chow. Predicting splicing from primary sequence with deep learning. *Cell*, 176(3):535–548, 2019.

[69] C.H. Jan, R.C. Friedman, J.G. Ruby, and D.P. Bartel. Formation, regulation and evolution of caenorhabditis elegans 3' utrs. *Nature*, 469(7328):97–101, 2011.

[70] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv*, 2016.

[71] M. Jenal, R. Elkon, F. Loayza-Puch, G. van Haaften, U. Kühn, F.M. Menzies, J.A.O. Vrielink, A.J. Bos, J. Drost, K. Rooijers, and D.C. Rubinsztein. The poly (a)-binding protein nuclear 1 suppresses alternative cleavage and polyadenylation sites. *Cell*, 149(3):538–553, 2012.

[72] Z. Ji, J.Y. Lee, Z. Pan, B. Jiang, and B. Tian. Progressive lengthening of 3' untranslated regions of mrnas by alternative polyadenylation during mouse embryonic development. *Proceedings of the National Academy of Sciences*, 106(17):7028–7033, 2009.

[73] S.S. Kalia, K. Adelman, S.J. Bale, W.K. Chung, C. Eng, J.P. Evans, G.E. Herman, S.B. Hufnagel, T.E. Klein, B.R. Korf, and K.D. McKelvey. Recommendations for reporting of secondary findings in clinical exome and genome sequencing, 2016 update (acmg sf v20): a policy statement of the american college of medical genetics and genomics. *Genetics in medicine*, 19(2):249–255, 2017.

[74] P. Kalita, A.K. Padhi, K.Y. Zhang, and T. Tripathi. Design of a peptide-based subunit vaccine against novel coronavirus sars-cov-2. *Microbial Pathogenesis*, 145, 2020.

[75] E. Karosiene, M. Rasmussen, T. Blicher, O. Lund, S. Buus, and M. Nielsen. Netmhciipan-3. 0, a common pan-specific mhc class ii prediction method including all three human mhc class ii isotypes, hla-dr, hla-dp and hla-dq. *Immunogenetics*, 65(10):711–724, 2013.

[76] I. Kaufmann, G. Martin, A. Friedlein, H. Langen, and W. Keller. Human fip1 is a subunit of cpsf that binds to u-rich rna elements and stimulates poly (a) polymerase. *The EMBO journal*, 23(3):616–626, 2004.

[77] D.R. Kelley, Y.A. Reshef, M. Bileschi, D. Belanger, C.Y. McLean, and J. Snoek. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome research*, 28:739–750, 2018.

[78] D.R. Kelley, J. Snoek, and J.L. Rinn. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research*, 26:990–999, 2016.

[79] P. Kheradpour and M. Kellis. Systematic discovery and characterization of regulatory motifs in encode tf binding experiments. *Nucleic acids research*, 42(5):2976–2987, 2014.

[80] N. Killoran, L.J. Lee, A. Delong, D. Duvenaud, and B.J. Frey. Generating and designing dna with deep generative models. *arXiv*, 2017.

[81] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.

[82] D.P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv*, 2013.

[83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[84] A. Korte and A. Farlow. The advantages and limitations of trait analysis with gwas: a review. *Plant methods*, 9(1):1–9, 2013.

[85] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv*, 2016.

[86] J. Lanchantin, R. Singh, Z. Lin, and Y. Qi. Deep motif: Visualizing genomic sequence classifications. *arXiv*, 2016.

[87] M.J. Landrum, J.M. Lee, M. Benson, G. Brown, C. Chao, S. Chitipiralla, B. Gu, J. Hart, D. Hoffman, J. Hoover, and W. Jang. Clinvar: public archive of interpretations of clinically relevant variants. *Nucleic acids research*, 44(D1):D862–D868, 2016.

[88] M.J. Landrum, J.M. Lee, M. Benson, G.R. Brown, C. Chao, S. Chitipiralla, B. Gu, J. Hart, D. Hoffman, W. Jang, and K. Karapetyan. Clinvar: improving access to variant interpretations and supporting evidence. *Nucleic acids research*, 46(D1):D1062–D1067, 2018.

[89] M.J. Landrum, J.M. Lee, G.R. Riley, W. Jang, W.S. Rubinstein, D.M. Church, and D.R. Maglott. Clinvar: public archive of relationships among sequence variation and human phenotype. *Nucleic acids research*, 42(D1):D980–D985, 2014.

[90] R.A. Langan, S.E. Boyken, A.H. Ng, J.A. Samson, G. Dods, A.M. Westbrook, T.H. Nguyen, M.J. Lajoie, Z. Chen, S. Berger, and V.K. Mulligan. De novo design of bioactive protein switches. *Nature*, 572(7768):205–210, 2019.

[91] M.K. Leung, A. Delong, and B.J. Frey. Inference of the human polyadenylation code. *Bioinformatics*, 34(17):2889–2898, 2018.

[92] S. Levy, G. Sutton, P.C. Ng, L. Feuk, A.L. Halpern, B.P. Walenz, N. Axelrod, J. Huang, E.F. Kirkness, G. Denisov, and Y. Lin. The diploid genome sequence of an individual human. *PLoS Biol*, 5(10):e254, 2007.

[93] C. Li and R.J. Samulski. Engineering adeno-associated virus vectors for gene therapy. *Nature Reviews Genetics*, 21(4):255–272, 2020.

[94] X.Q. Li and D. Du. Rna polyadenylation sites on the genomes of microorganisms, animals, and plants. *PLoS One*, 8(11):e79511, 2013.

[95] Z. Li, Y. Li, B. Zhang, Y. Li, Y. Long, J. Zhou, X. Zou, M. Zhang, Y. Hu, W. Chen, and X. Gao. Deerect-apa: Prediction of alternative polyadenylation site usage through deep learning. *Genomics, proteomics & bioinformatics*, 2021.

[96] J. Linder, N. Bogard, A.B. Rosenberg, and G. Seelig. A generative neural network for maximizing fitness and diversity of synthetic dna and protein sequences. *Cell Systems*, 11(1):49–62, 2020.

[97] J. Linder, A. La Fleur, Z. Chen, A. Ljubetič, D. Baker, S. Kannan, and G. Seelig. Interpreting neural networks for biological sequences by learning stochastic masks. *bioRxiv*.

[98] J. Linder, A. LaFleur, S. Kannan, Z. Chen, Ajasja Ljubetič, David Baker, and G. Seelig. Efficient inference of nonlinear feature attributions with scrambling neural networks. *In Proceedings of the 2nd Conference on Machine Learning in Computational Biology*, 2020 November.

[99] J. Linder and G. Seelig. Deep exploration networks for rapid engineering of functional dna sequences. *In Proceedings of the 1st Conference on Machine Learning in Computational Biology*, 2019 December.

[100] J. Linder and G. Seelig. Fast differentiable dna and protein sequence optimization for molecular design. *arXiv*, 2020.

[101] G. Liu, B. Carter, T. Bricken, S. Jain, M. Viard, M. Carrington, and D.K. Gifford. Robust computational design and evaluation of peptide vaccines for cellular immunity with application to sars-cov-2. *bioRxiv*, 2020.

[102] J. Ludwiczak, A. Winski, K. Szczepaniak, V. Alva, and S. Dunin-Horkawicz. Deepcoil—a fast and accurate prediction of coiled-coil domains in protein sequences. *Bioinformatics*, 35(16):2790–2795, 2019.

[103] S.M. Lundberg and S.I. Lee. A unified approach to interpreting model predictions. *In Advances in neural information processing systems*, pages 4765–4774, 2017.

[104] C. Lundegaard, K. Lamberth, M. Harndahl, S. Buus, O. Lund, and M. Nielsen. Netmhc-3.0: accurate web accessible predictions of human, mouse and monkey mhc class i affinities for peptides of length 8–11. *Nucleic acids research*, 36(suppl_2):W509–W512, 2008.

[105] J.B. Maguire, S.E. Boyken, D. Baker, and B. Kuhlman. Rapid sampling of hydrogen bond networks for computational protein design. *Journal of chemical theory and computation*, 14(5):2751–2760, 2018.

[106] G. Martin, A.R. Gruber, W. Keller, and M. Zavolan. Genome-wide analysis of pre-mrna 3' end processing reveals a decisive role of human cleavage factor i in the regulation of 3' utr length. *Cell reports*, 1(6):753–763, 2012.

[107] C.P. Masamha, Z. Xia, J. Yang, T.R. Albrecht, M. Li, A.B. Shyu, W. Li, and E.J. Wagner. Cfim25 links alternative polyadenylation to glioblastoma tumour suppression. *Nature*, 510(7505):412–416, 2014.

[108] K.A. Matreyek, L.M. Starita, J.J. Stephany, B. Martin, M.A. Chiasson, V.E. Gray, M. Kircher, A. Khechaduri, J.N. Dines, R.J. Hause, and S. Bhatia. Multiplex assessment of protein variant abundance by massively parallel sequencing. *Nature genetics*, 50(6):874–882, 2018.

[109] C. Medina-Trillo, J.D. Aroca-Aguilar, C.D. Méndez-Hernández, L. Morales, M. García-Antón, J. García-Feijoo, and J. Escribano. Rare foxc1 variants in congenital glaucoma: identification of translation regulatory sequences. *European Journal of Human Genetics*, 24(5):672–680, 2016.

[110] A. Melnikov, A. Murugan, X. Zhang, T. Tesileanu, L. Wang, P. Rogov, S. Feizi, A. Gnirke, C.G. Callan, J.B. Kinney, and M. Kellis. Systematic dissection and optimization of inducible enhancers in human cells using a massively parallel reporter assay. *Nature biotechnology*, 30(3):271–277, 2012.

[111] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[112] F. Mignone, C. Gissi, S. Liuni, and G. Pesole. Untranslated regions of mrnas. *Genome biology*, 3(3):1–10, 2002.

[113] A. Mobiny, P. Yuan, S.K. Moulik, N. Garg, C.C. Wu, and H. Van Nguyen. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, 11(1):1–14, 2021.

[114] R. Movva, P. Greenside, G.K. Marinov, S. Nair, A. Shrikumar, and A. Kundaje. Deciphering regulatory dna sequences and noncoding genetic variants using neural network models of massively parallel reporter assays. *PLoS One*, 14(6):e0218073, 2019.

[115] S.M. Mustaza, A.F.Z. Abidin, Z. Ibrahim, M.A. Shamsudin, A.R. Husain, and J.A.A. Mukred. A modified computational model of ant colony system in dna sequence design. *In 2011 IEEE Student Conference on Research and Development*, pages 169–173, 2011 December.

[116] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[117] A. Owen. Ch. 9: Importance sampling. *Monte Carlo theory, methods and examples*, 2013.

[118] R.P. Patwardhan, C. Lee, O. Litvin, D.L. Young, D. Pe'er, and J. Shendure. High-resolution analysis of dna regulatory elements by synthetic saturation mutagenesis. *Nature biotechnology*, 27(12):1173–1175, 2009.

[119] J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. *In Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007 June.

[120] P.A. Pinto, T. Henriques, M.O. Freitas, T. Martins, R.G. Domingues, P.S. Wyrzykowska, P.A. Coelho, A.M. Carmo, C.E. Sunkel, N.J. Proudfoot, and A. Moreira. Rna polymerase ii kinetics in polo polyadenylation signal selection. *The EMBO journal*, 30(12):2431–2444, 2011.

[121] V. Potapov, J.B. Kaplan, and A.E. Keating. Data-driven prediction and design of bzip coiled-coil interactions. *PLoS Comput Biol*, 11(2), 2015.

[122] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv*, 2015.

[123] S. Rafiq, C.S. Hackett, and R.J. Brentjens. Engineering strategies to overcome the current roadblocks in car t cell therapy. *Nature reviews Clinical oncology*, 17(3):147–167, 2020.

[124] D. Ray, H. Kazan, K.B. Cook, M.T. Weirauch, H.S. Najafabadi, X. Li, S. Gueroussov, M. Albu, H. Zheng, A. Yang, and H. Na. A compendium of rna-binding motifs for decoding gene regulation. *Nature*, 499(7457):172–177, 2013.

[125] D. Repecka, V. Jauniskis, L. Karpus, E. Rembeza, I. Rokaitis, J. Zrimec, S. Poviloniene, A. Laurynenas, S. Viknander, W. Abuajwa, and O. Savolainen. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence*, pages 1–10, 2021.

[126] M.T. Ribeiro, S. Singh, and C. Guestrin. 'why should i trust you?' explaining the predictions of any classifier. *In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016 August.

[127] A. Riesselman, J.E. Shin, A. Kollasch, C. McMahon, E. Simon, C. Sander, A. Manglik, A. Kruse, and D. Marks. Accelerating protein design using autoregressive generative models. *bioRxiv*, 2019.

[128] G.J. Rocklin, T.M. Chidyausiku, I. Goreshnik, A. Ford, S. Houliston, A. Lemak, L. Carter, R. Ravichandran, V.K. Mulligan, A. Chevalier, and C.H. Arrowsmith. Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science*, 357(6347):168–175, 2017.

[129] A.B. Rosenberg, R.P. Patwardhan, J. Shendure, and G. Seelig. Learning the sequence determinants of alternative splicing from millions of random sequences. *Cell*, 163(3):698–711, 2015.

[130] P.J. Sample, B. Wang, D.W. Reid, V. Presnyak, I.J. McFadyen, D.R. Morris, and G. Seelig. Human 5' utr design and variant effect prediction from a massively parallel translation assay. *Nature biotechnology*, 37(7):803–809, 2019.

[131] K.S. Sarkisyan, D.A. Bolotin, M.V. Meer, D.R. Usmanova, A.S. Mishin, G.V. Sharonov, D.N. Ivankov, N.G. Bozhanova, M.S. Baranov, O. Soylemez, and N.S. Bogatyreva. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

[132] J. Schreiber, Y.Y. Lu, and W.S. Noble. Ledidi: Designing genome edits that induce functional activity. *bioRxiv*, 2020.

[133] A.W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A.W. Nelson, A. Bridgland, and H. Penedones. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[134] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[135] E. Sharon, Y. Kalma, A. Sharp, T. Raveh-Sadka, M. Levo, D. Zeevi, L. Keren, Z. Yakhini, A. Weinberger, and E. Segal. Inferring gene regulatory logic from high-throughput measurements of thousands of systematically designed promoters. *Nature biotechnology*, 30(6):521–530, 2012.

[136] P.J. Shepard, E.A. Choi, J. Lu, L.A. Flanagan, K.J. Hertel, and Y. Shi. Complex and dynamic landscape of rna polyadenylation revealed by pas-seq. *Rna*, 17(4):761–772, 2011.

[137] Y. Shi. Alternative polyadenylation: new insights from global analyses. *RNA*, 18:2105–2117, 2012.

[138] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. *arXiv*, 2017.

[139] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv*, 2013.

[140] S. Singh, Y. Yang, B. Poczos, and J. Ma. Predicting enhancer-promoter interaction from genomic sequence with deep neural networks. *Quantitative Biology*, 7:122–137, 2019.

[141] I.V. Slutskin, A. Weinberger, and E. Segal. Sequence determinants of polyadenylation-mediated regulation. *Genome research*, 29(10):1635–1647, 2019.

[142] R.P. Smith, L. Taher, R.P. Patwardhan, M.J. Kim, F. Inoue, J. Shendure, I. Ovcharenko, and N. Ahituv. Massively parallel decoding of mammalian regulatory sequences supports a flexible organizational model. *Nature genetics*, 45(9):1021, 2013.

[143] J. Snoek, H. Larochelle, and R.P. Adams. Practical bayesian optimization of machine learning algorithms. *arXiv*, 2012.

[144] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams. Scalable bayesian optimization using deep neural networks. *In International conference on machine learning*, pages 2171–2180, 2015 June.

[145] F. Spitz and E.E. Furlong. Transcription factors: from enhancer binding to developmental control. *Nature reviews genetics*, 13(9):613–626, 2012.

[146] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv*, 2014.

[147] S.N. Stacey, P. Sulem, A. Jonasdottir, G. Masson, J. Gudmundsson, D.F. Gudbjartsson, O.T. Magnusson, S.A. Gudjonsson, B. Sigurgeirsson, K. Thorisdottir, and R. Ragnarsson. A germline variant in the tp53 polyadenylation signal confers cancer susceptibility. *Nature genetics*, 43(11):1098–1103, 2011.

[148] P.D. Stenson, M. Mort, E.V. Ball, K. Evans, M. Hayden, S. Heywood, M. Hussain, A.D. Phillips, and D.N. Cooper. The human gene mutation database: towards a comprehensive repository of inherited mutation data for medical research, genetic diagnosis and next-generation sequencing studies. *Human genetics*, 136(6):665–677, 2017.

[149] Y. Sun, Y. Zhang, K. Hamilton, J.L. Manley, Y. Shi, T. Walz, and L. Tong. Molecular basis for the recognition of the human aauaaa polyadenylation signal. *Proceedings of the National Academy of Sciences*, 115(7):E1419–E1428, 2018.

[150] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *arXiv*, 2017.

[151] A. Talukder, C. Barham, X. Li, and H. Hu. Interpretation of deep learning in genomics and epigenomics. *Briefings in Bioinformatics*, 2020.

[152] V. Tam, N. Patel, M. Turcotte, Y. Bossé, G. Paré, and D. Meyre. Benefits and limitations of genome-wide association studies. *Nature Reviews Genetics*, 20(8):467–484, 2019.

[153] B. Tian, J. Hu, H. Zhang, and C. S. Lutz. A large-scale analysis of mrna polyadenylation of human and mouse genes. *Nucleic Acids Research*, 33:201–212, 2005.

[154] B. Tian and J.L. Manley. Alternative polyadenylation of mrna precursors. *Nature reviews Molecular cell biology*, 18(1):18–30, 2017.

[155] J. Trigg, K. Gutwin, A.E. Keating, and B. Berger. Multicoil2: predicting coiled coils and their oligomerization states from sequence in the twilight zone. *PloS one*, 6(8), 2011.

[156] A.M. Valencia and C. Kadoch. Chromatin regulatory mechanisms and therapeutic opportunities in cancer. *Nature cell biology*, 21(2):152–161, 2019.

[157] D. Wang, P.W. Tai, and G. Gao. Adeno-associated virus vector as a platform for gene therapy delivery. *Nature reviews Drug discovery*, 18(5):358–378, 2019.

[158] Y. Wang, H. Wang, L. Liu, and X. Wang. Synthetic promoter design in escherichia coli based on generative adversarial network. *bioRxiv*, 2019.

[159] N. Whiffin, K.J. Karczewski, X. Zhang, S. Chothani, M.J. Smith, D.G. Evans, A.M. Roberts, N.M. Quaife, S. Schafer, O. Rackham, and J. Alföldi. Characterising the loss-of-function impact of 5'untranslated region variants in 15,708 individuals. *Nature communications*, 11(1):1–12, 2020.

[160] M.A. White, C.A. Myers, J.C. Corbo, and B.A. Cohen. Massively parallel in vivo enhancer assay reveals that highly local features determine the cis-regulatory function of chip-seq peaks. *Proceedings of the National Academy of Sciences*, 110(29):11952–11957, 2013.

[161] A. Wiestner, M. Tehrani, M. Chiorazzi, G. Wright, F. Gibellini, K. Nakayama, H. Liu, A. Rosenwald, H.K. Muller-Hermelink, G. Ott, and W.C. Chan. Point mutations and genomic deletions in ccnd1 create stable truncated cyclin d1 mrnas that are associated with increased proliferation rate and shorter survival. *Blood Hematology*, pages 4599–4606, 2007.

[162] X. Wu and D.P. Bartel. Widespread influence of 3'-end structures on mammalian mrna processing and stability. *Cell*, 169(5):905–917, 2017.

[163] M. Wylenzek, C. Geisen, L. Stapenhorst, K. Wielckens, and K.R. Klingler. A novel point mutation in the 3'region of the prothrombin gene at position 20221 in a lebanese/syrian family. *Thrombosis and haemostasis*, 85(5):943–944, 2001.

[164] J. Xiao, J. Xu, Z. Chen, K. Zhang, and L. Pan. A hybrid quantum chaotic swarm evolutionary algorithm for dna encoding. *Computers & Mathematics with Applications*, 57(11-12):1949–1958, 2009.

[165] H.Y. Xiong, B. Alipanahi, L.J. Lee, H. Bretschneider, D. Merico, R.K. Yuen, Y. Hua, S. Gueroussov, H.S. Najafabadi, T.R. Hughes, and Q. Morris. The human splicing code reveals new insights into the genetic determinants of disease. *Science*, 347(6218), 2015.

[166] H.Y. Xiong, Y. Barash, and B.J. Frey. Bayesian prediction of tissue-regulated splicing using rna sequence and cellular context. *Bioinformatics*, 27(18):2554–2562, 2011.

[167] J. Yang, I. Anishchenko, H. Park, Z. Peng, S. Ovchinnikov, and D. Baker. Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences*, 117(3):1496–1503, 2020.

[168] K.K. Yang, Z. Wu, and F.H. Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8):687–694, 2019.

[169] J. Yoon, J. Jordon, and M. van der Schaar. Invase: Instance-wise variable selection using neural networks. *In International Conference on Learning Representations*, 2018 September.

[170] M.D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *In European conference on computer vision. Springer Cham*, pages 818–833, 2014 September.

[171] W. Zeng, Y. Wang, and R. Jiang. Integrating distal and proximal information to predict gene expression via a densely connected convolutional neural network. *Bioinformatics*, 36:496–503, 2020.

[172] W. Zeng, M. Wu, and R. Jiang. Prediction of enhancer-promoter interactions via natural language processing. *BMC genomics*, 19:13–22, 2018.

[173] J. Zhou and O.G. Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931–934, 2015.

[174] Y. Zhu, X. Wang, E. Forouzmand, J. Jeong, F. Qiao, G.A. Sowd, A.N. Engelman, X. Xie, K.J. Hertel, and Y. Shi. Molecular mechanisms for cfim-mediated regulation of mrna alternative polyadenylation. *Molecular cell*, 69(1):62–74, 2018.

[175] J. Zou, M. Huss, A. Abid, P. Mohammadi, A. Torkamani, and A. Telenti. A primer on deep learning in genomics. *Nature genetics*, 51(1):12–18, 2019.

Appendix A

CODE REPOSITORIES AND WEB TOOL

The code, pre-trained models and processed data relevant to Chapter 3 (APARENT), 4 (SeqProp), 5 (Deep Exploration Networks) and 6 (Scramblers) are publicly available online. See below for links to the corresponding repositories:

- Github repository for APARENT. The processed data is available through this website.

  `https://github.com/johli/aparent`

- Github repository for SeqProp.

  `https://github.com/johli/seqprop`

- Github repository for Deep Exploration Networks.

  `https://github.com/johli/genesis`

- Github repository for Scrambler Networks.

  `https://github.com/johli/scrambler`

In addition to making the code, data and model available, we also developed an interactive web tool for predicting APA variants based on the model APARENT. This web tool is available at `https://apa.cs.washington.edu/`.

VITA

Johannes Linder received his Bachelor's degree in Computer Science from the Royal Institute of Technology (KTH) in Stockholm, Sweden in 2013, and received his Master's degree in Computer Science from KTH in 2015. Johannes attended the PhD program in Computer Science at the Paul G. Allen School of Computer Science, University of Washington, Seattle, US, in 2017, where he was advised by Prof. Georg Seelig. Johannes research interests are at the intersection of genomics, proteomics and machine learning, where he develops computational methods and models for increasing our understanding of gene regulation and protein function.