Master's Thesis

Using a Deep Learning Foundation Model for CRISPR Array Detection

Lukas Daugaard Schröder

Examiner: Prof. Dr. Rolf Backofen

Advisers: Alexander Mitrofanov, Ryan Köksal

University of Freiburg

Faculty of Engineering

Department of Computer Science

Bioinformatics Group Freiburg

September 22$^{\text{th}}$, 2025

Writing Period

20. 03. 2025 – 22. 09. 2025

Examiner

Prof. Dr. Rolf Backofen

Second Examiner

Dr. André Biedenkapp

Advisers

Alexander Mitrofanov, Ryan Köksal

Declaration

I hereby declare that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.

I hereby also declare that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Freiburg im Breisgau, 22.09.2025 — Lukas Schröder

Place, Date — Signature

Abstract

Existing computational methods for Clustered Regularly Interspaced Short Palindromic Repeats (CRISPR) array detection face significant challenges when applied to short-read sequencing data and degenerate repeat sequences. This thesis explores the application of deep learning foundation models to address these limitations. By adapting the Evo foundation model, a novel method for CRISPR array detection was developed to overcome these challenges.

Two model variants were implemented: an 8,192-nucleotide model for long sequences and a 150-nucleotide model tailored for Illumina short reads. Both classify nucleotides as part of a repeat, spacer, or non-CRISPR array regions. The 8,192-nucleotide and 150-nucleotide models achieved test accuracies of 98.16% and 90.03%, respectively. The 150-nucleotide model demonstrated the ability to directly analyze short reads, achieving a spacer precision of 24.31% and recall of 49.12% on simulated metagenomic data. Notably, 12.57% of spacers detected by this model were not found by Metagenomic CRISPR Array Analysis Tool (MCAAT) in the same reads, demonstrating its potential to complement existing tools. The 8,192-nucleotide model showed the capacity to identify 71 degenerate repeat candidates, of which 92.5% aligned significantly to consensus repeats, highlighting its ability to detect heterogeneous repeats.

This work establishes foundation models as effective tools for CRISPR array detection, enabling analysis of short sequencing reads and identification of degenerate repeats. The approach represents a significant advancement in computational genomics, with implications beyond CRISPR research.

Contents

List of Figures

List of Tables

List of Algorithms

1. Introduction

1.1. Background and Motivation

The discovery and characterization of Clustered Regularly Interspaced Short Palindromic Repeats and CRISPR-associated (CRISPR-Cas) systems have fundamentally transformed the study of prokaryotic immunity and enabled a revolution in genome engineering. CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) and their associated Cas proteins allow bacteria and archaea to acquire adaptive defense mechanisms against viruses and plasmids, acting as a molecular memory of infectious DNA [1, 2]. The accuracy and comprehensiveness of CRISPR array annotation are now critical not only for understanding microbial evolution, but also for the continued development of CRISPR-based biotechnological tools [3].

In recent years, the amount and diversity of genomic sequencing data have rapidly expanded, particularly through metagenomic studies using short-read technologies such as Illumina [4]. This growth has facilitated breakthroughs in microbial ecology and evolutionary biology, but it also poses computational challenges: new methods must be able to identify and characterize CRISPR arrays under diverse conditions and from incomplete or fragmented sequence data [5].

1.2. Problem Statement and Objectives

While existing computational approaches perform well on complete genomes with canonical CRISPR structures, they encounter critical limitations when applied to short-read sequencing data or arrays with degenerate repeats. Two primary challenges are:

1. **Short-Read Constraints:** Current methods require multiple repeats to recognize and utilize the repeat pattern, rendering them ineffective for Illumina short reads (75–300 bp), which are common in metagenomic studies [4, 6, 7, 8].

2. **Degenerate Repeat Sensitivity:** Heterogeneous CRISPR arrays—where repeats exhibit sequence variation—are frequently misclassified, or their terminal regions are excluded from classification by rule-based and machine learning approaches [7, 9].

These limitations hinder the ability to characterize CRISPR-Cas systems in their natural contexts, limiting insights into microbial immunity and evolution in complex environmental samples.

To address these specific challenges, this thesis develops a deep learning foundation model-based approach for CRISPR array detection with the following targeted objectives:

1. To leverage transfer learning from a large-scale foundation model (Evo) pretrained on prokaryotic genomes, viruses, and plasmids, thereby capturing evolutionary and contextual genomic knowledge [10].

2. To fine-tune models for per-nucleotide classification, tailored to CRISPR array detection, enabling robust predictions even in the presence of short or degenerated sequence reads that challenge existing methods.

1.3. Approach and Contributions

This work adapts the Evo foundation model—based on the StripedHyena hybrid attention-convolution architecture and capable of processing long nucleotide sequences—for the specialized task of CRISPR array detection. Using parameter-efficient techniques such as Low-Rank Adaptation (LoRA), the approach enables fine-tuning on curated and labeled datasets of bona fide CRISPR arrays [11]. The methodology includes systematic evaluation across varying input lengths and comprehensive error analyses highlighting both strengths and limitations of the approach.

The key contributions of this thesis are:

- **Foundation Model Adaptation:** Demonstrating that genomic foundation models can be successfully adapted to identify CRISPR arrays and classify their internal components with high accuracy.

- **Multi-Scale Sequence Analysis:** Developing domain-adapted foundation models capable of detecting CRISPR arrays in both long assembled sequences and short sequencing reads.

- **Degenerate Repeat Detection:** Showcasing the model's ability to identify degraded or variant repeats that are missed by conventional similarity-based approaches.

2. Related Work

2.1. Existing Methods for CRISPR Array Detection

The identification of CRISPR arrays in prokaryotic genomes has been a focus of bioinformatics research for over a decade. Early approaches predominantly relied on the detection of repetitive DNA patterns, specifically regularly interspaced short palindromic repeats. Tools such as CRISPR Recognition Tool (CRT) and PILER-CR exemplify this strategy, using k-mer analysis and local alignment, respectively, to identify candidate arrays based on repeat-spacer organization. These methods evaluate patterns using built-in scoring functions that consider features such as repeat length, similarity, and array structure [12, 13].

More advanced tools, such as CRISPRCasFinder, extend this paradigm by integrating repeat detection with Cas gene identification. They utilize Vmatch for repeat finding and combine this with protein profile searches to validate the presence of signature Cas genes, thereby improving specificity and enabling system classification [6]. Despite these improvements, rule-based approaches often struggle to distinguish functional CRISPR arrays from other repetitive elements, especially in fragmented or low-quality reads [7].

The problem of false positives and limited sensitivity in challenging genomic contexts has motivated the development of machine learning-based methods. CRISPRidentify represents a significant step forward by replacing manual scoring with a data-driven classifier trained on curated sets of positive and negative CRISPR array examples.

This approach extracts a comprehensive set of features from candidate arrays and uses them to train a model capable of distinguishing true arrays from spurious repeats. Benchmarking has demonstrated that CRISPRidentify achieves higher sensitivity and a drastically reduced false positive rate compared to previous tools, while also providing detailed annotations such as array orientation and leader sequence prediction [7].

More recently, Talibli et al. introduced the Metagenomic CRISPR Array Analysis Tool (MCAAT), addressing specific challenges of CRISPR array detection in metagenomic data. MCAAT employs a novel graph-based approach that exploits the structural properties of CRISPR arrays, which form multicycles in de Bruijn graphs, and is used in an strategy that can directly analyze sequencing reads. This method relies on repeating patterns of the repeat, which can lead to worse performance when dealing with degenerated repeats. Additionally, the method's reliance on k-mers can lead to issues with sequencing errors, which can create spurious cycles or false spacers [14].

2.2. Deep Learning in Genomics

Traditional machine learning approaches for CRISPR array detection, such as CRISPRidentify, rely on manual feature engineering, which can limit their ability to capture complex sequence dependencies [7, 15]. Deep learning models have emerged as a powerful alternative, capable of learning hierarchical representations directly from raw nucleotide sequences. Early applications in genomics, such as DeepSEA and DanQ, utilize convolutional and recurrent neural networks to predict regulatory elements and variant effects, demonstrating the capacity of deep models to extract meaningful biological features without explicit feature design [16, 17]. However, their "black-box" nature complicates biological interpretation, representing a trade-off for improved accuracy [18].

Recent advances have seen the adoption of deep learning architectures specifically

tailored to CRISPR-related tasks. For example, lightweight frameworks such as CRISPR-MFH have demonstrated that even compact deep learning models can achieve improved off-target prediction for CRISPR-Cas9 systems by leveraging multi-feature encoding strategies. While these models excel at tasks like off-target prediction, their application to CRISPR array detection remains limited by input constraints and the need for large labeled datasets [19].

2.3. Foundation Models in Bioinformatics

The introduction of foundation models marked a paradigm shift in computational genomics. These large-scale models are pretrained on massive and diverse genomic datasets, enabling them to create general-purpose sequence representations for a variety of downstream tasks [20, 10, 21]. The Nucleotide Transformer (NT), for example, is a family of models ranging from 50 million to 2.5 billion parameters, trained on collections spanning thousands of human and microbial genomes [10, 21]. NT and related DNA language models have demonstrated robust performance in tasks such as promoter and enhancer prediction, species differentiation, and the analysis of non-coding regions, with learned representations that transfer effectively across species and tasks [21, 10]. However, their computational cost limits accessibility for many labs [10].

In the context of CRISPR research, foundation models pretrained on prokaryotic genomes are particularly promising. By learning from the full diversity of bacterial and archaeal sequences, these models can capture complex sequence patterns that define CRISPR arrays and their evolutionary context. Furthermore, foundation models that also include prokaryotic viruses and plasmid sequences in their pretraining data may aid in the detection of spacers in the prokaryotic genome [22, 23, 24].

The Evo model, based on the StripedHyena architecture, is a promising foundational model candidate for CRISPR array detection. Evo is pretrained 300 billions

nucleotides from prokaryotic genomes, prokaryotic viruses and plasmid sequences. Furthermore, it can process sequences exceeding 100,000 bases, making it well-suited for analyzing the long repetitive structure characteristic of CRISPR loci [22]. The ability to fine-tune such models for specific tasks, even with limited labeled data, offers a compelling solution to the challenges faced by traditional and earlier deep learning approaches. In this thesis, we build upon Evo's architecture and pretrained knowledge to develop a CRISPR-specific model, addressing the unique requirements of array detection in genomic reads.

Summary

The field has progressed from alignment-driven and rule-based methods [12, 13, 6, 7] to machine learning for CRISPR array detection [7, 15], and more recently, to graph-based and deep learning methods for CRISPR array research [16, 17, 19, 25, 26, 27, 28]. While traditional CRISPR array detection approaches are effective for well-assembled genomes with canonical array structures, they are limited by reduced sensitivity in complex or fragmented data [7]. Graph-based approaches allow for assembly-free strategies, but face issues with degenerated repeats and sequencing errors [14]. Deep learning models can achieve improved prediction accuracy by learning from raw sequences but are often restricted by data requirements [25, 26].

Foundation models address this limitation by leveraging large-scale pretraining on diverse genomic data [20, 10, 21], enabling robust and generalizable sequence representations. Their capacity to process long sequences and to transfer knowledge across tasks and domains of life positions them as a transformative technology for CRISPR array detection and beyond [22, 23, 24]. While foundation models have shown promise for various genomic tasks, their application to CRISPR array detection remains largely unexplored, representing a significant opportunity to leverage evolutionary-scale pretraining for this specialized detection task [20, 10, 21, 22]. Building on these advances,

our work introduces a domain-adapted model optimized for CRISPR array detection, combining Evo's long-context processing with a classification head and CRISPR array detection data.

3. Background

3.1. CRISPR-Cas Systems

CRISPR and Cas proteins constitute an adaptive immune system in bacteria and archaea, providing defense against invading genetic elements such as bacteriophages and plasmids. The overall mechanism of CRISPR-Cas is shown in Figure 1, which visually depicts the three principal stages:

- **Adaptation:** Short fragments of foreign DNA, called spacers, are integrated into the CRISPR array within the host genome.

- **Expression:** The CRISPR array is transcribed into precursor CRISPR RNA (pre-crRNA), which is processed into mature crRNAs.

- **Interference:** These crRNAs guide Cas protein complexes to recognize and cleave complementary sequences in invading genetic material, neutralizing the threat.

This coordinated process enables prokaryotes to efficiently recognize and eliminate foreign genetic elements [1].

CRISPR-Cas systems are broadly classified into two classes. Class 1 systems (Types I, III, IV) utilize multi-protein effector complexes, while Class 2 systems (notably Cas9 and Cas12) employ single, large effector proteins [30, 31]. The simplicity and

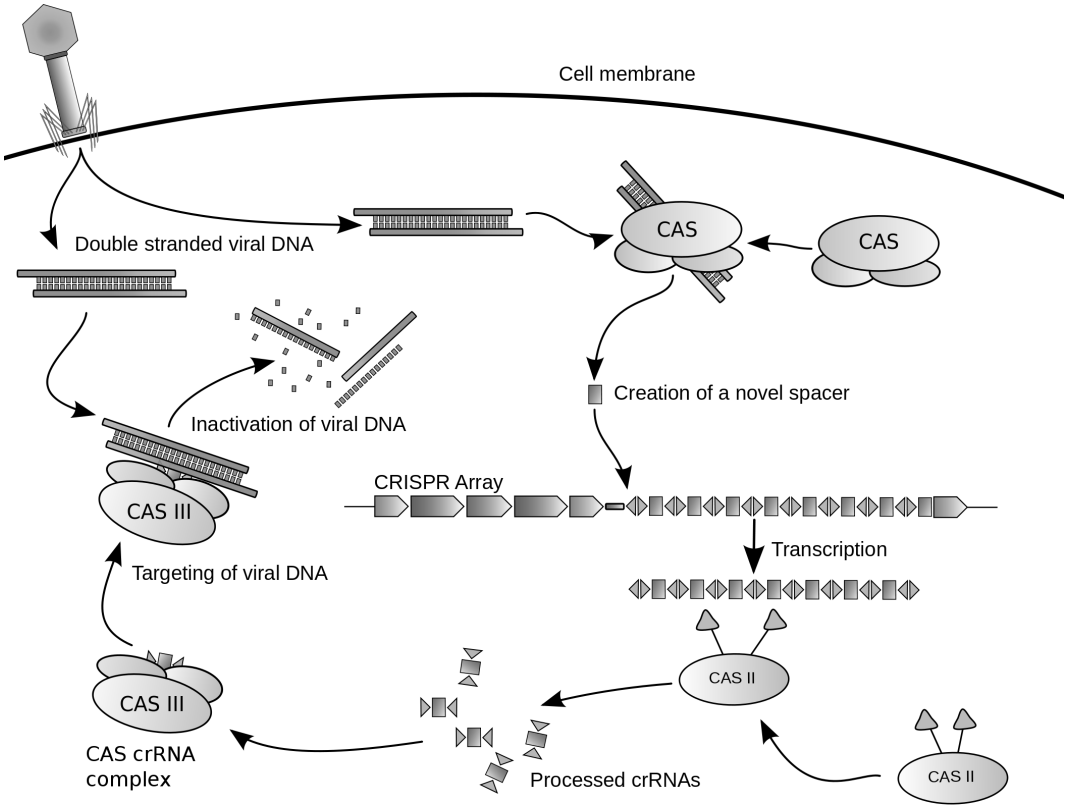

Figure 1.: Overview of the CRISPR-Cas Mechanism. The process includes phage infection, integration of viral DNA into the CRISPR array, transcription and processing of crRNAs, and targeting of matching viral DNA for inactivation by Cas protein complexes. Rectangles denote DNA, ovals represent Cas proteins (labeled by type), and arrows indicate process flow. In the CRISPR array, pairs of triangles mark repetitive DNA sequences (repeats), rectangles between them are spacers, and rectangles with a triangle at the end indicate flanking sequences.
Image by James Atmos, licensed under CC BY-SA 3.0 via Wikimedia Commons [29].

programmability of Class 2 systems, particularly CRISPR-Cas9, have revolutionized genome editing across a wide range of organisms [2].

3.2. Structure and Function of CRISPR Arrays

A CRISPR array is a specialized genomic locus that serves as a molecular memory of past infections. Its structure, shown in Figure 2, consists of direct repeats—short, palindromic, highly conserved sequences—interspersed with unique spacers derived from foreign genetic elements. Typically, a CRISPR array begins and ends with a repeat, with each spacer flanked by two repeats [32, 33]. While repeats are often homogeneous, some arrays exhibit sequence variations among repeats [7].

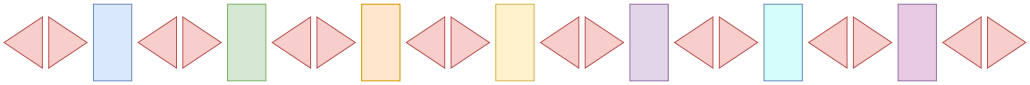

Figure 2.: Illustration of a CRISPR Array Structure. Each pair of adjacent triangles represents a repeat, while rectangles denote spacers. Repeats are shown in uniform colors to indicate how they often have sequence homogeneity, whereas spacers are depicted in different colors to reflect their diversity.

Upon transcription, the array yields a long precursor RNA, which is processed into individual crRNAs. Each crRNA contains a unique spacer, enabling surveillance and targeting of matching foreign DNA sequences [32, 33].

3.3. Computational Identification of CRISPR-Cas Systems

The rapid expansion of genomic data has necessitated robust computational approaches for the detection and classification of CRISPR-Cas systems [34]. These approaches generally use one or both of the following strategies:

- **CRISPR Array Detection:** CRISPR arrays are identified by searching for direct repeats (21–48 base pairs) interspersed with spacers (26–72 base pairs) [35]. Key methods include:
  - *K-mer-based approaches* (e.g., CRT [12])

- *Local alignment approaches* (e.g., PILER-CR [13])

- *Machine learning models* (e.g., CRISPRidentify [7])

Challenges include distinguishing functional arrays from tandem repeats and handling sequence degeneration in Type IV systems [7].

- **Cas Gene Identification:** Cas operons are detected using:

  - *Homology search tools* (Basic Local Alignment Search Tool (BLAST), HMMER) against protein databases (TIGRFAM, Pfam)

  - *Operon structure analysis* (e.g., CRISPRCasFinder [6])

  - *Proximity to CRISPR arrays* for functional validation

  Signature genes such as *cas1* and *cas2* (adaptation) and effector proteins (e.g., *cas3*, *cas9*) help classify systems as Class 1 or 2 [36].

Modern pipelines integrate array and Cas gene data for system detection and classification. Rule-based methods (e.g., CRISPRCasFinder) use protein profiles and array features, while machine learning approaches (e.g., CRISPRCasTyper) leverage genomic context features for high-accuracy subtype assignment [6, 37].

Limitations

Despite these advances, several challenges persist:

- **Assembly Quality:** Fragmented genomes may split arrays or Cas operons, complicating detection [38].

- **Mobile Systems:** Functional CRISPR-Cas systems embedded in transposons (e.g., Tn7-like) are often misclassified as nonfunctional [39].

- **Attenuated Systems:** Arrays lacking adaptation genes (*cas1/cas2*) may retain interference activity but evade detection [9].

- **Degenerated Repeats:** As shown in Section 5.9, state-of-the-art tools can overlook degenerated repeats due to their underlying use of repeat matching [7, 6].

3.4. Deep Learning Models

Traditional machine learning approaches for CRISPR array detection, such as CRISPRidentify, depend on accurate manual feature engineering [7]. Deep learning models overcome this limitation by learning hierarchical representations directly from raw nucleotide sequences, capturing complex spatial dependencies that are difficult to encode manually [2, 19].

Architectural Considerations

- **Attention Mechanisms:** Self-attention enables models to learn global dependencies, which is advantageous for genomic data with long-range interactions. However, computational costs scale quadratically with sequence length [40].

- **Convolutions:** Convolutional layers efficiently detect local motifs but have limited receptive fields. Techniques like dilated convolutions and residual connections extend their range but add complexity [41, 42].

Hybrid architectures, such as StripedHyena, combine these strengths and, thereby, limit their weaknesses by integrating gated convolutions and a limited number of self-attention layers. Evo, for instance, utilises 29 blocks with gated convolutions and three attention blocks, thus supporting the analysis of much longer sequences than standard transformer models [22].

3.5. Foundation Models in Genomics

Foundation models, pretrained on vast genomic datasets, have emerged as powerful tools for downstream tasks even when labeled data are limited [10, 20]. For example, the Evo model, built on the StripedHyena architecture, is pretrained to predict the next nucleotides in a sequence and can process sequences exceeding 100,000 bases. Its training on bacterial, archaeal, and viral genomes is particularly relevant for understanding CRISPR arrays, which are composed of both host-derived repeats and foreign-derived spacers [22, 23].

4. Approach

4.1. Overall Approach

Our methodology combines evolutionary-scale pretraining, parameter-efficient fine-tuning, and hierarchical classification to address CRISPR array detection in sequencing reads. To tackle the limitations of short-read constraints and degenerate repeat sensitivity, outlined in Section 1.2, the approach consists of three sequential phases that build upon each other to create a robust detection pipeline:

Foundation Model Selection and Adaptation

The first phase involves selecting a suitable genomic foundation model and adapting it for the CRISPR detection task:

- **Backbone Architecture:** We use Evo, a StripedHyena-based foundation model pretrained on 300 billion nucleotides from prokaryotic genomes, viruses, and plasmids. The Evo model version with an 8,192-nucleotide context window enables analysis of repeat-spacer patterns in reads, while lowering resource requirements compared to the 131k-nucleotide context window version [22].

- **Parameter-Efficient Fine-tuning:** We apply LoRA to Evo's query-key-value projection layers and all MLP layers, freezing 99.27% of the original parameters

while maintaining model capacity. This enables training on 80GB A100 GPUs with the full 8,192 context window.

Multi-Task Classification Architecture

Building on this foundation model adaptation, the sequence classification pipeline is structured as follows:

```
[Seq] → Evo → [Embeddings] → CLS_Head → [Repeat/Spacer/Non-Array]
```

where the sequence input is processed by Evo to generate position-wise embeddings, which are then classified by a custom classification head. The components are:

- **Input Processing:** Raw nucleotide sequences encoded using ASCII (e.g., A=65, C=67, G=71, T=84, etc.).
- **Feature Extraction:** Evo generates position-wise embeddings through its hybrid attention-convolution operators.
- **Classification (CLS) Head:**
  - Dropout layer for regularization.
  - Fully connected layer ($512 \rightarrow 3$ units) with softmax activation.
  - Per-nucleotide classification using cross-entropy loss.

Training and Validation Strategy

To ensure robust model performance, the final phase implements a comprehensive training approach:

- **Data Pipeline:** CRISPRidentify-labeled sequences selected from:

  - Prokaryotic genomes with CRISPR arrays classified as bona fide.
  - Up to 500 flanking nucleotides added to each side of the CRISPR array.
  - Sequences cropped or padded to match the target length.

- **Optimization:** The following hyperparameters were optimized via manual hyperparameter tuning: Learning rate, LoRA rank, classification head dropout probability and AdamW weight decay. The optimization process is detailed in Section 4.8.

Having outlined the overall methodology, the following sections provide detailed descriptions of each component, beginning with the pretraining data that forms the foundation of our approach.

4.2. Pretraining Data

As the foundation for our approach, the Evo foundation model was pretrained on OpenGenome [22], a comprehensive sub-dataset comprising three key types of genetic material relevant to CRISPR-Cas systems:

- **Prokaryotic Genomes:** 80,789 bacterial and 4,416 archaeal genomes with an average length of 3,214,184 bases from the Genome Taxonomy Database (GTDB) v214.1 [43].
- **Prokaryotic Viruses:** 2,653,046 loci with an average length of 13,658 bases from the Integrated Microbial Genomes/Virus (IMG/VR) v4 database [44].
- **Plasmids:** 214,950 sequences with an average length of 27,106 bases from the Integrated Microbial Genomes/Plasmid (IMG/PR) database [45].

For each source, only one representation was chosen for each species (for genomes), viral operational taxonomic unit (vOTU), and plasmid taxonomic unit (PTU). Moreover, viral and plasmid sequences were further filtered to include only those whose assigned taxonomic classification was found in a prokaryotic host at least twice. For viral sequences, additional curation excluded potential eukaryotic viruses and those with poor taxonomic specificity, following the protocol described by Nguyen et al. [22].

The resulting OpenGenome sub-dataset spans 300 billion nucleotides [22]. Figure 3 illustrates the nucleotide distribution by sequence type.

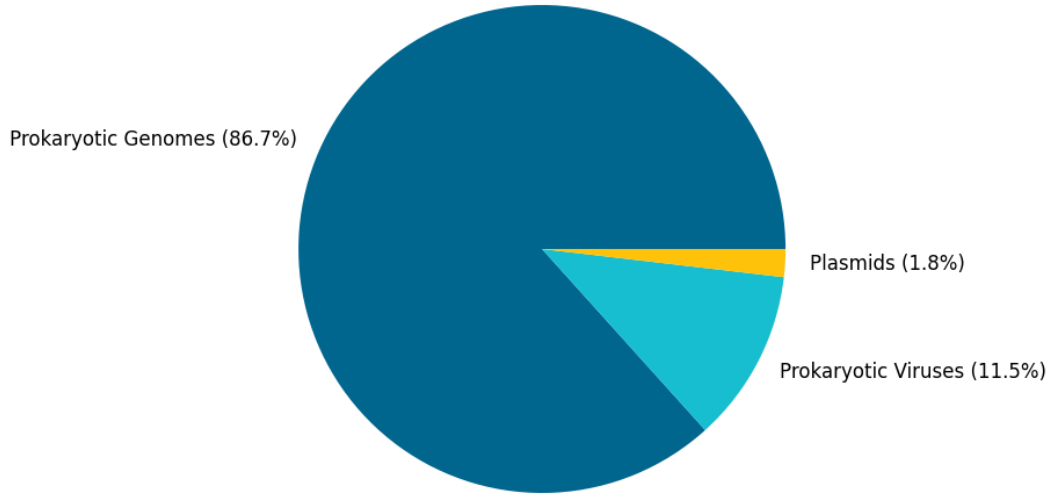

Figure 3.: Nucleotide Distribution by Sequence Type in the Evo Pretraining Dataset. The chart shows the proportion of nucleotides contributed by each sequence type: prokaryotic genomes (273,865 million bases), prokaryotic viruses (36,236 million bases), and plasmids (5,827 million bases). This distribution reflects the relative abundance of each sequence type in the OpenGenome dataset used for pretraining, highlighting the dominance of prokaryotic genome sequences [22].

While the pretraining data provides the foundation for general sequence understanding, task-specific fine-tuning requires CRISPR array annotations. The following section describes this specialized fine-tuning dataset and how it was constructed.

4.3. Fine-tuning Data

To adapt the Evo model for the specific task of CRISPR array detection, we assembled a labeled dataset of high-quality CRISPR arrays. Specifically, we selected 5,084 CRISPR arrays, each annotated by CRISPRidentify with a certainty score of at least 0.75, which CRISPRidentify classifies as bona fide CRISPR arrays [7].

Figure 4 illustrates the distribution of the lengths of the CRISPR arrays and their corresponding sequences. Each nucleotide within these arrays is labeled as either a repeat or spacer. To ensure the model learns to distinguish CRISPR array regions from non-array regions, we included up to 500 nucleotides of flanking sequence on each side of each array, where available. These flanking nucleotides are labeled as non-array. Consequently, the resulting dataset provides a balanced representation of repeat and spacer nucleotides, as well as a substantial amount of non-array nucleotides for model training and evaluation. Figure 5 illustrates the nucleotide label distribution in the fine-tuning dataset, showing this balanced class distribution.

With the training data established, the next component is the input processing pipeline, which must maintain compatibility with the pretrained Evo model while accommodating the specific requirements of CRISPR array detection.

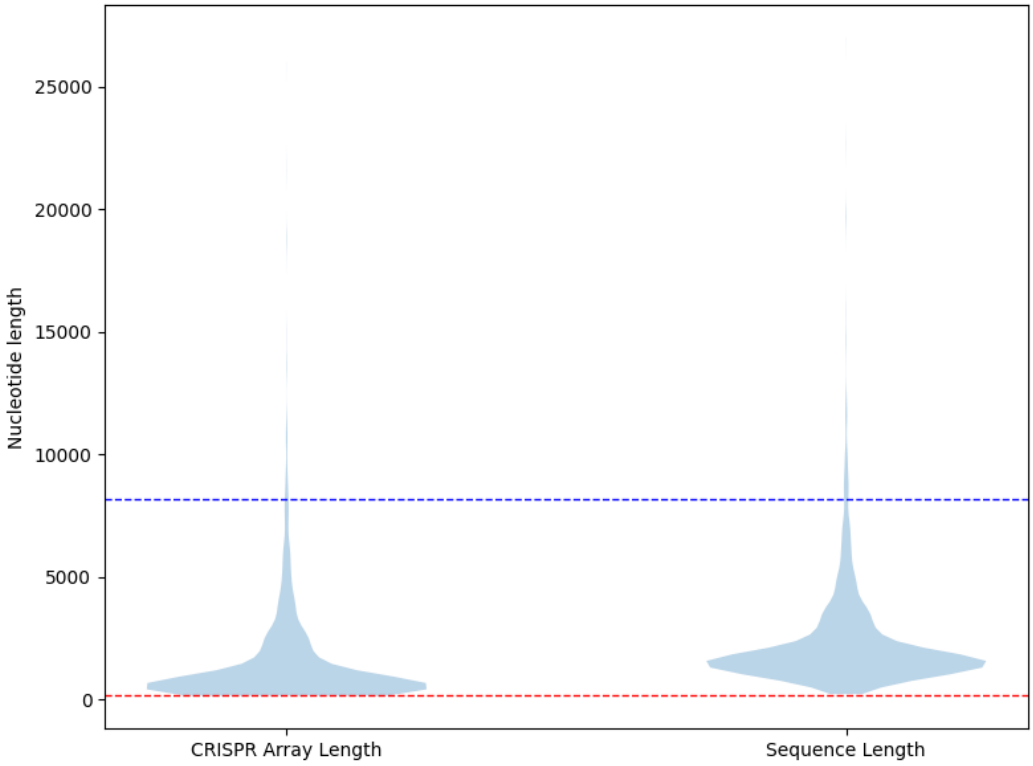

Figure 4.: Distribution of Finetune Dataset Lengths. Lengths of the 5,084 CRISPR arrays and the length of their sequences (CRISPR array length plus flanking nucleotides) in the fine-tuning dataset. The violin plot shows the distribution of lengths. The median lengths are 701 and 1,639, the longest are 26,001 and 27,001, and the shortest are 143 and 221, for CRISPR array length and sequence length, respectively. The red dashed line represents the target length of 150, and the blue line the 8,192. This distribution shows that the majority of CRISPR arrays fit within the 8,192 context window and that all sequences are longer than the 150 target length.

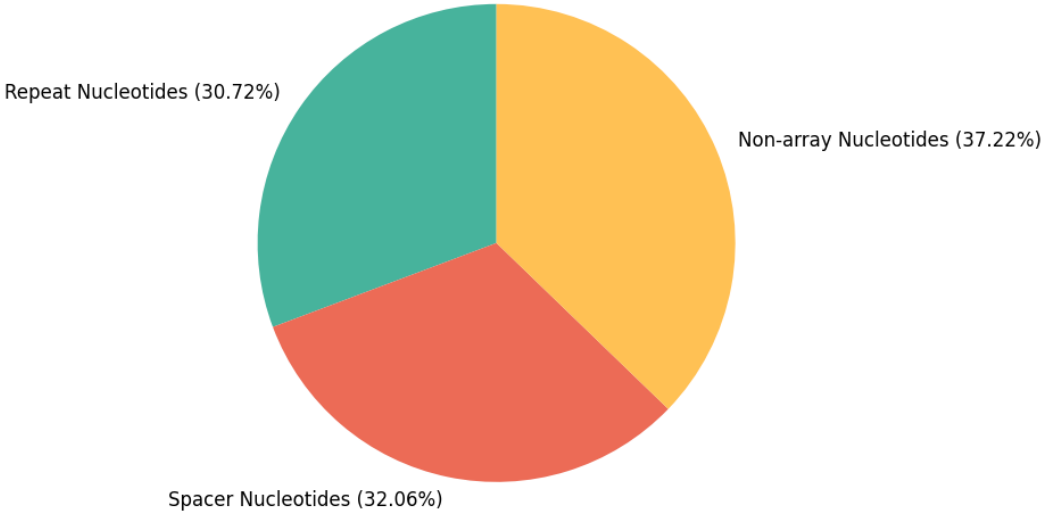

Figure 5.: Nucleotide Distribution by Class of the Finetune Dataset. Labels of nucleotides in the fine-tuning dataset. The 5,084 CRISPR arrays and their potential flanking regions. The chart shows the proportion of nucleotides by each label: repeat nucleotides (3,563,841), spacer nucleotides (3,719,636), and non-CRISPR array (non-array) nucleotides (4,318,435). This distribution shows that repeat and spacer nucleotide classes are balanced, while non-array nucleotides form the largest class.

4.4. Input Features

The input pipeline was carefully designed to maintain compatibility with Evo's pretraining framework while addressing the specific requirements of CRISPR array detection. To achieve this balance, our feature processing consists of two key components:

- **Raw Nucleotide Encoding:** Sequences are converted using byte-level, single-nucleotide encoding, where each character is converted to its ASCII byte representation (A=65, C=67, G=71, T=84, N=78, etc.) following Evo's pretraining protocol [22]. This sequence encoding is easy to process and allows extension up to 512 characters [22, 46].

- **Dynamic Sequence Handling:**

  - For sequences exceeding the target length (150, 300, 8,192, etc.), we employ random cropping with uniform sampling. This data augmentation strategy prevents overfitting by ensuring the model sees different array and non-array segments during training [47].

  - Shorter sequences are padded with neutral values to the target length, maintaining batch processing efficiency while avoiding introduction of artificial nucleotide signals [48].

The unified encoding strategy between pretraining and fine-tuning phases ensures seamless knowledge transfer from Evo's genome-scale next token (nucleotide) prediction pretraining task. By maintaining consistency with the original byte-level ASCII encoding and utilizing random cropping, we preserve the model's learned representations of prokaryotic sequence patterns while adapting to the CRISPR detection task.

Having established the input processing approach, we now turn to the backbone model that forms the core of our architecture.

4.5. Evo

We employed Evo as the backbone foundation model for our CRISPR array detection approach. Evo is a state-of-the-art genomic foundation model based on the StripedHyena architecture, designed for single-nucleotide resolution modeling of biological sequences [22].

Evo was pretrained on the OpenGenome dataset, comprising over 300 billion nucleotides from 12.2 million accessions spanning prokaryotic genomes, phages, and plasmids. This extensive pretraining enables Evo to learn the complex sequence patterns and evolutionary signatures present across diverse prokaryotic taxa [22]. A key feature of Evo is its long-context processing capability: depending on the model version, Evo supports input sequences up to 8,192 or 131,072 nucleotides in length. This allows the model to capture the repeat–spacer organization and long-range dependencies characteristic of CRISPR loci. Furthermore, Evo's hybrid architecture combines data-controlled convolutional operators with attention mechanisms, efficiently extracting both local and global sequence features critical for genomic analysis [22]. For this work, we utilize the 8,192-nucleotide context window version to balance computational feasibility and sequence coverage during fine-tuning.

During pretraining, Evo was optimized for next nucleotide prediction, learning to model the statistical properties of genomic sequences [22]. This foundational knowledge enables effective transfer learning for downstream tasks, such as per-nucleotide classification of CRISPR array components (repeat, spacer, non-array).

By leveraging Evo's pretrained representations, our approach benefits from robust, evolution-aware sequence modeling, which is particularly advantageous for detecting

CRISPR arrays in fragmented or complex genomic contexts where traditional methods may fail [22].

This backbone supports the subsequent classification and parameter-efficient fine-tuning strategies detailed in the following sections.

4.6. Classification Head

The classification head processes Evo's position-wise embeddings using a straightforward architecture tailored for per-nucleotide classification into three classes: repeat, spacer, and non-array. The design prioritizes simplicity and interpretability while leveraging Evo's rich sequence representations. The classification head consists of the following three components:

1. **Dropout Layer:** Applies dropout regularization to Evo's 512-dimensional token embeddings, mitigating overfitting by randomly zeroing features during training.

2. **Fully Connected Layer:** A linear transformation ($\mathbb{R}^{512} \to \mathbb{R}^3$) maps embeddings to class logits. The weight matrix $W \in \mathbb{R}^{512\times 3}$ and bias vector $b \in \mathbb{R}^3$ are learned during fine-tuning.

3. **Activation Function:** Softmax activation converts logits to class probabilities for each nucleotide position i:

$$\text{Softmax}(\mathbf{z}_i)_c = \frac{e^{z_{i,c}}}{\sum_{k=1}^{3} e^{z_{i,k}}}$$

   This ensures probabilities sum to 1 across classes (repeat, spacer, non-array), enforcing mutual exclusivity.

The final architecture implements:

$$\hat{Y} = \text{Softmax}(W \cdot \text{Dropout}(H_{\text{Evo}}) + b) \tag{1}$$

Classification Head

where $H_{\text{Evo}} \in \mathbb{R}^{L \times 512}$ contains Evo's embeddings for a sequence of length L, and $\hat{Y} \in \mathbb{R}^{L \times 3}$ contains per-nucleotide class probabilities. Cross-entropy loss compares $\hat{Y}$ against one-hot encoded labels:

$$\mathcal{L} = -\frac{1}{L} \sum_{i=1}^{L} \sum_{c=1}^{3} y_{i,c} \log(\hat{y}_{i,c}). \tag{2}$$

Fine-tuning Loss

where $y_{i,c}$ is the true label for nucleotide i and class c. The classification head's parameters (W, b) are optimized during fine-tuning, while Evo's parameters remain frozen except for those adapted via LoRA. This design enables effective per-nucleotide classification of CRISPR array components while leveraging Evo's pretrained sequence representations.

To enable efficient adaptation of the large foundation model, we employ a parameter-efficient fine-tuning strategy, detailed in the next section.

4.7. Low-Rank Adaptation (LoRA)

To enable efficient adaptation of the Evo backbone model for the CRISPR array detection task, LoRA is employed. LoRA is a parameter-efficient fine-tuning technique that introduces trainable low-rank matrices into selected layers of a large pretrained model while keeping the majority of the model's original parameters frozen. This

approach allows effective domain adaptation with minimal computational overhead, preserving the benefits of large-scale pretraining [11].

Application to Evo:

The implementation of LoRA in our system involves several key considerations:

- **Targeted Layers:** LoRA is applied to Evo's query, key, and value, as well as all MLP layers. These layers are critical for both local and global sequence modeling, maximizing adaptability while minimizing computational and memory overhead.

- **Low-Rank Decomposition:** For a frozen weight matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$, LoRA introduces trainable matrices $\mathbf{A} \in \mathbb{R}^{r \times k}$ and $\mathbf{B} \in \mathbb{R}^{d \times r}$, where $r \ll \min(d, k)$. The modified forward pass is given by:

$$\mathbf{h} = \mathbf{W}\mathbf{x} + \alpha \mathbf{B}\mathbf{A}\mathbf{x}$$

  where α is a scaling factor. Only $\mathbf{A}$ and $\mathbf{B}$ are updated during fine-tuning, while $\mathbf{W}$ remains fixed.

- **Fixed LoRA Scaling Factor α:** Following Hu et al., we also fix the scaling factor α and use the rank to determine the LoRA learning rate [11].

- **Parameter Efficiency:** With $r = 8$, the number of trainable parameters is reduced to approximately 0.63% of the full model, enabling fine-tuning of the full 8,192 context window Evo model on a single 80GB GPU.

Comparison to Alternatives:

To justify this approach, we considered several alternatives:

- **Full Fine-tuning:** Updating all model parameters is computationally expensive and requires significantly more memory, making it infeasible for large models like Evo on available hardware.

- **Adapter Layers:** While effective, adapter-based methods increase inference latency and add additional parameters at each layer [49].

- **Dynamic Rank Allocation:** Although dynamic rank allocation can further optimize parameter usage, it is not adopted here to maintain simplicity and reproducibility.

By leveraging LoRA, this work achieves efficient and effective adaptation of the Evo foundation model for CRISPR array detection. This approach preserves the pretrained knowledge of the backbone model while enabling domain-specific specialization and effective transfer learning for CRISPR array detection.

With the architectural components established, the next step is to optimize the model performance through hyper-parameter tuning, detailed in the following section.

4.8. Hyper-Parameter Optimization (HPO)

Hyper-parameter optimization (HPO) was conducted to establish stable and effective training dynamics for the 8,192-context Evo model. The process focused on the parameters most critical to model performance and accounted for hardware constraints imposed by training on a single NVIDIA A100 80GB GPU, following established HPO best practices [50, 51, 11, 52].

Optimized Parameters

The following hyperparameters were systematically explored in our optimization process:

- **Learning rate (α):** 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}
- **LoRA rank (r):** 4, 8, 16, 24, 32, 64
- **Classification head dropout probability:** 0%, 5%, 10%, 15%, 20%, 25%
- **AdamW weight decay:** 0, 10^{-3}, 10^{-2}, 10^{-1}

Constrained Parameters

Due to the large context size and memory limitations, several parameters were fixed for the 8,192 target length experiments:

- **Batch size:** 1
- **LoRA target modules:** Projection and MLP layers
- **Precision:** `bf16` (enabled)

For experiments with shorter target sequence lengths, batch size was maximized by choosing values in powers of 2^n, as suggested by Goodfellow et al. [52]. Other constraints remained in place.

Non-optimized Parameters

The following settings were kept fixed throughout HPO:

- **LoRA bias:** none
- **LoRA scaling factor (α):** 8
- **Adam β_1:** 0.9
- **Adam β_2:** 0.999

- **Adam ϵ:** 1×10^{-8}

- **Gradient clipping:** $|\nabla| \leq 1.0$

- **Learning rate schedule:** cosine decay

All hyperparameters except for cosine decay were the default values in the HuggingFace environment [53, 54, 55]. Cosine decay was chosen due to its proven success as demonstrated by Loshchilov et al. Annealing was not used due to resource constraints [56].

Optimization Procedure

Manual hyperparameter tuning was favored over exhaustive grid search, random search, or other AutoML HPO procedures due to resource constraints and prior knowledge of the model. First, the training loss was decreased with the non-regularization hyperparameters, and then the validation accuracy was increased through regularization, evaluating each configuration after 500 training steps [52]. The systematic process proceeded as follows:

1. Begin with a learning rate of 10^{-5}, increasing stepwise until training loss diverges, with LoRA r set to its default (8) and regularization (dropout and AdamW weight decay) disabled [54, 55].

2. For the learning rate with the lowest training loss, sweep LoRA r values to identify the rank with the lowest training loss.

3. Subsequently, introduce classification head dropout and select its value based on validation accuracy.

4. Then, optimize AdamW weight decay based on validation accuracy.

5. Finally, take the configuration with the highest validation accuracy. For the 8,192 target length, it is trained for three epochs (the default value [53, 55]), while a lower target length is trained for more than three epochs.

Each experiment was run with a fixed random seed to ensure reproducibility. The best hyperparameter set was selected based on the highest validation accuracy after 500 steps. Results are based on single runs per configuration due to computational constraints. Due to balanced class distribution, accuracy was used as the primary validation metric.

Computational Budget

Given hardware constraints, the total number of configurations evaluated was limited. Only the most promising parameter combinations were explored further with additional regularization tuning, reflecting practical trade-offs in HPO for deep learning [50, 51, 52].

Summary

This HPO protocol prioritizes efficient exploration of the most impactful hyperparameters while maintaining reproducibility and computational feasibility. The results of the optimization are detailed in Section 5.4.

Having established the optimization framework, we can now describe the specific model variants developed for different sequence lengths and their evaluation methodology.

4.9. Fine-tuned Models

This work develops fine-tuned models that specifically address the limitations of existing CRISPR array detection approaches, as outlined in Section 1.2. Specifically,

the fine-tuned models are designed to overcome challenges such as the inability of traditional tools to detect CRISPR arrays in short sequencing reads (e.g., Illumina reads) and reduced sensitivity to arrays with degenerate or heterogeneous repeats.

Target Sequence Lengths and Model Variants

Two primary fine-tuned model variants were developed to leverage the full capabilities of the Evo 8k backbone model and to address the requirements of different sequencing technologies:

- **8,192 Target Length Model:** Utilizes the maximum context window of the Evo 8k model, enabling detection of long-range repeat-spacer patterns characteristic of CRISPR arrays. This model is suitable for analyzing assembled genomes or long-read sequencing data, where capturing entire or large parts of the array structure is essential.

- **150 Target Length Model:** Tailored for Illumina sequencing data, which typically produces short reads ranging from 75 to 300 base pairs [57, 8]. By restricting the target length to 150 nucleotides, this model is optimized for high sensitivity and specificity in short-read contexts, directly addressing the limitations of existing methods on metagenomic or fragmented data.

Furthermore, additional experiments, detailed in Section 5.7, extend this approach to cover a broader range of Illumina read lengths, ensuring adaptability across various sequencing platforms and study designs.

Training and Optimization

All fine-tuned models were trained using LoRA, which allows effective domain adaptation while keeping the majority of backbone parameters frozen. HPO was conducted for each target length to maximize model performance, as described in Section 4.8.

Evaluation

Chapter 5 presents experiments designed to evaluate whether these models address the main weaknesses of state-of-the-art CRISPR array detection tools, including performance on short reads and robustness to repeat sequence heterogeneity.

While accuracy may be a good indication of overall model performance, an algorithm was developed to classify and quantify different error types for further analysis. The error types are categorized as follows:

1. **Non-array from start is classified as array:** At the start of the input sequence, nucleotides not belonging to the CRISPR array are classified as part of the array.

2. **Non-array from end is classified as array:** At the end of the input sequence, nucleotides not belonging to the CRISPR array are classified as part of the array.

3. **Too long repeat:** Nucleotides flanking the repeat are classified as repeat nucleotides.

4. **Too long spacer:** Nucleotides flanking the spacer are classified as spacer nucleotides.

5. **Misclassified repeat:** All nucleotides in a repeat are misclassified as either non-array or spacer.

6. **Misclassified spacer:** All nucleotides in a spacer are misclassified as either non-array or repeat.

7. **Start of array is classified as non-array:** At the start of the input sequence, nucleotides belonging to the CRISPR array are classified as non-array.

8. **End of array is classified as non-array:** At the end of the input sequence, nucleotides belonging to the CRISPR array are classified as non-array.

9. **Undetermined error:** Wrong classification of nucleotides that do not belong to any of the above categories.

Algorithm 1 presents pseudo-code for determining these error types. For the error types "Non-array from start is classified as array", "Non-array from end is classified as array", "Start of array is classified as non-array", and "End of array is classified as non-array" a tolerance window of three was applied. If a nucleotide was deemed to have that error type from the start or end, a nucleotide between one to three positions after (or before) could also be classified as that error, even if the nucleotide(s) in between did not have that or any error type. This ensures that in regions where the model is uncertain about nucleotide classification, errors are assigned to the correct type.

Having established this framework, Chapter 5 will demonstrate how these models perform in practice and evaluate their effectiveness in addressing the challenges of short-read CRISPR array detection and repeat heterogeneity.

Algorithm 1 Determine Error Types

Initialize empty error array **errors**$[0..n-1]$
$n \leftarrow$ length of sequence
for $i \leftarrow 0$ **to** $n-1$ **do**
    if $labels[i] \neq predictions[i]$ **then**
        errors$[i] \leftarrow$ "undetermined_error"
    end if
end for
for each $(start, end)$ in CRISPR array intervals **do**
    if all **errors**$[start..end] =$ "undetermined_error" **then**
        Set **errors**$[start..end]$ based on label type (repeat/spacer)
    end if
    Extend **too_long** errors before *start* and after *end*
end for
Check sequence start with 3-nucleotide tolerance window:
for $i \leftarrow 0$ **to** $n-1$ **do**
    if in non-array region **and** prediction = array **then**
        if $i \leq 3$ **or** has error in previous 3 positions **then**
            errors$[i] \leftarrow$ "non_array_from_start"
        end if
    end if
end for
Check sequence end with 3-nucleotide tolerance window:
for $i \leftarrow n-1$ **downto** 0 **do**
    if in non-array region **and** prediction = array **then**
        if $i \geq n-4$ **or** has error in next 3 positions **then**
            errors$[i] \leftarrow$ "non_array_from_end"
        end if
    end if
end for
return errors

5. Experiments

The present chapter is concerned with an experimental evaluation of the fine-tuned models for CRISPR array detection. In Section 5.1 and Section 5.2 , the capabilities of the pretrained Evo model were assessed in the context of next nucleotide prediction and zero-shot classification, respectively. Section 5.3 and Section 5.4 outline the data partitioning strategy and the HPO results for the fine-tuned models. In Sections 5.5, 5.6, and 5.7, the robustness of the model is analysed with regard to repeat similarity, input sequence shifts and varying target lengths. Finally, Sections 5.8 and 5.9 employ the models for two downstream tasks. The analysis of simulated metagenomic data and the detection of degenerate repeats.

5.1. Zero-Shot Next Nucleotide Prediction

To assess whether the Evo backbone model possesses inherent knowledge of CRISPR arrays, we evaluated its performance on the next nucleotide prediction task using the pretrained model prior to any fine-tuning. Specifically, we input the first 8,192 nucleotides of each sequence in the fine-tuning dataset (see Section 4.3) and measured the model's ability to predict the correct subsequent nucleotide at each position.

The Evo model achieved an average next nucleotide prediction probability of 57.22% on the fine-tuning dataset. While Nguyen et al. [22] report log probabilities as sequence scores, we present the standard probabilities here for improved interpretability.

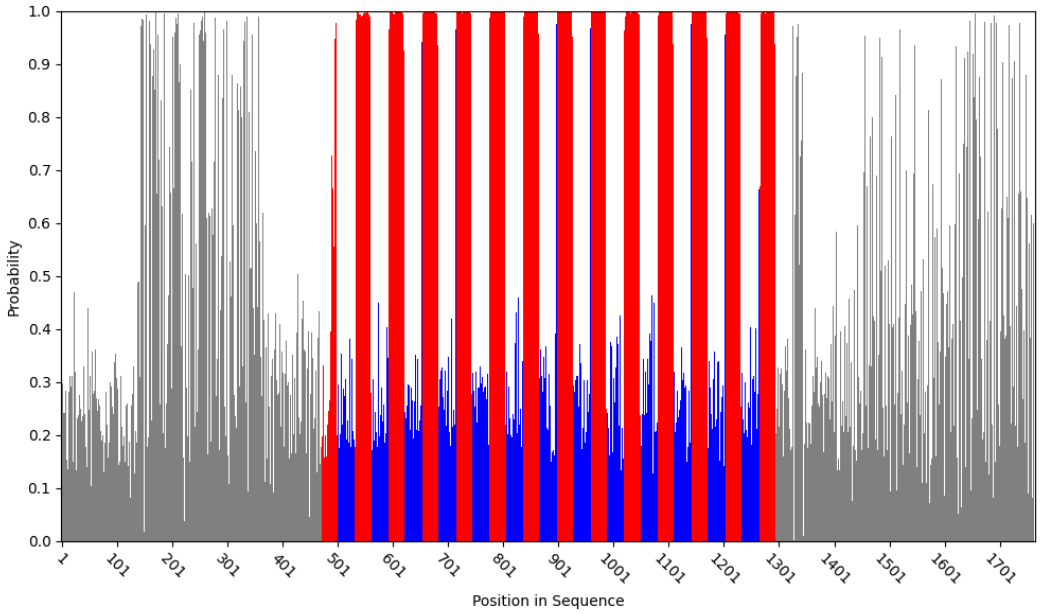

Figure 6.: Next Nucleotide Prediction Example. Example of next nucleotide prediction probabilities for a randomly selected CRISPR array, colored by CRISPRidentify-labeled nucleotide class (repeat is red, spacer is blue, and non-array is gray).

To dissect the model's predictive behavior, we analyzed the class distribution of nucleotides for which the predicted probability exceeded the average. Notably, 75.00% of these high-confidence predictions corresponded to repeat regions, 21.11% to non-array regions, and only 3.89% to spacers. This indicates that the model is particularly adept at recognizing and predicting repeats, a key structural feature of CRISPR arrays, which is congruent with the amount of viral nucleotides the model was pretrained on (see Figure 3). Figure 6 exemplifies this: the model achieves prediction probabilities exceeding 95% for most nucleotides within each repeat (except the first), suggesting that once the repeat structure is recognized, the model can reliably identify subsequent repeats. More examples are visualized in Section A.1.

These findings demonstrate that the Evo model, even before fine-tuning, captures salient sequence patterns characteristic of CRISPR arrays, especially repeat motifs. This inherent capability supports its suitability as a backbone for downstream CRISPR array detection tasks.

5.2. Zero-Shot Classification

An important question when leveraging large foundation models is whether task-specific fine-tuning is necessary or whether meaningful predictions can be obtained using the inherent representations learned during pretraining. This section explores whether a simple, unsupervised classification approach—based purely on the output probabilities of the pretrained Evo model—can be used to effectively detect CRISPR array components without any additional fine-tuning.

Methodology

Building upon the findings in Section 5.1, we observed that the pretrained model exhibits strong next nucleotide prediction performance for internal repeats but lower

confidence for the initial repeat of each CRISPR array. To address this bias, we augmented the prediction process by analyzing both the forward and reverse complements of each sequence. Thus, every input sequence is processed twice. While this approach enhances sensitivity, it doubles the required inference time, or requires two models instances to predict the sequences in parallel.

For classification, a sliding window strategy is adopted: for each nucleotide position, the 15-nucleotide window centered at that position (7 nucleotides to each side) is considered. If at least 5 out of these 15 positions have model probabilities exceeding 95%, the central position is labeled as part of a repeat. This procedure is run independently on both the original and reverse complement sequences; the predicted repeat regions are then merged by taking their union. Spacers are defined as regions between predicted repeats, and all remaining nucleotides are classified as non-array.

Results and Analysis

When applied to the CRISPR arrays in the fine-tuning dataset, this zero-shot classifier achieves an accuracy of 89.4%, demonstrating substantial ability to discriminate repeat and spacer regions within annotated arrays. However, when evaluated over the entire sequence (including flanking, non-array regions), the accuracy decreases significantly to 60.5%. This drop in performance highlights the model's limited ability to differentiate CRISPR repeats from non-CRISPR elements present in prokaryotic genomes.

The main limitation arises because the Evo foundation model, during pretraining, primarily learns general features of prokaryotic DNA—without explicit exposure to annotated repeat/non-repeat distinctions relevant to CRISPR arrays. As a result, while the model captures strong signals for canonical repeat structures, it cannot robustly distinguish CRISPR-specific patterns from other genomic regularities (see Section 4.2). Figure 7 illustrates a representative case where the classifier performs

well within the array but yields false positives in the flanking regions. More cases are shown in Section A.2.

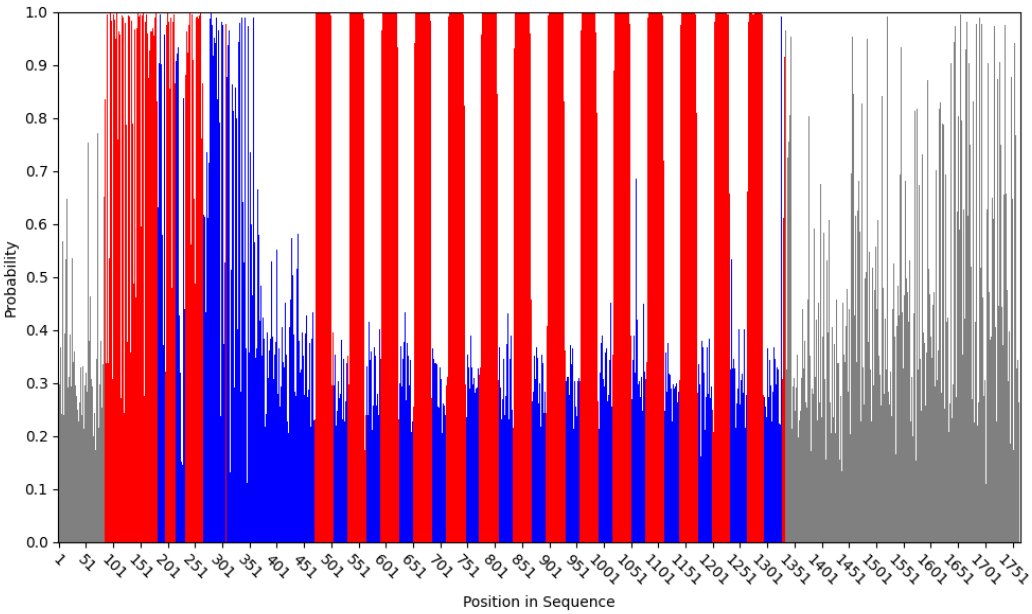

Figure 7.: Zero-Shot Classification Example. Example of zero-shot classification for the same CRISPR array shown in Figure 6. The plot visualizes the predicted nucleotide class for each position, computed utilizing the maximum probability from the original and reverse complement sequences: repeat (red), spacer (blue), and non-array (gray).

Conclusion

This experiment demonstrates that while the pretrained Evo model captures useful sequence motifs for CRISPR repeat identification, the lack of explicit task adaptation hinders robust generalization, especially with respect to array boundaries and distinguishing non-CRISPR elements. Therefore, fine-tuning remains essential to achieve reliable, high-precision CRISPR array detection across diverse genomic contexts.

5.3. Data Partitioning

To fine-tune and evaluate the models for each target length dataset, we partitioned the data into training, validation, and test sets using a 70-10-20 split, respectively. This split ensures that the majority of data is available for model training, while reserving sufficient validation data for hyperparameter tuning and an adequate test set for robust evaluation of model performance and error analysis.

To prevent data leakage between splits, duplicate input sequences were removed from the target length datasets. The partitioning was performed at the sequence level, ensuring that no identical sequence appears in more than one subset.

Furthermore, we assessed the possibility of data leakage due to sequence similarity—particularly among CRISPR repeats—across splits. Section 5.5 presents experiments quantifying the impact of similar repeat sequences between partitions and discusses their potential effect on model performance.

A fixed random seed was used during partitioning to ensure reproducibility. The number of sequences in each dataset is detailed in Table 1. The nucleotide distribution by class can be seen in Figure 8 and Figure 9 for 8,192 and 150 target lengths, respectively. Both datasets contain substantial numbers of all three classes; however, while the 8,192 nucleotide dataset has a class distribution deviation of 1.51%, the 150 target length dataset has a higher deviation of 8.30% due to the increased proportion of non-array nucleotides. This is partly due to the ability to crop shorter sequences such that they may not contain any array nucleotides, which is not possible for target lengths above 500 nucleotides.

This rigorous partitioning and class distribution analysis ensures that subsequent model evaluation is both fair and reflective of real-world sequence diversity, supporting robust and interpretable experimental results.

Table 1.: Number of Sequences in Dataset Splits by Target Length. Number of sequences in each dataset split after deduplication. N_{train}, $N_{validation}$, and N_{test} are the numbers of sequences in the training, validation, and test splits, respectively, and N denotes the total number of sequences.

Target length	N_{train}	$N_{validation}$	N_{test}	N
8,192	3,553	507	1,016	5,076
150	3,484	498	1,016	4,995

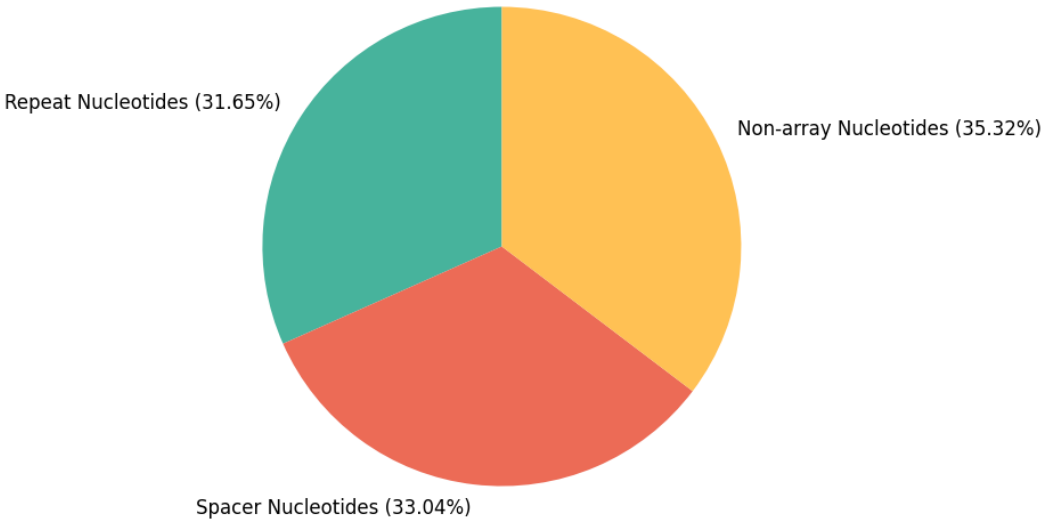

Figure 8.: 8,192 Target Length Nucleotide Distribution by Class. Labels of nucleotides in the 8,192 target length dataset. The chart shows the proportion of nucleotides by each label: repeat nucleotides (2,992,945), spacer nucleotides (3,124,066), and non-array nucleotides (3,339,725). This distribution shows that while still substantial, the non-array nucleotide percentage is smaller than in Figure 5, partly due to the cropping of CRISPR arrays that are over 8,192 in length, which contain mainly array nucleotides.

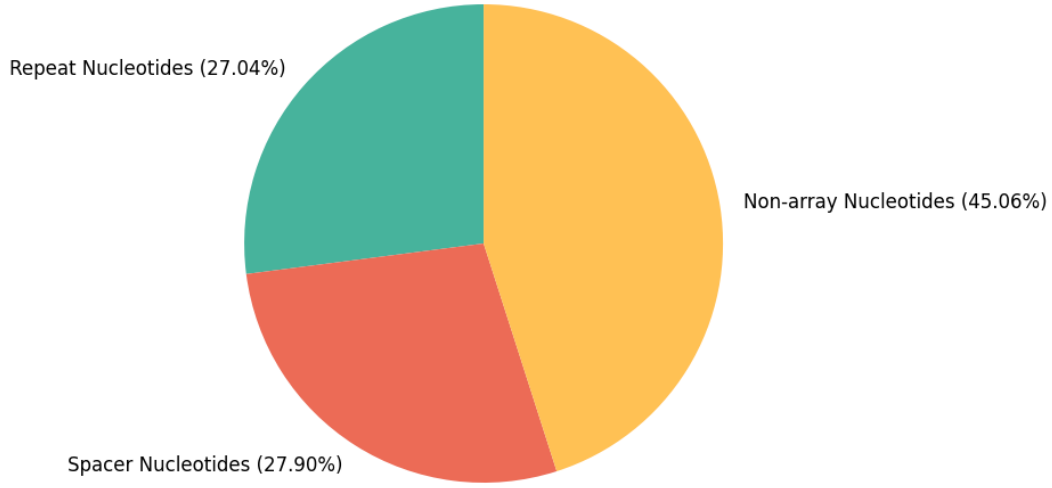

Figure 9.: 150 Target Length Nucleotide Distribution by Class. Labels of nucleotides in the 150 target length dataset. The chart shows the proportion of nucleotides by each label: repeat nucleotides (202,698), spacer nucleotides (209,178), and non-array nucleotides (337,824). This distribution is less similar to the original fine-tuning dataset than to the 8,192 target length dataset (see Figure 5 and Figure 8, respectively), partly due to the capability of creating input sequences with only non-array nucleotides.

5.4. Hyper-Parameter Optimization (HPO) Results

As detailed in Section 4.8, HPO was conducted to develop models for the target lengths. This section presents the results of the HPO process, summarizing the impact of each hyperparameter on model performance and describing and evaluating the final configurations selected for further experiments.

Introduction and Rationale

The performance of deep learning models is highly sensitive to hyperparameter choices, especially for large-scale genomic sequence tasks. To ensure robust and reproducible results, key hyperparameters—including learning rate, LoRA rank, dropout probability, and weight decay—were systematically tuned for each target length. Due to computational constraints, a stepwise manual search was employed, prioritizing the most influential parameters as established in prior work [50, 51, 11, 52].

Optimization Procedure

For each target length, the HPO process was structured as follows:

1. Proceed stepwise from the lowest to the highest learning rate until the training loss diverges, using the default LoRA rank and no regularization [54, 55].

2. For the best learning rate, vary the LoRA rank to minimize training loss.

3. Introduce and tune dropout probability based on validation accuracy.

4. Tune AdamW weight decay for further regularization.

5. For the 8,192 target length, retrain the best configuration for three epochs.

Training results for 8,192 Target Length

Table 2 summarizes the HPO results for the 8,192 target length. The initial learning rate determination revealed that $\alpha = 10^{-4}$ provided the lowest training loss. Further tuning of the LoRA rank identified $r = 4$ as optimal. Introducing dropout regularization improved validation accuracy, with 15% dropout achieving the best results. Weight decay regularization was also explored but did not yield further improvements over the dropout-only configuration.

Table 2.: Hyper-Parameter Optimization (HPO) Results for 8,192 Target Length Model. Hyper-Parameter Optimization (HPO) results for models training for the 8,192 target length, where Loss is the training loss, α is the learning rate, r is the LoRA rank, Accuracy is the validation accuracy, Dropout is the classification head dropout percentage, and Weight Decay is the AdamW weight decay.

Step	α	r	Dropout	Weight Decay	Loss	Accuracy
1. α	10^{-5}	8	0%	0	1.0693	56.79%
	10^{-4}	8	0%	0	0.2200	94.13%
	10^{-3}	8	0%	0	0.5743	61.22%
2: r	10^{-4}	4	0%	0	0.1586	94.96%
	10^{-4}	16	0%	0	0.1709	94.36%
	10^{-4}	24	0%	0	0.4015	94.31%
	10^{-4}	32	0%	0	0.2082	94.58%
	10^{-4}	64	0%	0	0.1975	94.02%
3. Dropout	10^{-4}	4	5%	0	0.2856	95.49%
	10^{-4}	4	10%	0	0.1953	95.35%
	10^{-4}	4	15%	0	0.2375	95.61%
	10^{-4}	4	20%	0	0.1823	94.92%
	10^{-4}	4	25%	0	0.2847	93.00%
4. Weight Decay	10^{-4}	4	15%	10^{-3}	0.2160	95.06%
	10^{-4}	4	15%	10^{-2}	0.2003	94.10%
	10^{-4}	4	15%	10^{-1}	0.1281	95.64%
5. Final model	$\mathbf{10^{-4}}$	**4**	**15%**	**0**	**0.0525**	**97.78%**

The final configuration—$\alpha = 10^{-4}$, $r = 4$, 15% dropout, and no weight decay—achieved a validation accuracy of 95.61%. It was then retrained for three epochs and achieved a validation accuracy of 97.78%.

Training results for 150 Target Length

Table 3 presents the HPO results for the 150 target length, which is tailored for short-read Illumina data. Here, a higher learning rate ($\alpha = 10^{-3}$) yielded the best results. LoRA rank $r = 32$ was found to be optimal and, with no dropout or weight decay, achieved the best validation accuracy.

The best configuration for the 150 target length—$\alpha = 10^{-3}$, $r = 32$, with no dropout and weight decay—achieved a validation accuracy of 88,57%.

Evaluation

The final models for the main target lengths 8,192 and 150 achieve 98.16% and 90.03% accuracy on their test data, respectively. The reason for the lower test accuracy of the 150-nucleotide model is not immediately apparent from these results.

As can be seen from Figure 10 and Figure 11, a significant number of the model errors are due to classifying nucleotides as being part of the CRISPR array when CRISPRidentify has not classified them as part of it. Section 5.9 explores this to show that these models can be used to predict degenerated repeats, even though they are trained on data from tools that cannot find them.

The biggest error type for the 150-nucleotide model is “Start of array is classified as non-array” with 7,171 falsely classified nucleotides, whereas the corresponding “End of array is classified as non-array” only has 279 falsely classified nucleotides (see Figure 11). A similar pattern is also observed with the 8,192-nucleotide model. This

Table 3.: Hyper-Parameter Optimization (HPO) Results for 150 Target Length Model. Hyper-Parameter Optimization HPO results for models training for the 150 target length with the batch size of 64, where Loss is the training loss, α is the learning rate, r is the LoRA rank, Accuracy is the validation accuracy, Dropout is the classification head dropout percentage, and Weight Decay is the AdamW weight decay. The model with the highest validation accuracy was selected as the final model, highlighted in bold text.

Step	α	r	Dropout	Weight Decay	Loss	Accuracy
1. α	10^{-5}	8	0%	0	1.0235	44.71%
	10^{-4}	8	0%	0	0.2770	87.01%
	10^{-3}	8	0%	0	0.0804	87.42%
	10^{-2}	8	0%	0	1.0662	39.12%
2. r	10^{-3}	4	0%	0	0.0711	87.49%
	10^{-3}	16	0%	0	0.0504	88.49%
	10^{-3}	24	0%	0	0.0870	87.23%
	10^{-3}	**32**	**0%**	**0**	**0.1026**	**88.57%**
	10^{-3}	64	0%	0	0.0571	87.97%
3. Dropout	10^{-3}	32	5%	0	0.0565	86.66%
	10^{-3}	32	10%	0	0.0617	87.29%
	10^{-3}	32	15%	0	0.0540	87.68%
	10^{-3}	32	20%	0	0.1708	87.02%
	10^{-3}	32	25%	0	0.0740	87.27%
4. Weight Decay	10^{-3}	32	0%	10^{-3}	0.0513	87.32%
	10^{-3}	32	0%	10^{-2}	0.0618	87.92%
	10^{-3}	32	0%	10^{-1}	0.1002	87.96%

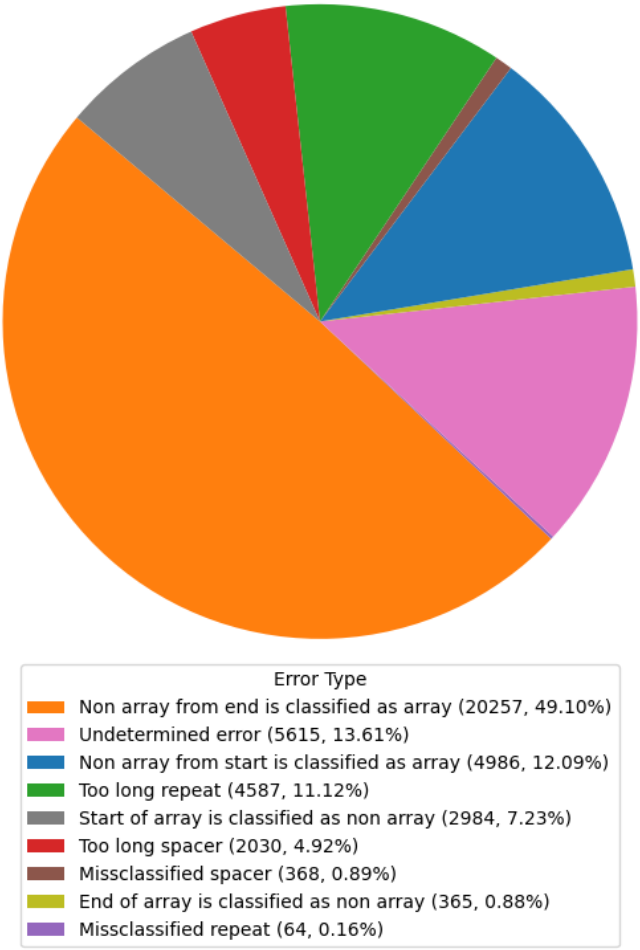

Figure 10.: Distribution of Error Types for the 8,192-Nucleotide Model. Error types of the falsely classified nucleotides in the 8,192 target length test dataset. The chart shows the proportion of error types by each nucleotide. The first number in the parentheses of each legend label denotes the number of misclassified nucleotides followed by its percentage of all nucleotide errors.

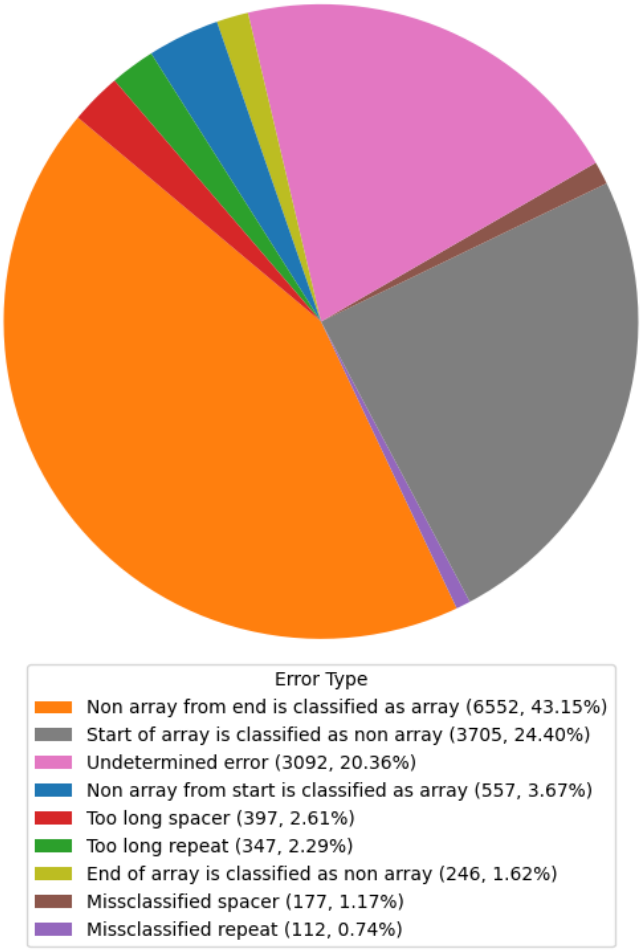

Figure 11.: Distribution of Error Types for the 150-Nucleotide Model. Error types of the falsely classified nucleotides in the 150 target length test dataset. The chart shows the proportion of error types by each nucleotide. The first number in the parentheses of each legend label denotes the number of misclassified nucleotides followed by its percentage of all nucleotide errors.

suggests that the models are better at predicting the later parts of the array than the beginning.

Due to computational constraints each configuration was evaluated only once and the search space was limited to the most promising parameter combinations. While this approach prioritizes efficiency, it may not capture the full variability of model performance across random seeds or alternative hyperparameter values. Nevertheless, the selected configurations consistently outperformed alternatives in preliminary runs and provided stable training dynamics.

5.5. Repeat Similarities Between Splits

To ensure that potential data leakage did not affect model performance, we conducted an experiment in which sequences in the training set containing repeats similar to those in the validation or test set were excluded. Repeat similarity was determined using the Levenshtein edit distance, chosen for its computational efficiency and interpretability [58, 59]. We used the 8,192 target length dataset to maximize the likelihood of detecting similar repeats and to ensure that most sequences contained full CRISPR arrays (see Figure 4). Only repeats entirely contained within the input sequence were considered, to avoid excessive sequence removal due to partial repeat similarity. Due to computational constraints, models were trained for three epochs using the final hyperparameter configuration (Table 2).

To assess whether model accuracy depended on repeat similarity, we calculated the correlation between the prediction accuracy of each test sequence and the weighted percent identity of its repeats compared with those in the training set. The FASTA algorithm was used to compute percent identity, which was then weighted by alignment length and the minimum repeat length to account for differences in repeat size [60].

The weighted percent identity was calculated as:

$$w = p \cdot \frac{N_{alignment}}{min\{N_{test}, N_{train}\}} \tag{3}$$

Weighted Percent Identity

where w is the weighted percent identity, p is the percent identity, $N_{alignment}$ is the alignment length, and N_{test}, N_{train} are the lengths of the repeats from the test and training splits, respectively. An E-value threshold of 0.01 was applied to retain only significant matches [61]. To assess whether model accuracy depends on repeat similarity, we calculated the correlation between the weighted percent identity of matched test repeats and the prediction accuracy of the test sequences containing them.

Table 4.: Model Test Accuracy on Datasets Excluding Similar Repeats. Overview over the model test accuracy of datasets where splits does not contain similar repeats. Edit Distance denotes the maximum edit distance that is not allowed between the splits, Dataset Size is the amount of sequences in the entire dataset, Model Test Accuracy is the accuracy on the test set after training the model, and Correlation is the correlation between the matched repeats weighted percent identity and the accuracy of the sequences containing them.

Edit Distance	Dataset Size	Model Test Accuracy	Correlation
0	3593	98.35%	0.10
1	3586	98.11%	0.13
2	3582	98.39%	0.10
3	3580	98.34%	0.09
4	3576	97.98%	0.08
5	3568	98.22%	0.10
6	3567	98.27%	0.09
7	3564	98.19%	0.10
8	3552	98.26%	0.08
9	3488	97.96%	0.07
10	3288	98.12%	0.15

Table 4 summarizes the results. Test accuracy remained stable across all edit distance thresholds, varying by less than ±0.23% from the standard test accuracy of 98.16%. As the allowed edit distance increased, the dataset size decreased, with an edit distance of 10 reducing the dataset to about two-thirds of its original size. This threshold was chosen to balance computational feasibility with dataset size and exceeds the default edit distance of 6 used by CRISPRidentify [7]. The correlation between test sequence accuracy and repeat similarity was consistently low (±0.15), indicating little to no relationship between repeat similarity and model performance.

5.6. Shift Input Sequences

To rigorously assess model robustness to small perturbations in input sequences—crucial for reliable array detection in real-world scenarios—we systematically evaluated both the 150-nucleotide and 8,192-nucleotide models by introducing controlled nucleotide shifts of ±1, ±5, and ±50 positions. For shifts extending beyond sequence boundaries, padding was applied to maintain consistent input length. Model output stability was quantified using the *consistency score* for overlapping regions, defined as:

$$c = \frac{\sum_{i=1}^{N} \mathbb{1}_{t_i = s_i}}{N} \tag{4}$$

Consistency Score

where c is the consistency score, N is the overlap length, and t_i, s_i denote the predictions for the original and shifted sequences at position i, respectively.

Key Observations

Both models maintain high consistency (>93%) for ±1 and ±5 nucleotide shifts. The 150-nucleotide model exhibits nearly symmetrical consistency across positive

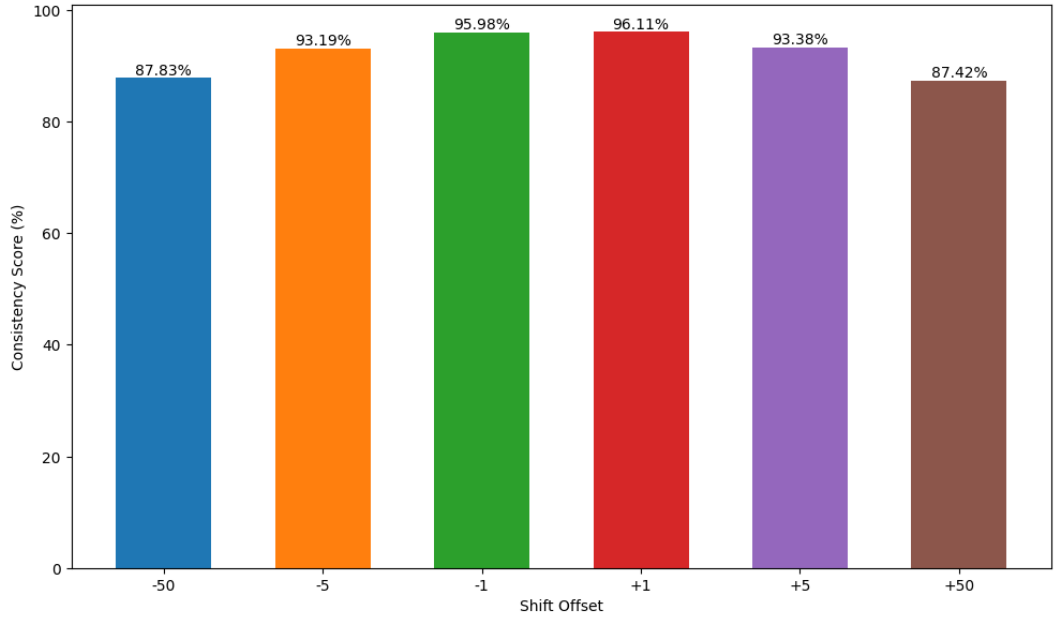

Figure 12.: Consistency Score of Shifted Sequences for 150 Target Length. Average consistency scores of the shifted sequences for the 150-nucleotide model.

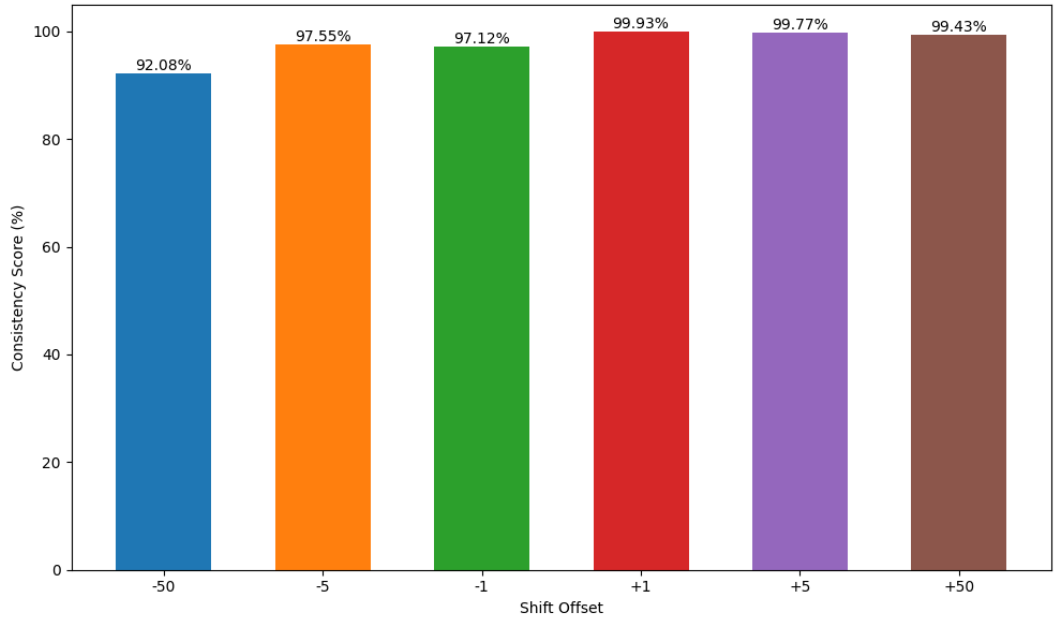

Figure 13.: Consistency Score of Shifted Sequences for 8,192 Target Length. Average consistency scores of the shifted sequences for the 8,192-nucleotide model.

and negative shifts (Figure 12). In contrast, the 8,192-nucleotide model displays pronounced asymmetry, with a notable drop in consistency for the −50 nucleotide shift (Figure 13).

Interpretation of Asymmetries

The observed asymmetries between offsets, shown in Figures 12 and 13, can be explained by fundamental differences in the training procedures for each model. For the 150-nucleotide model, the observed symmetry is attributable to randomized cropping during training, which exposes the model to diverse sequence windows and mitigates positional bias. In contrast, the 8,192-nucleotide model exhibits pronounced asymmetry because training was performed exclusively with end-padding. Consequently, negative shifts (such as the −50 nucleotide shift) introduce start-padding—an unseen configuration during training—leading to reduced consistency. Additionally, the larger sequence overlap in the 8,192-nucleotide model likely contributes to its generally higher consistency, except when unfamiliar padding configurations are encountered.

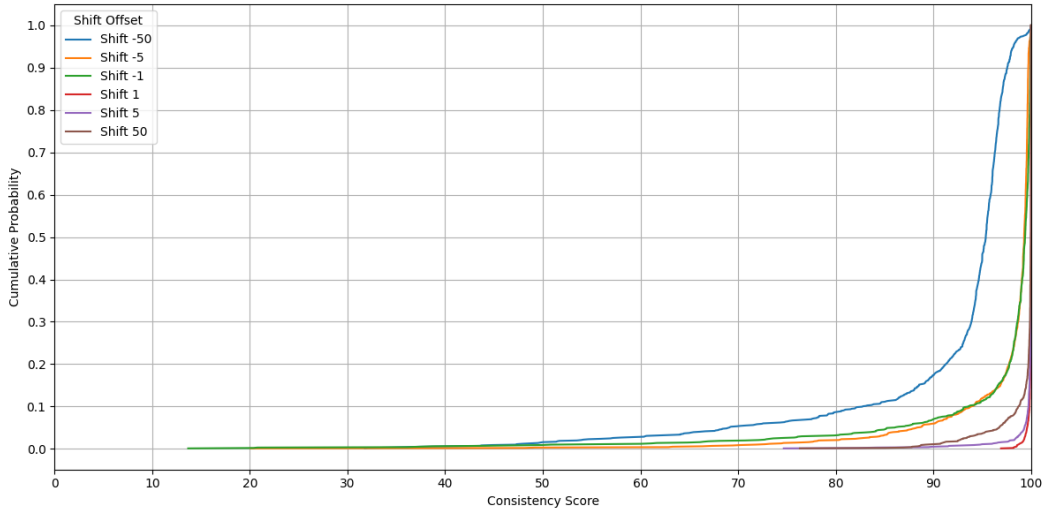

Figure 14.: CDF Plot of Consistency Scores by Shift Offset for the 8,192-Nucleotide Model. Cumulative distribution function plot of the consistency scores of the shifted sequences for the 8,192-nucleotide model.

Figure 14 demonstrates that, for the 8,192-nucleotide model, all shifts (except the −50 nucleotide shift) result in at least 90% of sequences achieving a consistency score above 90%. For positive shifts, all sequences exceed 97% consistency, indicating that only a small subset is affected by large negative shifts.

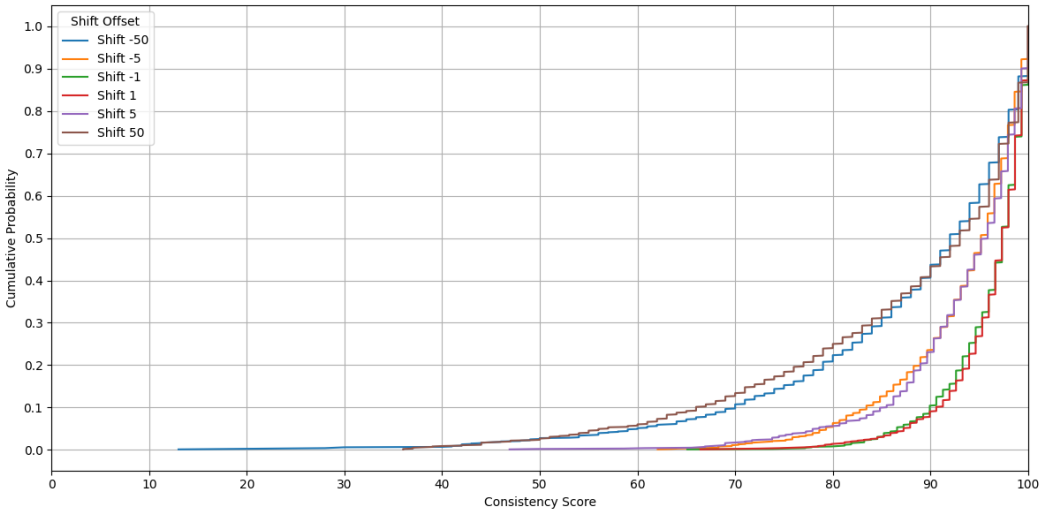

Figure 15.: CDF Plot of Consistency Scores by Shift Offset for the 150-Nucleotide Model. Cumulative distribution function plot of the consistency scores of the shifted sequences for the 150-nucleotide model.

In contrast, the 150-nucleotide model (Figure 15) shows similar distributions for positive and negative shifts. For the ±50 nucleotide shifts, only 80% of sequences maintain a consistency above 75%. For ±5 and ±1 nucleotide shifts, 80% of sequences exceed 87% and 91% consistency, respectively, but these results still indicate that even small shifts can substantially alter predictions.

Table 5 reveals strong correlations between negative shift offsets for the 8,192-nucleotide model, supporting the interpretation that the lack of start-padding exposure during training makes certain sequences more difficult to predict under such conditions.

Table 6 shows no strong correlations for the 150-nucleotide model, indicating that prediction instability is not due to specific sequences but rather reflects general model

Table 5.: Correlation Matrix of Consistency Scores by Shift Offset for the 8,192-Nucleotide Model. The correlation matrix quantifies the pairwise similarity of consistency scores across different shift offsets.

Shift Offset	-50	-5	-1	1	5	50
-50	1.00	0.74	0.78	0.49	0.51	0.79
-5	0.74	1.00	0.83	0.45	0.42	0.63
-1	0.78	0.83	1.00	0.44	0.65	0.64
1	0.49	0.45	0.44	1.00	0.35	0.52
5	0.51	0.42	0.65	0.35	1.00	0.40
50	0.79	0.63	0.64	0.52	0.40	1.00

Table 6.: Correlation Matrix of Consistency Scores by Shift Offset for the 150-Nucleotide Model. The correlation matrix quantifies the pairwise similarity of consistency scores across different shift offsets.

Shift Offset	-50	-5	-1	1	5	50
-50	1.00	0.69	0.55	0.47	0.45	0.34
-5	0.69	1.00	0.57	0.54	0.48	0.26
-1	0.55	0.57	1.00	0.54	0.45	0.29
1	0.47	0.54	0.54	1.00	0.48	0.33
5	0.45	0.48	0.45	0.48	1.00	0.27
50	0.34	0.26	0.29	0.33	0.27	1.00

sensitivity to input shifts.

Shift Augmentation Experiments

To address the observed sensitivity, we augmented the training set by including shifted versions (by one nucleotide) of each sequence, exposing the models to both start- and end-padding scenarios. For the 150-nucleotide model, the increased data volume necessitated switching from 500 training steps to 6 epochs, calculated as:

$$e = \text{round}\left(\frac{s \cdot b}{N}\right) = \text{round}\left(\frac{500 \times 64}{5076}\right) = 6 \tag{5}$$

Epoch Calculation

where e is the number of epochs, s the number of steps, b the batch size, and N the number of fine-tuning sequences.

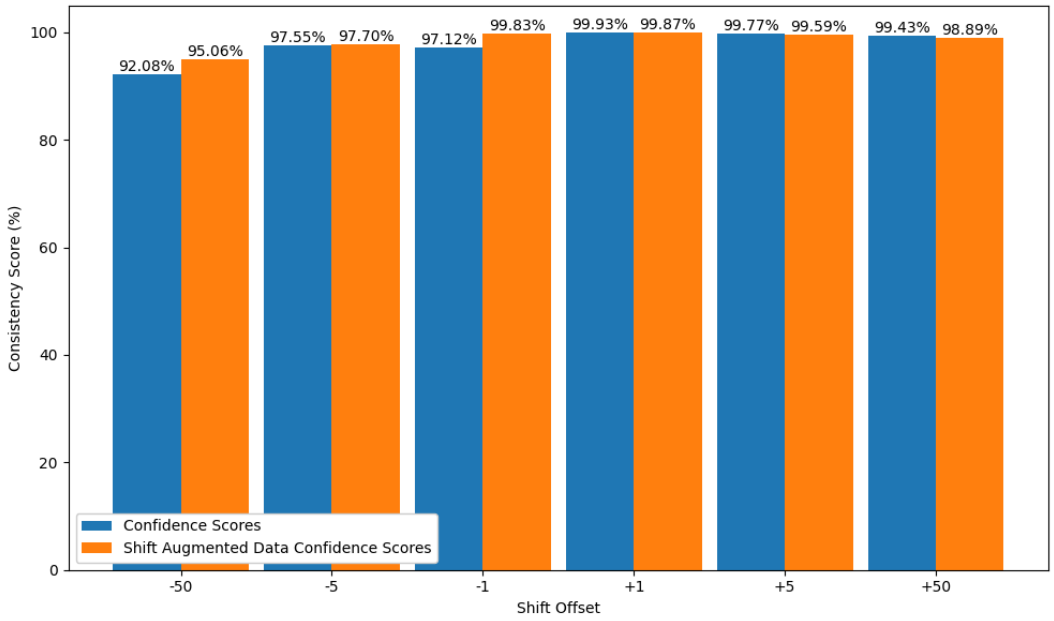

Figure 16.: Consistency Score of Shifted Sequences Shift Data Augmentation Comparison for 8,192 Target Length. Comparison of average consistency scores of the shifted sequences with and without shift data augmentation for the 8,192-nucleotide model.

With shift augmentation, the 8,192-nucleotide model (Figure 16) shows improved consistency for negative shifts, and the performance for the -1 nucleotide shift approaches that of positive shifts.

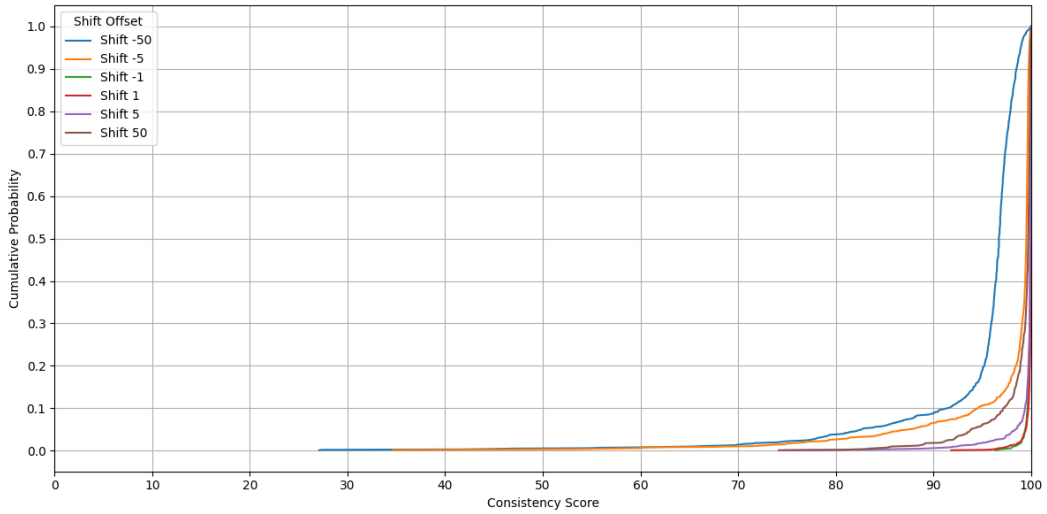

Figure 17.: CDF Plot of Consistency Scores by Shift Offset for the 8,192-Nucleotide Model with 1 Nucleotide Shift. Cumulative distribution function plot of the consistency scores for the 8,192-nucleotide model with 1 nucleotide shift data augmentation.

Figure 17 further demonstrates that, after augmentation, the CDFs for ± 1 nucleotide shifts closely align, indicating improved handling of both positive and negative offsets. This suggests that the original model's inconsistency was primarily due to lack of exposure to start-padding, rather than an inherent inability to generalize across shifts.

For the 150-nucleotide model (Figure 18), shift augmentation yields only marginal improvements for small positive offsets, while negative offsets perform slightly worse, suggesting limited benefit from this strategy.

To further probe this limitation, we extended shift augmentation to include offsets from 1 to 15 nucleotides in both directions for the 150-nucleotide model. Due to computational constraints and the strong performance gains already observed for the

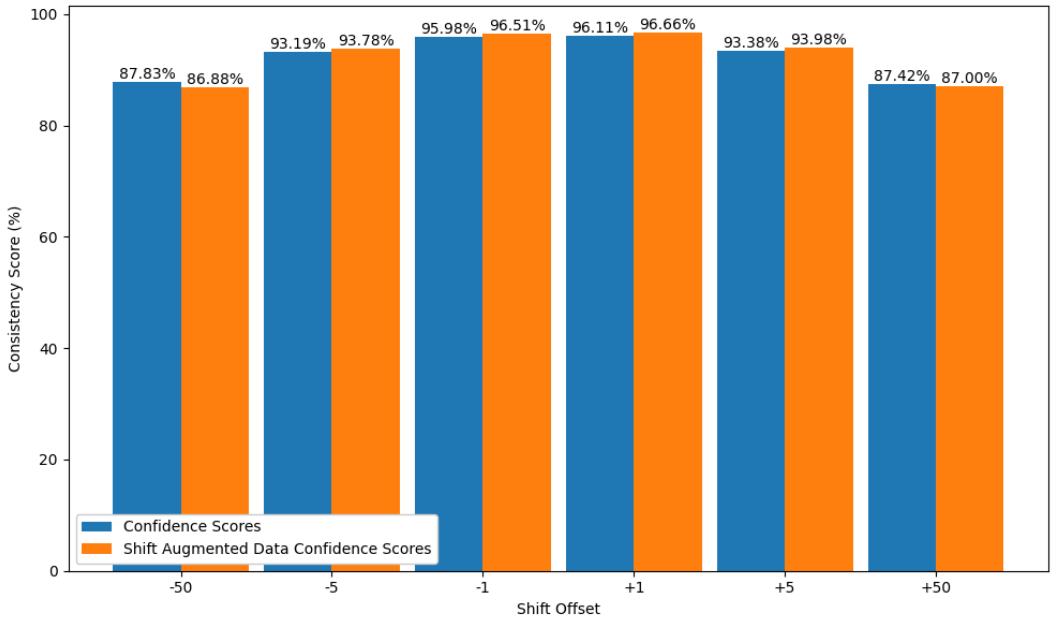

Figure 18.: Consistency Score of Shifted Sequences Shift Data Augmentation Comparison for 150 Target Length. Comparison of average consistency scores of the shifted sequences with and without shift data augmentation for the 150-nucleotide model.

8,192-nucleotide model, this was not pursued for the longer context model.

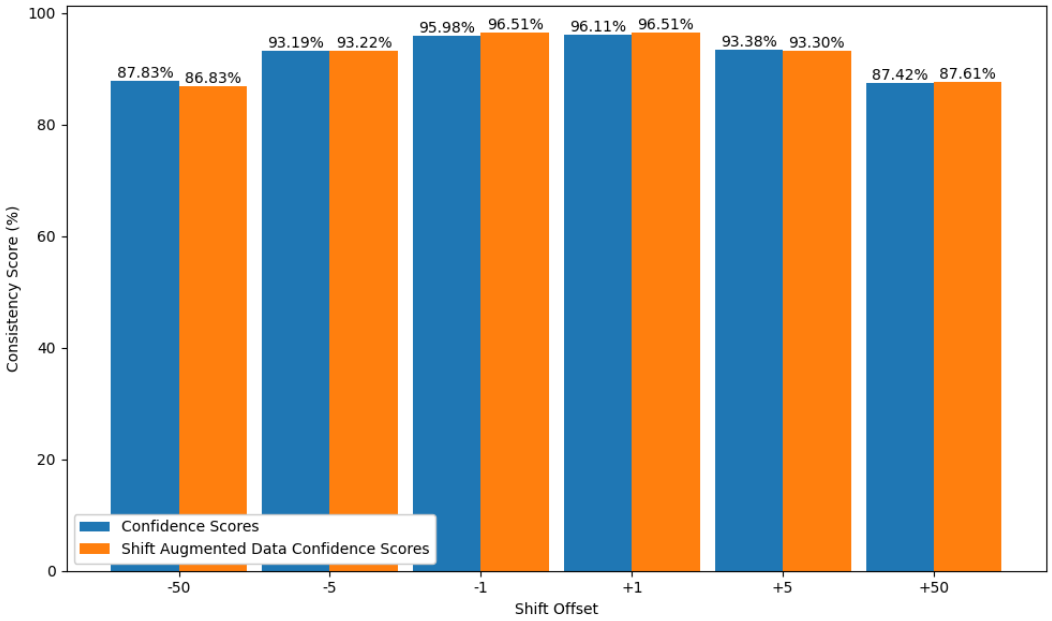

Figure 19.: Consistency Score of Shifted Sequences Extended Shift Data Augmentation Comparison for 150 Target Length. Comparison of average consistency scores of the shifted sequences with and without shift data augmentation (offsets 1–15 nucleotides) for the 150-nucleotide model.

As shown in Figure 19, even with extensive shift augmentation, the 150-nucleotide model remains inconsistent for both large and small shifts. This highlights a limitation of the approach: short-context models are inherently less robust to input perturbations.

5.7. Variable Short Target Lengths

Illumina sequencing platforms, the dominant technology in second-generation sequencing, typically produce short reads ranging from 75 to 300 base pairs [57, 8]. While previous experiments (see Section 5.4) established the model's effectiveness at a 150-nucleotide input length, it is important to assess whether the model can

generalize across the spectrum of Illumina read lengths. To this end, we evaluated model performance on three additional target lengths: 75, 225, and 300 nucleotides. These lengths were selected to reflect common Illumina read configurations and to demonstrate the model's adaptability to different sequencing setups. Due to computational constraints, only these three additional lengths were explored.

Table 7.: Number of Sequences in Each Dataset Split for Variable Short Target Lengths. The table enumerates, for each considered target length, the post-deduplication sequence count assigned to train, validation, and test sets, where N_{train}, $N_{validation}$, and N_{test} is the number are sequences in the training, validation and test splits, respectively, and N is the total number of sequences.

Target length	N_{train}	$N_{validation}$	N_{test}	N
75	3,483	490	1,000	4,973
225	3,531	495	1,000	5,026
300	3,526	495	1,000	5,021

For each target length, the dataset was partitioned into training, validation, and test sets following the same deduplication and splitting strategy described in Section 5.3. The number of sequences in each split is summarized in Table 7.

Hyperparameter Optimization

For each target length, hyperparameter optimization (HPO) was performed using the same stepwise manual protocol as for the primary models (Section 4.8). The hyperparameters tuned were learning rate (α), LoRA rank (r), dropout probability in the classification head, and AdamW weight decay. See Section A.3 for the HPO results for each target length.

Model Performance

The final test accuracies for the best models at each target length were 82.12%, 92.73%, and 93.61% for 75, 225, and 300 nucleotides, respectively.

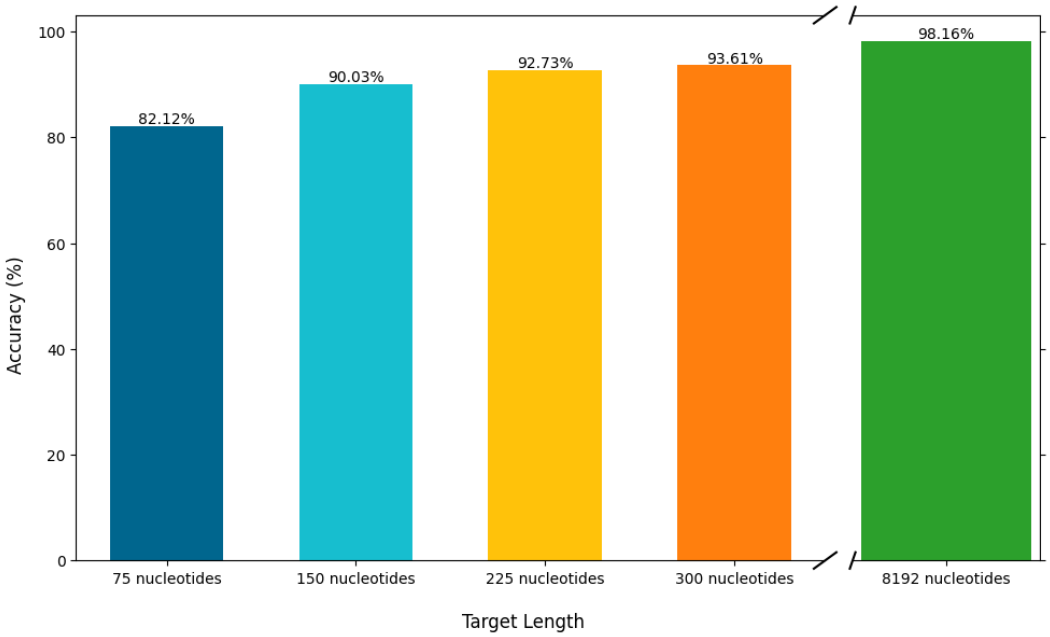

Figure 20.: Test Accuracy of Fine-Tuned Models. Test accuracy of fine-tuned models by target length.

These results demonstrate that the approach can robustly detect CRISPR arrays across a range of short read lengths. Notably, there is a clear trend of increasing accuracy with longer input sequences, as illustrated in Figure 20. This suggests that providing more sequence context improves the model's ability to detect CRISPR arrays and accurately classify their components.

5.8. Metagenomic Analysis

While long-read sequencing is developing, short-read sequencing technologies are still the main sequencing approaches [4]. Short reads from metagenomic analysis

can be assembled into longer sequences on which state-of-the-art CRISPR array detection methods can be used. However, our approach enables detection of CRISPR arrays in small reads directly. To show that the 150-nucleotide model is capable of detecting CRISPR arrays in small reads, we have used the model to detect the CRISPR arrays in the simulated dataset of Talibli et al. and compare it to their MCAAT approach [14].

We use the following filtering criteria for selecting spacers that we then compare to spacers in the database CRISPRCasDB, whose spacer and repeat sets were assembled using CRISPRFinder [62, 63]. We require spacers to have at least 23 consecutive nucleotides, allowing up to three nucleotides within this region to not be predicted as part of a spacer. We chose to focus on spacers due to their heterogeneity, compared to repeats, and to demonstrate that it does not rely on the repeat pattern as with other tools [7, 6, 14].

In order to quantize the findings we compute a precision metric by seeing how many found spacers are in the CRISPRCasDB, similarly we compute a recall metric by seeing how many of the spacers in the CRISPRCasDB are found in our dataset. We compare the spacers from the two sets by using BLAST with the minimum percent identity set to 95%, the minimum fraction of query aligned set to 90% and an E-value threshold of 10^{-5}, to only consider very significant matches [64].

The method achieves a precision of 24.31% and a recall of 49.12%. In contrast MCAAT achieves a precision of 65.25% and a recall of 70.89%. While the metrics of our approach are low in comparison to MCAAT, it is important to consider that this is the first method (to our knowledge) that can classify the sequences without doing any kind of assembly and processing each input sequence in isolation. Furthermore, our model is trained on CRISPRidentify bona fide data while the CRISPRCasDB utilized CRISPRFinder for its spacer set. Both approaches therefore inherit the biases of the algorithms they use.

To demonstrate that our approach discovers spacers that were not discovered by

MCAAT we combined the recall of both approaches and illustrated the overlap in Figure 21. Together they contain a recall of 83.45% with this demonstrates that our approach is able to recall 12.57% of the spacers that MCAAT cannot, as illustrated in Figure 21.

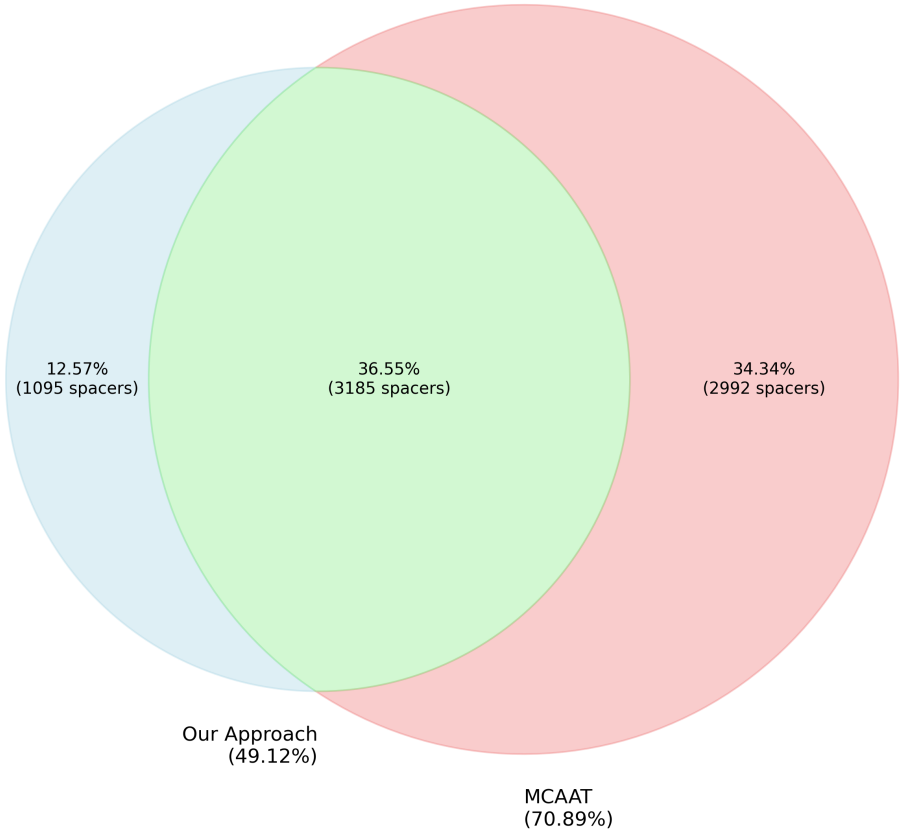

Figure 21.: Overlap of Covered Spacers. An Euler diagram displaying the recall of our approach and MCAAT and their overlap, with a combined recall of 83.45% (7272/8714 spacers).

Irregardless of the metrics there are several strong signals, which we exemplify in Tables 8 and 9. Additional strong signals are shown in Section A.4. The CRISPRCasFinder spacer-classified sequence GAAATACTGGAAATCAATGCGC-CTCGGTTTTCTAT is not classified as a spacer by our approach in Table 8 (see part 1/3), but is in Table 9 (see part 3/3), indicating that the model is not merely

memorising spacer representations.

Table 8.: Comparison of Array Detection Algorithms (AP022323.1_2083_12/1). Detected array regions by both our approach and CRISPRCasFinder are shown for the same continuous sequence, split across three table parts for readability. Nucleotides are colored: red for repeats, blue for spacers, and black for non-CRISPR array classified bases. CRISPRCasFinder used the full array and flanking context, while the foundation model input was a 150-nucleotide sequence.

Algorithm	Part	Subsequence
CRISPRCasFinder	1/3	AGATAAAAACGAAATACTGGAAATCAATGCGCCTCGGTTTTCTATGTTTC
Our Approach	1/3	AGATAAAAACGAAATACTGGAAATCAATGCGCCTCGGTTTTCTATGTTTC
CRISPRCasFinder	2/3	AATTCCTTATAGTTAAGATAAAAACCTTGTATAGCAATCTATTATGTTTA
Our Approach	2/3	AATTCCTTATAGTTAAGATAAAAACCTTGTATAGCAATCTATTATGTTTA
CRISPRCasFinder	3/3	CCGAGTTATTTTGTTTCAATTCCTTATAGTTAAGATAAAAACACGAAATA
Our Approach	3/3	CCGAGTTATTTTGTTTCAATTCCTTATAGTTAAGATAAAAACACGAAATA

While this approach is feasible, one must consider the high computational complexity of using a foundation model to run on all short reads versus Talibli et al.'s approach versus assembling these short reads into longer sequences that can be used with state-of-the-art methods, such as CRISPRidentify. Furthermore, as seen in Section 5.4, the full context window performs better at identifying CRISPR arrays, and as shown in Section 5.7, even for small target lengths, a longer context window size seems to indicate better detection performance.

Table 9.: Comparison of Array Detection Algorithms (AP022323.1_3391_20/1). Detected array regions by both our approach and CRISPRCasFinder are shown for the same continuous sequence, split across three table parts for readability. Nucleotides are colored: red for repeats, blue for spacers, and black for non-CRISPR array classified bases. CRISPRCasFinder used the full array and flanking context, while the foundation model input was a 150-nucleotide sequence.

Algorithm	Part	Subsequence
CRISPRCasFinder	1/3	GAGCCGAACGTTTCAATTCCTTATAGTTAAGATAAAAACATGAGGGAAAC
Our Approach	1/3	GAGCCGAACGTTTCAATTCCTTATAGTTAAGATAAAAACATGAGGGAAAC
CRISPRCasFinder	2/3	TAAACAAATGTATGAAGAAATCAAAAGTTTCAATTCCTTATAGTTAAGAT
Our Approach	2/3	TAAACAAATGTATGAAGAAATCAAAAGTTTCAATTCCTTATAGTTAAGAT
CRISPRCasFinder	3/3	AAAAACGAAATACTGGAAATCAATGCGCCTCGGTTTTCTATGTTTCAATT
Our Approach	3/3	AAAAACGAAATACTGGAAATCAATGCGCCTCGGTTTTCTATGTTTCAATT

5.9. Degenerated Repeat Detection

Detecting degenerated CRISPR repeats poses a significant challenge for current state-of-the-art tools such as CRISPRidentify and CRISPRCasFinder, which rely on sequence similarity among repeats [7, 6]. If repeat sequences have diverged, these tools can fail to delineate accurate CRISPR array boundaries. Notably, in our 8,192-nucleotide model, the most prevalent and third most prevalent error types involve misclassifying non-array nucleotides at the sequence termini (see Figure 10), suggesting that the model may identify elements associated with CRISPR arrays that are missed by existing tools such as CRISPRidentify.

To systematically explore the model's ability to detect degenerated repeats, we analyzed the test set and extracted candidate regions labeled as *repeat* by the model but not annotated as such by CRISPRidentify. Only candidates comprising at least 24 nucleotides were considered, corresponding to the lower end of the range of 95% of repeats, reported by Biswas et al. [65]. To account for minor inconsistencies in labeling, we employed a tolerance window of 3 nucleotides, allowing for short

interruptions within predicted repeat regions.

To assess whether the retrieved candidates correspond to actual degenerate variants of known repeats, each was aligned against the consensus repeat of the respective array using the FASTA algorithm with an E-value threshold of 0.01 [61]. The consensus repeats were extracted using CRISPRidentify.

In total, 71 degenerate repeat candidates were detected, of which 65 (92.5%) produced significant FASTA alignments to their respective array's consensus repeat. These candidates were distributed across 33 of the 1,016 test sequences (3.25% prevalence). This demonstrates that the model is capable of identifying repeat-like signals outside the annotated array bounds, likely corresponding to biologically relevant but sequence-diverged repeat units.

Table 10 provides an example of such a candidate. The repeat exhibits multiple mismatches and deletions relative to the consensus, potentially explaining why it was excluded from the original annotation.

Table 10.: CRISPR Array with Degenerated Repeat Candidate (APTL01000115:16,220–16,36). An example of a CRISPR array detected in accession APTL01000115 (positions 16,220–16,336), illustrating repeat and spacer organization. Repeats are shown on the left and spacers or intervening sequences on the right. Differences from the consensus repeat (as defined by CRISPRidentify) are indicated in red to mark a candidate degenerated repeat. Only nucleotide positions that deviate from the consensus are displayed for clarity.

Repeats	Spacers / Sequences in between
GTTCTAAGGTCTA.....-	AATCCATCCGCACG(...)ACAGACCTA
.....	GTCTTTTATAATCTT(...)CTAAGGTCTA
.....	TTTAGAAGTTGC(...)AACAAGCCG
.....	
GATTCCCGCCTGCGCGGGAATGACGG	

In 16 of the 33 sequences with predicted degenerate repeat candidates, the surrounding region contained unknown nucleotides (ambiguous base 'N'). Table 11 shows such a

case, demonstrating that the model can still detect likely degenerated repeats even within low-quality or masked genomic regions.

Table 11.: CRISPR Array with Degenerated Repeat Candidate (GL872110:347,341–347,672). An example of a CRISPR array detected in accession GL872110 (positions 347,341–347,672), illustrating repeat and spacer organization. Repeats are shown on the left and spacers or intervening sequences on the right. Differences from the consensus repeat (as defined by CRISPRidentify) are indicated in red to mark a candidate degenerated repeat. Only nucleotide positions that deviate from the consensus are displayed for clarity.

Repeats	Spacers / Sequences in between
GT..... CC	GCGTCCGTCAAAGAGGAAGCCAAAGCCATGCA
GT.....G.....AC	CGNNNN(...)NTAGCAGCACTCCACCAGCTCGAGCAGCAGGT
....	ACCGCAAAAAAATAATATCCGGCAGTCTGTACGGTAGT
....	CCGTCGGTTTGCGCGTATTGTTGTTCGGGGTCTATGT
....	CCGTAGGTAAATCACAGCTATTTGATAAGGGCGTGTGT
....	CCGAATAGCAATAGTCCATAGATTTGCGAAAACAGGT
....	CCGGAGCCTGACGAGACTACTGAGGCCGTTCTGTCGA
....	
GTTCCCCGCGCCAGC GGGGATAAA	

Further examples of predicted degenerate repeat candidates can be found in Section A.5.

These findings highlight the model's potential as a complementary tool for identifying unconventional or degenerated repeats missed by rule-based detectors. For future work, a dedicated post-processing pipeline could be developed to systematically validate, cluster, and refine degenerate repeat predictions. Such a pipeline might incorporate alignment quality metrics, structural context, and secondary filters based on CRISPR array grammar (e.g., repeat-consensus adjacency or terminal symmetry).

6. Conclusion

6.1. Summary

This thesis demonstrates the application of deep learning foundation models for CRISPR array detection, addressing limitations of existing computational approaches. By adapting the Evo foundation model through parameter-efficient fine-tuning techniques, we developed a novel method utilizing foundation models that overcomes two key challenges: short read constraints and degenerate sensitivity.

Key Achievements:

- **Foundation Model Adaptation:** We successfully fine-tuned the Evo model using LoRA for per-nucleotide classification of CRISPR array components. The 8,192-nucleotide model achieved 98.16% test accuracy, while the 150-nucleotide model optimized for Illumina reads achieved 90.03% accuracy.

- **Metagenomic Analysis Capabilities:** The 150-nucleotide model demonstrated the ability to analyse short reads directly in isolation. Achieving a spacer precision of 24.31% and recall of 49.12% on simulated metagenomic data. 12.57% of detected spacers where not found by MCAAT in the same reads, demonstrating how the model can be used to complement existing tools.

- **Degenerate Repeat Detection:** The 8,192-nucleotide model demonstrated the capability to identify 71 degenerate repeat candidates, of which 92.5%

aligned significantly to consensus repeats, showcasing its ability to detect variant repeats missed by conventional similarity-based approaches.

6.2. Limitations

Despite the promising results, several limitations constrain the current approach:

- **Computational Complexity:** The foundation model approach requires substantial computational resources compared to traditional methods. While LoRA reduces training parameters to 0.63% of the full Evo model, inference still demands significant GPU memory and processing time.

- **Training Data Dependency:** Model performance is inherently limited by the quality and comprehensiveness of the bona-fide CRISPRidentify-labeled training data. While our approach can detect degenerate repeats missed by the labeling tool, it may also inherit biases present in the training annotations.

- **Input Sequence Sensitivity:** The models exhibit sensitivity to input sequence shifts, particularly the 150-nucleotide model, which showed reduced consistency scores for shifts as small as ± 5 nucleotides. While shift data augmentation improved robustness for the 8,192 model, the short target length model remained inherently less stable.

- **Context Window Constraints:** The 8,192 context window limitation of the Evo model restricts analysis of exceptionally long CRISPR arrays, though this affects only a small fraction of arrays.

6.3. Future Work

Several promising directions emerge from this research:

- **Next-Generation Foundation Models:** Evaluate performance using newer foundation models such as Evo1.5 and Evo2, which may offer improved sequence understanding and longer context windows given the same amount of compute [66, 67]. Additionally, explore intermediate layer representations rather than final output embeddings to capture hierarchical sequence features.

- **Bidirectional Sequence Processing:** Implement explicit reverse complement sequence processing during training and inference to improve model robustness to sequence orientation.

- **Knowledge Distillation:** Train a smaller model utilizing the foundation model to achieve similar performance with a smaller model.

- **Specialized Architecture Development:** If computational resources permit, develop BERT-like bidirectional models pretrained on an OpenGenome dataset to capture array-specific patterns more effectively.

- **Computational Cost Optimization:** Conduct comprehensive analysis comparing the computational costs of direct short-read analysis versus long-read analysis versus assembly-based pipelines followed by traditional CRISPR detection methods. This analysis should inform optimal processing strategies for different sequencing scenarios.

- **Extended Validation:** Evaluate model performance on diverse prokaryotic taxa and environmental samples to assess generalizability beyond the training distribution. Particular emphasis should be placed on Type IV CRISPR systems known for their degenerate repeat patterns [68].

- **Real-World Deployment:** Transition from proof of concept to production-ready tools, including optimization for different hardware configurations and development of user-friendly interfaces for the broader research community.

Outlook

This work establishes foundation models as viable tools for CRISPR array detection and opens new avenues for applying deep learning to prokaryotic immunity research. The demonstrated ability to detect degenerate repeats and process short reads represents a significant advancement in computational genomics, with implications extending beyond CRISPR research to other repetitive genomic element detection tasks.

7. Acknowledgments

First and foremost, I would like to thank my advisors, Alexander Mitrofanov and Ryan Köksal, for their mentorship, insightful feedback, and continuous support throughout this research journey. Their expertise in CRISPR and deep learning, respectively, was instrumental in shaping this work. I am particularly grateful to Alexander Mitrofanov for providing data and assisting with the verification of experimental results.

I would also like to thank Prof. Dr. Rolf Backofen and Dr. André Biedenkapp for serving as examiners.

Special appreciation goes to Dr. Michael Uhl for his suggestions on zero-shot comparison and reverse complement potential, and to Prof. Dr. Omer Alkhnbashi for supervising and guiding the project's direction.

I gratefully acknowledge the support provided by the state of Baden-Württemberg through bwHPC and by the German Research Foundation (DFG) under grant INST 35/1597-1 FUGG.

I would also like to thank my brother, Jascha Laurits Maximilian Schäfer, and Erich Russell Choudhury for their careful proofreading and valuable suggestions, which significantly improved the clarity of this thesis.

Finally, my heartfelt thanks go to my family and Violetta Diana for their unwavering encouragement and support during the intensive writing period of this thesis.

A. Supplementary Materials

This appendix contains supplementary figures, tables, and experimental results that support the findings presented in the main body of the thesis. The materials are organized into distinct sections to facilitate navigation and reference. While this supplementary information is not essential for understanding the core contributions of the work, it provides additional context and detailed examples that may be of interest to readers seeking a more comprehensive view of the experimental methodology and results.

The appendix is structured as follows: Section A.1 presents additional examples of zero-shot next nucleotide prediction to demonstrate the Evo model's behavior across different CRISPR arrays. Section A.2 provides supplementary zero-shot classification examples showing the zero-shot classification model's performance on different CRISPR arrays. Section A.3 includes the hyperparameter optimization results for variable short target lengths. Section A.4 shows additional strong signals of metagenomic analysis classification and Section A.5 presents further examples of degenerated repeat detection capabilities.

A.1. Additional Zero-Shot Next Nucleotide Prediction Examples

The following figures provide additional examples of the pretrained Evo model's next nucleotide prediction capabilities on CRISPR arrays, supplementing the analysis

presented in Section 5.1.

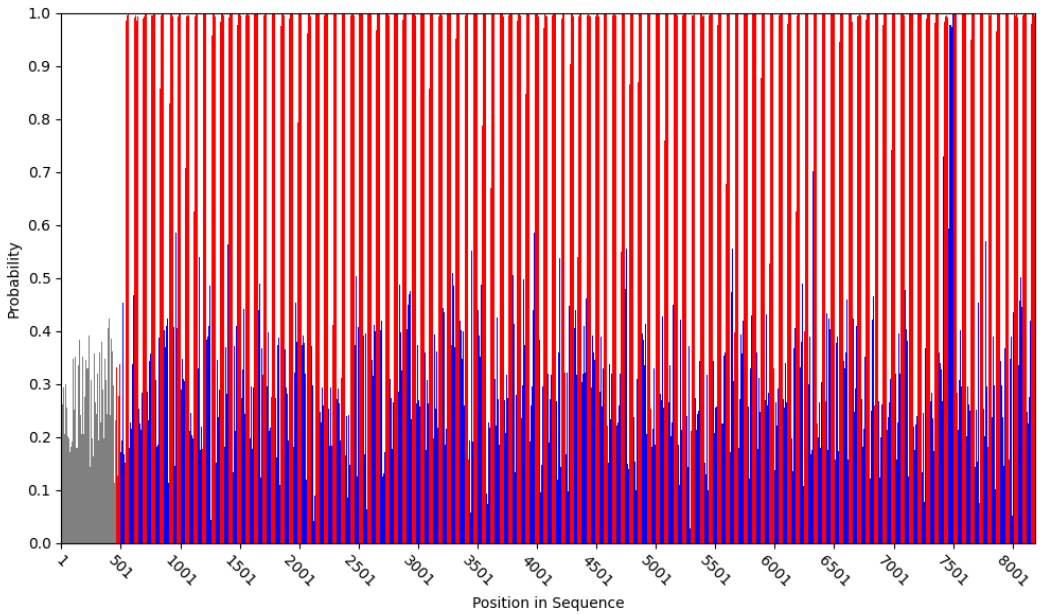

Figure 22.: First Additional Example of Next Nucleotide Prediction. Next nucleotide prediction probabilities for a randomly selected CRISPR array from the accession number CP027433 (positions 1417640–1425518), colored by CRISPRidentify-labeled nucleotide class (repeat is red, spacer is blue, and non-array is gray).

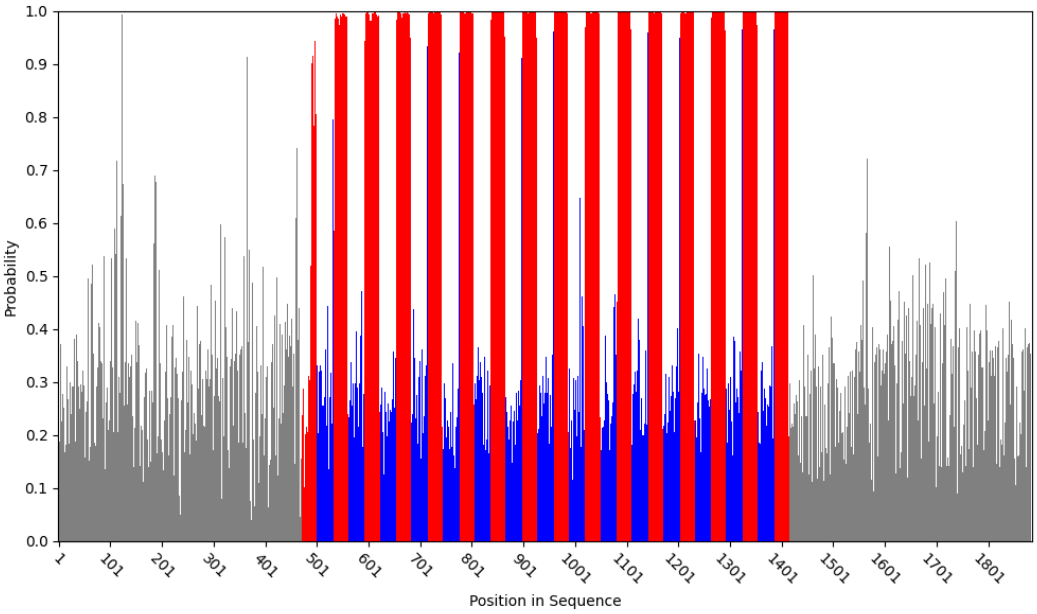

Figure 23.: Second Additional Example of Next Nucleotide Prediction. Next nucleotide prediction probabilities for a randomly selected CRISPR array from the accession number APZF01000100 (positions 167361–168247), colored by CRISPRidentify-labeled nucleotide class (repeat is red, spacer is blue, and non-array is gray).

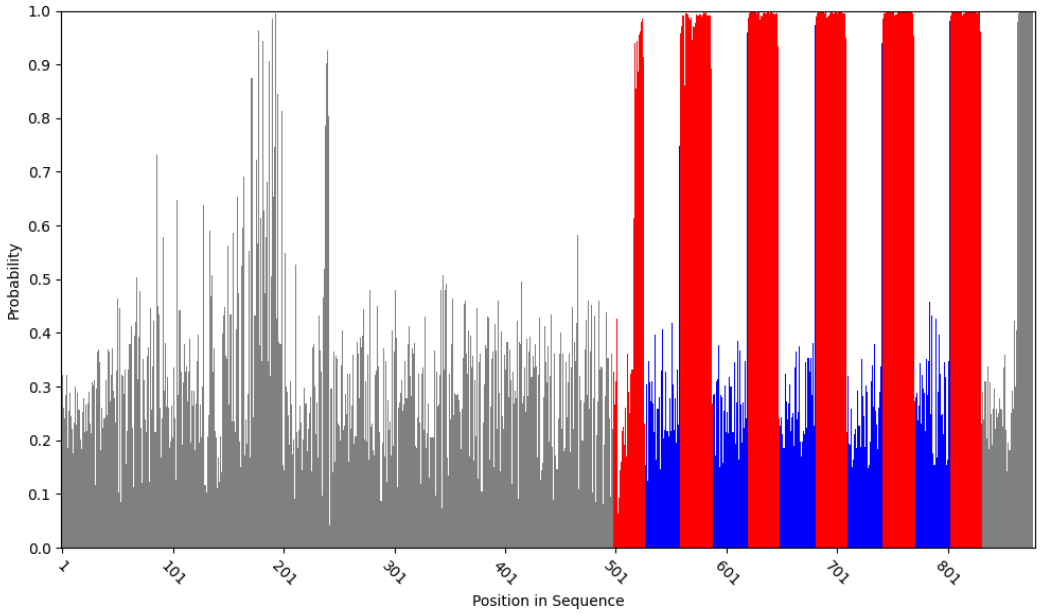

Figure 24.: Third Additional Example of Next Nucleotide Prediction. Next nucleotide prediction probabilities for a randomly selected CRISPR array from the accession number AETS01000059 (positions 44789–45123), colored by CRISPRidentify-labeled nucleotide class (repeat is red, spacer is blue, and non-array is gray).

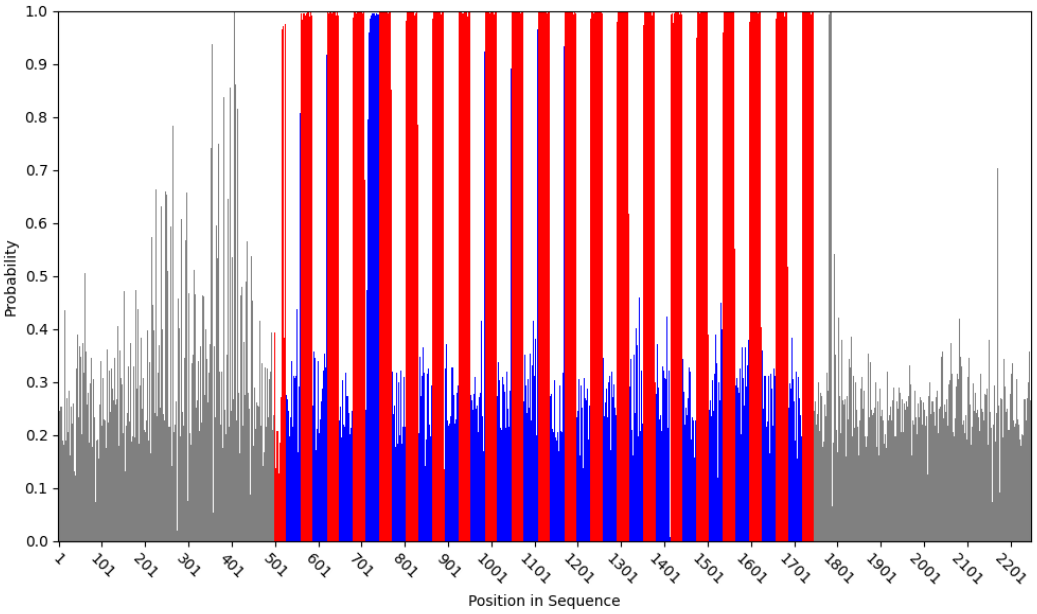

Figure 25.: Fourth Additional Example of Next Nucleotide Prediction. Next nucleotide prediction probabilities for a randomly selected CRISPR array from the accession number CP000423 (positions 356874–358123), colored by CRISPRidentify-labeled nucleotide class (repeat is red, spacer is blue, and non-array is gray).

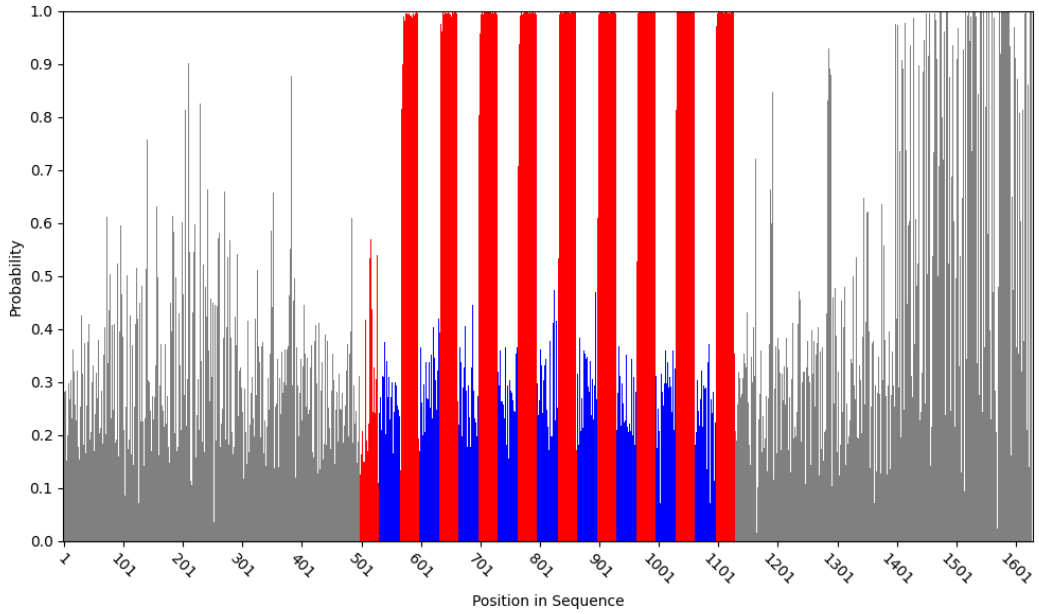

Figure 26.: Fifth Additional Example of Next Nucleotide Prediction. Next nucleotide prediction probabilities for a randomly selected CRISPR array from the accession number CP006953 (positions 733510–734140), colored by CRISPRidentify-labeled nucleotide class (repeat is red, spacer is blue, and non-array is gray).

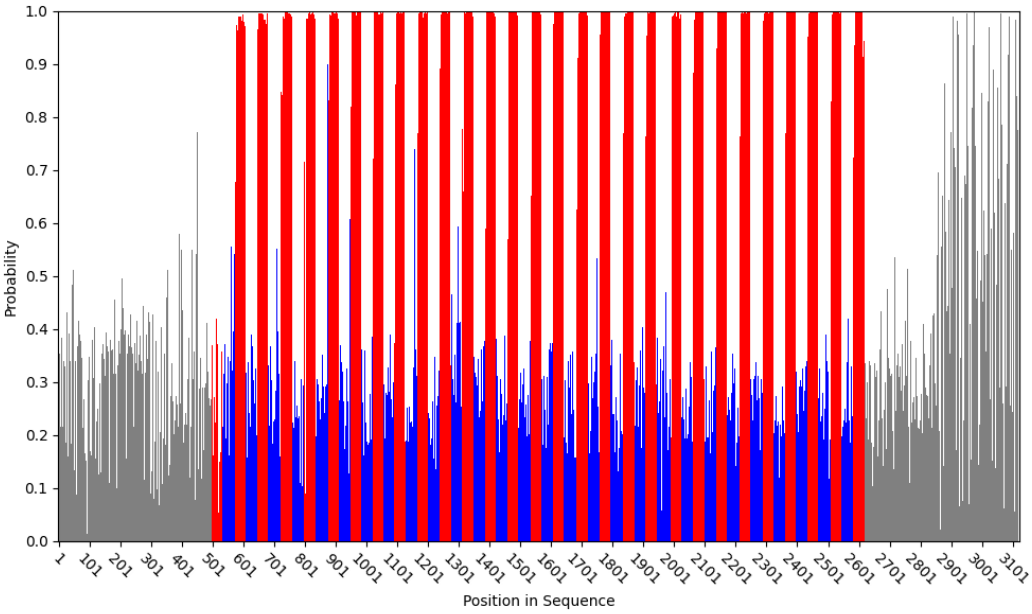

Figure 27.: Sixth Additional Example of Next Nucleotide Prediction. Next nucleotide prediction probabilities for a randomly selected CRISPR array from the accession number CP091284 (positions 64251–66372), colored by CRISPRidentify-labeled nucleotide class (repeat is red, spacer is blue, and non-array is gray).

A.2. Additional Zero-Shot Classification Examples

This section presents supplementary examples of zero-shot classification performance, extending the analysis from Section 5.2.

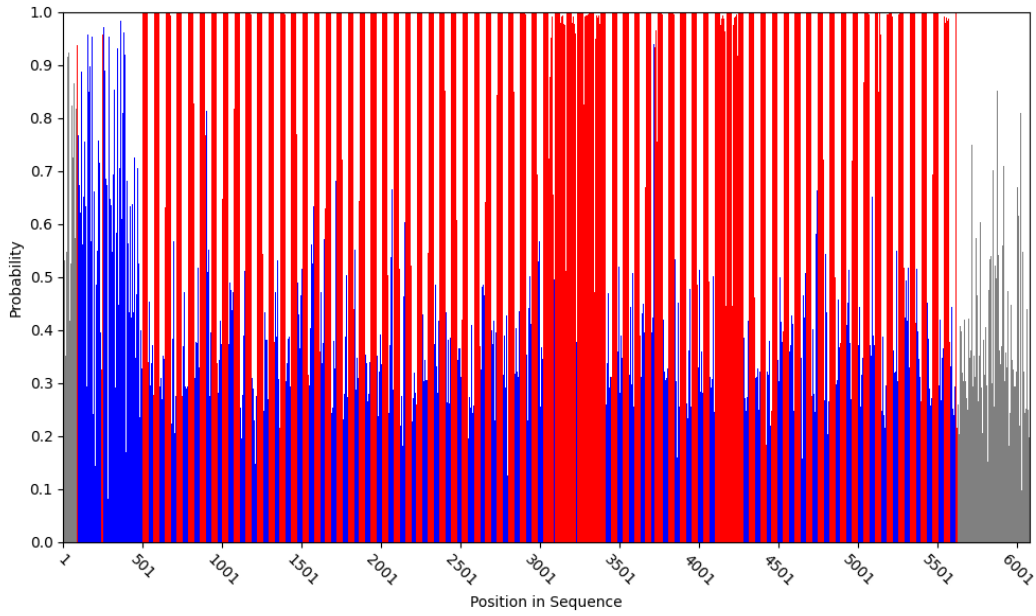

Figure 28.: First Additional Example of Zero-Shot Classication. Zero-shot classification for a random CRISPR array and flanking regions from the accession number ACVN02000303 (positions 16038–21118). The plot visualizes the predicted nucleotide class for each position, computed utilizing the maximum probability from the original and reverse complement sequences: repeat (red), spacer (blue), and non-array (gray).

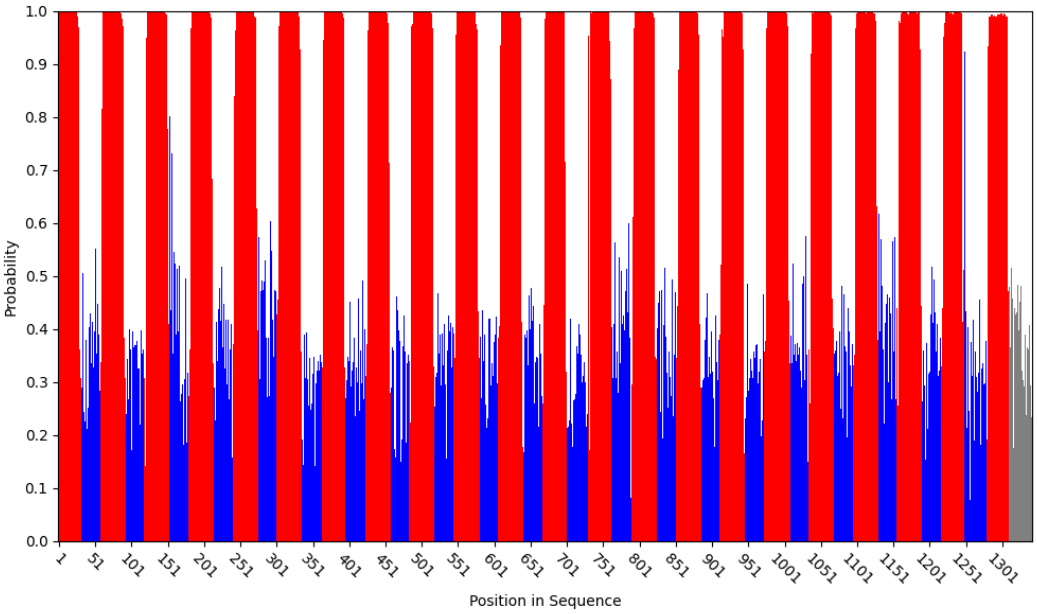

Figure 29.: Second Additional Example of Zero-Shot Classication. Zero-shot classification for a random CRISPR array and flanking regions from the accession number BAHD01000148 (positions 1–1311). The plot visualizes the predicted nucleotide class for each position, computed utilizing the maximum probability from the original and reverse complement sequences: repeat (red), spacer (blue), and non-array (gray).

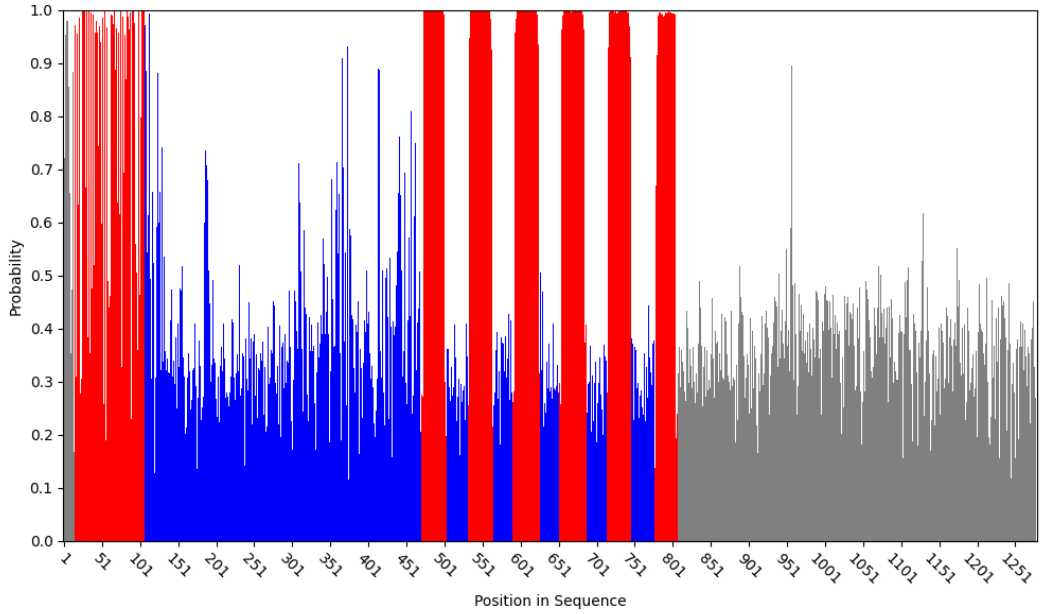

Figure 30.: Third Additional Example of Zero-Shot Classication. Zero-shot classification for a random CRISPR array and flanking regions from the accession number CP021532 (positions 4014916–4015195). The plot visualizes the predicted nucleotide class for each position, computed utilizing the maximum probability from the original and reverse complement sequences: repeat (red), spacer (blue), and non-array (gray).

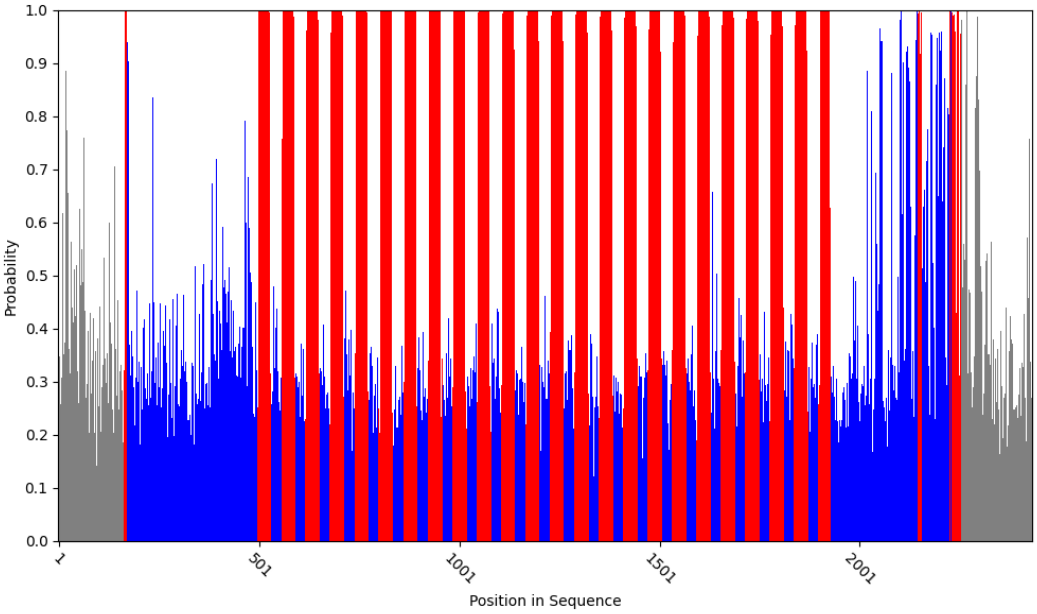

Figure 31.: Fourth Additional Example of Zero-Shot Classication. Zero-shot classification for a random CRISPR array and flanking regions from the accession number LR134190 (positions 2042753–2044184). The plot visualizes the predicted nucleotide class for each position, computed utilizing the maximum probability from the original and reverse complement sequences: repeat (red), spacer (blue), and non-array (gray).

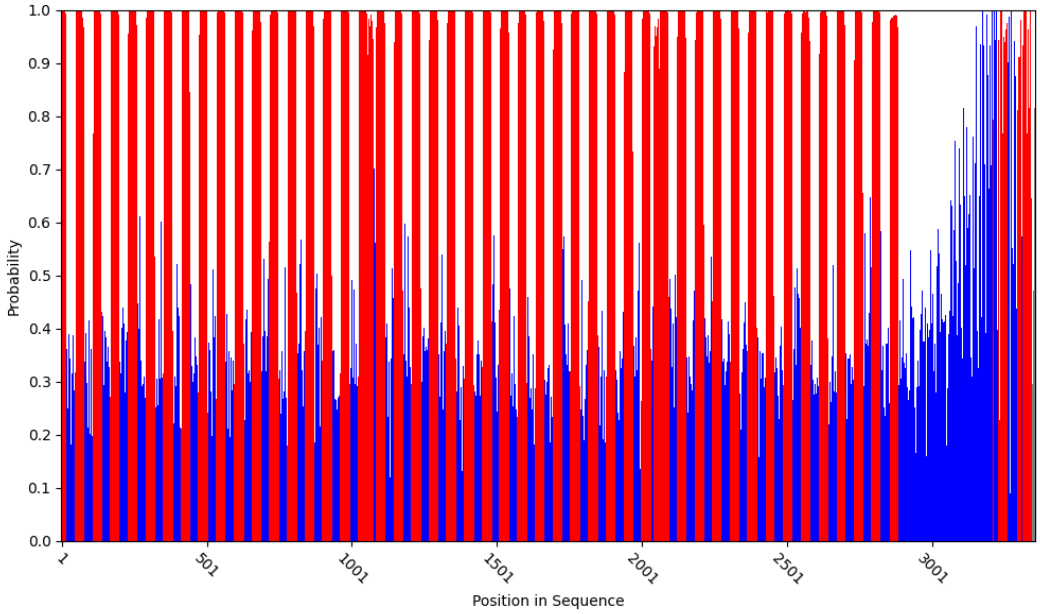

Figure 32.: Fifth Additional Example of Zero-Shot Classication. Zero-shot classification for a random CRISPR array and flanking regions from the accession number LRDG01000013 (positions 24897–27675). The plot visualizes the predicted nucleotide class for each position, computed utilizing the maximum probability from the original and reverse complement sequences: repeat (red), spacer (blue), and non-array (gray).

A.3. Hyperparameter Optimization Results for Variable Short Target Lengths

This section provides the hyperparameter optimization (HPO) results for the variable short target length experiments described in Section 5.7.

Table 12.: Hyper-Parameter Optimization (HPO) Results for 75 Target Length Model. Hyper-Parameter Optimization (HPO) results for models training for the 75 target length, where Loss is the training loss, α is the learning rate, r is the LoRA rank, Accuracy is the validation accuracy, Dropout is the classification head dropout percentage, and Weight Decay is the AdamW weight decay. The model with the highest validation accuracy was selected as the final model, highlighted in bold text.

Step	α	r	Dropout	Weight Decay	Loss	Accuracy
1. α	10^{-5}	8	0%	0	1.0481	47.42%
	10^{-4}	8	0%	0	0.4316	80.44%
	10^{-3}	8	0%	0	0.3374	82.99%
	10^{-2}	8	0%	0	1.0652	46.50%
2: r	10^{-3}	4	0%	0	0.1080	82.80%
	10^{-3}	16	0%	0	0.2316	82.33%
	10^{-3}	24	0%	0	0.1067	81.98%
	10^{-3}	32	0%	0	0.1672	82.42%
	10^{-3}	64	0%	0	0.1259	82.20%
3. Dropout	10^{-3}	24	5%	0	0.1550	82.08%
	10^{-3}	24	10%	0	0.2668	82.50%
	10^{-3}	**24**	**15%**	**0**	**0.2293**	**82.84%**
	10^{-3}	24	20%	0	0.3118	82.49%
	10^{-3}	24	25%	0	0.1438	81.40%
4. Weight Decay	10^{-3}	24	15%	10^{-3}	0.3182	82.82%
	10^{-3}	24	15%	10^{-2}	0.2831	82.72%
	10^{-3}	24	15%	10^{-1}	0.1056	82.12%

Table 13.: Hyper-Parameter Optimization (HPO) Results for 225 Target Length Model. Hyper-Parameter Optimization (HPO) results for models training for the 225 target length, where Loss is the training loss, α is the learning rate, r is the LoRA rank, Accuracy is the validation accuracy, Dropout is the classification head dropout percentage, and Weight Decay is the AdamW weight decay. The model with the highest validation accuracy was selected as the final model, highlighted in bold text.

Step	α	r	Dropout	Weight Decay	Loss	Accuracy
1. α	10^{-5}	8	0%	0	1.0701	50.80%
	10^{-4}	8	0%	0	0.3086	87.26%
	10^{-3}	8	0%	0	0.1253	91.96%
	10^{-2}	8	0%	0	1.0733	41.23%
2: r	10^{-3}	4	0%	0	0.1520	91.85%
	10^{-3}	16	0%	0	0.1520	91.82%
	10^{-3}	24	0%	0	0.1532	92.12%
	10^{-3}	32	0%	0	0.1190	92.10%
	10^{-3}	64	0%	0	0.1518	92.09%
3. Dropout	$\mathbf{10^{-3}}$	**32**	**5%**	**0**	**0.1327**	**92.44%**
	10^{-3}	32	10%	0	0.1504	91.95%
	10^{-3}	32	15%	0	0.1191	91.87%
	10^{-3}	32	20%	0	0.1264	92.06%
	10^{-3}	32	25%	0	0.1782	91.75%
4. Weight Decay	10^{-3}	32	5%	10^{-3}	0.1100	92.01%
	10^{-3}	32	5%	10^{-2}	0.1141	92.01%
	10^{-3}	32	5%	10^{-1}	0.1081	92.08%

Table 14.: Hyper-Parameter Optimization Results for 300 Target Length Model. HPO results for models training for the 300 target length, where Loss is the training loss, α is the learning rate, r is the LoRA rank, Accuracy is the validation accuracy, Dropout is the classification head dropout percentage, and Weight Decay is the AdamW weight decay,. The model with the highest validation accuracy was selected as the final model, highlighted in bold text.

Step	α	r	Dropout	Weight Decay	Loss	Accuracy
1. α	10^{-5}	8	0%	0	1.0701	43.03%
	10^{-4}	8	0%	0	0.2095	91.97%
	10^{-3}	8	0%	0	0.1216	93.33%
	10^{-2}	8	0%	0	1.0794	40.96%
2: r	10^{-3}	4	0%	0	0.1492	93.13%
	10^{-3}	16	0%	0	0.1146	93.08%
	10^{-3}	24	0%	0	0.1193	93.35%
	10^{-3}	32	0%	0	0.1098	93.16%
	10^{-3}	64	0%	0	0.1225	93.30%
3. Dropout	$\mathbf{10^{-3}}$	**32**	**5%**	**0**	**0.1052**	**93.41%**
	10^{-3}	32	10%	0	0.1124	93.34%
	10^{-3}	32	15%	0	0.0916	93.04%
	10^{-3}	32	20%	0	0.1027	93.36%
	10^{-3}	32	25%	0	0.1140	93.22%
4. Weight Decay	10^{-3}	32	5%	10^{-3}	0.1082	93.16%
	10^{-3}	32	5%	10^{-2}	0.1067	93.13%
	10^{-3}	32	5%	10^{-1}	0.1049	93.30%

A.4. Additional Metagenomic Analysis Strong Signals

This section presents supplementary strong signals of CRISPR array detection in metagenomic analysis data, extending the analysis from Section 5.8.

Table 15.: First Additional Comparison of CRISPR Array Detection Algorithms (AP022323.1_622_1/1). Detected array regions by both our approach and CRISPRCasFinder are shown for the same continuous sequence, split across three table parts for readability. Nucleotides are colored: red for repeats, blue for spacers, and black for non-CRISPR array classified bases. CRISPRCasFinder used the full array and flanking context, while the foundation model input was a 150-nucleotide sequence.

Algorithm	Part	Subsequence
CRISPRCasFinder	1/3	AAACGAAATACTGGAAATCAATGCGCCTCGGTTTTCTATGTTTCAATTCC
Our Approach	1/3	AAACGAAATACTGGAAATCAATGCGCCTCGGTTTTCTATGTTTCAATTCC
CRISPRCasFinder	2/3	TTATAGTTAAGATAAAAACCTTGTATAGCAATCTATTATGTTTACCGAGT
Our Approach	2/3	TTATAGTTAAGATAAAAACCTTGTATAGCAATCTATTATGTTTACCGAGT
CRISPRCasFinder	3/3	TATTTTGTTTCAGTTCCTTATAGTTAAGATAAAAACACGAAATAAACATG
Our Approach	3/3	TATTTTGTTTCAGTTCCTTATAGTTAAGATAAAAACACGAAATAAACATG

Table 16.: Second Additional Comparison of CRISPR Array Detection Algorithms (CP000679.1_3469_12/1). Detected array regions by both our approach and CRISPRCasFinder are shown for the same continuous sequence, split across three table parts for readability. Nucleotides are colored: red for repeats, blue for spacers, and black for non-CRISPR array classified bases. CRISPRCasFinder used the full array and flanking context, while the foundation model input was a 150-nucleotide sequence.

Algorithm	Part	Subsequence
CRISPRCasFinder	1/3	ATATCAAGTTTCAATTCCCAAAGGGGAGGCTACAAACTATCAAAGCAAAT
Our Approach	1/3	ATATCAAGTTTCAATTCCCAAAGGGGAGGCTACAAACTATCAAAGCAAAT
CRISPRCasFinder	2/3	GCAAATATGGGTGATGAGCAAATTTTCAATCCCCAAAGGGAAGGCTACAA
Our Approach	2/3	GCAAATATGGGTGATGAGCAAATTTTCAATCCCCAAAGGGAAGGCTACAA
CRISPRCasFinder	3/3	ACTTGCTGTGTCAAACCACGTTCTTTTCTCAAGTCTTTGTATCAATCCCC
Our Approach	3/3	ACTTGCTGTGTCAAACCACGTTCTTTTCTCAAGTCTTTGTATCAATCCCC

A.5. Additional Degenerated Repeat Examples

This section presents supplementary examples of degenerated repeat detection, extending the analysis from Section 5.9.

Table 17.: First Additional Example of CRISPR Array Annotation with Degenerated Repeat Candidate. An example of a CRISPR array detected in accession CASA01000002 (positions 108,834–109,415), illustrating repeat and spacer organization. Repeats are shown on the left and spacers or intervening sequences on the right. Differences from the consensus repeat (as defined by CRISPRidentify) are indicated in red to mark a candidate degenerated repeat. Only nucleotide positions that deviate from the consensus are displayed for clarity.

Repeats	Spacers / Sequences in between
---.....	GTCATTTTCTTGTTAATGGCGCTTGCATTAAC
.....	GTAGCCTTCAGACACTTTAGCCAGGTTCCCTA
.....--	NNNNNNNNNNNNNNN(...)CTTGCATTAAC
.....	CGCACCTGCCTGGACATGACCCTGCCGGAGCT
.....	GTCATTTTCTTGTTAATGGCGCTTGCATTAAC
.....	CTGTGTGGTCTGTGCATAACGGTGTAACAGAG
.....	CTGGGAAGATTGGGCGCTTTCAATATCTTCCA
.....	AAAAATAGTCCTGAACGATAGCCCGCGCGGTC
.....	CAGGAGTGGCTGGAGACTGTCGTCAAGCCTGA
.....	ATTTTAACGGCAAGGTCGACAGGAGATACCAC
.....	TTTTATAACGACACTAAAACCCGCCAGTTATA
.....	AATCCCGATGATCATGAATATTTCTGGCGCCA
.....	GATCATCACCCAGGCATTTTCTGGAATATGAA
.....	

GTGTTCCCCGCGCCAGCGGGGATAAACCG

Table 18.: Second Additional Example of CRISPR Array Annotation with Degenerated Repeat Candidate. An example of a CRISPR array detected in accession CP034192 (positions 1,672,883–1,674,884), illustrating repeat and spacer organization. Repeats are shown on the left and spacers or intervening sequences on the right. Differences from the consensus repeat (as defined by CRISPRidentify) are indicated in red to mark a candidate degenerated repeat. Only nucleotide positions that deviate from the consensus are displayed for clarity.

Repeats	Spacers / Sequences in between
.....	ACTGCCGTTGCTCGTCCAGCCAGCGCGCGAAGCGG
.....	CCGAAATCCCGGCCATAGCATATCGACCGGCGCGTC
.....	CCCACCACATCTGCCGTGGGGCGCGTTTTGAAAA
.....	TCCGCGTCCATGCTGCCGGTGGCCTTGTGGTCGGCC
.....	ATCTTGGTATCAGATTAAAGCGCTCCCCCCTTT
.....	GGCCCGAGGCCTGCACAGCGTGGACGTGGAAATCT
.....	ATGCAGGTGTGGGGGTCGTATGATACAATATTCG
.....	CTTAAGGCGCGGCAGGCGGCGGCAACCGAAATCA
.....	AGCAGGGAGTAGGCGGTATTGTAGCGTGTCCCGTAT
.....	AGCTGGCGGGGGCTGCCACCAACCGTGGATGAAC
.....	ACGTTGATATCAGATTAAGGCGCGCTCCCCACTTT
.....	ATCTACAACGGCACCGGTTACTATTCGCCCTTT
.....	CGGAGCTCATCGCCAACATCGCCATCAATCGCGG
.....	ACCACTCCCATTGGATTACGGGAGGCGATGGCGGT
.....	CTCCGCTGTACGCCTTAGATAGATGAACTGCCACC
.....	AGCCTGTTGGGGCGGCGGGTGGGACTGCGCACAG
.....	GCCCACACCGTCATCCTGCTCACCGTCGAAGCCGC
.....	CTTTCATCTGATTGGAACGACGCTTCCGATTCT
.....	TCGCATGGTCCCGCCTTCGAGGACGAGCATGGTG
.....	GTTGCGCTACTAGCCGCGTCTAACGTGAAGTGAA
.....	TCGTGGCTGATGACGCTGGCCCTGTGCAGGGCGGC
.....	ACCGGCTGGAACACGCAGCCGGATGGCAGGGGCCAG
.....	TGTGTACGGATACATGGCCGTGGCCGCAACGTTCACC
.....	GCCGCGAAAGCAAGGCGAGCAAGCGGATCGGCAAA
.....	CTGTCATTTTGGCTCCTTTATTAAATGGCGAACATT
.....	TGTGGTTATCGGTGGTCTTGTGTGACGGTGAGTGTT
.....	CTGGAAGAAGGCATGCGAGGATCCGGACGTGGA
.....	CCGCGAGTCGCCCGACCATGATATGGAACGTGATATG
.....	TGGCTCGTCCACAACTGGTACTAAGGAGCGGGACA
.....T.....	CTGTACGGCAAGGCTCAGACCGGCAGCGAGGGCTG
.....	CAAGCCCCCATACCCCGCGCAGCCCTCAAATCCAG
.....T.....-------	

GTCGCTCCCCATCCGGGGAGCGTGGATTGAAAT

Table 19.: Third Additional Example of CRISPR Array Annotation with Degenerated Repeat Candidate. An example of a CRISPR array detected in accession CP011799 (positions 7,225,314–7,225,535), illustrating repeat and spacer organization. Repeats are shown on the left and spacers or intervening sequences on the right. Differences from the consensus repeat (as defined by CRISPRidentify) are indicated in red to mark a candidate degenerated repeat. Only nucleotide positions that deviate from the consensus are displayed for clarity.

Repeats	Spacers / Sequences in between
--CCCCGCGGGTGCGGGGAGCAGC	CGCCACACCTATGCATCACGACCGACGGGGTGG
GTTCTAAGGTCTA.....-	GCGTCCGTCAAAGAGGAAGCCAAAGCCATGCA
.....	GGTCCCACGGTCAGGCTCGCGGCGGCCTTGCGGGTGGC
.....	GGTCCCGGCCCCACCGGCAACGAGACCGCGCCTCACCC
.....	AGTCCCCGCTTCCGCCGTGAGGTGCTTCACCTGACCGG
.....	GGTCCCCCTTCGGCCCGGTACACATGTTCCTGAACCAC
.....	

CTGCTCCCCGCACCCGCGGGGAT

Table 20.: Fourth Additional Example of CRISPR Array Annotation with Degenerated Repeat Candidate. An example of a CRISPR array detected in accession GL635657 (positions 194,439–194,935), illustrating repeat and spacer organization. Repeats are shown on the left and spacers or intervening sequences on the right. Differences from the consensus repeat (as defined by CRISPRidentify) are indicated in red to mark a candidate degenerated repeat. Only nucleotide positions that deviate from the consensus are displayed for clarity.

Repeats	Spacers / Sequences in between
.....-	CAAACAGATATTTAGAGAACTTGGAATAAAAGAG
-.....-	CTGAAACTCCTTTATCTTGGCTN×270AT(...)AA
.....	TTGGTATATTACGTATTTTTTCCTTTGTAATTGG
.....	CCTAATGTATCTTTAAAGAAAAACGCGACATCAC
.....	TTTCAGGAGAAAGTTCGGTGTCGAGATAGAAGCCTA
.....	ATTAGAAGAAAAAAAAGAAGCCTATGAAGAGGCT
.....	TTTTGTCTCCTTTCTCATTATACAAAAATTATA
.....	AAGCAGGTAGCTCCAACATCACGCTCAATCTGCG
.....	CTGAAAAATGGGAATTGGACTCTTACGATGGCGAT
.....	

CTGCTCCCCGCACCCGCGGGGAT

Bibliography

[1] P. Horvath and R. Barrangou, "Crispr/cas, the immune system of bacteria and archaea," *Science*, vol. 327, no. 5962, pp. 167–170, 2010.

[2] M. Lee, "Deep learning in crispr-cas systems: a review of recent studies," *Frontiers in Bioengineering and Biotechnology*, vol. Volume 11 - 2023, 2023.

[3] F. Zhang, S. Zhao, C. Ren, Y. Zhu, H. Zhou, Y. Lai, F. Zhou, Y. Jia, K. Zheng, and Z. Huang, "Crisprminer is a knowledge base for exploring crispr-cas systems in microbe and phage interactions," *Communications Biology*, vol. 1, Oct. 2018.

[4] R. Eisenhofer, J. Nesme, L. Santos-Bay, A. Koziol, S. J. Sørensen, A. Alberdi, and O. Aizpurua, "A comparison of short-read, hifi long-read, and hybrid strategies for genome-resolved metagenomics," *Microbiology Spectrum*, vol. 12, Apr. 2024.

[5] R. Bharti and D. G. Grimm, "Current challenges and best-practice protocols for microbiome analysis," *Briefings in Bioinformatics*, vol. 22, p. 178–193, Dec. 2019.

[6] D. Couvin, A. Bernheim, C. Toffano-Nioche, M. Touchon, J. Michalik, B. Néron, E. P. C. Rocha, G. Vergnaud, D. Gautheret, and C. Pourcel, "CRISPRCasFinder, an update of CRISRFinder, includes a portable version, enhanced performance and integrates search for cas proteins," *Nucleic Acids Res.*, vol. 46, pp. W246–W251, July 2018.

[7] A. Mitrofanov, O. S. Alkhnbashi, S. A. Shmakov, K. S. Makarova, E. V. Koonin, and R. Backofen, "CRISPRidentify: identification of CRISPR arrays using machine learning approach," *Nucleic Acids Res.*, vol. 49, p. e20, Feb. 2021.

[8] J. Zhang, K. Kobert, T. Flouri, and A. Stamatakis, "Pear: a fast and accurate illumina paired-end read merger," *Bioinformatics*, vol. 30, pp. 614–620, 10 2013.

[9] S. Shmakov, A. Smargon, D. Scott, D. Cox, N. Pyzocha, W. Yan, O. O. Abudayyeh, J. S. Gootenberg, K. S. Makarova, Y. I. Wolf, K. Severinov, F. Zhang, and E. V. Koonin, "Diversity and evolution of class 2 crispr–cas systems," *Nature Reviews Microbiology*, vol. 15, p. 169–182, Jan. 2017.

[10] H. Dalla-Torre, L. Gonzalez, J. Mendoza-Revilla, N. Lopez Carranza, A. H. Grzywaczewski, F. Oteri, C. Dallago, E. Trop, B. P. de Almeida, H. Sirelkhatim, G. Richard, M. Skwark, K. Beguir, M. Lopez, and T. Pierrot, "Nucleotide transformer: building and evaluating robust foundation models for human genomics," *Nature Methods*, vol. 22, p. 287–297, Nov. 2024.

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *CoRR*, vol. abs/2106.09685, 2021.

[12] C. Bland, T. L. Ramsey, F. Sabree, M. Lowe, K. Brown, N. C. Kyrpides, and P. Hugenholtz, "Crispr recognition tool (crt): a tool for automatic detection of clustered regularly interspaced palindromic repeats," *BMC Bioinformatics*, vol. 8, p. 209, Jun 2007.

[13] R. C. Edgar, "Piler-cr: Fast and accurate identification of crispr repeats," *BMC Bioinformatics*, vol. 8, Jan. 2007.

[14] F. Talibli and B. Voß, "Metagenomic crispr array analysis tool: a novel graph-based approach to finding crispr arrays in metagenomic datasets," *microLife*, vol. 6, 2025.

[15] W. Alanazi, D. Meng, and G. Pollastri, "Advancements in one-dimensional protein structure prediction using machine learning and deep learning," *Computational and Structural Biotechnology Journal*, vol. 27, pp. 1416–1430, 2025.

[16] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning–based sequence model," *Nature Methods*, vol. 12, p. 931–934, Aug. 2015.

[17] D. Quang and X. Xie, "Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences," *Nucleic Acids Research*, vol. 44, p. e107–e107, Apr. 2016.

[18] A. Talukder, C. Barham, X. Li, and H. Hu, "Interpretation of deep learning in genomics and epigenomics," *Briefings in Bioinformatics*, vol. 22, Aug. 2020.

[19] Y. Zheng, Q. Zou, J. Li, and Y. Yang, "Crispr-mfh: A lightweight hybrid deep learning framework with multi-feature encoding for improved crispr-cas9 off-target prediction," *Genes*, vol. 16, p. 387, Mar. 2025.

[20] F. Guo, R. Guan, Y. Li, Q. Liu, X. Wang, C. Yang, and J. Wang, "Foundation models in bioinformatics," *National Science Review*, vol. 12, Jan. 2025.

[21] B. Yan, Y. Nam, L. Li, R. A. Deek, H. Li, and S. Ma, "Recent advances in deep learning and language models for studying the microbiome," *Frontiers in Genetics*, vol. 15, Jan. 2025.

[22] E. Nguyen, M. Poli, M. G. Durrant, B. Kang, D. Katrekar, D. B. Li, L. J. Bartie, A. W. Thomas, S. H. King, G. Brixi, J. Sullivan, M. Y. Ng, A. Lewis, A. Lou, S. Ermon, S. A. Baccus, T. Hernandez-Boussard, C. Ré, P. D. Hsu, and B. L. Hie, "Sequence modeling and design from molecular to genome scale with evo," *Science*, vol. 386, no. 6723, p. eado9336, 2024.

[23] W. Li, X. Jiang, W. Wang, L. Hou, R. Cai, Y. Li, Q. Gu, Q. Chen, P. Ma, J. Tang, M. Guo, G. Chuai, X. Huang, J. Zhang, and Q. Liu, "Discovering

crispr-cas system with self-processing pre-crrna capability by foundation models," *Nature Communications*, vol. 15, Nov. 2024.

[24] J. W. Modell, W. Jiang, and L. A. Marraffini, "Crispr–cas systems exploit viral dna injection to establish and maintain adaptive immunity," *Nature*, vol. 544, p. 101–104, Mar. 2017.

[25] G. Benegas, C. Ye, C. Albors, J. C. Li, and Y. S. Song, "Genomic language models: opportunities and challenges," *Trends in Genetics*, vol. 41, p. 286–302, Apr. 2025.

[26] v. Avsec, V. Agarwal, D. Visentin, J. R. Ledsam, A. Grabska-Barwinska, K. R. Taylor, Y. Assael, J. Jumper, P. Kohli, and D. R. Kelley, "Effective gene expression prediction from sequence by integrating long-range interactions," *Nature Methods*, vol. 18, p. 1196–1203, Oct. 2021.

[27] H. A. Gündüz, M. Binder, X.-Y. To, R. Mreches, B. Bischl, A. C. McHardy, P. C. Münch, and M. Rezaei, "A self-supervised deep learning method for data-efficient training in genomics," *Communications Biology*, vol. 6, Sept. 2023.

[28] N. Sapoval, A. Aghazadeh, M. G. Nute, D. A. Antunes, A. Balaji, R. Baraniuk, C. J. Barberan, R. Dannenfelser, C. Dun, M. Edrisi, R. A. L. Elworth, B. Kille, A. Kyrillidis, L. Nakhleh, C. R. Wolfe, Z. Yan, V. Yao, and T. J. Treangen, "Current progress and open challenges for applying deep learning across the biosciences," *Nature Communications*, vol. 13, Apr. 2022.

[29] J. Atmos, "Diagram of the possible mechanism for crispr," 2009. Licensed under CC BY-SA 3.0.

[30] Y. Chen, J. Liu, S. Zhi, Q. Zheng, W. Ma, J. Huang, Y. Liu, D. Liu, P. Liang, and Z. Songyang, "Repurposing type i–f crispr–cas system as a transcriptional activation tool in human cells," *Nature Communications*, vol. 11, no. 1, p. 3136, 2020.

[31] Y. Xu and Z. Li, "Crispr-cas systems: Overview, innovations and applications in human disease research and gene therapy," *Computational and Structural Biotechnology Journal*, vol. 18, pp. 2401–2415, 2020.

[32] J. Grainy, S. Garrett, B. R. Graveley, and M. P. Terns, "Crispr repeat sequences and relative spacing specify dna integration by pyrococcus furiosus cas1 and cas2," *Nucleic Acids Research*, vol. 47, pp. 7518–7531, 06 2019.

[33] P. D. Hsu, E. S. Lander, and F. Zhang, "Development and applications of CRISPR-Cas9 for genome engineering," *Cell*, vol. 157, pp. 1262–1278, June 2014.

[34] F. Gómez-Vela, F. Divina, and M. García-Torres, "Computational methods for the analysis of genomic data and biological processes," *Genes*, vol. 11, p. 1230, Oct. 2020.

[35] V. Anand, H. S. Prabhakaran, A. Prakash, M. S. Hussain, and M. Kumar, "Differential processing of crispr rna by lincas5c and lincas6 of leptospira," *Biochimica et Biophysica Acta (BBA) - General Subjects*, vol. 1867, no. 12, p. 130469, 2023.

[36] S. S. Abby, B. Néron, H. Ménager, M. Touchon, and E. P. C. Rocha, "Macsyfinder: A program to mine genomes for molecular systems with an application to crispr-cas systems," *PLoS ONE*, vol. 9, p. e110726, Oct. 2014.

[37] J. Russel, R. Pinilla-Redondo, D. Mayo-Muñoz, S. A. Shah, and S. J. Sørensen, "Crisprcastyper: Automated identification, annotation, and classification of crispr-cas loci," *The CRISPR Journal*, vol. 3, p. 462–469, Dec. 2020.

[38] O. S. Alkhnbashi, A. Mitrofanov, R. Bonidia, M. Raden, V. D. Tran, F. Eggenhofer, S. A. Shah, E. Öztürk, V. A. Padilha, D. S. Sanches, A. C. P. L. F. de Carvalho, and R. Backofen, "<tt>crisprloci:</tt> comprehensive and accurate annotation of crispr–cas systems," *Nucleic Acids Research*, vol. 49, p. W125–W130, June 2021.

[39] J. E. Peters, K. S. Makarova, S. Shmakov, and E. V. Koonin, "Recruitment of crispr-cas systems by tn7-like transposons," *Proceedings of the National Academy of Sciences*, vol. 114, Aug. 2017.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[42] K. Sytwu, C. Groschner, and M. C. Scott, "Understanding the influence of receptive field and network complexity in neural network-guided tem image analysis," *Microscopy and Microanalysis*, vol. 28, p. 1896–1904, Dec. 2022.

[43] D. H. Parks, M. Imelfort, C. T. Skennerton, P. Hugenholtz, and G. W. Tyson, "Checkm: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes," *Genome Research*, vol. 25, p. 1043–1055, May 2015.

[44] A. P. Camargo, S. Nayfach, I.-M. A. Chen, K. Palaniappan, A. Ratner, K. Chu, S. J. Ritter, T. B. K. Reddy, S. Mukherjee, F. Schulz, L. Call, R. Y. Neches, T. Woyke, N. N. Ivanova, E. A. Eloe-Fadrosh, N. C. Kyrpides, and S. Roux, "Img/vr v4: an expanded database of uncultivated virus genomes within a framework of extensive functional, taxonomic, and ecological metadata," *Nucleic Acids Research*, vol. 51, p. D733–D743, Nov. 2022.

[45] A. P. Camargo, L. Call, S. Roux, S. Nayfach, M. Huntemann, K. Palaniappan, A. Ratner, K. Chu, S. Mukherjeep, T. B. K. Reddy, I.-M. A. Chen, N. N. Ivanova, E. A. Eloe-Fadrosh, T. Woyke, D. A. Baltrus, S. Castañeda-Barba, F. de la

Cruz, B. E. Funnell, J. P. J. Hall, A. Mukhopadhyay, E. P. C. Rocha, T. Stalder, E. Top, and N. C. Kyrpides, "Img/pr: a database of plasmids from genomes and metagenomes with rich annotations and metadata," *Nucleic Acids Research*, vol. 52, p. D164–D173, Nov. 2023.

[46] A. Malusare, H. Kothandaraman, D. Tamboli, N. A. Lanman, and V. Aggarwal, "Understanding the natural language of dna using encoder–decoder foundation models with byte-level precision," *Bioinformatics Advances*, vol. 4, no. 1, 2024.

[47] R. Takahashi, T. Matsubara, and K. Uehara, "Ricap: Random image cropping and patching data augmentation for deep cnns," in *Proceedings of The 10th Asian Conference on Machine Learning* (J. Zhu and I. Takeuchi, eds.), vol. 95 of *Proceedings of Machine Learning Research*, pp. 786–798, PMLR, 14–16 Nov 2018.

[48] A. Lopez-del Rio, M. Martin, A. Perera-Lluna, and R. Saidi, "Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction," *Scientific Reports*, vol. 10, Sept. 2020.

[49] A. Rücklé, G. Geigle, M. Glockner, T. Beck, J. Pfeiffer, N. Reimers, and I. Gurevych, "Adapterdrop: On the efficiency of adapters in transformers," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2021.

[50] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," 2020.

[51] J. A Ilemobayo, O. Durodola, O. Alade, O. J Awotunde, A. T Olanrewaju, O. Falana, A. Ogungbire, A. Osinuga, D. Ogunbiyi, A. Ifeanyi, I. E Odezuligbo, and O. E Edu, "Hyperparameter tuning in machine learning: A comprehensive review," *Journal of Engineering Research and Reports*, vol. 26, p. 388–395, Jun. 2024.

[52] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[53] Hugging Face, "Trainer — transformers 4.48.2 documentation." https://huggingface.co/docs/transformers/v4.48.2/en/main_classes/trainer#transformers.TrainingArguments, 2025. Accessed: 2025-06-07.

[54] Hugging Face, "Loraconfig — peft 0.14.0 documentation." https://huggingface.co/docs/peft/v0.14.0/en/package_reference/lora#peft.LoraConfig, 2025. Accessed: 2025-06-07.

[55] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2019.

[56] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2016.

[57] B. E. Slatko, A. F. Gardner, and F. M. Ausubel, "Overview of next-generation sequencing technologies," *Current Protocols in Molecular Biology*, vol. 122, p. e59, Apr 2018.

[58] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb. 1966.

[59] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, p. 31–88, Mar. 2001.

[60] D. J. Lipman and W. R. Pearson, "Rapid and sensitive protein similarity searches," *Science*, vol. 227, p. 1435–1441, Mar. 1985.

[61] S. McClain, "Bioinformatic screening and detection of allergen cross-reactive ige-binding epitopes," *Molecular Nutrition & Food Research*, vol. 61, Mar. 2017.

[62] C. Pourcel, M. Touchon, N. Villeriot, J.-P. Vernadet, D. Couvin, C. Toffano-Nioche, and G. Vergnaud, "Crisprcasdb a successor of crisprdb containing crispr arrays and cas genes from complete genome sequences, and tools to download and query lists of repeats and spacers," *Nucleic Acids Research*, Oct. 2019.

[63] I. Grissa, G. Vergnaud, and C. Pourcel, "Crisprfinder: a web tool to identify clustered regularly interspaced short palindromic repeats," *Nucleic Acids Research*, vol. 35, p. W52–W57, May 2007.

[64] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, p. 403–410, Oct. 1990.

[65] A. Biswas, R. H. Staals, S. E. Morales, P. C. Fineran, and C. M. Brown, "Crisprdetect: A flexible algorithm to define crispr arrays," *BMC Genomics*, vol. 17, p. 356, May 2016.

[66] A. T. Merchant, S. H. King, E. Nguyen, and B. L. Hie, "Semantic mining of functionalde novogenes from a genomic language model," *bioRxiv*, Dec. 2024.

[67] G. Brixi, M. G. Durrant, J. Ku, M. Poli, G. Brockman, D. Chang, G. A. Gonzalez, S. H. King, D. B. Li, A. T. Merchant, M. Naghipourfar, E. Nguyen, C. Ricci-Tam, D. W. Romero, G. Sun, A. Taghibakshi, A. Vorontsov, B. Yang, M. Deng, L. Gorton, N. Nguyen, N. K. Wang, E. Adams, S. A. Baccus, S. Dillmann, S. Ermon, D. Guo, R. Ilango, K. Janik, A. X. Lu, R. Mehta, M. R. Mofrad, M. Y. Ng, J. Pannu, C. Ré, J. C. Schmok, J. S. John, J. Sullivan, K. Zhu, G. Zynda, D. Balsam, P. Collison, A. B. Costa, T. Hernandez-Boussard, E. Ho, M.-Y. Liu, T. McGrath, K. Powell, D. P. Burke, H. Goodarzi, P. D. Hsu, and B. L. Hie, "Genome modeling and design across all domains of life with evo 2," *bioRxiv*, Feb. 2025.

[68] A. Moya-Beltrán, K. S. Makarova, L. G. Acuña, Y. I. Wolf, P. C. Covarrubias, S. A. Shmakov, C. Silva, I. Tolstoy, D. B. Johnson, E. V. Koonin, and

R. Quatrini, "Evolution of type iv crispr-cas systems: Insights from crispr loci in integrative conjugative elements of acidithiobacillia," *The CRISPR Journal*, vol. 4, p. 656–672, Oct. 2021.