Enhancing mRNA Translation Efficiency through Deep Learning

Siavash Khalaj

A Thesis submitted to the University of Ottawa in partial fulfillment of the requirements for the degree of Master of Computer Science and Concentration Applied AI

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

uOttawa

Abstract

The 5′ Untranslated Region (5′ UTR) of mRNA plays a key role in regulating translation efficiency. Consequently, optimizing this region is important in synthetic biology and therapeutic mRNA design for increasing protein yield and functional potency. Despite advances in computational methods for modeling 5′ UTR translation efficiency and sequence design, existing approaches do not account for mRNA secondary structure, provide limited control over sequence modification, and remain inefficient for bulk optimization. This thesis proposes a secondary structure-informed framework that combines accurate translation efficiency modeling with controllable, large-scale optimization of 5′ UTR sequences.

A Graph Attention Network (GAT) encoder integrating nucleotide identity, positional information, and predicted mRNA secondary structure was first trained to accurately predict 5′ UTR translation efficiency. The encoder was then extended into a multitask autoencoder by adding an autoregressive LSTM decoder. The autoencoder achieved near-perfect sequence reconstruction accuracy while maintaining the encoder's performance on translation efficiency prediction. The decoder was then fine-tuned using reinforcement learning to generate 5′ UTR variants with higher predicted translation efficiency. Fine-tuning the decoder using the REINFORCE algorithm substantially increased the number of generated sequences with improved predicted translation efficiency, while Difference between Augmented and Posterior (DAP)-regularized fine-tuning delivered improvements through smaller, more controlled edits that maintained greater similarity to the original sequences. Incorporating curriculum learning in DAP-regularized fine-tuning substantially increased the proportion of improved sequences with limited disruption to composition and entropy. Interpretability analyses confirmed that the framework captures biologically meaningful

determinants of translation initiation and applies optimization strategies consistent with known regulatory mechanisms.

Overall, this framework presents a novel approach to RNA sequence optimization and extends to regulatory elements beyond the $5'$ UTR.

Acknowledgements

I wish to thank my supervisor, Professor Marcel Turcotte, for his support and guidance throughout my studies.

Contents

List of Figures

List of Tables

List of Abbreviations

5′ UTR 5′ Untranslated Region ii, iii, 1–10, 14, 16–18, 31, 32, 35–47, 53, 55, 57, 64, 67, 70–73, 78–81, 83, 89, 90, 93, 106, 112, 116, 117, 119–123, 126–128, 130–133

3′ UTR 3′ Untranslated Region 14, 132

ARE AU-rich element 117

Cas9 CRISPR-associated protein 9 5, 6

CDS Coding Sequence 14

CE Categorical Cross-Entropy 84

CNN Convolutional Neural Network 20, 36–39, 42, 43, 49, 68–73, 96, 127, 129, 130

CRISPR Clustered Regularly Interspaced Short Palindromic Repeats 5

DAP Difference between Augmented and Posterior ii, 8–10, 29, 30, 83, 89, 90, 93–95, 98–104, 112, 114, 127–130

DNA Deoxyribonucleic Acid 5, 12–14, 25

GA Genetic Algorithm 42–44, 47, 50

GAN Generative Adversarial Network 44–46, 50, 128

GAT Graph Attention Network ii, 7, 9, 10, 53, 54, 58–61

GATv2 Graph Attention Network v2 54, 60, 61, 64, 67, 70–73, 76–78, 81

GCN Graph Convolutional Network 59

GFP green fluorescent protein 36, 64, 70, 71

GNN Graph Neural Network 7, 9, 20, 21, 53, 57–59, 61, 62, 70–72, 79, 80, 89, 90, 96, 97, 100, 118, 123, 126, 127, 129–131

IRES internal ribosome entry site 3, 39, 132

LSTM Long Short-Term Memory 9, 22–25, 29, 77, 79–82, 89, 90, 95, 126, 127, 129–131

mCherry monomeric Cherry 36, 70, 71

MFE Minimum Free Energy 19, 22, 107–111, 113, 116

MPRA massively parallel reporter assay 36, 37

MRL Mean Ribosome Load 7–11, 15–17, 27, 36, 37, 39, 40, 42, 43, 46–48, 53, 54, 62–65, 67, 68, 70, 72, 73, 77–81, 84, 86, 87, 89, 90, 93, 95–104, 106–118, 120–123, 128–130, 132

mRNA messenger RNA ii, 1–4, 6–8, 10, 12, 14, 15, 36, 39, 42, 43, 46, 47, 70, 117, 131–133

MSE Mean Squared Error 64, 70, 84, 86

RBP RNA-binding protein 3, 117

RBS ribosome-binding site 5

RL Reinforcement Learning 7–9, 26, 30, 89, 128, 130, 133

RNA Ribonucleic Acid 2, 4, 5, 10, 12–14, 16–22, 25, 37, 39–41, 43, 45–50, 54, 57, 77, 87, 88, 96, 116, 117, 126, 127, 131, 132

RNN Recurrent Neural Network 20

t-SNE t-distributed stochastic neighbor embedding 41, 108

TE translation efficiency 44

uAUG upstream AUG 16, 36, 37, 39, 70, 107–115

UMAP Uniform Manifold Approximation and Projection 107–111

uORF upstream Open Reading Frame 3, 16, 36, 37, 43, 70, 107–113

UTR Untranslated Region 16, 64, 133

1 Introduction

The 5′ Untranslated Region (5′ UTR) region of messenger RNA (mRNA) contains regulatory features that influence translation initiation through nucleotide composition, folding patterns, and start-codon context. Beyond translation efficiency, the 5′ UTR affects transcript stability, immune recognition, and tissue-specific expression. Thus, the 5′ UTR has become a major focus of mRNA therapeutic design. Pharmaceutical and biotechnology companies invest in the design of optimized 5′ UTRs, and several synthetic sequences are now patented for improved translation and reduced innate immune activation [1–3].

However, despite the growing availability of large-scale data on 5′ UTR activity and advances in predictive modeling, current design approaches continue to rely on heuristic search or generative models that provide limited control over biological outcomes. These limitations underscore the need for frameworks that can optimize 5′ UTRs efficiently while maintaining biological fidelity. To address this gap, this thesis presents a structure-aware generative framework that enables tunable and scalable optimization of empirically derived 5′ UTRs for translation efficiency while maintaining compositional similarity to the original designs.

1.1 Problem

Designing 5′ UTRs for enhanced translation efficiency involves balancing sequence composition, structural stability, and regulatory context. Existing generative frameworks are generally designed to generate plausible sequences or model the underlying sequence distribution. As a result, they typically do not search for sequence variants that improve a target property while remaining close to a given input, do not explicitly enforce sequence similarity to the original designs, often do not incorporate secondary-structure information, and provide limited control over the magnitude of functional improvement. They are also not designed for efficient bulk optimization across large libraries of empirically derived sequences.

1.2 Motivation

1.2.1 Scientific Rationale

Improving 5′ UTR design increases protein synthesis by improving ribosome recruitment, stabilizing translation initiation, and minimizing inhibitory secondary structures. The structural features of mRNA also influence innate immune sensing through secondary structures that resemble viral Ribonucleic Acid (RNA) or form stable double-stranded regions [4, 5]. Careful optimization of this region therefore benefits the therapeutic and industrial applications of mRNA technology.

The 5′ UTR is not a passive leader sequence or a simple determinant of baseline translation efficiency. It is a densely encoded regulatory region containing overlapping cis-regulatory elements that jointly influence ribosome recruitment, scanning, start-codon selection, cap-independent initiation, and mRNA stability. These elements include upstream Open Reading Frames (uORFs) [6], structural motifs such as stem-loops and G-quadruplexes, RNA-binding protein (RBP) motifs, and chemical modifications such as m^6A that can alter initiation dynamics and transcript fate [7, 8]. Importantly, these features do not act independently. Instead, they interact in a context-dependent manner, giving rise to coupled regulatory effects on both translation efficiency and transcript stability [8].

Given the superimposition of regulatory codes in the 5′ UTR, even small sequence changes can produce large and sometimes unintended effects on translation efficiency and mRNA stability [9]. Consequently, large-scale sequence alterations carry a high risk of pleiotropic disruption. Aggressive generative changes intended solely to maximize ribosome load may inadvertently obliterate unannotated stabilizing motifs, destroy stress-responsive internal ribosome entry sites (IRESs), or create cryptic uORFs that prematurely target the transcript for nonsense-mediated decay [6, 9]. These constraints motivate a minimum-edit approach that improves protein output while maintaining the structural and regulatory features of the native transcript. Such an approach provides a safer and more predictable alternative to unconstrained de novo sequence design or search.

1.2.2 Vaccines

mRNA vaccines deliver synthetic mRNA encoding viral or tumor antigens (in clinical development for cancer immunotherapy) to host cells, where the translation of the encoded protein elicits an adaptive immune response. The effectiveness of mRNA vaccines depends on the level and duration of antigen expression after vaccination. Higher translation efficiency leads to stronger antigen expression, which enhances immune priming and can prolong cellular and humoral immune responses. Efficient expression also allows lower mRNA doses to achieve the same immune potency, reducing both the manufacturing cost and the reactogenicity (transient inflammatory side effects such as fever or soreness). At the same time, innate sensing of vaccine mRNA as foreign RNA can modulate translation and inflammation. Accordingly, rational 5′ UTR optimization enables fine-tuning of translation efficiency and immunogenicity, a principle already reflected in patented 5′ UTR designs that support high protein yield and reduced innate activation in commercial mRNA vaccine platforms [1–3, 10].

1.2.3 Organisms

The 5′ UTR regulates the expression in both prokaryotic and eukaryotic cells and is widely exploited in biotechnology to enhance the production of recombinant proteins. In mammalian expression systems used for large-scale manufacturing of therapeutic proteins, both defined structural motifs and high-throughput 5′ UTR libraries have been shown

to systematically increase protein expression [11–14]. In bacteria, 5′ UTR engineering, also referred to as ribosome-binding site (RBS) optimization, has been a central focus of biotechnology and synthetic biology for more than two decades. Predictive frameworks such as RBS calculators are routinely used to optimize translation initiation and protein yield in *E. coli* and related bacterial hosts used for recombinant production of therapeutics such as insulin, growth factors, and enzymes.

Together, these examples demonstrate that 5′ UTR optimization is a general and scalable strategy for enhancing translation efficiency, and that systematic, large-scale optimization of available 5′ UTR libraries represents a promising direction for accelerating protein production in biotechnology.

1.2.4 CRISPR/Cas9 Engineering Therapy

The Clustered Regularly Interspaced Short Palindromic Repeats (CRISPR)/CRISPR-associated protein 9 (Cas9) system enables targeted genome editing by combining a guide RNA, which can be programmed to bind any desired Deoxyribonucleic Acid (DNA) sequence, with the Cas9 nuclease RNA. Once translated, the Cas9 protein forms a complex with the guide RNA and introduces a site-specific double-stranded break in the target DNA, which is then repaired by the cell's endogenous pathways to modify the gene. This approach underlies a range of gene-therapy strategies, including the correction of inherited mutations and the reprogramming of immune cells in cancer.

CRISPR/Cas9 and related RNA-encoded gene-editing systems depend on efficient

translation of the nuclease and guide machinery. Studies using high-throughput 5′ UTR libraries have demonstrated that 5′ UTR design strongly influences editing efficiency, protein yield, and cellular response. In particular, machine-learning-guided optimization of 5′ UTRs has been shown to improve translation of Cas9 transcripts across diverse mammalian cell types [13, 15]. These insights motivate the development of data-driven frameworks for systematic 5′ UTR optimization in mRNA therapeutics.

1.2.5 mRNA Therapeutics for Chronic and Metabolic Diseases

Beyond vaccines and gene editing, 5′ UTR engineering is an active area of research in mRNA therapeutics targeting chronic and metabolic conditions. Unlike the transient expression required for vaccines, these applications often demand sustained protein production and high potency to overcome complex physiological barriers. Recent evidence suggests that 5′ UTR optimization can transition from generic high-expression designs toward disease-specific architectures. For instance, tailored sequences derived from ribosomal protein transcripts have been shown to enhance both the level and duration of protein expression in preclinical models of aging and obesity [16]. By fine-tuning ribosome recruitment, such designs allow for effective therapeutic dosing while minimizing the inflammatory risks associated with higher mRNA concentrations.

However, the utility of the 5′ UTR as a regulatory hub extends beyond expression enhancement; it also functions as a programmable control mechanism for conditions characterized by protein overproduction. In neurodegenerative disorders such as Parkinson's and

Alzheimer's diseases, toxic α-synuclein and amyloid aggregates remain a major therapeutic challenge. Because these proteins are often intrinsically disordered and difficult to target with small molecules, the $5'$ UTR of the mRNA transcript offers a high-leverage site for translational attenuation. Emerging platforms such as PROTEIMERs, engineered proteins that bind structured motifs within the $5'$ UTR, are designed to inhibit translation or induce targeted mRNA decay [17].

Together, these features highlight the $5'$ UTR as a versatile control element, allowing researchers to either increase translation to overcome metabolic resistance or suppress pathogenic protein synthesis.

1.3 Research Questions

This thesis integrates predictive modeling and generative optimization into an autoencoder framework fine-tuned with Reinforcement Learning (RL) to systematically optimize $5'$ UTR sequences for improved translation efficiency. The following research questions were formulated to develop and evaluate this framework:

1. How accurately can a Graph Neural Network (GNN) encoder based on the GAT architecture predict translation efficiency of $5'$ UTRs, as measured by Mean Ribosome Load (MRL)?

2. Can a multitask autoencoder that combines a GAT-based encoder with an autoregressive decoder compatible with the REINFORCE algorithm—a policy-gradient

method in RL—learn latent representations that capture essential 5′ UTR features and faithfully reconstruct empircal 5′ UTRs?

3. Can fine-tuning the autoregressive decoder using the REINFORCE algorithm improve predicted translation efficiency (MRL) by generating optimized 5′ UTR sequences that outperform their original counterparts?

4. Can fine-turning the autoregressive decoder using the Difference between Augmented and Posterior (DAP) objective improve compositional and structural similarity to the original 5′ UTRs while enhancing translation efficiency (MRL)?

5. What is the effect of employing a curriculum-based training strategy on optimization dynamics and translation efficiency outcomes during reinforcement-learning fine-tuning of the autoregressive decoder?

6. What mechanistic determinants of translation efficiency are revealed by model explanations and empirical analyses of sequence features, structure, and biophysical properties?

1.4 Contributions

The following list outlines the contributions of this thesis, which collectively advance computational modeling and data-driven optimization of 5′ UTRs in mRNA-based expression systems:

1. **Graph-based sequence representation:** A novel GAT-based GNN encoder that models both nucleotide composition and 5′ UTR secondary structure for accurate prediction of translation efficiency (MRL).

2. **Multitask autoencoder for joint prediction and sequence reconstruction:** A novel architecture combining a GAT encoder with an autoregressive Long Short-Term Memory (LSTM) decoder for joint MRL prediction and faithful sequence reconstruction. The model learns compact latent representations that capture biologically relevant sequence and structural information and support the reconstruction of 5′ UTRs.

3. **RL fine-tuning for sequence optimization:** A novel application of the REINFORCE policy-gradient algorithm for fine-tuning an autoregressive decoder to generate optimized 5′ UTRs with increased MRL from learned latent representations.

4. **DAP-regularized fine-tuning for controllable optimization:** A novel use of the DAP objective for fine-tuning the decoder with a tunable reward-weight parameter that balances MRL improvements with sequence fidelity. While the DAP objective has been utilized in REINVENT [18] and recently PepINVENT [19] for molecular and peptide design, to the best of our knowledge, this work represents its first application to nucleic acid sequences. Specifically, the DAP objective enforces a limited Hamming distance between original and optimized sequences, maintains diversity among generated variants, and ensures optimization remains within biologically relevant bounds.

5. **Curriculum learning for progressive reinforcement optimization:** Introduced

a novel application of curriculum learning during DAP-regularized fine-tuning that stages decoder exposure from low- to high-efficiency sequences. This approach yields higher overall MRL gains while preserving compositional and structural integrity.

6. **Interpretability and mechanistic insights:** Applied latent space visualization and feature-level regression to relate model predictions to measurable sequence and structural features. Complementary compositional and structural analyses of optimized sequences and applying graph explainers indicated that the fine-tuned decoder optimized sequences through biologically meaningful mechanisms.

1.5 Organization of the Thesis

This thesis is organized into eight chapters. Chapter 1 introduces the motivation and problem statement, outlining the scientific, therapeutic, and biotechnological importance of 5′ UTR optimization, followed by specific research questions and contributions. Chapter 2 provides biological and computational background, covering translation mechanisms, RNA structure, and the machine-learning methods used in this thesis. Chapter 3 reviews related literature, including prior models for mRNA translation efficiency prediction and generative approaches for sequence design. Chapter 4 presents the GAT encoder for predicting MRL from 5′ UTR graphs, together with experimental results and ablation analyses. Chapter 5 introduces the multitask autoencoder architecture for joint 5′ UTR reconstruction and MRL prediction, including its evaluation results and ablation studies. Chapter 6 builds on

this framework by fine-tuning the autoencoder using reinforcement learning to generate optimized MRL sequences for enhanced translation efficiency, and presents evaluation results demonstrating the effectiveness of this optimization strategy. Chapter 7 interprets the biological features of the optimized sequences, linking observed improvements to structural and compositional changes. Finally, Chapter 8 concludes with a discussion of the findings, limitations, and directions for future research.

2 Background

2.1 Introduction

This chapter reviews essential biological concepts underlying mRNA structure and translation, followed by the deep learning and reinforcement learning methods that support the models developed in this thesis.

2.2 Biology

2.2.1 Nucleic Acids and mRNA

Nucleic acids are fundamental biological molecules responsible for storing, transmitting, and expressing genetic information in all forms of life [20]. They exist in two main forms: DNA and RNA. DNA functions as the stable and long-term repository of genetic information, while RNA is involved in several steps of gene expression and regulation. mRNA represents a class of RNA molecules responsible for delivering genetic information from DNA to ribosomes for protein synthesis. mRNA is therefore a key link in the flow of genetic

information and a central focus of research on translation efficiency and gene regulation.

Nucleic acids are chains of repeating units called nucleotides. Each nucleotide consists of a five-carbon sugar (ribose in RNA and deoxyribose in DNA), a phosphate group, and a nitrogenous base [20]. The primary nitrogenous bases found in nucleic acids include adenine (A), guanine (G), cytosine (C), thymine (T), and uracil (U). These bases are classified into two groups: purines and pyrimidines. Purines (A and G) are bicyclic, containing a five-membered ring fused to a six-membered ring. In contrast, pyrimidines (C and T) are monocyclic with a single six-membered ring (Figure 2.1). In canonical Watson-Crick pairing, each base pair consists of one purine and one pyrimidine: A pairs with T via two hydrogen bonds, and G pairs with C via three hydrogen bonds. In RNA, U replaces T and forms two hydrogen bonds with A. However, in bioinformatics, T and U are often used interchangeably in RNA sequences for simplicity and compatibility with DNA-based tools.

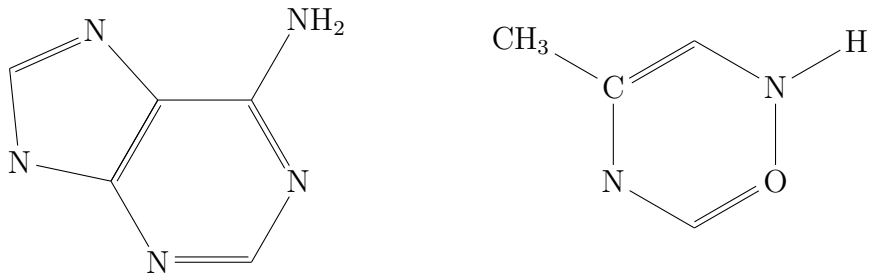


Figure 2.1: Representative structures of adenine (purine, left) and thymine (pyrimidine, right), two complementary nucleic acid bases.

2.2.2 Translation

In eukaryotic cells, genetic information flows according to the central dogma of biology. During transcription, DNA is transcribed into mRNA. This transcript is then used during translation, where ribosomes, the cellular organelles responsible for protein synthesis, decode mRNA to assemble proteins. Each mRNA molecule has a defined directionality determined by its sugar-phosphate backbone. The $5'$ end carries a free phosphate group attached to the fifth carbon of the ribose sugar, while the $3'$ end contains a free hydroxyl group on the third carbon. Translation initiates at the $5'$ end, with ribosomes progressing to the $3'$ end to synthesize the polypeptide chain from the N-terminus to the C-terminus [20].

mRNA molecules contain both translated and untranslated regions. The Coding Sequence (CDS) represents the translated region that encodes the polypeptide chain. The $5'$ UTR lies upstream of the start codon, whereas the $3'$ Untranslated Region ($3'$ UTR) is located downstream of the CDS [20]. These untranslated regions are not mere spacers; they play important regulatory roles. The $5'$ UTR influences how efficiently ribosomes bind to mRNA and initiate translation. Meanwhile, the $3'$ UTR affects mRNA stability, localization, and interactions with regulatory proteins such as RNA-binding proteins and microRNAs [20].

2.2.3 Mean Ribosome Load (MRL)

MRL is a standard measure of translation efficiency that quantifies the average number of ribosomes bound to an mRNA transcript [14]. MRL is estimated using polysome profiling, a technique that separates transcripts into sucrose gradient fractions according to ribosome occupancy. Transcripts found in lighter fractions are associated with fewer ribosomes and lower translation efficiency, whereas transcripts found in denser fractions are associated with more ribosomes and higher translation efficiency.

Let R_{nm} denote the read count of transcript n in fraction m, where fraction m corresponds to transcripts bound by approximately m ribosomes. Relative read abundance for each transcript in each fraction is first computed to normalize differences in total read counts between fractions:

$$\text{Relative } R_{nm} = \frac{R_{nm}}{\sum_n R_{nm}}$$

Normalized R_{nm} is then computed to define the distribution of each transcript across fractions:

$$\text{Normalized } R_{nm} = \frac{\text{Relative } R_{nm}}{\sum_m \text{Relative } R_{nm}}$$

MRL is subsequently calculated as the weighted sum of Normalized R_{nm} values:

$$\text{MRL}_n = \sum_m (\text{Normalized } R_{nm} \times m)$$

2.2.4 5′ UTR-Mediated Regulation of Translation

Translation proceeds through three stages: initiation, elongation, and termination, with initiation often being the rate-limiting step in eukaryotic cells. Multiple features of the 5′ UTR influence the initiation efficiency. In particular, the Kozak sequence, the nucleotide context surrounding the start codon, modulates ribosome recognition and initiation of translation [21]. Stable secondary structure in the 5′ UTR can impede ribosomal scanning [22]. Upstream AUGs (uAUGs) and uORFs also regulate initiation via leaky scanning or reinitiation [23, 24]. Moreover, RNA-binding proteins bind to Untranslated Region (UTR) elements to enhance or repress translation initiation [25, 26].

While the above factors provide important guidance for 5′ UTR design, they do not fully determine the optimal sequence. Translation efficiency is highly context dependent and shaped by interactions with the downstream coding sequence, local RNA structure, and the cellular environment [27]. Consequently, a 5′ UTR that performs well in one setting may not generalize to others. Because these interactions are numerous and difficult to model analytically, data-driven approaches such as machine learning are useful for discovering sequence patterns that optimize translation for specific genes and cellular contexts.

Since translation initiation strongly influences protein output, this work focuses on optimizing 5′ UTR sequences to improve ribosome recruitment and increase MRL, a key proxy for translation efficiency (Section 2.2.3).

2.2.5 RNA Secondary Structure and Translation Regulation

RNA secondary structure is the pattern of base-pairing interactions within a single RNA molecule, causing it to fold into structural elements such as stems, loops, bulges, and hairpins. These structures are formed through both canonical Watson–Crick base pairs (A–U and G–C) and non-canonical interactions such as G–U wobble. In essence, the secondary structure represents an intermediate level of organization that defines which nucleotides pair without specifying the complete three-dimensional arrangement described by the tertiary structure (Figure 2.2).

In the context of the 5′ UTR, secondary structure strongly influences translation initiation. Stable structures near the 5′ cap or start codon can impede ribosome access, slow scanning, or even block initiation entirely [22]. Conversely, unstructured regions generally facilitate ribosome recruitment and promote efficient translation initiation [28].

Certain structural motifs within the 5′ UTR exert strong regulatory effects on translation initiation. Hairpins near the 5′ cap or start codon can impede ribosome scanning [22], while GC-rich stems often form highly stable structures that limit initiation efficiency [9, 29]. G-quadruplexes can also form especially stable folds that disrupt scanning and influence start-codon recognition [30].

In this work, secondary structure predictions from the ViennaRNA package [31] are used to build RNA graphs (Section 4.2.1.1). These graphs serve as input to our encoder and autoencoder models. This approach allows the integration of structural information into the prediction of MRL, a measure of the efficiency of translation initiation, and guides

the optimization of 5′ UTRs to improve translation.

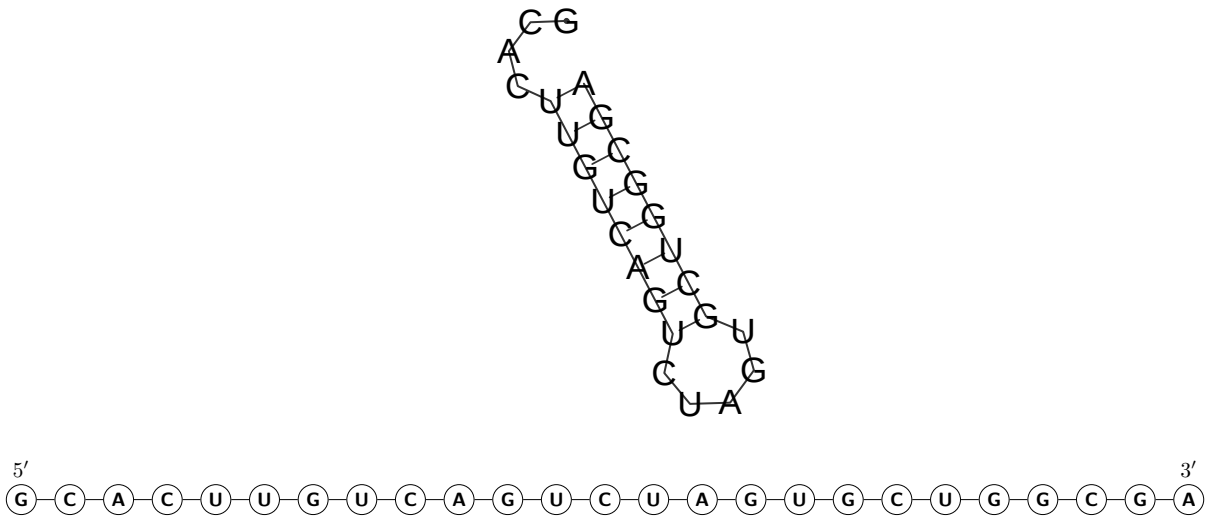


Figure 2.2: Primary and secondary structures of an RNA sequence.

2.2.5.1 Predicting RNA Secondary Structure with ViennaRNA

RNA secondary structure was predicted using **ViennaRNA** [31, 32], a widely used framework for modeling RNA folding. ViennaRNA is based on the principle that RNA folds into minimal Gibbs free energy conformations, as described by the nearest-neighbor thermodynamic model. In this model, the stability of a given helix is determined not only by Watson–Crick (A–T, G–C) and wobble (G–T) base pairs, but also by the stacking interactions of adjacent pairs. The associated energy parameters are derived from optical melting experiments [33].

RNA secondary structures are predicted using dynamic programming algorithms orig-

inally introduced by Zuker and Stiegler in the early 1980s [33] and subsequently refined in the ViennaRNA package [34]. Instead of enumerating the vast space of all possible foldings, the algorithm decomposes the sequence into smaller subintervals and calculates the Minimum Free Energy (MFE) achievable within each. These local solutions are then combined to yield the MFE structure. This recursive strategy achieves polynomial-time complexity while maintaining thermodynamic accuracy.

To improve biological realism, ViennaRNA also enforces constraints such as minimum hairpin loop sizes and penalties for isolated pairs. These measures help avoid predictions of sterically implausible or thermodynamically unstable structures [31].

Beyond predicting the single MFE structure, ViennaRNA derives base-pairing probabilities using McCaskill's partition function algorithm [35], as implemented in the package by Lorenz et al. [31]. The partition function evaluates the entire Boltzmann ensemble of possible RNA secondary structures. Each structure is weighted by its free energy, making lower-energy conformations exponentially more probable. This approach yields base-pairing probabilities that quantify structural diversity and uncertainty.

ViennaRNA returns secondary structures in dot-bracket string notation, where matching brackets denote paired nucleotides and dots indicate unpaired ones. These MFE-derived structures provide the basis for constructing RNA secondary structure graphs.

2.3 Machine Learning

2.3.1 Deep Learning

2.3.1.1 Graph Neural Networks

GNNs are a class of deep learning architectures designed for data that can be represented as graphs, where nodes denote entities and edges capture relationships between them. In molecular biology and chemistry, these entities may be atoms, nucleotides, or amino acids, while the edges might represent chemical bonds, base-pairing interactions, or spatial proximity. Unlike Convolutional Neural Networks (CNNs) that operate on grid-structured data or Recurrent Neural Networks (RNNs) that process ordered sequences, GNNs can directly model the irregular relational structure found in biomolecular data [36]. This capability makes GNNs ideal for representing RNA sequences as graphs enriched with secondary structure information.

The core principle of a GNN is the message-passing mechanism. In this iterative process, each node updates its feature vector (embedding) by aggregating information from its neighbors. After k iterations, the embedding of a node incorporates information from all nodes within its k-hop neighborhood. This ability to integrate local and global contexts is particularly relevant for RNA modeling: the influence of a nucleotide on translation efficiency can depend on both immediate neighbors in the sequence and distant bases that form structural contacts in the folded RNA. GNN aggregation functions (such as

aggregation based on mean, sum, or learnable attention) and update functions are designed to be permutation invariant, ensuring that GNN captures structural information without being affected by the order in which neighbors of a node are processed [36]. Mathematically, the message-passing mechanism is formulated as follows:

$$\mathbf{h}_i^{(\ell+1)} = \phi^{(\ell)}\Big(\mathbf{h}_i^{(\ell)},\ \mathrm{AGG}_{j\in\mathcal{N}(i)}\psi^{(\ell)}(\mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)}, \mathbf{e}_{ij})\Big),$$

where $\mathbf{h}_i^{(\ell)}$ and $\mathbf{h}_j^{(\ell)}$ represents the feature vector of nodes i and j in layer ℓ, $\mathbf{h}_i^{(\ell+1)}$ represents the updated feature vector of node i after message passing, $\mathbf{e}_{ij}$ is the edge feature between nodes i and j, $\psi^{(\ell)}$ is the message function that computes messages from neighbors $j \in \mathcal{N}(i)$, AGG is a permutation-invariant aggregation operator, and $\phi^{(\ell)}$ is the update function that integrates the aggregated messages with $\mathbf{h}_i^{(\ell)}$. In graph attention networks, neighbor-specific attention coefficients derived from $\mathbf{e}_{ij}$ and node features are used to weight the contribution of each message before aggregation; see details in Section 4.2.2.

GNNs have been applied to a wide range of biological prediction tasks. In molecular property prediction, atoms are represented as nodes with chemical descriptors, and bonds as edges with bond-type attributes. This graph formulation allows an accurate estimation of physicochemical properties, bioactivity, and toxicity [37]. In structural biology, amino acid residues or nucleotides can be modeled as nodes, whereas edges may encode hydrogen bonding, base pairing, or three-dimensional spatial proximity. Models such as ZHMolGraph [38] extend this paradigm by representing RNA molecules and proteins as graphs for the accurate modeling of RNA–protein interactions. In this thesis, a similar representation

strategy is applied to RNA: nodes containing biochemical and positional features correspond to nucleotides, while edges encode both sequential adjacency and base-pairing interactions derived from the MFE secondary structure predicted by ViennaRNA.

2.3.1.2 Long Short-Term Memory Networks

LSTM networks are a special type of recurrent neural network (RNN) designed to model sequential dependencies while addressing the vanishing and exploding gradient problems that affect standard RNNs. They were introduced by Hochreiter and Schmidhuber [39] to enable learning of long-range temporal dependencies.

Unlike standard recurrent neural networks (RNNs), which have a simple recurrent update, LSTM networks maintain an explicit *memory cell* c_t alongside a *hidden state* h_t. The cell state c_t acts as a highway for information, allowing gradients to flow over long time spans without vanishing. In contrast, the hidden state h_t represents the current output of the LSTM at each time step. This architecture enables LSTMs to learn when to store, overwrite, or expose information using gates controlled by learned parameters [39].

Gate equations and state updates. Given the current input x_t, the previous hidden state h_{t-1}, and the previous cell state c_{t-1}:

Forget gate — decides what to erase:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)\,.$$

Input gate — decides what to write:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)\,.$$

Candidate cell update — proposes new content:

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)\,.$$

Update cell state — combines old and new information:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t.$$

Output gate — controls what to reveal:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)\,.$$

Hidden state — exposes the output:

$$h_t = o_t \odot \tanh(c_t)\,.$$

Here, $\sigma(\cdot)$ is the sigmoid activation function, $\tanh(\cdot)$ denotes the hyperbolic tangent nonlinearity, and $\odot$ represents element-wise multiplication. These equations follow the reference `torch.nn.LSTM` implementation as documented in the official PyTorch manual [40].

Figure 2.3 illustrates the internal computations and state transitions within a single LSTM cell.

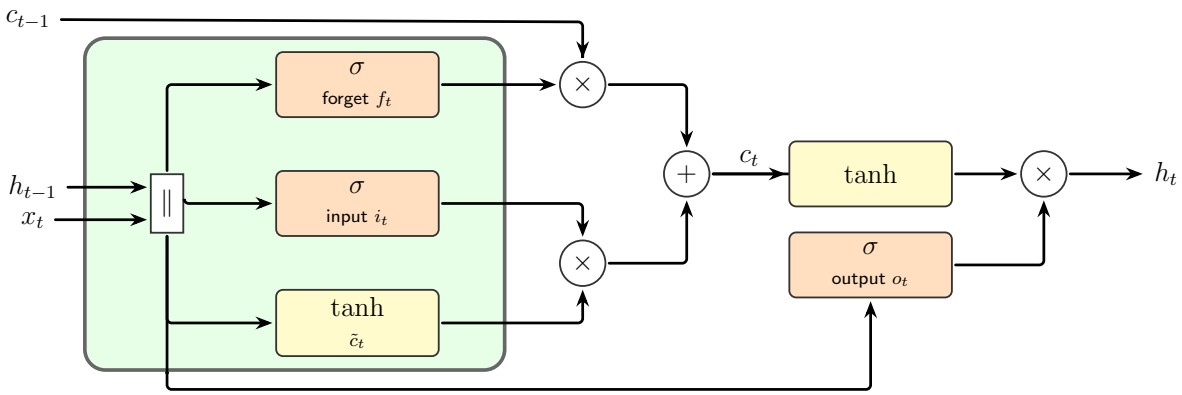


Figure 2.3: Architecture of a Long Short-Term Memory cell. The LSTM cell maintains two forms of state: the cell state (c_t) and the hidden state (h_t). At each time step, the previous hidden state (h_{t-1}) and the current input (x_t) are combined through learned linear transformations—conceptually equivalent to concatenating $[h_{t-1}, x_t]$—to compute four key components: the forget gate (f_t), which produces a gating vector that determines how much of the previous cell state (c_{t-1}) is retained; the input gate (i_t) and the candidate cell state ($\tilde{c}_t$), which together regulate how much new information is added; and the output gate (o_t), which determines how much of the transformed updated cell state $\tanh(c_t)$ contributes to the new hidden state. The cell state is updated by combining retained memory and new information, while the hidden state is derived by applying a non-linear transformation on the updated cell state, gated by o_t. The green-shaded area highlights the internal state computations of the LSTM cell, while external connections represent inputs, outputs, and state transitions. Figure created by the author.

Before the advent of Transformers, LSTM networks were widely used for sequence modeling tasks such as language modeling, translation, and speech recognition. Transformers leverage multi-head self-attention to capture relationships between all tokens simultaneously, enabling more effective modeling of long-range dependencies [41]. However, this capability comes at the cost of requiring substantially more training data, greater computational

resources, and more complex optimization strategies. In contrast, LSTMs process data sequentially and update their internal state at each step. This stepwise processing simplifies training and makes them more suitable for tasks with localized dependencies and relatively small datasets. Given these properties, we opted for an LSTM-based decoder within the autoencoder framework developed in this study. However, transformers are also a valid option worth exploring.

LSTMs have been widely used in bioinformatics to model biological sequences. For example, DanQ combined convolutional layers with bidirectional LSTMs to predict DNA regulatory activity [42], and LSTM-based models have been applied to predict RNA–protein binding preferences [43]. LSTM-containing architectures have also been used to predict long non-coding RNA–protein interactions [44]. More recent deep learning models continue to incorporate LSTM layers for sequence-based prediction tasks such as nucleic acid–binding protein identification [45]. Beyond these classification tasks, LSTM generators have been explored for de novo sequence design, including aptamers and protein-binding RNA motifs [46, 47]. Although a previous study [48] used an LSTM decoder within an autoencoder to classify DNA sequences, to our knowledge, this is the first application of an LSTM decoder within an autoencoder framework for the generative optimization of nucleic acid sequences.

2.3.2 Reinforcement Learning

Reinforcement Learning (RL) is a framework in which an agent learns an action policy by interacting with an environment to maximize cumulative rewards [49]. In the context of sequence generation, the decoder acts as an agent that sequentially generates tokens to form an output sequence, while the reward measures the quality of the generated sequence relative to a task-specific objective. RL is particularly suitable when the optimization objective is not differentiable with respect to the model parameters. In the autoencoder framework developed in this study, this is the case because rewards are computed from *discrete tokens* sampled by the decoder.

2.3.2.1 REINFORCE Algorithm

The REINFORCE algorithm [50] is a policy-gradient method used to optimize the expected reward of a stochastic policy π_θ, where θ denotes the trainable parameters of the model. In sequence generation, the policy π_θ defines the probability of generating a sequence $T = (t_1, t_2, \ldots, t_\ell)$. Because the decoder is autoregressive, this probability factorizes into a product of conditional probabilities:

$$\pi_\theta(T) = \prod_{i=1}^{\ell} \pi_\theta(t_i \mid t_{<i}) \,,$$

where $t_{<i} = \{t_1, \ldots, t_{i-1}\}$ denotes all previously generated tokens.

The objective is to maximize the expected reward:

$$J(\theta) = \mathbb{E}_{T\sim\pi_\theta}[R(T)],$$

where $R(T)$ evaluates the quality of the generated sequence, for example, the predicted MRL. Expanding the expectation explicitly gives

$$J(\theta) = \sum_T \pi_\theta(T)\, R(T),$$

which corresponds to a probability-weighted sum over all possible sequences.

Directly differentiating this objective is generally not be possible for two reasons. First, the decoder generates discrete tokens, which prevents gradients from propagating through the sampling process. Second, the reward $R(T)$ may be obtained from a nondifferentiable source, such as a rule-based model, or from a pre-trained predictor whose parameters are frozen during optimization. To address this problem, REINFORCE relies on the *log-likelihood gradient identity* [49], which follows directly from the derivative of the natural logarithm:

$$\nabla_\theta \log \pi_\theta(T) = \frac{\nabla_\theta \pi_\theta(T)}{\pi_\theta(T)} \quad \Longrightarrow \quad \nabla_\theta \pi_\theta(T) = \pi_\theta(T)\, \nabla_\theta \log \pi_\theta(T).$$

This simple identity allows us to express the derivative of the probability in terms of the

derivative of its logarithm. Substituting into the gradient of the objective gives

$$\nabla_\theta J(\theta) = \sum_T \pi_\theta(T)\, \nabla_\theta \log \pi_\theta(T)\, R(T).$$

Here, $R(T)$ is treated as a constant with respect to θ because of the two reasons mentioned above. The expression above is a probability-weighted sum of a function of T, which by definition is equivalent to an expectation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{T\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(T)\, R(T)]\,.$$

This formulation avoids differentiating through either the reward function or the discrete sampling process. The reward simply acts as a scaling factor: sequences with higher rewards increase their likelihood under the policy, whereas those with lower rewards decrease it. To further reduce the variance of the gradient estimate, a baseline b (for example, the moving average of past rewards) is introduced [50]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{T\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(T)\, (R - b)]\,.$$

Because the policy $\pi_\theta(T)$ factorizes autoregressively, the log-probability of a sequence can be written as

$$\log \pi_\theta(T) = \log \prod_{i=1}^{\ell} \pi_\theta(t_i \mid t_{<i}) = \sum_{i=1}^{\ell} \log \pi_\theta(t_i \mid t_{<i})\,,$$

where we use the logarithmic identity $\log(ab) = \log a + \log b$. Taking the gradient of this expression with respect to θ is straightforward, because differentiation distributes over summation:

$$\nabla_\theta \log \pi_\theta(T) = \sum_{i=1}^{\ell} \nabla_\theta \log \pi_\theta(t_i \mid t_{<i}) \,.$$

Substituting this into the REINFORCE update and distributing the expectation over the summation gives

$$\nabla_\theta J(\theta) = \sum_{i=1}^{\ell} \mathbb{E}_{T\sim\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(t_i \mid t_{<i}) \; (R - b)\Big].$$

This token-level formulation is particularly useful for autoregressive decoders such as the LSTM decoder used in this study, as it enables the model to assign credit or blame to individual token-level decisions based on the reward associated with the entire generated sequence.

2.3.2.2 Augmented Likelihood and DAP Loss

To keep the generated sequences close to the distribution of the original sequences, we adopt an augmented likelihood regularization strategy inspired by the REINVENT framework [18]. This strategy is implemented via the Difference between Augmented and Posterior (DAP) loss, which regularizes the generative model's policy towards a *pretrained prior* and prevents excessive divergence from biologically plausible sequences. For each generated sequence T,

the augmented log-likelihood is defined as

$$\log P_{\text{aug}}(T) = \log P_{\text{prior}}(T) + \sigma R(T).$$

Here, $\log P_{\text{prior}}(T)$ is the log-likelihood of the sequence under the fixed prior model, $R(T)$ is a reward representing sequence quality, and $\sigma \geq 0$ controls the trade-off between staying close to the prior and maximizing rewards. When $\sigma = 0$, the model behaves exactly like the prior, whereas larger σ values encourage deviation towards higher-scoring sequences.

The DAP loss is defined as the squared difference between the agent's log-likelihood and the augmented log-likelihood:

$$\mathcal{L}(T) = \left(\log P_{\text{aug}}(T) - \log P_{\text{agent}}(T)\right)^2.$$

Minimizing this loss drives the policy towards generating sequences that achieve high predicted rewards and remain consistent with the prior distribution.

Comparison with Pure REINFORCE. Pure RL using the REINFORCE algorithm optimizes the expected reward directly by maximizing

$$J(\theta) = \mathbb{E}_{T \sim P_{\text{agent}}}\left[R(T)\right].$$

In this approach, policy gradients are driven solely by the reward signal, without any constraint from a prior. While this strategy can lead to discovering high-reward sequences,

it often causes the model to drift toward unrealistic or implausible solutions (e.g., sequences containing long runs of adenine and thymine; see Section 6.6.2.1). This issue becomes more pronounced when the reward estimates are uncertain or when most generated sequences receive similar predicted rewards.

In contrast, augmented likelihood regularization integrates the prior model's likelihood into the optimization process. This strategy offers two key benefits. First, it preserves the similarity between generated sequences and those in the original dataset. Second, it stabilizes training by preventing policy collapse, which is a common issue in pure REINFORCE when the model overfits to extreme high-reward regions. Thus, prior-augmented log-likelihood regularization balances *exploitation* (reward optimization) and *exploration* (maintaining diversity), making it particularly suitable for optimizing $5'$ UTRs, where both translation efficiency and structural plausibility must be preserved.

2.3.2.3 Curriculum Learning

Curriculum learning is a machine learning paradigm in which a model is exposed to tasks in a structured order rather than being trained on the full complexity of the problem from the start [51]. In its classical form, the curriculum follows a progressive order of difficulty, where the model first focuses on easier examples and gradually incorporates more challenging cases. Variants of this approach, including reverse curricula, have also been explored depending on the optimization objective [52].

Curriculum learning can be particularly valuable for optimization of biological sequences.

Optimizing sequences such as 5′ UTR regions requires navigating a high-dimensional search space, where small changes in sequence composition can produce nonlinear and often unpredictable effects on target properties. Designing a curriculum by grouping sequences based on a chosen property and introducing them in a controlled order helps the model explore the sequence space more effectively, avoid overfitting to a narrow set of high-scoring motifs, and generate sequences that are both diverse and biologically plausible.

2.4 Performance Evaluation

We define metrics and protocols used throughout the thesis. For predictive models: mean squared error (MSE), Pearson's r, and R^2 on held-out sets. For generative models: improvement in predicted MRL, success rate under constraints (e.g., Hamming distance thresholds, motif preservation), diversity measures, and ablation protocols that isolate the contribution of secondary-structure information. We also outline cross-validation or split strategies and report computational considerations relevant for fair comparisons.

2.4.1 Pearson's r and Coefficient of Determination R^2

Pearson's correlation r and the coefficient of determination R^2 are reported on the test set. Pearson's r quantifies the strength of the linear association between observed and predicted values (range -1 to 1), while R^2 gives the proportion of variance in y explained by the predictions. Because targets and predictions are standardized with the same z-score of the

training set, r and R^2 are invariant to the inverse transformation.

$$r = \frac{\sum_{i=1}^{n}(y_i - \bar{y})\,(\hat{y}_i - \overline{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}\,\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \overline{\hat{y}})^2}} \qquad R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

where y_i are targets, $\hat{y}_i$ are predictions, $\bar{y}$ is the mean of $\{y_i\}$, and $\overline{\hat{y}}$ is the mean of $\{\hat{y}_i\}$.

2.4.2 Entropy and Sequence Diversity

Entropy, introduced by Shannon [53], is a fundamental measure of uncertainty or variability in a probability distribution. For a discrete alphabet $\mathcal{A}$, the Shannon entropy (in bits) is defined as

$$H(p) = -\sum_{s \in \mathcal{A}} p(s)\,\log_2 p(s),$$

where $p(s)$ is the probability of observing symbol s. To allow comparison across settings, the normalized entropy is reported as

$$\tilde{H} = \frac{H}{\log_2 |\mathcal{A}|},$$

where $\mathcal{A}$ denotes the alphabet of possible symbols (e.g., $\mathcal{A} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ for nucleotide sequences). This quantity ranges from 0 (no diversity) to 1 (maximum diversity).

Shannon entropy is a standard measure used in nucleic acid information content analysis [54] and to quantify sequence diversity [55]. Two complementary approaches are commonly used to characterize sequence diversity. Total sequence entropy captures the global

compositional diversity by computing the base frequencies for a sequence T of length L:

$$p(s) = \frac{c_s + \alpha}{L + \alpha|\mathcal{A}|},$$

where c_s is the count of symbol s and $\alpha > 0$ is a small pseudocount to avoid zero probabilities, which would otherwise make the entropy undefined because of the $\log p(s)$ term. The whole-sequence entropy is then

$$H_{\text{seq}}(T) = H(p(\cdot)),$$

which reflects the overall diversity of a sequence.

In contrast, sliding-window entropy measures local variability. Using a window of width w, for each window starting at position i, the base frequencies are estimated as

$$p_i(s) = \frac{c_s^{(i:i+w-1)} + \alpha}{w + \alpha|\mathcal{A}|}, \qquad H_{\text{win}}(i) = H(p_i(\cdot)).$$

The average sliding-window entropy, defined as

$$\overline{H}_{\text{win}} = \frac{1}{L - w + 1} \sum_{i=1}^{L-w+1} H_{\text{win}}(i),$$

provides a summary of local sequence complexity, while the profile $H_{\text{win}}(i)$ can reveal conserved regions, repetitive motifs, or low-diversity segments.

3 Literature Review

3.1 Introduction

This chapter reviews computational methods for predicting and optimizing translation efficiency using 5′ UTR sequences. First, we examine regressor-only models that map sequences to translation-related properties, followed by generative approaches for sequence design, including adversarial frameworks, evolutionary algorithms, and optimization in latent space. We conclude by summarizing the strengths and limitations of these methods to provide context for this work.

3.2 Regressor-Only Models

3.2.1 Convolutional Neural Networks

Early studies on modeling translation efficiency were based on regression methods and hand-crafted sequence features. Classic studies by Kozak established the role of the initiation context around the AUG start codon in shaping translation initiation efficiency [21, 56].

Subsequent regression analyses showed that codon usage and local mRNA secondary structure affect protein output in bacteria [57] and yeast [58, 59]. Ribosome profiling studies in yeast further refined these models, with Weinberg et al. [60] demonstrating that much of the gene-to-gene variance in initiation efficiency could be explained by features such as uAUGs, uORFs, GC content, and 5′ UTR structure.

The advent of massively parallel reporter assays (MPRAs) enabled systematic measurements of mRNA regulatory activity in human cells. Sample et al. [14] developed a polysome-profiling MPRA to quantify the MRL of 280,000 unique, randomly synthesized 5′ UTRs in HEK293T cells using a green fluorescent protein (GFP) reporter gene. This large-scale dataset provided the depth and diversity necessary to train a CNN model directly on one-hot encoded 5′ UTRs. The model, named Optimus 5-Prime, was the first to predict translation efficiency from 5′ UTRs without reliance on hand-crafted features, k-mer representations, or regulatory motifs.

Optimus 5-Prime substantially outperformed linear k-mer–based model by explaining more than 90% of the variance in MRLs in the held-out test set. The model further generalized beyond the training dataset to assess chemically modified mRNAs and estimate the effects of endogenous 5′ UTRs and disease-associated variants. Moreover, validation in a separate randomly synthesized library with a monomeric Cherry (mCherry) reporter confirmed that Optimus 5-Prime can generalize in different reporter contexts with high predictive accuracy. Despite these advances, the model was constrained by the fixed 50-nucleotide input length and exhibited reduced reliability for sequence patterns sparsely represented in the training data, most notably 5′ UTRs containing long poly(U) runs.

Subsequent work extended CNN-based approaches beyond synthetic MPRA datasets. Karollus et al. [61] introduced FramePool, a model that preserves codon-frame context by dividing the convolutional feature map into three offsets relative to the canonical start codon. This enables the network to distinguish whether motifs such as uAUGs occur in-frame or out-of-frame, a distinction important for translation. To accommodate variable-length sequences during batching, inputs were padded with zeros but accompanied by a mask to ensure that pooling operations ignored the padding. By pooling features within each offset separately, FramePool removed the fixed-length constraint of earlier models while maintaining biologically meaningful frame information. The model performed comparably to Optimus on matched test sets but generalized better to variable-length sequences and endogenous transcripts. These results show how architectural modifications can improve CNN performance and enable a more biologically faithful modeling of 5′ UTR regulatory mechanisms.

Although convolutional models accurately predicted MRL in randomly synthetesized MPRA libraries, their ability to generalize to endogenous human 5′ UTRs remained unclear. Korbel et al. [62] addressed this question by training CNNs on random or human reporter libraries and using integrated gradients and meta-attribution maps to interpret the models. They showed that models trained solely on synthetic data overestimated the repressive effects of uAUGs and uORFs while failing to capture sequence contexts enriched in natural transcripts, such as GC- and U-rich motifs associated with the structure of RNA. The observed mismatch highlights the inherent limitation of relying solely on synthetic libraries to capture the determinants of translation efficiency.

Korbel and colleagues addressed the above limitations by developing OptMRL [62], a CNN fine-tuned on human 5′ UTR reporters after initial training on synthetic data. OptMRL preserved performance on synthetic data while improving generalization to human sequences by reducing bias and capturing key regulatory features. This study showed that combining randomly synthesized and natural libraries expands sequence space coverage and produces models that more faithfully capture the regulatory architecture of endogenous 5′ UTRs. It also underscores the value of explainable AI for validating learned features and for guiding architectural and dataset choices in translation modeling.

Together, these studies established CNNs as the first generation of end-to-end models for 5′ UTR translation efficiency. Optimus demonstrated feasibility on randomly synthesized datasets, FramePool extended applicability to variable-length and endogenous transcripts, and OptMRL improved interpretability by linking predictions to known regulatory features. Despite these advances over handcrafted regressors, CNNs remain limited in their ability to capture global structure and long-range dependencies. This limitation underscores the need to develop more expressive architectures.

3.2.2 Nucleic Acid Language Models

Pretraining on large 5′ UTRs datasets allows language models to learn embeddings that encode compositional, positional, and structural signals. Attention mechanisms enable these models to capture long-range dependencies that were difficult to represent with CNNs. When fine-tuned, 5′ UTR language models provide robust baselines for predicting

translation outcomes such as ribosome load, efficiency, and expression. Taken together, these advances establish nucleic acid language models as natural successors to convolutional approaches for modeling 5′ UTRs regulatory codes.

UTR-LM [63] is a transformer-based language model pretrained on multi-species 5′ UTR datasets using a masked nucleotide prediction objective. In addition, the authors incorporated auxiliary tasks for predicting secondary structure and minimum free energy to provide structural signals that enrich the learned embeddings. When fine-tuned for downstream tasks such as predicting MRL, translation efficiency, mRNA expression, and IRES activity, the model achieved state-of-the-art performance. It surpassed both CNN-based approaches, including Optimus and FramePool, as well as RNA foundation models such as RNA-FM [64] and RNABERT [65]. The model also generalized to variable-length human 5′ UTRs and, in zero-shot tests, accurately transferred to an independent luciferase reporter assay.

Beyond predictive accuracy, Chu et al. showed the practical design utility of UTR-LM by generating 211 new 5′ UTRs with high predicted translation efficiency and confirming higher protein expression in reporter assays. Attention analyses identified established biological features, including Kozak consensus enrichment near the start codon and depletion of uAUGs in highly expressed UTRs. The analyses also highlighted potential novel sequence motifs that may influence translation efficiency, although these remain unvalidated. Overall, these findings show the potential of 5′ UTR–specific language models to combine predictive accuracy with interpretability and design utility.

Language models extend beyond 5′ UTRs to the broader RNA landscape. RNA-

FM (RNA Foundation Model) [64] is a 12-layer transformer pretrained with masked nucleotide modeling on over 20 million non-coding RNAs from RNAcentral [66]. The resulting embeddings encode biologically meaningful features and are useful for downstream structural and functional analyses. When reduced in dimensionality and projected into a two-dimensional "RNA Atlas," these embeddings cluster RNAs into functional classes such as housekeeping and regulatory. They also reveal structural, functional, and evolutionary relationships among RNAs.

RNA-FM embeddings have been reported to provide features for downstream models of RNA structure. They show high accuracy in secondary structure prediction and improvements in tertiary structure benchmarks. In addition, RNA-FM has been applied to genome analyses of RNA viruses, such as SARS-CoV-2, and to protein–RNA interaction models. In the latter case, the embeddings performed comparably to experimental structural measurements.

Although RNA-FM was not designed for $5'$ UTR modeling, its embeddings have the potential to complement UTR-specific models and support tasks such as predicting MRL. This view is supported by the model's reported success in other functional applications, including the prediction of RNA–protein interactions.

A recently introduced model, RiNALMo [67], extends RNA language modeling by pretraining a 650M-parameter Transformer on over 36 million non-coding RNAs aggregated from RNAcentral [66] and its partner databases. Using recent advances in Transformer design, the model yields embeddings that are more informative than those from earlier

approaches. t-distributed stochastic neighbor embedding (t-SNE) analyses demonstrated clear family-level clustering of the embeddings, indicating that the embeddings reflect biologically relevant structural and functional features.

In structure prediction, RiNALMo achieves strong generalization to RNA families excluded from training, outperforming thermodynamic and probabilistic baselines. These findings indicate that RNA language models can transfer learned principles to unseen families.

In functional tasks, RiNALMo matched the performance of RNA-FM and specialized models on supervised benchmarks such as splice-site detection and non-coding RNA classification. Important for this section, despite being trained on non-coding RNAs, the model generalized to the prediction of 5′ UTR translation, where it showed strong performance across several human data sets.

Together, these models represent complementary advances in RNA language modeling. RNA-FM provided broadly useful embeddings from non-coding RNAs; UTR-LM demonstrated the benefits of 5′ UTR-specific pretraining for translation tasks; and RiNALMo showed that scaling with modern Transformer design enables both structural generalization and functional prediction.

3.3 Generative Models

3.3.1 Genetic Algorithms

Genetic Algorithms (GAs) are stochastic optimization methods inspired by the principles of natural selection. In a genetic algorithm, candidate solutions are organized into a population of chromosomes that undergo iterative refinement through crossover and mutation. In each iteration, a fitness function evaluates evolved candidates and preferentially propagates those that better solve the problem. Because GAs do not require gradient information, they are particularly effective in solving problems that involve exploring vast combinatorial search spaces, including biological sequence design [68].

Sample et al. [14] developed a GA guided by their Optimus 5-Prime CNN to engineer 50-nt 5′ UTRs with predicted ribosome loading close to predefined MRL values, where MRL is a polysome profiling-based measure of the average number of ribosomes associated with an mRNA (Section 2.2.3). The algorithm iteratively edited random 50-nucleotide sequences until Optimus 5-Prime predicted the desired MRL. Two experimental strategies were evaluated. First, the GA generated sequences that targeted specific MRL values (3 through 9, plus a no-limit maximum), which were then synthesized and assayed. Second, the GA was run in a stepwise manner: for the first 800 generations it selected low-MRL sequences, after which the target was switched to high MRL. Each unique sequence along this trajectory was synthesized and assayed to provide an experimental record of the transition from low to high translation. Because predictive performance deteriorated at

very high MRLs for GA-designed 5′ UTRs with long poly(U) tracts, Optimus 5-Prime was retrained for four additional rounds on a GA-enriched library containing sequences with frequent homopolymer stretches that were rare in the original random library. The retrained CNN improved agreement with observed MRLs and was subsequently applied to both the fixed-target and stepwise evolution experiments.

Although influential, Sample et al.'s generative model was limited by the fixed 50-nucleotide input window of Optimus 5-Prime and showed reduced robustness for sequence patterns sparsely represented in the training data, such as long poly(U) tracts. Moreover, the GA procedure was computationally intensive and followed a search-based optimization strategy rather than a sequence-to-sequence formulation. As a result, the generated sequences could differ substantially from the starting inputs, and the study did not report similarity metrics such as Hamming distance. It must also be noted that 5′ UTRs in eukaryotes vary widely in length, ranging from tens to several thousand nucleotides, with a median length of approximately 200 nucleotides in humans [29]. However, given the constraints of available datasets and in line with recent computational studies, the present work focuses on fixed-length 50-nt sequences.

In another study, Cao et al. [13] used a GA guided by a random forest model to design 100-nucleotide 5′ UTRs that increase protein output from a human cytomegalovirus promoter. The random forest regression model was trained on matched human Ribo-seq and RNA-seq data to predict translation efficiency and mRNA abundance from sequence features including k-mer frequency, RNA folding energy, 5′ UTR length, and number of uORFs. Using this surrogate as a fitness function, the GA iteratively mutated and recombined

endogenous human 5′ UTRs to optimize predicted translation efficiency (TE) and protein production. The resulting candidate 5′ UTRs were inserted upstream of a reporter gene and screened to identify three synthetic sequences that consistently boosted protein production across cell types. Overall, while Cao et al. showed that genetic algorithms guided by a surrogate model can efficiently discover high-performing 5′ UTRs, their approach was limited by the fixed 100-nucleotide design window and reliance on a hand-crafted random forest predictor.

Together, GA-based approaches highlight the potential of evolutionary search for sequence design. However, these methods do not yield a trained model that can be reused or generalized to other tasks. They are also not true sequence-to-sequence models, since they refine candidate populations rather than learning a reusable mapping from arbitrary inputs to optimized outputs. The iterative search and fitness scoring is computationally expensive and may settle into local optima if diversity is not maintained. Even with extensive computational resources, it is not guaranteed that a GA explores the best regions of the sequence space.

3.3.2 Generative Adversarial Networks

Generative Adversarial Networks (Generative Adversarial Networks (GANs)) are adversarial learning frameworks in which a generator and a discriminator compete in a minimax game. This process drives the generator to produce synthetic samples that are indistinguishable from real data [69]. Adversarial learning models have shown potential for *de novo* generation

of nucleic acid sequences with desirable properties. However, these models struggle with sequence tasks because the discrete token sampling step is non-differentiable and blocks backpropagation. Workarounds such as policy-gradient training and Gumbel–Softmax relaxations [70] partially address this issue, but instability and mode collapse remain recurring challenges.

Barazandeh et al. recently introduced UTRGAN, the first GAN-based model specifically aimed at the design of synthetic human 5′ UTRs [71]. In this study, the model was trained in two phases. In the adversarial pretraining phase, the generator learned to produce natural-like 5′ UTRs by competing against a critic network. In the subsequent optimization phase, latent noise vectors were iteratively updated using gradients backpropagated from frozen predictors of translation efficiency. Experimental validation confirmed that the resulting sequences improved protein output relative to the benchmark constructs. This work showed that adversarial frameworks can be applied to the design of regulatory sequences.

In another recent study, Riley et al. proposed a combined predictive–generative framework for functional RNAs design [72]. First, a predictor network was trained to map the sequence and structural features of RNA to functional readouts. A GAN was then trained to reproduce the sequence and structural patterns of the training data. Optimization was achieved by iteratively adjusting latent input vectors using gradients backpropagated from the frozen predictor. As a proof of concept, the approach was applied to 5′ UTR design and later extended to more structurally constrained regulatory RNA elements, highlighting its flexibility across RNA classes.

Overall, both studies demonstrate the potential of GANs in designing regulatory RNA sequences. However, the latent space learned by GANs is often irregular and complex, which limits fine-grained control over outputs. In addition, because GANs lack true sequence-to-sequence mappings, they cannot directly optimize existing sequences for specific functional properties.

3.3.3 Latent-Space Optimization

A practical strategy for sequence design is to operate in a continuous latent space learned by autoencoders and related generative models. During optimization, latent vectors are updated by gradients backpropagated from a frozen predictor or perturbed directly to explore promising directions. This setup facilitates smooth exploration of the design space and supports the efficient discovery of candidate sequences with improved predicted properties.

Tang et al. introduced Smart5UTR, a multi-task autoencoder designed to optimize synthetic $5'$ UTRs for mRNA vaccines [73]. The model combines an encoder for predicting MRL with a sequence reconstruction decoder, shaping a latent space that reflects both translation efficiency and sequence features. To improve predictive accuracy in the high-expression range most relevant for vaccine design, the authors used a custom loss that penalized errors more heavily for high-MRL sequences.

Following training, the learned representations served as a foundation for optimization. Latent vectors were iteratively perturbed with Gaussian noise and a strengthening

coefficient, decoded back into sequences, and scored for predicted MRL. Repeating this perturbation–decode–score optimization loop progressively directed candidates toward higher predicted translational outputs.

Smart5UTR combined high predictive accuracy with faithful sequence reconstruction, ensuring that optimization remains closely linked to underlying sequence features. During optimization, the model successfully improved translational output across almost the entire test set, surpassing the GA baseline by a wide margin. Experimental validation confirmed optimization gains, with designed sequences driving enhanced Spike expression and stronger immune responses relative to an endogenous control 5′ UTR.

Building on the latent space optimization paradigm, Liao et al. introduced RNAGenScape, a framework that shifts the focus from *de novo* sequence generation to the refinement of existing RNAs [74]. This shift addresses a key limitation of purely generative models, whose outputs often diverge from natural biological patterns and are therefore less interpretable and harder to validate experimentally. Although designed as a general mRNA framework, the approach was evaluated on optimizing existing 5′ UTRs to enhance translation efficiency.

At the core of RNAGenScape is an organized autoencoder that simultaneously learns to reconstruct sequences and predict functional properties such as translation efficiency. The encoder compresses the input sequences into compact latent representations, the predictor organizes these representations around desired properties, and the decoder reconstructs them back into full-length RNA sequences.

Optimization proceeds through property-guided Langevin dynamics, where a drift term directs latent embeddings along the gradient of the property predictor (e.g., MRL) and Gaussian noise introduces exploration to avoid local minima. After each update, a manifold projector — implemented as a denoising autoencoder — projects the perturbed embedding back onto the learned manifold, ensuring that the search remains within biologically meaningful regions of latent space.

Through repeated drift–noise–projection updates, RNAGenScape progressively steers latent representations toward improved properties while preserving their biological plausibility. The final embedding and intermediate optimization states can be decoded into sequences, offering an interpretable view of how incremental changes shape function.

Empirically, RNAGenScape generated smooth, manifold-aligned optimization trajectories and consistently outperformed both *de novo* generative models and conventional optimization baselines. It achieved substantial improvements in MRL while remaining highly efficient at inference. These findings highlighted the effectiveness of manifold-constrained refinement as a practical alternative to unconstrained generation in RNA design.

Latent space optimization transforms the discrete nucleotide space into a continuous structured representation that enables smooth exploration, gradient-based updates, and biologically meaningful refinements. When combined with appropriate regularization or manifold constraints, this approach can guide the search toward biologically plausible regions of sequence space and yield interpretable trajectories toward improved properties such as translation efficiency. Despite these advantages, latent space optimization does not

produce a single model capable of directly mapping an input sequence to an optimized output in one step. Instead, each candidate must undergo an iterative refinement process, which can be computationally expensive. Furthermore, enforcing similarity to the original sequence is challenging, and optimization may remain confined to regions near the training distribution, potentially limiting the discovery of more distant but superior solutions.

3.4 Limitations and Trade-offs

The methods reviewed in this chapter have advanced RNA modeling and optimization in complementary ways. At the same time, they involve inherent trade-offs that shape their suitability and performance in different contexts. In this section, we examine these strengths and limitations to better understand how various strategies compare to our framework in terms of design choices and trade-offs.

- **CNN regressors:** CNN-based regressors use local convolutional filters to capture short-range sequence motifs and positional patterns. They offer strong performance and scalability in sequence-based prediction tasks. However, their reliance on local context limits their ability to capture global structural influences or long-range dependencies. In addition, their internal representations are often difficult to interpret in a biological context, making it challenging to link specific learned features to functional outcomes.

- **Nucleotide language models:** Nucleic acid language models provide context-

sensitive embeddings that capture long-range dependencies and subtle contextual patterns, allowing improved generalization and transfer between tasks. However, they require large-scale pretraining data and substantial computational resources. Moreover, their high-dimensional embeddings lack explicit structural information and remain inherently difficult to interpret, limiting biological interpretability without supplementary modeling.

- **GANs:** GANs excel at producing native-like RNA sequences that closely match the distribution of their training data. However, several important limitations remain. Training instability and mode collapse can reduce sequence diversity, and the complex and entangled structure of the learned latent space restricts precise control over specific functional attributes. Moreover, because GANs do not provide explicit sequence-to-sequence mappings, they are poorly suited to tasks that require optimization of existing sequences. Finally, linking generated designs to the underlying regulatory mechanisms is challenging due to limited interpretability.

- **GAs:** GAs provide a flexible, model-free framework for RNA sequence optimization, can be readily extended to multi-objective tasks, and can efficiently explore a large sequence space without requiring labeled training data [75]. However, they do not produce a trained model that can be reused, so each optimization problem must be solved from scratch. In addition, because GAs refine candidate populations rather than learning explicit sequence-to-sequence mappings, they are not suitable for tasks that involve direct optimization of existing sequences. The iterative search and fitness

evaluation process is also computationally expensive and may converge to local optima if population diversity is not maintained. Finally, even with substantial computational resources, the evolutionary search may not converge on the most functionally relevant regions of the sequence space.

- **Latent optimization:** Latent space optimization enables smooth, gradient-based navigation of the complex sequence space and can uncover interpretable trajectories toward improved functional properties. However, this approach does not produce a single model capable of directly mapping an input sequence to an optimized output; instead, each candidate must undergo an iterative and computationally expensive refinement process. It is also challenging to preserve sequence similarity or explore regions far from the training distribution without violating biological constraints. Moreover, the fidelity of the decoder strongly affects optimization outcomes, as decoding errors can lead to suboptimal or invalid sequences. Finally, the effectiveness of latent space optimization depends heavily on the structure and quality of the learned latent space, which may not capture all relevant functional dependencies.

The approaches discussed above offer important strengths, spanning predictive power in convolutional models, contextual understanding in language models, generative capabilities in adversarial and evolutionary algorithms, and the continuous exploration of sequence space through latent space optimization. However, each approach has its own shortcomings, such as limited incorporation of structural information, the inability to directly optimize existing sequences, difficulty constraining how the input sequence is modified, slow or

iterative optimization, and challenges in interpretability. In the following chapters, we present a unified framework that integrates the strengths of these approaches while addressing their limitations. Our model incorporates structural context, preserves similarity to original sequences, enables direct sequence-to-sequence optimization, and combines efficient training with fast inference. These features bring predictive accuracy, data-driven sequence optimization, and scalability together in a single system, enabling the rapid optimization of tens of thousands of sequences within minutes.

4 Graph Attention 5′ UTR Encoder

4.1 Introduction

This chapter presents the GAT encoder designed to model 5′ UTR sequences and their predicted secondary structures for estimating translation efficiency, quantified by MRL. A GNN-based model was selected for its ability to model both sequential relationships and long-range structural dependencies of the folded 5′ UTR. These properties were expected to support the effective reconstruction of sequences from the encoder's latent representations and were hypothesized to enhance the accuracy of translation efficiency prediction by incorporating 5′ UTR secondary structure graphs (see Section 2.2.5). Additionally, GNNs can be interpreted using graph explanation methods to reveal the features that influence model predictions.

4.2 Encoder Overview

The encoder represents each 5′ UTR sequence as a graph whose nodes are nucleotides and whose edges capture both the backbone adjacency and the base-pairing from the

secondary structure. Node features combine base identity, a biochemical flag, and positional information; edge attributes distinguish backbone versus base-pair links. A stack of Graph Attention Network v2 (GATv2) [76] layers produces node embeddings that are aggregated by Set2Set [77] into a graph embedding for scalar MRL prediction. Sections 4.2.2 and 4.2.3 provide detailed descriptions of the Graph Attention Networks (GAT and GATv2) and the Set2Set aggregation mechanism.

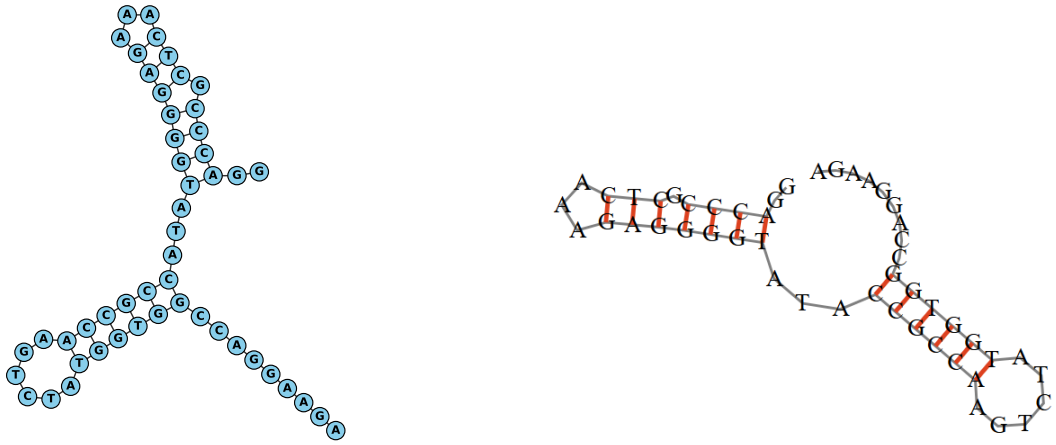
4.2.1 RNA Graph Construction and Feature Representation

4.2.1.1 RNA Graph Definition (nodes, edges, attributes)

RNA is represented as a graph in which each nucleotide is a node, and edges connect consecutive positions or base pairs derived from the ViennaRNA dot-bracket structure. Figure 4.1 illustrates this mapping for the same sequence. Details of the node features and edge attributes used by the model are given in the following sections.

4.2.1.2 Edge Types and Attributes

Two edge types are defined in the RNA graph: backbone (sequential) and base-pair. Backbone edges link nucleotide i to $i{+}1$ along the sequence; base-pair edges link nucleotides that pair in the predicted secondary structure. Each edge has a two-dimensional type attribute encoded as a one-hot vector: $[1, 0]$ for backbone and $[0, 1]$ for base-pair. Distinguishing edge types allows message passing to treat sequence adjacency and base pairing differently



(A) (B)

Figure 4.1: (A) RNA graph built from the ViennaRNA secondary structure prediction shown in (B) for a sequence from the dataset. Backbone edges indicate sequence adjacency (gray). Base-pair edges represent predicted pairing contacts and appear as ladder-like rungs (red). Visualizations were generated using NetworkX [78] and RNAplot from the ViennaRNA package [31].

to capture both local context and long-range dependencies.

4.2.1.3 Node Feature Vector

Each nucleotide node is represented by an 11-dimensional feature vector obtained by concatenating three components: (i) a 4-dimensional one-hot encoding of nucleic acid base identity (A, C, G, T); (ii) a 6-dimensional sinusoidal encoding of the base position in the 5′ UTR sequence (Section 4.2.1.4); and (iii) a biochemical class indicator distinguishing purines from pyrimidines (A/G $\rightarrow$ 0, C/T $\rightarrow$ 1). The resulting concatenation is shown in Figure 4.2.

Purine/Pyrimidine Indicator. Section 2.2.1 introduced the biochemical distinction

$$\Big[\underbrace{1\quad 0\quad 0\quad 0}_{\text{One-hot encoding (A/C/G/T)}}\qquad\underbrace{0.12\quad -0.88\quad 0.54\quad 0.33\quad -0.21\quad 0.77}_{\text{Sinusoidal positional encoding}}\qquad\underbrace{1}_{\text{Purine flag}}\Big]$$

Figure 4.2: Illustration of a node embedding vector formed by the concatenation of one-hot nucleotide encoding, sinusoidal positional encoding, and purine/pyrimidine flag.

between purines and pyrimidines. The purine/pyrimidine indicator encodes nucleotide base class information and is expected to help the model detect class-dependent patterns, including oligopyrimidine tracts or purine-rich segments where either A or G can appear. These patterns can influence translation efficiency [15].

4.2.1.4 Sinusoidal Positional Encoding

Sinusoidal positional encoding was originally introduced by Vaswani et al. [41] in the Transformer architecture as a parameter-free method to represent positions in a sequence. For a given position pos and positional encoding vector dimension i, the i-th component of the positional encoding vector is defined as

$$\mathrm{PE}(\mathrm{pos}, 2i) = \sin\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right), \quad \mathrm{PE}(\mathrm{pos}, 2i+1) = \cos\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right),$$

where d_{model} is the dimensionality of the positional encoding vector. Figure 4.3 shows how each dimension follows a sine or cosine wave with a distinct wavelength, supporting the encoding of positional information across multiple spatial scales.

This encoding scheme has several advantages over representing position with a single integer index or a one-hot vector. A scalar index makes the relative distance a nonlinear, difficult-to-learn function, while a one-hot representation fixes the sequence length and lacks any notion of distance between positions. In contrast, sinusoidal positional encoding uses multiple sine–cosine components at different wavelengths, producing smooth, bounded features in which nearby positions have similar representations. This multifrequency structure makes relative offsets easier for attention and message-passing layers to learn and enables the model to capture both short-range (short wavelength) and long-range (long wavelength) positional relationships.

GNNs are permutation-invariant with respect to node ordering. Consequently, when representing RNA sequences as graphs, nucleotide positions cannot be inferred from the graph structure alone. Incorporating positional encodings is therefore necessary to provide explicit information about the relative or absolute positions of nucleotides along the sequence.

The proposed encoder incorporates a six-dimensional sinusoidal positional encoding as part of the node feature embedding, corresponding to three sine–cosine frequency pairs. Unlike the original Transformer architecture, where positional encodings match the full embedding dimensionality to compensate for permutation-invariant self-attention [41], a compact sinusoidal positional encoding is sufficient in this context given the short length of $5'$ UTR sequences and the availability of structural information through graph connectivity. During development, this encoding proved to be the most influential feature: when removed, the model failed to learn meaningful patterns and the predictive performance collapsed (ablation results not shown). This observation highlights the essential role of

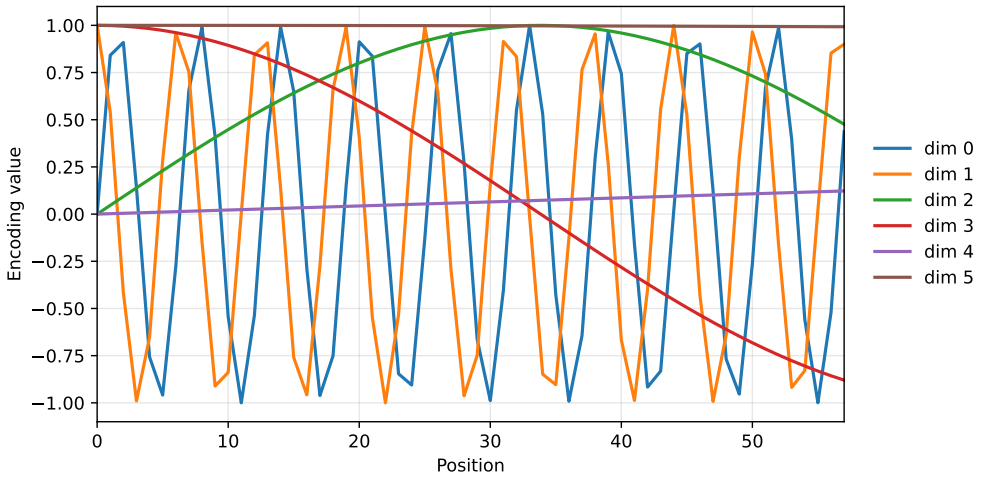


Figure 4.3: Sinusoidal positional encoding values for six dimensions ($d_{\text{model}} = 6$) across positions 0–60. Each curve corresponds to a distinct positional encoding dimension, illustrating how it varies as a function of sequence position.

positional information in enabling the GNN to capture sequence order and distance-dependent relationships.

4.2.1.5 Graph Building Algorithm

Base-pair indices are obtained from the ViennaRNA dot–bracket string using `RNA.ptable`, which produces a pair table. For each matched base pair (i, j), edges $i \to j$ and $j \to i$ are inserted. Likewise, bidirectional backbone edges are added between consecutive positions.

4.2.2 Graph Attention Networks: GAT and GATv2

GATs are message passing GNN layers that replace uniform aggregate functions (e.g., mean or max) with masked self-attention over each node's 1-hop neighborhood. Compared

to uniform-aggregation GNNs (e.g., Graph Convolutional Network (GCN)), GATs learn edge-specific attention weights so each node emphasizes informative neighbors while keeping per-head time complexity comparable to GCN-style layers. Formally, a single GAT attention head computes *message passing* as follows:

$$
\begin{aligned}
e_{ij} &= \mathrm{LeakyReLU}(a^{\top}[\, W\mathbf{h}_i \,\|\, W\mathbf{h}_j \,]), \\
\alpha_{ij} &= \mathrm{softmax}_{j\in\mathcal{N}_i}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik})}, \\
\mathbf{h}'_i &= \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\, W\mathbf{h}_j\right).
\end{aligned}
$$

With K attention heads, hidden layers typically *concatenate*:

$$
\mathbf{h}'_i = \left[\sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{(1)}\, W^{(1)}\mathbf{h}_j\right),\ \ldots,\ \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{(K)}\, W^{(K)}\mathbf{h}_j\right)\right],
$$

whereas the final layer usually *averages* before the task nonlinearity:

$$
\mathbf{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{(k)}\, W^{(k)}\mathbf{h}_j\right).
$$

Here, $\mathbf{h}_i \in \mathbb{R}^F$ are node features, $W \in \mathbb{R}^{F'\times F}$ is the weight matrix, $a \in \mathbb{R}^{2F'}$ is the attention vector, $\mathcal{N}_i$ denotes the 1-hop neighborhood of nodes (including i when self-loops are used), $[u\|v]$ is vector concatenation, and σ is an element-wise activation function. The cost per attention head is comparable to uniform aggregation of GNN layers, and the mechanism naturally handles variable-size neighborhoods and generalizes to unseen graphs [79].

GATv2 introduced by Brody, Alon, and Yahav [76] uses the same aggregation as GAT but changes the order of operations so that attention is truly query-conditioned (dynamic). Formally, a single head computes message passing as follows:

$$
\begin{aligned}
e_{ij} &= a^{\top}\, \phi\!\left(W\,[\mathbf{h}_i \| \mathbf{h}_j]\right), \\
\alpha_{ij} &= \mathrm{softmax}_{j\in\mathcal{N}_i}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik})}, \\
\mathbf{h}'_i &= \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\, W_v\, \mathbf{h}_j\right).
\end{aligned}
$$

Here, $\mathbf{h}_i \in \mathbb{R}^F$ are node features, $W \in \mathbb{R}^{F''\times 2F}$ is the weight matrix, $W_v \in \mathbb{R}^{F'\times F}$ is the value projection, $a \in \mathbb{R}^{F''}$ is the attention vector, ϕ is a pointwise nonlinear function, and $[u\|v]$ denotes concatenation.

Multihead attention uses the same scheme as GAT: concatenate intermediate heads in hidden layers and average heads in the last layer before the output activation.

Why changing the order of operations matters: In standard GAT, the attention score for edge (i, j) can be written as

$$
e_{ij} = \mathrm{LeakyReLU}(a_1^{\top} W\,\mathbf{h}_i + a_2^{\top} W\,\mathbf{h}_j),
$$

which produces a *query-independent (static)* neighbor ranking: Each neighbor j has a node-specific score $s_j = a_2^{\top} W\,\mathbf{h}_j$. As a result, the ordering of neighbors is identical for every

query node i. In contrast, GATv2 calculates the attention score as

$$e_{ij} = a^\top \phi(W[\mathbf{h}_i \| \mathbf{h}_j]),$$

so the score depends on the pair $(\mathbf{h}_i, \mathbf{h}_j)$ jointly. Consequently, the preferred neighbor can change with the query node i, yielding *dynamic* (query-conditioned) attention.

Empirically, GATv2 is more robust to structural noise and often outperforms GAT across standard benchmarks [76]. Therefore, in this work, GATv2 message-passing layers are used in the GNN encoder. Details of the GATv2 regression architecture are given in Section 4.2.4.

4.2.3 Set2Set Graph Readout and Regression Layer

A graph readout function aggregates node-level embeddings into a single vector representation of the entire graph for use in downstream tasks. Set2Set is a readout method that performs this aggregation through an iterative, permutation-invariant attention mechanism. Unlike uniform pooling methods such as mean or sum, Set2Set uses a recurrent controller to generate query vectors that attend to all nodes across processing steps. At each step, the model produces weighted combinations of node embeddings that can shift focus over successive iterations. This mechanism enables the encoder to emphasize biologically important motifs and reduce the influence of less informative regions. The method was originally proposed in the *Order Matters: Sequence to Sequence for Sets* paper by Vinyals et al. [77].

Formally, let x_i denote the embeddings of the nodes in the last layer of the GNN encoder. Set2Set introduces a recurrent query vector q_t, updated by an LSTM controller, to calculate attention on the set of node embeddings. At each processing step $t \in \{1, \ldots, T\}$, the operations are implemented in PyTorch Geometric [80] using the following equations:

$$q_t = \mathrm{LSTM}(q^*_{t-1}), \quad \alpha_{i,t} = \mathrm{softmax}(x_i \cdot q_t),$$

$$r_t = \sum_{i=1}^{N} \alpha_{i,t}\, x_i, \quad q^*_t = [q_t \,\|\, r_t].$$

Here, $\alpha_{i,t}$ are the attention weights that indicate the relative contribution of each node to the readout, r_t is the resulting weighted combination of node embeddings, and q^*_t is the concatenation of the query and the readout, which is fed back into the LSTM in the next step. The final graph-level embedding is taken as the last state q^*_T, which has dimensions $2 \times$ hidden dimensions, independent of the number of nodes in the graph.

During development of the GNN encoder, Set2Set pooling outperformed other graph readout methods including global attention, mean, and max pooling (comparison results not shown) and was therefore chosen as the graph readout mechanism. The output was then passed through a fully connected regression layer, implemented as a linear transformation from $2 \times$ hidden dimensions to a single scalar value, to produce final MRL predictions. A detailed evaluation of model performance is presented in Section 4.5.

4.2.4 GATv2 Regression Architecture

A three-layer GATv2 encoder with Set2Set readout and a scalar regressor is used to predict MRL. Inputs are 11-dimensional node features and 2-dimensional edge attributes on a directed, bidirectional graph.

Layer 1 (GATv2Conv; attention heads = 4). Input dimension 11; out_channels = 128 per head; head concatenation enabled, yielding a 512-dimensional node embedding. `edge_dim` = 2; internal attention dropout 0.2; followed by ReLU and feature dropout 0.2.

Layer 2 (GATv2Conv; attention heads = 4). Input dimension 512; out_channels = 128 per head; head concatenation enabled, yielding a 512-dimensional node embedding. `edge_dim` = 2; internal attention dropout 0.2; followed by ReLU and feature dropout 0.2.

Layer 3 (GATv2Conv; attention heads = 1). Input dimension 512; one attention head, producing a 128-dimensional node embedding. `edge_dim` = 2; followed by ReLU and feature dropout 0.2.

Readout. `Set2Set` with hidden size 128 and `processing_steps` = 5, yielding a 256-dimensional graph vector.

Regressor. Linear layer mapping 256 to 1.

Notes. No residual connections or normalization layers are used; Batching follows the PyG `batch` vector.

4.3 Dataset

This thesis utilizes data from GEO `GSM3130435`, part of the `GSE114002` series generated in the high-throughput polysome profiling study by Sample et al. [14, 81]. The dataset contains polysome profiling measurements for 300,000 randomly synthesized, 50-nucleotide 5′ UTRs cloned upstream of an eGFP coding sequence in HEK293T cells. Translation efficiency is quantified by sequencing UTRs from ribosome-associated fractions and computing the MRL as the weighted mean of fraction-specific abundances.

4.4 Training Setup

Objective and optimizer. The GATv2 regressor was trained using Mean Squared Error (MSE) and optimized with Adam at a learning rate of 1×10^{-3}. The number of training epochs was experiment-specific and reported with the corresponding results.

Batching. Batches of 64 graphs were created using the `torch_geometric` `DataLoader`. The training set was shuffled every epoch, while the test set order was preserved.

Target scaling. The MRL target was standardized with a z-score transform using `StandardScaler`. The scaler was fit on the training partition only ($\mu_{\text{train}}, \sigma_{\text{train}}$) and the same parameters were applied to validation/test targets to avoid leakage. Training optimized MSE on the standardized targets.

Train–test splits. Two strategies were used in the experiments:

(a) *Random split by sequence.* Disjoint training and test sets were created at the sequence level using a fixed random seed.

(b) *Read-count split.* Following Sample et al. [14], the dataset was sorted by per-sequence read count; the top 20,000 sequences were assigned to the test set and the bottom 260,000 to the training set. This approach ensures that the test set contains the most reliable measurements, as high read counts provide more accurate estimates of MRL and reduce the impact of experimental noise during evaluation.

All splits ensured that no sequence appeared in more than one partition.

4.5 Experiments and Results

4.5.1 Main Results

4.5.1.1 Random Split

The dataset was split into 70% training, 15% validation, and 15% test. Early stopping on validation loss (patience 10, minimum improvement 10^{-4}) was applied; the remaining training settings follow Section 4.4.

Training dynamics. Validation loss decreased rapidly during the first 10 epochs and then plateaued; early stopping triggered at epoch 48. Validation metrics stabilized around $r \approx 0.94$ and $R^2 \approx 0.87$ (Fig. 4.4).

Test performance. The model achieved $R^2 = 0.88$ and Pearson's $r = 0.94$ (Fig. 4.5).

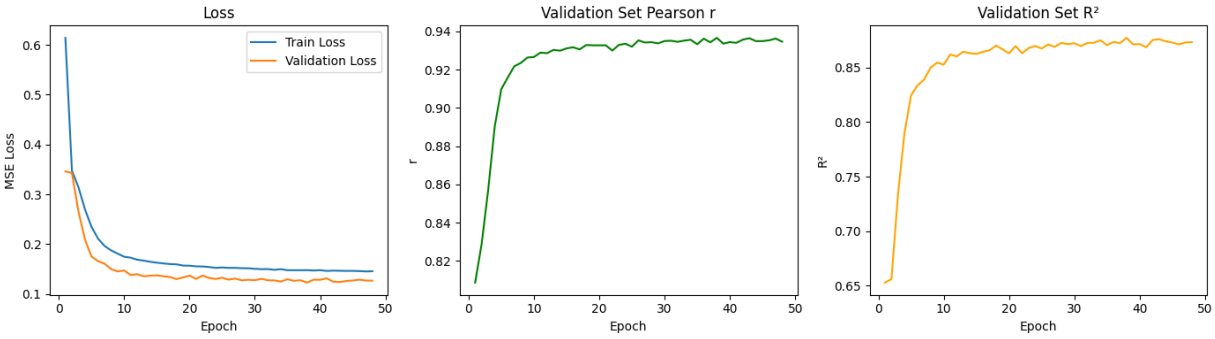


Figure 4.4: Training and validation curves for the random split: MSE loss (left) and validation metrics r (middle) and R^2 (right). Early stopping occurs at epoch 48.

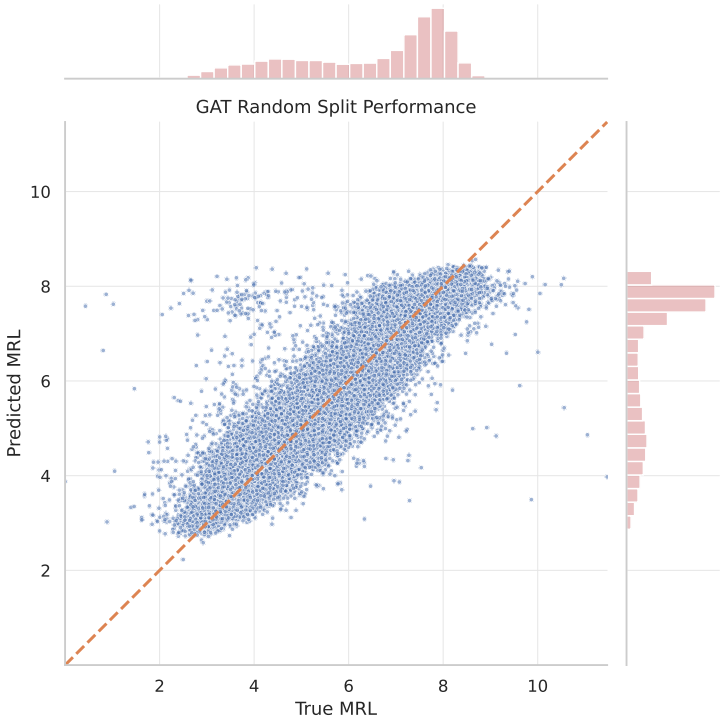


Figure 4.5: Test-set predictions versus ground truth for the random split; the dashed line indicates $y = x$.

4.5.1.2 Read-Count Split (Sample et al. Protocol)

Following Sample et al. [14], 5′ UTR sequences were sorted by total sequencing reads in non-ascending order. The top 20,000 sequences were reserved for testing, and the remaining 260,000 were used for training. As justified in their study, high sequencing reads yield higher fidelity (lower noise) estimates of MRL.

The GATv2 encoder was trained for 10 epochs. This number of epochs was chosen from random split learning curves, where the validation loss plateaued and began to increase

marginally after ~ 10 epochs; the remaining training settings follow Section 4.4.

Test performance. In the read-count split, the model achieved Pearson's $r = 0.95$ and $R^2 = 0.90$. Figure 4.6 shows the scatter plot of predicted versus observed MRL on the high-read-count test set.

For comparison, under a random split, the corresponding metrics were $r = \mathbf{0.94}$ and $R^2 = \mathbf{0.88}$, performing essentially as well as the read-count split. This mirrors the observation by Sample et al. that their CNN model performance under a random split is nearly as good as the read-count split [14].

The read count split scheme was adopted for final reporting and training of the multi-task autoencoder in Chapter 5 to maximize confidence when results may inform wet-lab validation.

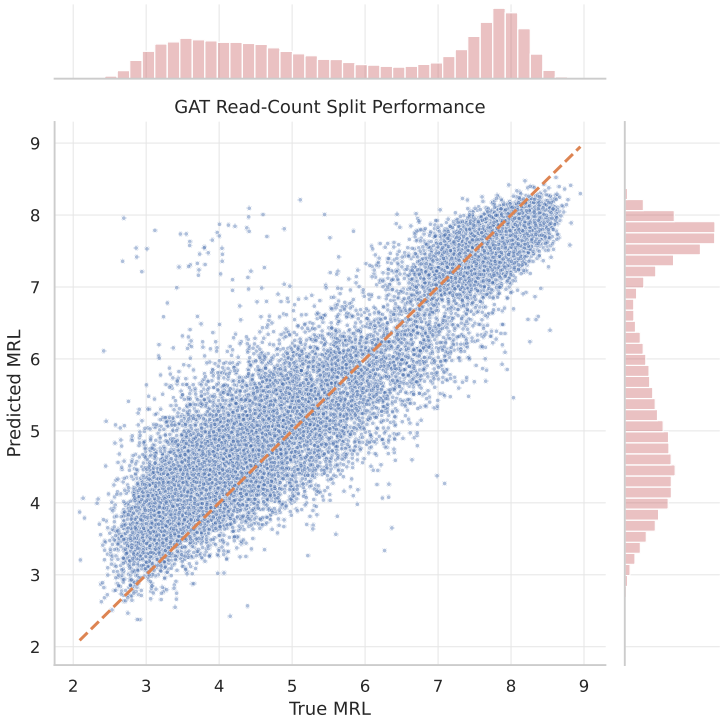


Figure 4.6: Test predictions vs. ground truth under the read-count split (Sample et al. protocol); the dashed line shows $y = x$.

4.5.2 Comparison with a Reference CNN Predictor

To establish a performance baseline, we implemented the Optimus 5-Prime CNN predictor originally proposed by Sample et al. [14], with architectural modifications adopted from Tang et al. [73] to increase model capacity. The network consists of four sequential 1-D convolutional layers (kernel size 8, 160 filters, reduced to 80 in the final layer), each followed by ReLU activation, batch normalization, and dropout. The resulting feature representations are passed to a fully connected regression head (latent dimension 80) for

MRL prediction. After partitioning the dataset using the read-count split protocol, training was performed for 3 epochs with a batch size of 64 using the Adam optimizer ($lr = 10^{-3}$) to minimize the MSE loss.

The CNN predictor achieved high predictive accuracy on the held-out test set ($r = 0.97$, $R^2 = 0.94$). These results support using the model as a benchmark for quantifying MRL gains in Chapter 6, where the performance of GNNs for designing 5′ UTR sequences is evaluated.

4.5.3 Generalization Across Coding Sequences

A key question is whether a model that maps 5′ UTR sequence to translation efficiency (MRL) learns rules that transfer across different coding sequences. Following Sample et al. [14], the GATv2 encoder was trained on the eGFP library and evaluated zero-shot on an independent mCherry library that used different sucrose gradient conditions for polysome profiling. This deliberate distribution shift tests if learned signals such as uAUGs, uORFs, Kozak context, and secondary structure generalize beyond a single reporter gene, which is important for mRNA design where the coding sequence can change.

The GATv2 encoder trained on the eGFP dataset was evaluated on a library of 269 114 random 50-nucleotide 5′ UTRs upstream of the mCherry coding sequence. The pretrained model's raw MRL predictions showed strong correlation with measurements (Pearson's $r = 0.84$) but a large negative coefficient of determination ($R^2 = -11.18$), consistent with a cross-assay scale mismatch. Using an 80/20 calibration–test split, a post hoc regression

calibrator was fitted on the 80% calibration subset to absorb the cross-assay scale differences. The resulting mapping was then applied to the held-out 20% test split for evaluation. After calibration, the model explained 70% of variance ($R^2 = 0.70$) with unchanged correlation ($r = 0.84$).

For reference, the CNN encoder from Sample et al. was trained on the eGFP library and evaluated on the mCherry library using the same procedure; it achieved $R^2 = 0.72$ and $r = 0.85$.

4.5.4 Evaluation on Variable-Length 5′ UTRs

To evaluate the performance of the GATv2 encoder on variable-length 5′ UTRs, the experimental setup introduced by Sample et al. [14] was replicated. The encoder was trained for 40 epochs using the Adam optimizer with a learning rate of 1×10^{-3} on a random library of 79,400 5′ UTRs spanning 25–100 nucleotides. Two independent test sets of (i) 7,600 randomly generated sequences and (ii) 7,600 human-derived 5′ UTRs were used to evaluate the trained model. Each nucleotide length was represented by 100 samples in both the random and human test sets.

To accommodate variable-length 5′ UTRs, Sample et al. [14] fixed the input width of their Optimus 5-Prime CNN encoder to 100 nucleotides and padded shorter sequences with zeros. This approach allowed uniform input size but introduced artificial signals unrelated to biological properties. In contrast, GNNs naturally handle variable-length inputs without padding, thereby avoiding the introduction of non-biological signals.

The GATv2 encoder achieved an overall performance of $R^2 = 0.66$ and $r = 0.83$ when all sequences in the test set were combined. Following the approach of Sample et al. [14], the test sequences were partitioned into bins of 25–44, 45–64, 65–84, and 85–100 nucleotides. Across these bins, the model achieved R^2 values ranging from 0.64 to 0.67 and correlation coefficients (r) between 0.81 and 0.83 (Table 4.1). Notably, the model maintained stable predictive accuracy across all bins, with no substantial drop in performance as 5′ UTR length increased. Moreover, the GATv2 encoder consistently achieved higher performance on human-derived sequences compared to random 5′ UTRs in all bins.

For comparison, the Optimus 5-Prime CNN encoder trained on the same variable-length 5′ UTR dataset achieved higher performance on random sequences (R^2 ranging from 0.79 to 0.87) than on human-derived 5′ UTRs (R^2 ranging from 0.72 to 0.79), showing the reverse trend of the GNN results. Moreover, the performance of the CNN dropped more noticeably with increasing sequence length, particularly for human-derived 5′ UTRs (Figure 4.7).

Although the CNN encoder performed better in this experiment, zero-padding shorter sequences implicitly exposed sequence length. As a result, the CNN model relied on sequence length as a predictive cue for MRL, which reduced accuracy for longer inputs. In contrast, the GATv2 encoder preserved biological fidelity by avoiding length-encoded artifacts and maintained stable accuracy on human-derived 5′ UTRs. This observation is particularly promising for applications involving human transcripts or other biologically complex 5′ UTRs. Future improvements may be achieved by training the GNN on larger and more diverse datasets and incorporating richer structural and contextual representations of 5′ UTRs.

Bin (nt)	Random (R^2, r)	Human (R^2, r)	Overall (R^2, r)
25–44	0.65, 0.82	0.69, 0.84	0.67, 0.83
45–64	0.64, 0.81	0.69, 0.85	0.66, 0.83
65–84	0.59, 0.77	0.71, 0.86	0.65, 0.81
85–100	0.58, 0.77	0.68, 0.84	0.64, 0.81

Table 4.1: GATv2 encoder performance on variable-length test sets. R^2 and r values for MRL prediction are reported for random and human 5′ UTRs across length bins.

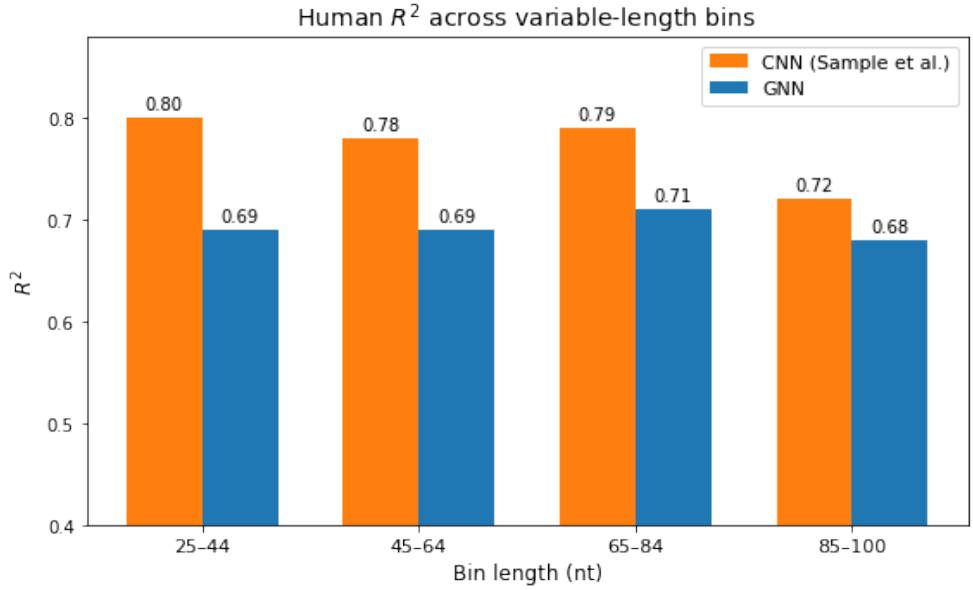


Figure 4.7: Comparison of R^2 values for MRL prediction across variable-length human 5′ UTRs between zero-padded CNN and GATv2 encoders.

4.5.5 Ablation Studies

4.5.5.1 Effect of Training Set Size

The initial train/test partition followed the read count protocol of Sample et al. [14]: sequences were sorted by total reads in descending order, the top 20,000 were held out as a fixed test set, and the remaining 260,000 formed the training pool. From this pool, random subsets of size 10k–260k (step 10k) were drawn. Each configuration was trained for 10 epochs with the training settings discussed in Section 4.4. Performance was evaluated on the common test set using R^2 and Pearson's r. The trend is summarized in Fig. 4.8, with exact values in Table 4.2.

The results show steady gains with more data followed by saturation. With 10k samples, the model achieved $R^2 \approx 0.62$ and $r \approx 0.84$; at 100k this rose to $R^2 \approx 0.86$ and $r \approx 0.93$. Beyond $\sim$200k, improvements were marginal, stabilizing around $R^2 \approx 0.90$ and $r \approx 0.95$, with small nonmonotonic fluctuations attributable to random sampling and the limited number of training epochs.

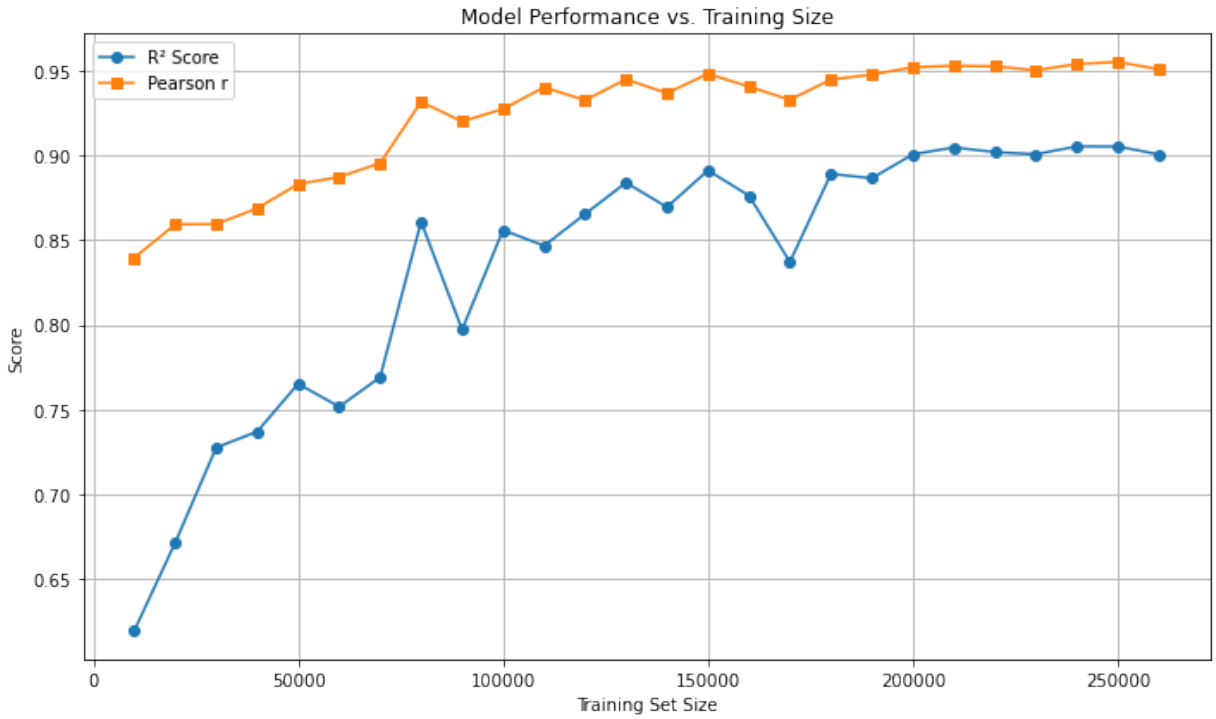


Figure 4.8: Model performance versus training-set size for the GATv2 regressor (random split). Curves report test-set R^2 (circles) and Pearson's r (squares) as the number of training examples increases; both improve with more data and begin to level off beyond ∼200,000 sequences.

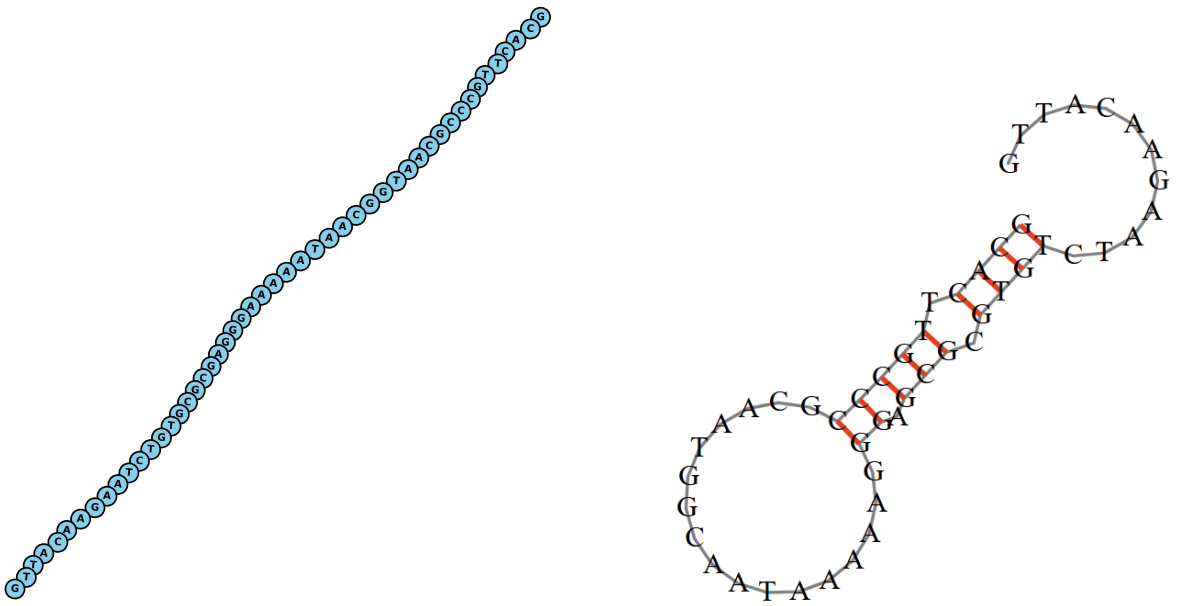
Train size	$\mathbf{R^2}$	**Pearson** r
260,000	0.90	0.95
250,000	0.91	0.96
240,000	0.91	0.95
230,000	0.90	0.95
220,000	0.90	0.95
210,000	0.90	0.95
200,000	0.90	0.95
190,000	0.89	0.95
180,000	0.89	0.94
170,000	0.84	0.93
160,000	0.88	0.94
150,000	0.89	0.95
140,000	0.87	0.94
130,000	0.88	0.95
120,000	0.87	0.93
110,000	0.85	0.94
100,000	0.86	0.93
90,000	0.80	0.92
80,000	0.86	0.93
70,000	0.77	0.90
60,000	0.75	0.89
50,000	0.77	0.88
40,000	0.74	0.87
30,000	0.73	0.86
20,000	0.67	0.86
10,000	0.62	0.84

Table 4.2: Effect of training-set size on GATv2 regressor performance (random split). Metrics shown are R^2 and Pearson's r, rounded to two decimals.

4.5.5.2 Secondary Structure Contribution

The role of secondary structure information in encoder performance was investigated using *sequence-only* RNA graphs. Backbone edges were kept, and base-pairing edges were removed to isolate the effect of secondary structure (Figure 4.9). The train/test split, the number of training epochs, and other training parameters followed the read-count split setup (Sample et al. protocol; Section 4.5.1.2).

Test performance. Ablating secondary structure information from RNA graphs had no impact on encoder accuracy: the GATv2 encoder achieved $R^2 = 0.90$ and Pearson's $r = 0.95$, identical to the results obtained when training the encoder using secondary structure graphs. However, as discussed in Section 5.8.2, removing secondary structure markedly reduces the LSTM decoder's sequence reconstruction accuracy. Thus, secondary structure is essential for faithful decoding, but not for encoder-side MRL prediction.



(A) (B)

Figure 4.9: **(A)** Sequence-only RNA graph in the secondary-structure ablation: the graph is built with backbone (adjacent-nucleotide) edges only. Base-pair edges are removed, eliminating long-range contacts. **(B)** Predicted ViennaRNA secondary structure for the same 5′ UTR sequence, shown for reference.

4.6 Summary

This chapter established the GATv2 encoder as a model for predicting translation efficiency. The encoder accurately predicted MRL by integrating structural and contextual features of 5′ UTRs. The following chapter extends the encoder into a multitask autoencoder that jointly performs MRL prediction and sequence reconstruction.

5 Multitask 5′ UTR Autoencoder

5.1 Introduction

This chapter introduces the multitask autoencoder designed for simultaneous 5′ UTR sequence reconstruction and prediction of translation efficiency, measured as MRL. As illustrated in Figure 5.1, the proposed model consists of three main components:

1. **GNN Encoder**: encodes 5′ UTR sequences into latent embeddings that capture structural features relevant to MRL.
2. **MRL Prediction Head**: predicts MRL from latent embeddings.
3. **Autoregressive LSTM Decoder**: reconstructs the input 5′ UTRs from latent embeddings.

The autoregressive decoder forms the basis for generative modeling aimed at designing sequences with optimized translation initiation efficiency. Chapter 6 describes and evaluates the Reinforcement Learning strategies used to fine-tune the decoder for this optimization.

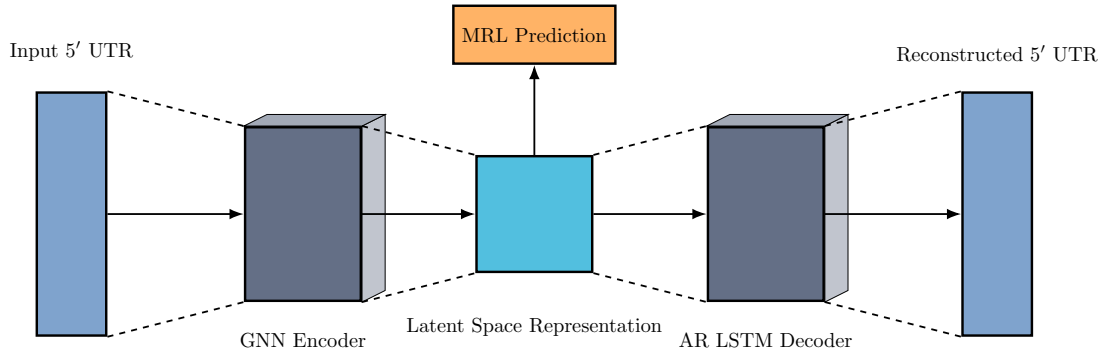


Figure 5.1: Architecture of the multitask autoencoder. The GNN encoder compresses 5′ UTR sequences into latent representations. MRL is predicted from the latent representation using a fully connected layer. The autoregressive LSTM decoder decompresses the latent representation to reconstruct the original sequence.

5.2 Input Representation

Each 5′ UTR sequence is represented as a graph, following the construction described in Section 4.2.1. Nucleotides are represented as nodes with features including one-hot base identity, sinusoidal positional encodings, and a purine/pyrimidine indicator. Edges represent either backbone connections or predicted base-pair interactions and carry a single binary type attribute: $[1, 0]$ for backbone edges and $[0, 1]$ for base-pair edges. These graphs are used as input to the GNN encoder of the multitask autoencoder.

5.3 Architecture

5.3.1 Encoder Architecture

The encoder adopts the three-layer GATv2 architecture with Set2Set readout described in Section 4.2.4. This architecture uses multihead attention to integrate node and edge features into latent representations used for downstream MRL prediction and sequence reconstruction.

5.3.2 Auxiliary MRL Head

The auxiliary MRL prediction head consists of a single linear regression layer applied to the latent representations produced by the encoder. It is identical to the regression layer described in Section 4.2.4 and outputs the predicted MRL.

5.3.3 Autoregressive LSTM Decoder

The autoregressive LSTM decoder reconstructs 5′ UTR sequences token by token from the latent representations produced by the encoder. It consists of:

1. **Latent projection layer**: a fully connected layer projects the concatenation of the latent embedding z and the predicted MRL $\hat{m}$ into the initial hidden state h_0 of the

LSTM:

$$h_0 = W_h[z \,\|\, \hat{m}] + b_h,$$

where $[z \,\|\, \hat{m}]$ denotes concatenation, W_h is the learnable projection matrix, and b_h is the learnable bias vector used to construct the initial hidden state h_0. The initial cell state c_0 is initialized to zeros. For details on the roles of hidden and cell states in LSTMs, refer to Section 2.3.1.2.

2. **Stacked LSTM layers**: a two-layer LSTM with a hidden size of 256 autoregressively models sequence dependencies. At each decoding step t, the LSTM updates the recurrent states (h_t, c_t) using the current token embedding x_t:

$$(h_{t+1}, c_{t+1}) = \text{LSTM}(x_t, (h_t, c_t)).$$

The updated hidden state h_{t+1} is used to predict the next nucleotide.

3. **Output projection layer**: a fully connected layer followed by a softmax maps the hidden state h_t to a four-class categorical distribution over nucleotides $\{A, C, G, T\}$.

Decoding begins with a special start token $\langle\text{START}\rangle$. At each step, the decoder predicts the most probable nucleotide, one-hot encodes it, and uses this as the next input token x_{t+1}. This iterative process continues until the full sequence length L is reached.

During decoder fine-tuning in Chapter 6, token selection shifts from a greedy argmax operation to stochastic sampling from the categorical distribution over nucleotides. In

addition, DAP-regularized fine-tuning evaluates a frozen copy of the decoder (the prior) on the sequences sampled by the trainable decoder (the agent). The prior operates under teacher forcing, meaning that at each time step, the current token (nucleotide) from the agent's sampled sequence is used as the next input. Using the probability assigned to that token, this setup allows computing the prior log-likelihood of the sampled sequence.

5.4 Objectives

The multitask autoencoder is optimized using two complementary objectives: sequence reconstruction and translation efficiency prediction. Together, these objectives encourage the encoder to learn latent representations that capture both structural features of 5′ UTR sequences and information relevant to translation efficiency.

5.4.1 Sequence Reconstruction Objective

The first objective ensures that the decoder learns to accurately reconstruct the original 5′ UTR sequence from its latent representation. At each decoding step t, the model predicts a probability distribution $\hat{\mathbf{y}}_t$ over the four nucleotides $\{A, C, G, T\}$, and the reconstruction loss is defined as

$$\mathcal{L}_{\text{seq}} = -\frac{1}{L} \sum_{t=1}^{L} \sum_{k=1}^{4} \mathbf{1}[y_t = k] \, \log \hat{y}_{t,k},$$

where L is the sequence length, y_t is the true nucleotide label at position t, $\hat{y}_{t,k}$ is the predicted probability of nucleotide k at step t, and $\mathbf{1}[y_t = k]$ is an indicator function selecting the correct nucleotide's log-probability. This formulation is equivalent to the Categorical Cross-Entropy (CE) loss commonly used for multi-class classification tasks.

Minimizing $\mathcal{L}_{\text{seq}}$ encourages the decoder to assign high probabilities to the correct nucleotides, resulting in faithful reconstruction of input sequences.

5.4.2 Translation Efficiency Prediction Objective

The second objective ensures that the encoder learns embeddings predictive of translation efficiency, measured as MRL. A regression layer maps each latent embedding to a predicted $\hat{m}_i$, and the MSE between predicted and true MRL values is minimized:

$$\mathcal{L}_{\text{mrl}} = \frac{1}{B} \sum_{i=1}^{B} (\hat{m}_i - m_i)^2 ,$$

where B is the batch size, $\hat{m}_i$ is the predicted MRL for the i-th sequence, and m_i is the corresponding ground-truth MRL. Minimizing $\mathcal{L}_{\text{mrl}}$ encourages the encoder to capture features relevant to translation efficiency alongside structural information.

5.4.3 Combined Multitask Objective

The total training objective combines sequence reconstruction and translation efficiency prediction into a single multitask loss:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{seq}} \mathcal{L}_{\text{seq}} + \lambda_{\text{mrl}} \mathcal{L}_{\text{mrl}},$$

where $\lambda_{\text{seq}} = 5.0$ and $\lambda_{\text{mrl}} = 1.5$ are weighting coefficients chosen empirically to balance the contributions of both tasks. This joint optimization shapes a latent space that supports accurate sequence reconstruction and translation efficiency prediction.

5.5 Dataset

All experiments in this chapter use the `GSM3130435` dataset described in Section 4.3.

5.6 Training Procedure

Following Sample et al. [14], the dataset was sorted in non-ascending order by sequence read counts. Using the same data-splitting strategy employed by Tang et al. [73] when training their autoencoder, the top 20,000 sequences were reserved for testing, sequences from index 20,000 to 200,000 were used for training, and the remaining sequences beyond index 200,000 formed the validation set.

A StandardScaler was fitted on the MRL values of the training set and used to normalize the MRL values across the entire dataset. The model was trained for a total of 40 epochs using the Adam optimizer with a learning rate of 1×10^{-3} and a batch size of 128. The model was optimized using the combined multitask loss described in Section 5.4. After each epoch, performance on the validation set was monitored using MSE for MRL prediction and cross-entropy loss for sequence reconstruction.

5.7 Evaulation Metrics

The multitask autoencoder was evaluated on the held-out test set using metrics specific to each learning objective. For the MRL prediction task, performance was assessed using Pearson's correlation coefficient r and the R^2 coefficient of determination (See section 2.4.1). For the sequence reconstruction task, performance was measured using token-level accuracy, defined as the proportion of nucleotides correctly reconstructed across all positions in the output sequences.

5.8 Experiment and Results

5.8.1 Main Results

The model achieved a Pearson correlation coefficient of $r = 0.94$ and an R^2 value of 0.86 for the MRL prediction task. These results are comparable to the results obtained when

the encoder is trained individually using the same test set splitting strategy described in Section 4.5.1.2. For the sequence reconstruction task, the model achieved a token-level accuracy of 99%, demonstrating near-perfect recovery of input sequences from the learned latent representations.

5.8.2 Secondary Structure Ablation Study

Replicating the encoder ablation setup from Section 4.5.5.2, we repeated the experiment for the multitask autoencoder using secondary-structure-ablated RNA graphs. All other settings, including the data splitting strategy, loss functions, optimizer, and training configuration, remained unchanged from Section 5.6.

For the MRL prediction task, the model achieved a Pearson correlation of $r = 0.95$ and $R^2 = 0.90$, comparable to the results obtained with full secondary structure graphs and consistent with the encoder ablation study in Section 4.5.5.2. This finding confirms that RNA secondary structure has little impact on MRL prediction.

In contrast, the model achieved only 67% token-level accuracy in sequence reconstruction. During training, the reconstruction loss on both training and validation sets plateaued after approximately 20 epochs and showed no further improvement for the remainder of the 40 epochs (Figure 5.2). The validation reconstruction loss was slightly lower than the training loss because dropout was active during training but disabled during evaluation. Recomputing the training loss in evaluation mode yielded nearly identical curves (data not shown).

These results suggest that incorporating RNA secondary structure improves sequence reconstruction performance.

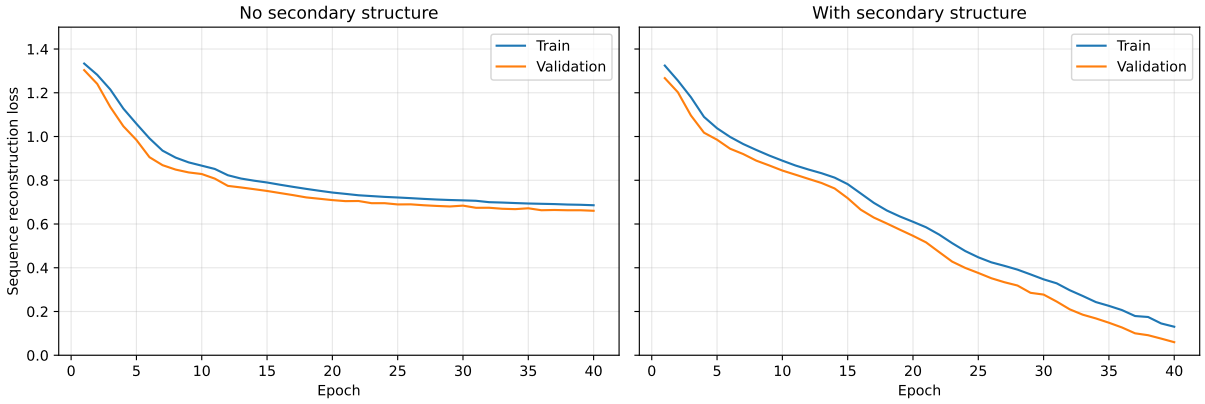


Figure 5.2: Training and validation sequence reconstruction loss for autoencoders trained on sequence-only graphs (left) and secondary structure graphs (right).

6 Reinforcement Learning for 5′ UTR Optimization

6.1 Introduction

This chapter presents Reinforcement Learning (RL) strategies to optimize 5′ UTR sequences for enhanced translation efficiency, quantified by MRL. Building upon the pre-trained multi-task autoencoder from Chapter 5, we keep the GNN encoder frozen while the autoregressive LSTM decoder is fine-tuned to generate sequences predicted to have higher MRL.

Two fine-tuning strategies are explored: (i) REINFORCE-based fine-tuning and (ii) DAP-regularized fine-tuning.

This dual setup allows us to evaluate the trade-off between aggressive optimization for MRL and maintaining biological plausibility.

6.2 Decoding Strategies in Pre-training and Fine-Tuning

During pretraining, the autoregressive LSTM decoder learns to reconstruct the original $5'$ UTR sequences from GNN-derived latent representations. At each timestep, the decoder outputs four logits corresponding to the nucleotide alphabet $\{A, C, G, T\}$, applies a softmax to obtain probabilities, and selects the nucleotide with the highest predicted probability (argmax decoding). This deterministic decoding strategy ensures accurate sequence reconstruction and prepares the model for downstream fine-tuning.

In contrast, during fine-tuning, the decoder adopts a stochastic sampling strategy. At each timestep, a nucleotide is sampled from the predicted probability distribution to introduce variability in the generated sequences. This approach promotes the exploration of alternative sequence candidates and ensures that the gradient estimates remain unbiased under the REINFORCE and DAP objectives (Figure 6.1).

6.3 REINFORCE-Based Fine-Tuning

REINFORCE (Section 2.3.2.1) is a policy-gradient method used to fine-tune the decoder parameters θ by maximizing the expected reward, where the reward $R(T)$ corresponds to the predicted MRL of a generated sequence T.

During fine-tuning, the decoder defines a stochastic policy π_θ over the nucleotide alphabet

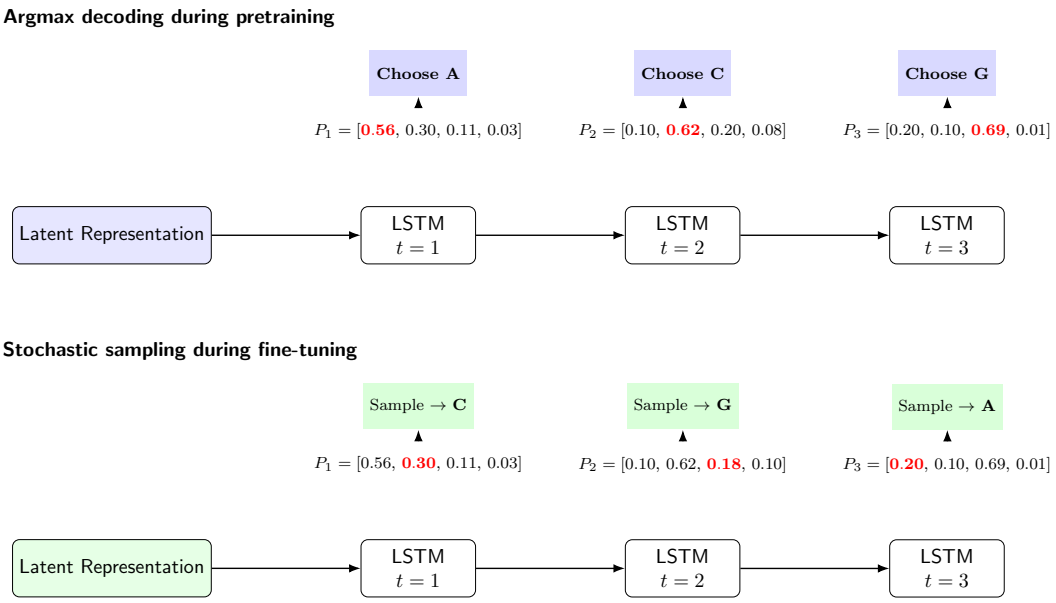


Figure 6.1: Decoding strategies during pretraining and fine-tuning.

$\{A, C, G, T\}$. Following the chain rule of probability, the probability of generating a sequence $T = \{t_1, t_2, \ldots, t_\ell\}$ of length ℓ is expressed as a product of conditional probabilities:

$$\pi_\theta(T) = \prod_{i=1}^{\ell} \pi_\theta(t_i \mid t_{<i}),$$

where t_i is the nucleotide generated at position i, $t_{<i} = \{t_1, t_2, \ldots, t_{i-1}\}$ denotes all previously generated nucleotides, and $\pi_\theta(t_i \mid t_{<i})$ is the probability of generating t_i conditioned on the prefix $t_{<i}$. To apply the log-derivative trick, REINFORCE operates on log-probabilities, yielding $\log \pi_\theta(T) = \sum_{i=1}^{\ell} \log \pi_\theta(t_i \mid t_{<i})$. This log-likelihood is used in the REINFORCE loss to update the decoder parameters.

At each fine-tuning step, a batch of B sequences $\{T^{(j)}\}_{j=1}^{B}$ is sampled from the decoder.

Each sequence is scored by the frozen encoder to obtain the predicted reward $R(T^{(j)})$. The REINFORCE loss is then defined as

$$\mathcal{L}_{\mathrm{RL}} = -\frac{1}{B}\sum_{j=1}^{B}\Big(R(T^{(j)}) - b\Big)\sum_{i=1}^{\ell}\log \pi_\theta(t_i^{(j)} \mid t_{<i}^{(j)}),$$

where $\mathcal{L}_{\mathrm{RL}}$ is the reinforcement learning loss minimized during training, B is the batch size, $T^{(j)} = \{t_1^{(j)}, \ldots, t_\ell^{(j)}\}$ is the j-th generated sequence, $t_i^{(j)}$ is the nucleotide generated at position i of $T^{(j)}$, $\pi_\theta(t_i^{(j)} \mid t_{<i}^{(j)})$ is the conditional probability of generating $t_i^{(j)}$, $R(T^{(j)})$ is the predicted reward, and b is an exponential moving-average baseline used to reduce variance in the policy gradient estimate.

Moving-average Baseline. To stabilize training, the baseline b is maintained as an exponential moving average (EMA) of past rewards:

$$b = \beta b + (1-\beta)\,\bar{R}, \qquad \bar{R} = \frac{1}{B}\sum_{j=1}^{B} R(T^{(j)}),$$

where $\bar{R}$ is the mean reward of the current batch, and $\beta \in (0,1)$ controls the smoothness of updates. A larger β results in slower updates and lower variance, while a smaller β allows the baseline to react more quickly to recent rewards. Using the difference $R(T^{(j)}) - b$, known as *advantage*, encourages sequences with rewards above the baseline and discourages those below, while keeping the gradient estimate unbiased.

REINFORCE-based fine-tuning allows the decoder to learn a policy that favors generat-

ing sequences predicted to have higher translation efficiency. Since the encoder remains frozen and only provides the reward signal, the gradients propagate exclusively through the decoder. The REINFORCE framework naturally handles the discrete nature of nucleotide generation.

6.4 DAP-Regularized Fine-Tuning

REINFORCE-based fine-tuning focuses solely on maximizing the predicted MRL of generated 5′ UTR sequences. However, optimizing exclusively for reward often causes decoder collapse, leading to low-diversity, biologically implausible sequences with repetitive motifs and homopolymeric runs. To address this issue, we incorporate a regularization term based on the DAP objective (Section 2.3.2.2).

Following the DAP formulation, the decoder is trained to match the log-likelihood of the sampled sequences to an augmented target distribution that combines the frozen prior and the reward signal. The corresponding loss is defined as

$$\mathcal{L}_{\text{DAP}} = \frac{1}{B}\sum_{j=1}^{B}\Big(\log P_{\text{agent}}(T^{(j)}) - [\log P_{\text{prior}}(T^{(j)}) + \sigma R(T^{(j)})]\Big)^2,$$

where B is the batch size, $T^{(j)}$ is the j-th generated sequence, $\log P_{\text{agent}}(T^{(j)})$ is the log-likelihood of the sequence under the current decoder, $\log P_{\text{prior}}(T^{(j)})$ is the log-likelihood under a frozen pretrained prior decoder, $R(T^{(j)})$ is the reward given by the predicted MRL, and σ is a scaling factor controlling the influence of the reward term.

Minimizing this loss aligns the agent decoder with an augmented target distribution $P_{\text{aug}}(T)$ defined in the log-probability space as

$$\log P_{\text{aug}}(T) = \log P_{\text{prior}}(T) + \sigma R(T).$$

Exponentiating both sides yields the expression in probability space:

$$P_{\text{aug}}(T) \propto P_{\text{prior}}(T) \cdot \exp(\sigma R(T)).$$

In this formulation, sequences with higher predicted rewards become exponentially more likely in the augmented target distribution. As a result, DAP-regularized fine-tuning encourages the decoder to explore the sequence space to improve translation efficiency while preserving biological plausibility and proximity to the original sequences.

6.5 Dataset

All experiments in this chapter use the `GSM3130435` dataset described in Section 4.3. The same dataset was also used for training the encoder and multitask autoencoder in Chapters 4 and 5.

6.6 Experiments and Results

The decoder component of the pre-trained autoencoder in Chapter 5 was fine-tuned to generate sequences with improved translation efficiency using two strategies: non-curriculum learning and curriculum learning. Both approaches build upon the pretrained multitask autoencoder introduced in Chapter 5, which serves several key roles in the training process:

(i) **Encoder-derived features:** The encoder component predicts MRL values and latent vectors, which are saved in a DataFrame. The predicted MRL values are used to perform stratified train–test–validation splitting in the non-curriculum learning setup and to group the training data into difficulty-based bins in curriculum learning, while the latent vectors are reused as inputs to the autoregressive LSTM decoder during fine-tuning.

(ii) **Reward estimation:** During training, the frozen encoder predicts MRL of sequences generated by the decoder, which serve as rewards to guide fine-tuning.

(iii) **Decoder weight initialization:** The pretrained decoder weights initialize the fine-tuning process, ensuring continuity between pretraining and fine-tuning.

(iv) **Prior-guided optimization:** In the DAP-regularized fine-tuning strategy, the frozen pretrained decoder acts as the prior model to constrain the fine-tuning.

6.6.1 Performance Evaluation

The performance of the fine-tuned decoder was evaluated using the CNN predictor originally introduced by Sample et al. [14] and later enhanced by Tang et al. [73]. The CNN predictor was trained on the same dataset, with details on its architecture, training, and performance provided in Section 4.5.2. This external predictor was selected over our GNN encoder to ensure an independent and unbiased evaluation of improvements in the predicted MRL of the generated sequences. Furthermore, the CNN predictor has been experimentally validated in wet laboratory assays in both cited studies.

In addition to improvements in predicted MRL, the fine-tuned decoder was evaluated by quantifying the changes in total sequence entropy, sliding-window entropy, Hamming distance, and GC content between the optimized sequences generated and their original counterparts. Here, entropy measures nucleotide diversity, with low entropy indicating highly constrained or repetitive sequences that may reflect reduced sequence complexity. While more sophisticated models for comparing RNA sequences exist [82, 83], this work adopts a unit-cost Hamming distance to measure the divergence between input sequences and generated variants for simplicity and computational efficiency.

6.6.2 Non-Curriculum Learning

6.6.2.1 Preliminary Experiments

Preliminary experiments evaluated REINFORCE fine-tuning using the full training set obtained from a 70/15/15 train/validation/test split of the dataset, stratified using MRL values predicted by the GNN encoder. In this setting, the fine-tuned decoder frequently collapsed to low-complexity sequences containing long A/T homopolymer tracts and generated many duplicate outputs when evaluated on the held-out test set (e.g., only 10,299 unique sequences among 42,000 sampled). Restricting training to a stratified subset of 50,000 sequences or fewer drawn from the 70% training set substantially reduced this degeneracy and consistently yielded fully unique optimized sequences while maintaining stable improvements in predicted MRL.

6.6.2.2 Training Procedure

The dataset was divided into training, validation, and test sets using a 70/15/15 stratified split based on the MRL values predicted by the GNN encoder. A fixed random seed of 42 was used for splitting to ensure consistent dataset partitions when comparing curriculum and non-curriculum learning strategies. From the training set, a stratified subset of 10,000 sequences was selected for training. This number of training samples was empirically chosen to balance optimization effectiveness and sequence diversity.

The decoder was fine-tuned for three epochs using the Adam optimizer with a learning

rate of 1×10^{-3} and a batch size of 128. Model performance was monitored in the validation set based on the average predicted MRL of the generated sequences.

Decoder fine-tuning was performed using two strategies: REINFORCE (Section 6.3) and DAP-regularized fine-tuning (Section 6.4). The same dataset splits and training configurations were used for both approaches to ensure a fair comparison.

6.6.2.3 Results

Under DAP-regularized fine-tuning, the fine-tuned decoder generated sequences with an average predicted MRL of 7.58, compared to 6.45 for the original sequences. Approximately 75.53% of generated sequences achieved higher predicted MRL values relative to their original counterparts. The GC content (0.49 vs. 0.50), total entropy (1.94 vs. 1.94), and sliding-window entropy (1.72 vs. 1.74) remained largely unchanged, indicating that the overall structural complexity of the sequences was preserved. Moreover, the mean Hamming distance between generated and original sequences was relatively low (2.92), suggesting that DAP-regularized fine-tuning introduces only modest and targeted modifications. Representative sequences generated by the DAP-regularized fine-tuned decoder are presented in Figure 6.3, showing how the targeted modifications introduced by the model result in substantial improvements in the predicted MRL.

In contrast, the REINFORCE fine-tuned decoder generated sequences with a higher predicted average MRL of 7.98, compared to 6.45 for the original sequences, with 95.35% of generated sequences showing improvements. However, these gains came at the cost of

more substantial sequence modifications, as indicated by a much higher mean Hamming distance (18.03). REINFORCE also induced notable shifts in GC content (0.34 vs. 0.50) and slightly reduced both total entropy (1.87 vs. 1.94) and sliding-window entropy (1.58 vs. 1.74), suggesting a tendency toward simpler and compositionally altered sequences. A comparative summary of the evaluation metrics is provided in Table 6.1.

Metric	**Original**	**DAP**	**REINFORCE**
Percentage of sequences with improved MRL	–	75.53%	95.35%
Mean Hamming distance	–	2.92	18.03
Mean predicted MRL	6.45	7.58	7.98
Mean GC content	0.50	0.49	0.34
Mean total entropy	1.94	1.94	1.87
Mean sliding-window entropy	1.74	1.72	1.58

Table 6.1: Comparison of results for DAP-regularized and REINFORCE fine-tuning.

To complement the summary statistics, Figure 6.2 illustrates the distributions of the evaluation metrics using histograms and violin plots under the two fine-tuning strategies.

(a, f) Predicted MRL distributions for original and generated sequences.

(b, g) Density and spread of predicted MRL values shown using violin plots.

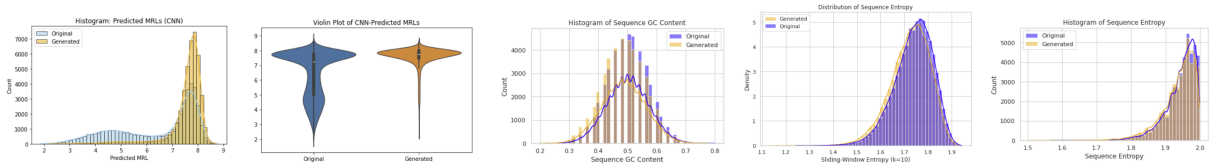
(c, h) GC content distributions based on the fraction of G and C nucleotides.

(d, i) Sliding-window entropy (Window Size = 10) measuring local sequence diversity.

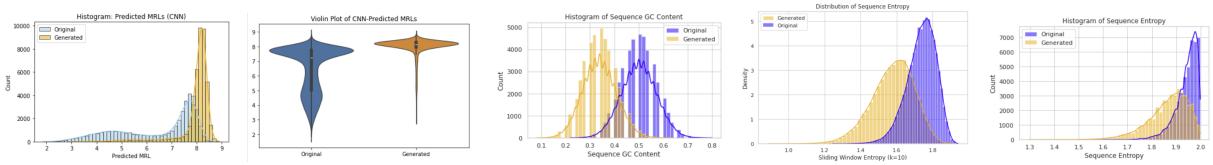
(e, j) Total sequence entropy reflecting overall compositional diversity.

Comparison of distributions shows that DAP-regularized fine-tuning improves predicted MRL with minimal changes to GC content and sequence entropy, while REINFORCE yields

stronger gains accompanied by pronounced shifts in composition and reduced diversity. Given this balance between performance and sequence stability, DAP-regularized fine-tuning was selected for curriculum learning in the following section.



a: MRL b: MRL (Violin c: GC content d: Window entropy e: Total entropy



f: MRL g: MRL (Violin) h: GC content i: Window entropy j: Total entropy

Figure 6.2: Non-curriculum decoder fine-tuning results. Top row (a–e): DAP-regularized fine-tuning. Bottom row (f–j): REINFORCE fine-tuning. Generated sequences are yellow/orange; original sequences are blue.

6.6.3 Curriculum Learning

6.6.3.1 Training Procedure

The training data was divided into training, validation, and test sets using a stratified split of 70/15/15 based on the predicted MRL values from the GNN encoder. The same fixed random seed of 42 as in Section 6.6.2 was used to split the dataset to ensure identical partitions between curriculum and non-curriculum learning. Within the training set, the

Optimized 5′ UTRs (Original top, Generated Bottom) with ΔMRL
GACGCACAACATGCGTTACAAGAAGAAGCGTTACGCAAGCCTCGAAAAAA
GACGCACAACA<u>C</u>GCGTTACAAGAAGAAGCGTTACGCAAGCCTCGAAAAAA
ΔMRL = 4.91
CCTACAAAACCTGAAAATGCATCGCGAGACGGGGGATCCAAAAAACAGTC
CCTACAAAACCTGAAAAT<u>T</u>CATCGCGAGACGGGGGATCCAAAAAACAGTC
ΔMRL = 4.85
CCCATCGACCGAAAGGGTGGGGAAAACACATTTAGTAGAACGAAACATGG
CCCATCGACCGAAAGGGTGGGGAAAACACATTTAGTAGAACGAAACA<u>G</u>GG
ΔMRL = 4.83
GACGTGAGCGACCATGGCCGTCAGCCGAGGAAACCGTTATTGCATTCATT
GACGTGAGCGACCA<u>G</u>GGCCGTCAGCCGAGGAAACCGTTATTGCATTCATT
ΔMRL = 4.82
TTACACACAGCGAAAGAAAAAGTGACAATTAACAATTACAGGAGACATGG
TTACACACAGCGAAAGAAAAAGTGACAATTAACAATTACAGGAGACA<u>G</u>GG
ΔMRL = 4.81

Figure 6.3: Representative examples of sequences generated by the DAP-regularized fine-tuned decoder. Modified nucleotides in the generated sequences are underlined.

sequences were further grouped into ten quantile-based bins (bins 0–9) according to their predicted MRL values, sorted in ascending order. These bins were then used to implement a staged curriculum learning strategy.

Training progressed sequentially from bin 0 to bin 9. At each stage, the decoder was fine-tuned using latent vectors from sequences in the current bin. For bins 0–8, the reward threshold was set to the 80^{th} percentile of the next bin's predicted MRL values. For bin 9, in the absence of a higher reference bin, the threshold was set to the 90^{th} percentile of

the same bin to establish a discriminative target within the highest MRL range. A binary reward of 1 was assigned to generated sequences with predicted MRL values $\geq$ threshold. Sequences with predicted MRL values below the threshold received a reward of 0.

Fine-tuning used a DAP-regularized objective described in Section 6.4. At each stage, the decoder was trained for three epochs using a frozen copy of the initial decoder as the prior. The trainable decoder was updated with the Adam optimizer ($lr = 1 \times 10^{-4}$, batch size = 128).

After each stage, performance was monitored on the validation set by tracking the average predicted MRL of generated sequences and the success rate relative to the stage-specific threshold.

6.6.3.2 Results

Under curriculum learning with DAP-regularized fine-tuning, the fine-tuned decoder generated sequences with an average predicted MRL of 7.73, compared to 6.45 for the original sequences. Approximately 96.98% of the generated sequences achieved higher predicted MRL values relative to their original counterparts, representing a substantial improvement over the 75.53% observed under non-curriculum DAP fine-tuning.

The mean Hamming distance between generated and original sequences was 6.17, which is higher than the 2.92 observed under non-curriculum DAP but still indicates relatively modest modifications. The mean GC content decreased slightly under curriculum learning (0.40 vs. 0.50 for the original sequences), while non-curriculum DAP preserved GC content

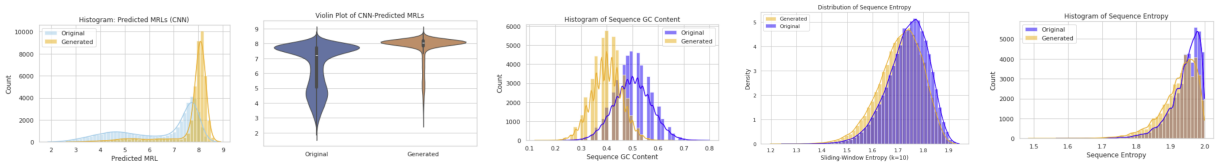
more closely (0.49 vs. 0.50). Measures of sequence diversity remained stable: the total entropy showed minimal change (1.92 vs. 1.94), and the sliding-window entropy decreased only slightly (1.71 vs. 1.74), suggesting that both local and global sequence complexity were largely maintained.

Figure 6.4 illustrates the distributions of evaluation metrics under curriculum learning compared to non-curriculum DAP fine-tuning. Panels (a, f) show the distributions of predicted MRL values, while (b, g) present violin plots of their density and spread. Panels (c, h) display GC content distributions, (d, i) show sliding-window entropy, and (e, j) depict total sequence entropy. The distributions confirm that curriculum learning achieved higher predicted MRL values while maintaining compositional diversity, at the cost of slightly larger sequence modifications compared to non-curriculum DAP.

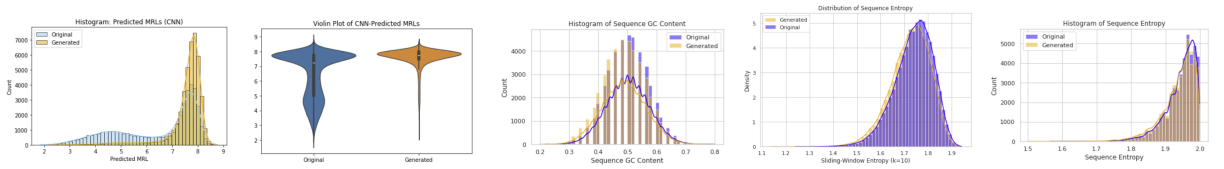
A detailed comparison of the results between curriculum learning and non-curriculum DAP fine-tuning is provided in Table 6.2.

Metric	**Original**	**Curriculum DAP**	**DAP**
Percentage of sequences with improved MRL	–	96.98%	75.53%
Mean Hamming distance	–	6.17	2.92
Mean predicted MRL	6.45	7.73	7.58
Mean GC content	0.50	0.40	0.49
Mean total entropy	1.94	1.92	1.94
Mean sliding-window entropy	1.74	1.71	1.72

Table 6.2: Comparison of results for curriculum learning DAP and non-curriculum DAP fine-tuning.



a: MRL b: MRL (Violin) c: GC content d: Window entropy e: Total entropy



f: MRL g: MRL (Violin) h: GC content i: Window entropy j: Total entropy

Figure 6.4: Curriculum learning decoder fine-tuning results. Top row (a–e): curriculum learning DAP-regularized fine-tuning. Bottom row (f–j): non-curriculum DAP fine-tuning. Generated sequences are yellow/orange; original sequences are blue.

6.7 Summary

This chapter evaluated decoder fine-tuning strategies to improve translation efficiency, measured by predicted MRL. Among the approaches tested, DAP-regularized fine-tuning achieved a balanced trade-off between increasing predicted MRL values and preserving sequence composition. REINFORCE produced stronger MRL gains at the cost of substantial sequence alterations and compositional shifts. Incorporating curriculum learning into DAP-regularized fine-tuning increased the percentage of improved sequences to 96.98% while maintaining overall sequence diversity. However, this gain came at the cost of slightly larger sequence modifications compared to the non-curriculum DAP.

These results motivate a deeper investigation into the regulatory mechanisms responsible

for improved translation efficiency in the generated sequences. This analysis is presented in the following chapter.

7 Model Interpretability and Biological Insights

7.1 Introduction

This chapter links computational interpretability with biological mechanisms underlying translation efficiency. The first part examines latent representations and feature-level attributions from non-optimized 5′ UTRs to reveal how the model encodes determinants of translation initiation. Subsequently, comparative analyses of optimized 5′ UTR identify compositional and structural refinements associated with increased MRL.

7.2 Representation and Feature-Level Interpretability

7.2.1 Dimensionality Reduction of Latent Representations

The organization of sequence embeddings produced by the encoder was analyzed using a two-dimensional Uniform Manifold Approximation and Projection (UMAP) projection. UMAP is a nonlinear manifold learning technique that preserves both local and global structure in the data, making it suitable for visualizing complex feature spaces produced by deep neural networks. The projection was computed using cosine distance, 30 nearest neighbors, and a minimum distance of 0.2 for the embeddings of 280,000 sequences in the dataset. The resulting two-dimensional map revealed two distinct regions in the latent manifold, suggesting that the encoder organized the sequences according to differences in sequence composition or regulatory features.

To further examine whether biological or compositional properties are reflected in this organization, k-means clustering ($k = 2$) was applied to predicted MRL, uAUG count, uORF count, GC content, and MFE to define low and high feature groups. The resulting cluster labels were overlaid on the UMAP projection by coloring points according to their label (Figure 7.1). Examination of the plots revealed that the latent organization aligns most strongly with predicted MRL, uAUG count, and uORF count. In contrast, GC content and MFE exhibited more diffuse color distributions. This pattern suggests

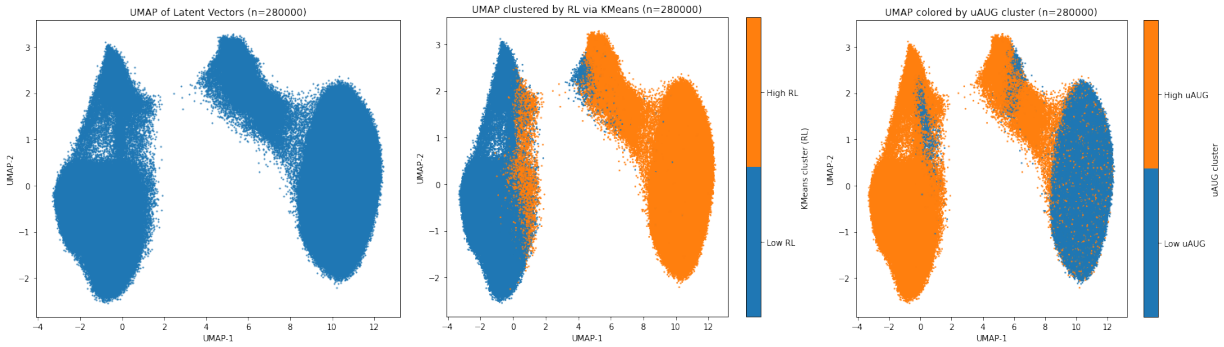
that the encoder's learned representation is more strongly structured around functional and positional features (e.g., number uAUGs and uORFs) than around global sequence composition or thermodynamic stability.

Consistent patterns were also observed in t-SNE projections of the latent representation colored by the same features. Stronger organization was observed for predicted MRL, uAUG count, and uORF count than for GC content or MFE (data not shown).

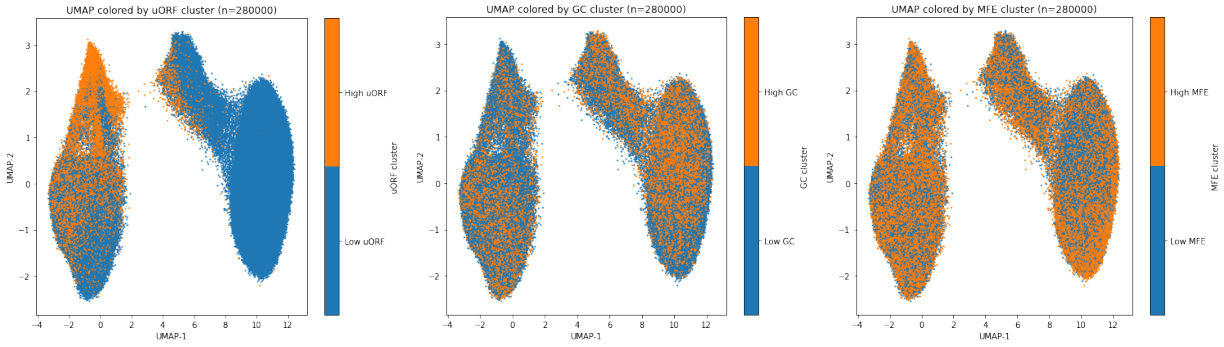
7.2.2 Quantitative Analysis of Feature Relationships

Qualitative visualization of UMAP projections (Figure 7.1) revealed that latent vectors corresponding to high uAUG and high uORF counts are almost exclusively overlaid on regions associated with low predicted MRL. In contrast, GC content and MFE exhibited scattered color distributions with no clear spatial alignment. To quantitatively confirm these visual patterns, pairwise cross-tabulation of the feature clusters showed strong co-occurrence between low MRL and high uAUG/uORF clusters, but weaker dependencies for GC content and MFE. Aggregation of predicted MRL values across all four-way feature-cluster combinations further confirmed that sequences with high regulatory burden (high uAUG and/or uORF) consistently exhibited the lowest predicted translation efficiency (Figure 7.2).

To statistically validate the observed differences in predicted MRL across biological feature combinations, a non-parametric Kruskal–Wallis test was performed. The test indicated a highly significant overall difference among combinations ($H = 159{,}272$, $p < 0.001$).



a: UMAP (uncolored) b: MRL clusters c: uAUG clusters



d: uORF clusters e: GC clusters f: MFE clusters

Figure 7.1: Two-dimensional UMAP embedding of encoder latent vectors. (a) Uncolored latent projection. Panels (b–f): k-means ($k = 2$) clustering by predicted MRL, uAUG count, uORF count, GC content, and MFE. Points are colored by cluster-mean feature value (orange = high, blue = low).

Pairwise post-hoc comparisons using Dunn's test with Bonferroni correction (Figure 7.3) revealed that nearly all combinations differ significantly in their predicted MRL values ($p \leq 0.05$), confirming systematic variation in translation efficiency predicted by the model.

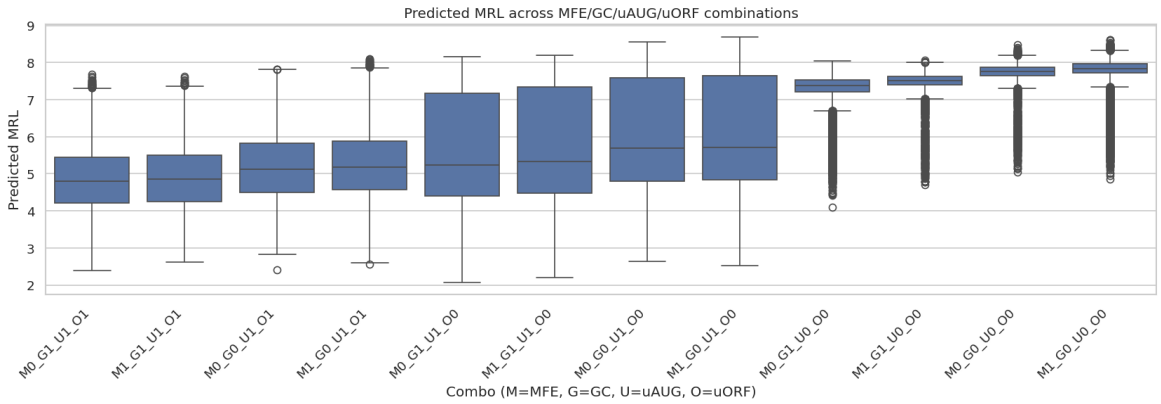


Figure 7.2: Distribution of predicted MRL across combinations of feature-cluster labels for MFE (M), GC content (G), uAUG (U), and uORF (O). Combinations are ordered by mean predicted MRL. Sequences belonging to high-uAUG and/or high-uORF clusters consistently exhibit lower predicted translation efficiency.

7.2.3 Feature-Level Regression Analysis

UMAP visualizations and accompanying statistical analyses indicated that compositional and regulatory features are associated with distinct patterns of translation efficiency. To quantify these relationships and test whether feature effects differ across latent-space regions, a robust Huber regression analysis was applied to estimate standardized feature coefficients and their interactions with the latent-side grouping.

In the baseline model of Huber regression, uAUG count, uORF count, GC content, and MFE collectively explained approximately half of the variance in the predicted MRL ($R^2 = 0.49$). uAUG count showed strong negative effects on predicted MRL (-1.07 per +1 SD), consistent with its known repressive roles in translation initiation. GC content and uORF count had a milder negative effect (-0.25 and -0.22 per +1 SD, respectively).

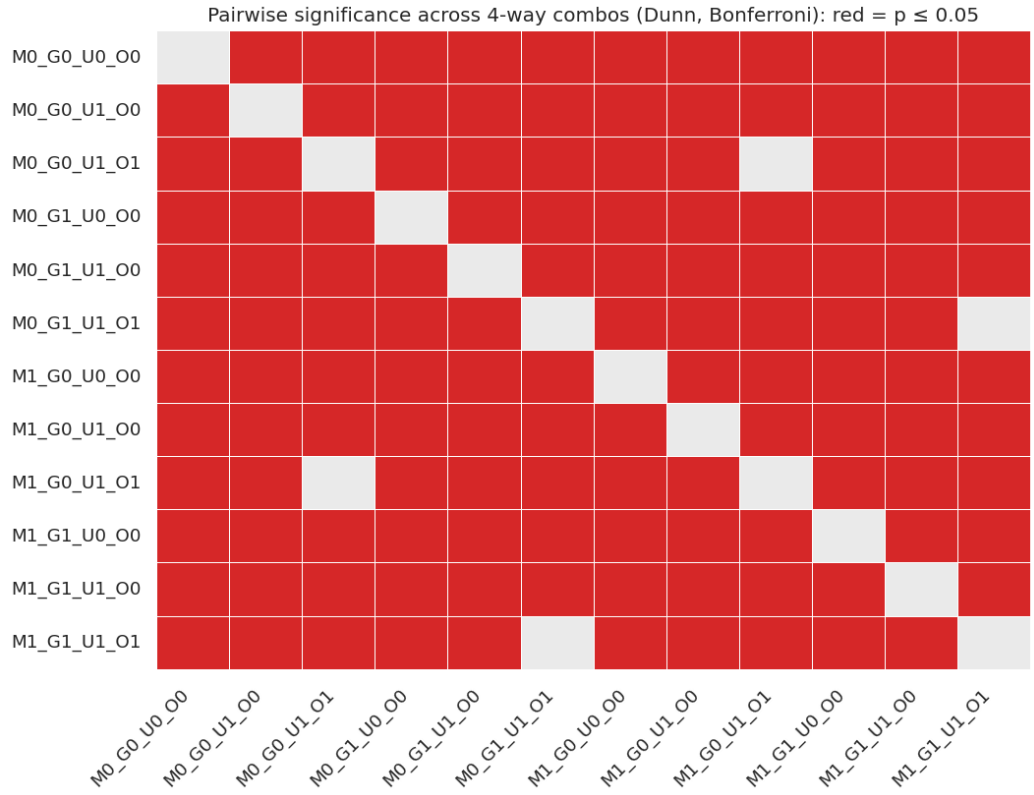


Figure 7.3: Pairwise significance of predicted MRL differences across four-way feature-cluster combinations (MFE (M), GC content (G), uAUG (U), and uORF (O)) based on Dunn's post-hoc test (Bonferroni-adjusted). Red indicates significant differences ($p \leq 0.05$). Almost all pairs show significant separation, confirming that predicted MRL values differ systematically across biological feature combinations.

MFE contributed minimally (+0.004 per +1 SD), reflecting its partial collinearity with GC content.

To capture potential differences between latent space regions identified by the UMAP projection, sequences were divided into two clusters using k-means clustering on the UMAP coordinates (`umap_x`, `umap_y`). The resulting clusters corresponded to the low and high

MRL regions observed in Figure 7.1. Incorporating the binary latent-side variable and its interaction terms substantially improved the explained variance in predicted MRLs ($R^2 = 0.88$). On the low MRL side, high GC content and multiple uAUGs were strongly inhibitory. On the high MRL side, these inhibitory effects were attenuated, while uORF count shifted from mildly positive to negative (Figure 7.4). These context-dependent relationships imply that the encoder's latent representations capture nonlinear interactions and additional regulatory mechanisms such as structural accessibility, motif positioning, or compensatory sequence context. The improvement in R^2 from 0.49 to 0.88 reflects this additional explanatory power contributed by the latent-space representations.

7.3 Mechanistic Interpretation of Learned Sequence Optimization

Fine-tuning the decoder led to a global redistribution of predicted MRL values, with generated sequences occupying the higher-efficiency regions of the histogram and exhibiting reduced variance relative to the original distribution (Figure 7.5). This section explores how the fine-tuned decoder transforms latent representations of the original 5′ UTRs into variants with improved predicted translation efficiency. The DAP fine-tuned decoder is used for analysis since it generated sequences with the smallest average Hamming distance from the original 5′ UTRs.

Among 42,000 sequences in the test set, the 2,000 showing the largest and smallest

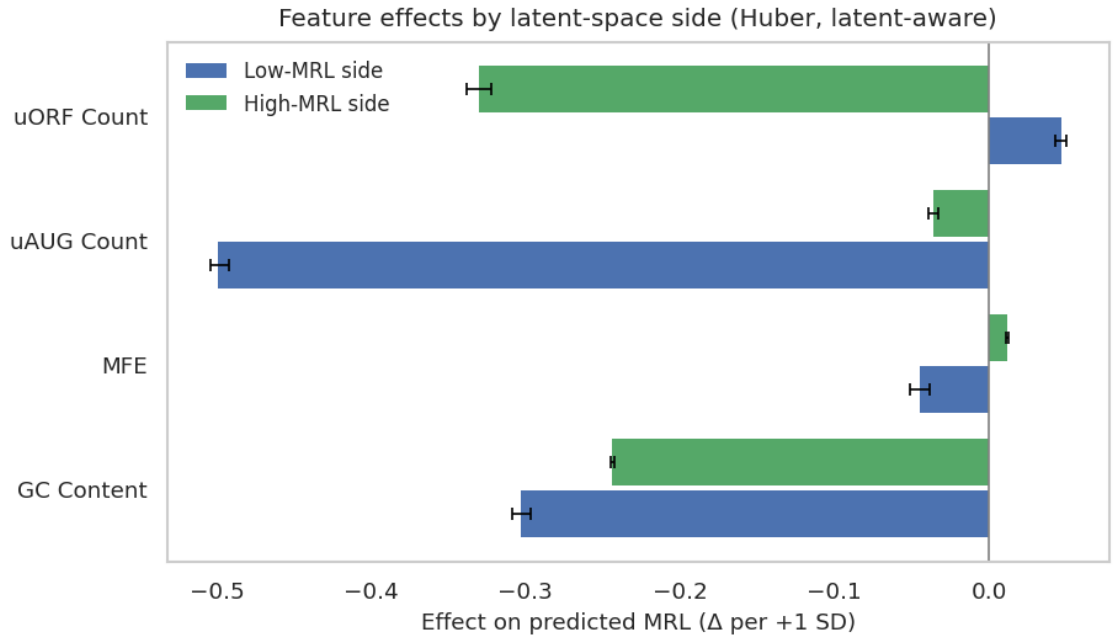


Figure 7.4: Feature effects by latent-space side estimated with a latent-aware Huber regression. Bars show the standardized coefficient (change in predicted MRL per +1 SD) for GC content, MFE, uAUG count, and uORF count, plotted separately for the low (blue) and high (green) MRL sides of the latent space.

gains in predicted MRL were compared to characterize the optimization strategies learned by the decoder. The fine-tuned decoder preferentially improved translation efficiency for sequences with initial low MRL values. In contrast, already efficient sequences experienced slight reductions (Table 7.1).

The above pattern suggests that optimization was primarily directed towards suboptimal regulatory regions, consistent with a saturation effect at the high-efficiency end of the distribution where diminishing reward signals limit further optimization.

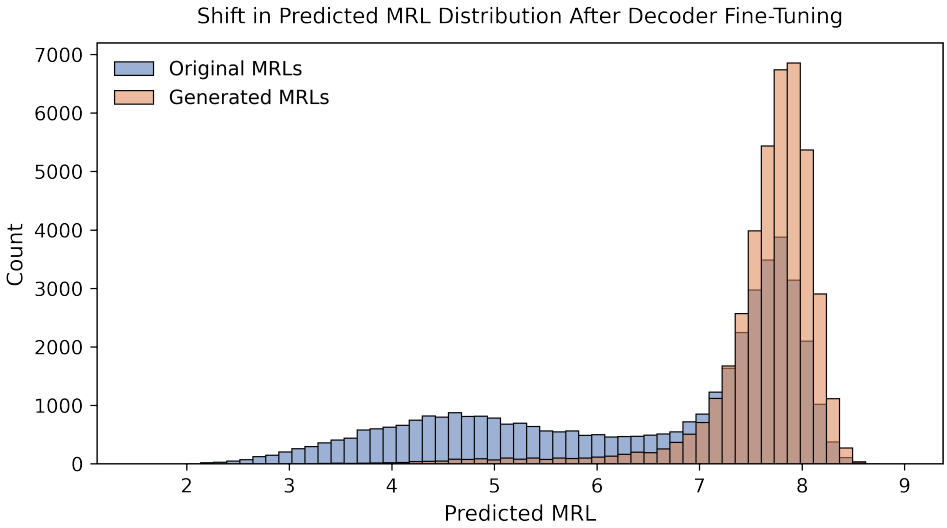


Figure 7.5: Distribution of predicted MRL values for original sequences and sequences generated by the decoder after DAP-regularized fine-tuning.

	Mean (before)	**Mean (after)**	**Mean ΔMRL**	**SD ΔMRL**
Full test set	6.45	7.58	1.13	0.85
Top 2K	3.27	7.80	4.53	0.33
Bottom 2K	7.28	6.77	-0.51	0.63

Table 7.1: Descriptive statistics of predicted MRL values for the full test set and for the top and bottom 2,000 sequences ranked by ΔMRL.

7.3.1 Upstream AUG Disruption

uAUG codons are known to reduce translation efficiency by initiating upstream open reading frames. To examine how these motifs are disrupted, substitutions converting an existing AUG triplet to a non-AUG triplet were counted separately for the top and bottom 2,000 original–generated sequence pairs ranked by ΔMRL.

The top ΔMRL group showed a marked increase in direct uAUG-disruptive substitutions, particularly G to T changes in the proximal region (positions 10–25), compared to the low ΔMRL group (Figure 7.6). This pattern suggests that uAUG removal is one of the mechanisms by which the decoder improves the predicted translation efficiency.

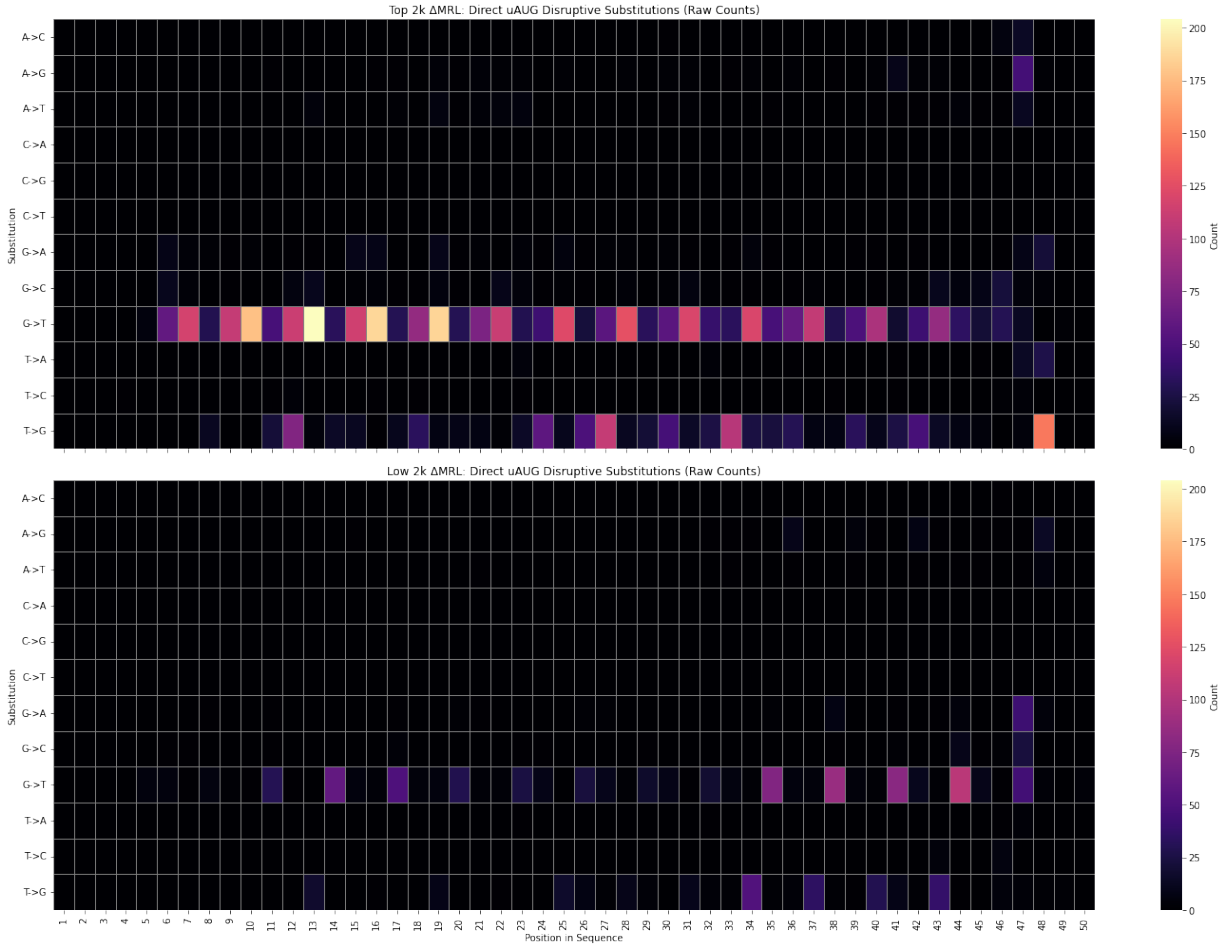


Figure 7.6: Positional distribution of nucleotide substitutions that directly disrupt uAUGs in the top and bottom 2,000 sequences ranked by ΔMRL. Heatmaps display raw counts per position and substitution type with a shared color scale across panels.

7.3.2 Reduced Inhibitory Structure Near the Start Codon

Stable local structures close to the 5′ cap or around the start codon can impede ribosomal scanning and start codon recognition. To test whether the decoder improves translation by altering local RNA accessibility, base-pairing probability was compared around the 3′ end (positions 45–50) between the original and generated sequences. In the top ΔMRL group, pairing probability decreased significantly after decoding (t-test, $p = 2.9 \times 10^{-7}$), while the bottom ΔMRL group showed a slight increase. These results indicate that the fine-tuned decoder tends to reduce inhibitory structure near the start codon as an optimization mechanism.

7.3.3 Reduced Cap-Proximal Structure and Start-Site Accessibility

The impact of the fine-tuned decoder on the structural stability of RNA was evaluated by comparing MFE values in the cap-proximal (1–30) and start-proximal (30–50) regions of the 5′ UTR. In the top ΔMRL group, MFE comparison between original and generated sequences showed a significant increase in both regions (t-test, $p < 10^{-4}$), indicating reduced structural stability and increased local accessibility. In contrast, the low ΔMRL group showed minimal or opposite changes. These results suggest that the decoder preferentially relaxes stable secondary structures near the 5′ cap and start codon to enhance translation initiation.

7.3.4 Insertion of Candidate RNA-Binding Protein Motifs

RBPs interact with AU-rich element (ARE) motifs in the untranslated regions of mRNA to modulate transcript stability, localization, and translation efficiency. Among generated 5′ UTRs in the test set with increased predicted MRL, insertion of ARE motifs resembling known RBP recognition sites such as AUUUA, UUAUU, and AUUUAA was associated with an average 1.5-fold higher ΔMRL relative to other improved sequences. Notably, the canonical AUUUA pentamer, a known binding site for several ARE-binding proteins including ELAVL1/HuR [84, 85], was inserted in 969 cases. These findings suggest that the decoder may have learned to introduce AU-rich patterns characteristic of RBP-regulated motifs to increase translation potential.

7.3.5 Pseudoknot Reduction

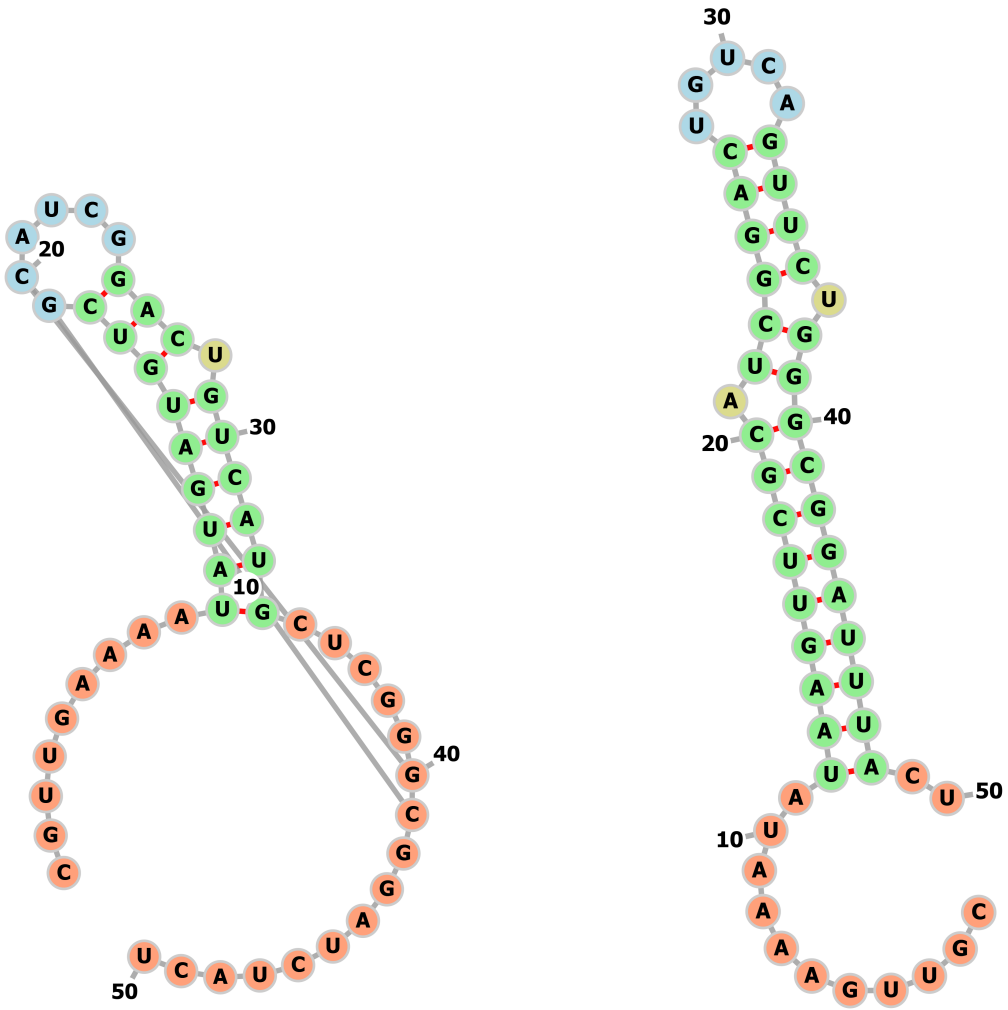
Pseudoknots form when nucleotides in a loop pair with a distant region of the RNA strand, producing overlapping stems that resemble a knot. These knot-like structures can interfere with translation in both prokaryotic and eukaryotic cells by pausing ribosomes or causing frameshifts [86–88].

To investigate the contribution of removing pseudoknots to the observed increase in the predicted MRL, IPknot [89] predictions were compared between the original and generated sequences (Figure 7.7). Among the 42,000 generated sequences in the test set, pseudoknots were completely removed in 15.4%, reduced in 8.8%, increased in 22.8%, and

retained in 53.0%. Sequences in which pseudoknots were completely removed showed the highest predicted mean ΔMRL (1.34), followed by those with increased (1.25), partially reduced (1.19), and retained (0.86) pseudoknots. The difference in mean ΔMRL among the four groups was highly significant (Kruskal–Wallis test, $H = 1329.8$, $p < 10^{-208}$). Post-hoc Holm–Bonferroni comparisons confirmed that all categories differed significantly (Figure 7.8). These findings indicate that while pseudoknot removal correlates with greater mean ΔMRL, it is not necessary for optimization. Accordingly, the comparable mean ΔMRL in the reduced and increased pseudoknot categories suggests that the decoder's edits were context-dependent, balancing local accessibility with other structural or compositional factors influencing translation efficiency.

7.4 Evidence from Model Attributions

Like other neural architectures, GNNs do not directly expose the features that drive their predictions. Graph explainers such as GNNExplainer [91] address this limitation by providing post hoc interpretability through learned importance masks over nodes, edges, and node features. The following subsections apply GNNExplainer to quantify how nucleotide positions and node features contribute to predicted translation efficiency in both original and optimized sequences.



(A) Original 5′ UTR with pseudoknot.

(B) Generated 5′ UTR without pseudoknot.

Figure 7.7: Comparison of secondary structures before and after pseudoknot removal. Structures were visualized using `Forna` [90] from the ViennaRNA web suite (`http://rna.tbi.univie.ac.at/forna/`). The removal of pseudoknots results in a planar secondary structure.

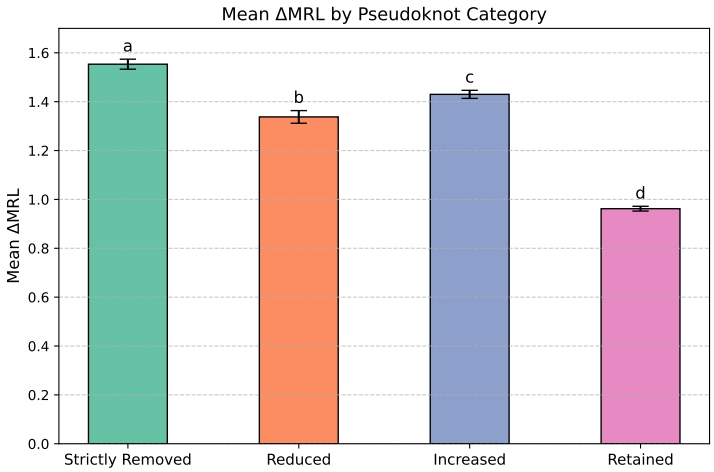


Figure 7.8: Mean predicted MRL gain by pseudoknot category. Different letters indicate statistically significant differences between groups.

7.4.1 Positional Attributions from GNNExplainer

GNNExplainer was used to identify nucleotide positions that most strongly contributed to the predicted translation efficiency. Sequences were stratified into ten bins based on the predicted MRL prior to optimization. Ten sequences were randomly sampled from each bin, yielding 100 original–generated sequence pairs. For each 5′ UTR graph, node-level importance values were obtained and normalized to sum to one. Node importance attributions were then aggregated into biologically meaningful positional bands corresponding to the cap-proximal region (positions 1–15), middle region (16–30), and start-proximal region (31–50).

Node importance values revealed a consistent positional hierarchy in both original and optimized sequences (Figure 7.9). The start-proximal region received the highest attribution,

followed by the middle and cap-proximal regions. This pattern was statistically significant in both original and optimized 5′ UTRs (paired Wilcoxon tests; start > cap and start > middle, $P < 10^{-7}$; middle > cap, $P < 5 \times 10^{-3}$).

The magnitude and ordering of regional attributions did not differ significantly between original and generated sequences, indicating that gains in predicted MRL did not shift the encoder's positional focus. Instead, optimization refined nucleotide content within the pre-existing regions of 5′ UTRs. The consistent prioritization of the start-proximal region aligns with the established roles of the Kozak context and local start-site accessibility for efficient translation initiation.

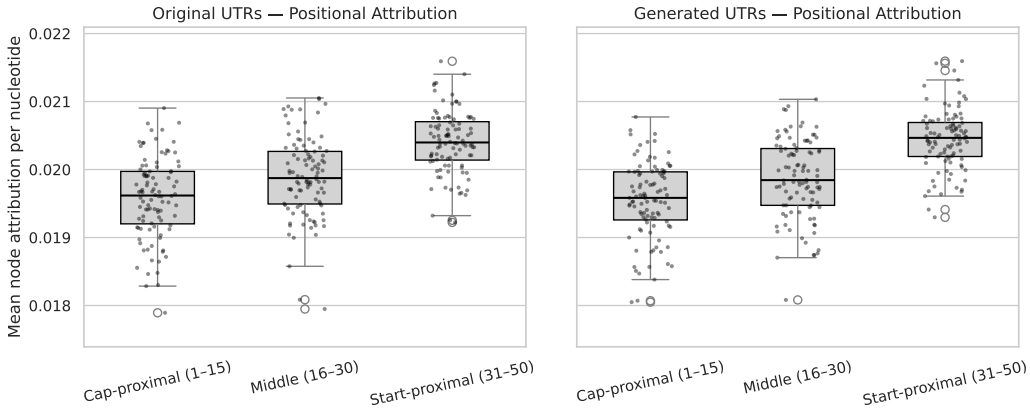


Figure 7.9: Positional node importance for original and generated 5′ UTR sequences. The encoder consistently assigns highest importance to nucleotides proximal to the start codon, followed by the middle and cap-proximal regions. Optimization does not significantly alter this positional preference profile.

A representative GNNExplainer heatmap illustrates the learned importance of individual nucleotides for an original and its corresponding optimized 5′ UTR (Figure 7.10).

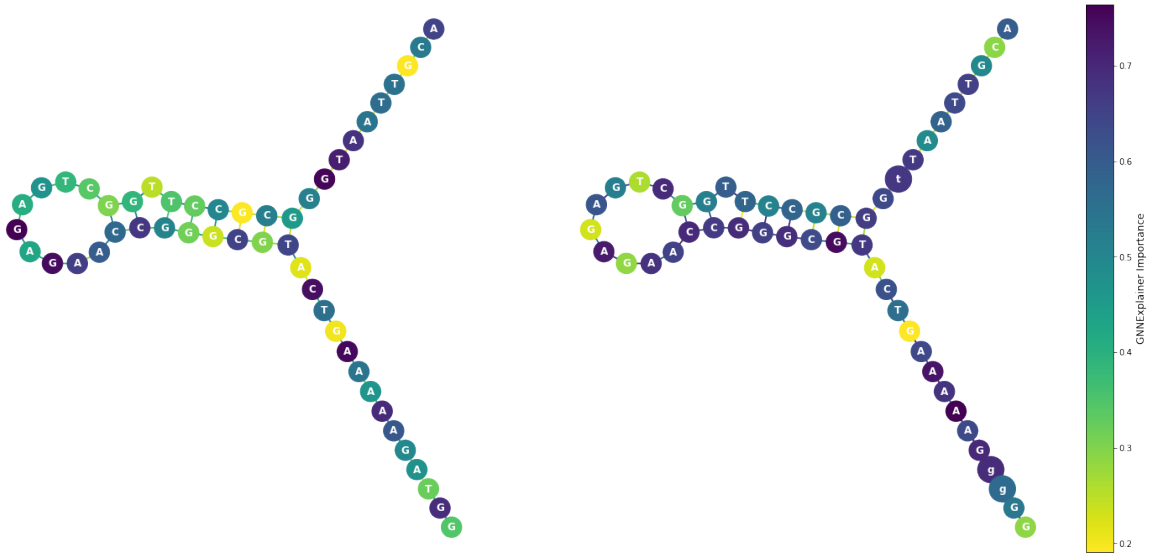


Figure 7.10: A GNNExplainer heatmap showing node importance values for an original (left) and its corresponding optimized 5′ UTR (right). Colors indicate the relative contribution of each nucleotide to the predicted MRL. Modified positions in the optimized sequence are shown in lowercase and with larger node size.

7.4.2 Feature Attributions from GNNExplainer

As described in Section 4.2.1.3, each node in the 5′ UTR graph was represented by a feature vector combining nucleotide identity, sinusoidal positional encoding, and purine/pyrimidine indicator. GNNExplainer feature importance masks for the same set of 100 original and generated sequence pairs sampled in the previous subsection were analyzed to estimate the relative contribution of each features to the predicted translation efficiency.

Feature attribution analysis revealed that positional encoding consistently dominated model predictions in both original and optimized sequences (Figure 7.11). Among all feature

groups, positional encoding accounted for over 70% of the total importance, followed by nucleotide identity (15%) and purine/pyrimidine indicators (14%). This pattern suggests that the GNN encoder primarily relied on sequential and positional features to predict translation efficiency.

Following optimization, the relative importance of positional encoding and nucleotide identity increased slightly ($p = 4.5 \times 10^{-6}$ and $p = 0.0081$, Wilcoxon signed-rank test), whereas the purine/pyrimidine contribution decreased slightly ($p = 4.17 \times 10^{-10}$, Wilcoxon signed-rank test). Although all differences were statistically significant, the magnitude of change in feature importance was minimal, indicating that the decoder's edits did not redefine the features important for predicting translation efficiency.

7.5 Summary

This chapter examined how the autoencoder's representations of $5'$ UTRs and the decoder's optimization strategies reflect known biological mechanisms. The encoder's latent geometry and feature correlations captured biologically meaningful distinctions, while the decoder modified sequences in ways consistent with established translation-enhancing mechanisms. GNNExplainer provided further insight, showing that start-proximal nucleotides and positional encodings are most influential for predicting MRL.

Several limitations should be noted. Predicted MRL is a surrogate for true translation efficiency and reflects biases from the encoder and training data. Moreover, in-vivo regulatory

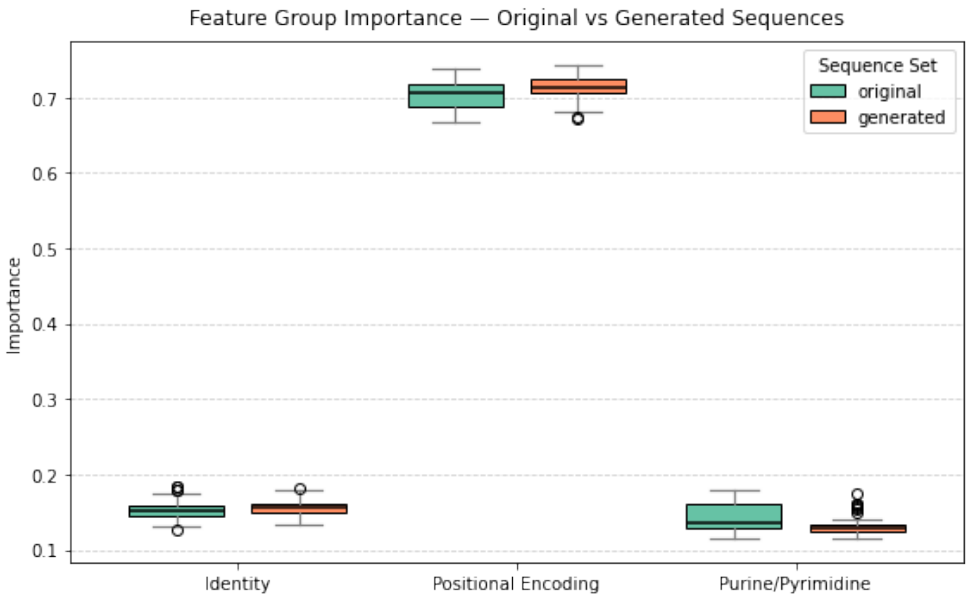


Figure 7.11: Comparison of node feature importance between original and generated sequences. Each box shows the distribution of feature attributions for nucleotide identity, positional encoding, and purine/pyrimidine indicators as quantified by GNNExplainer.

mechanisms likely act in combination rather than isolation. Despite these constraints, the analyses collectively support an interpretable and biologically consistent model behavior.

8 Discussion, Conclusion, and Future Work

8.1 Introduction

This chapter brings together the findings presented in the thesis and reflects on their implications for computational RNA modeling and design.

The following sections discuss the main results, acknowledge key limitations, and outline directions for extending this research toward experimentally validated and clinically relevant applications.

8.2 Discussion

The framework developed in this work combines a multitask autoencoder with reinforcement learning to improve 5′ UTR translation efficiency. The results show that a GNN encoder paired with an LSTM decoder can learn regulatory patterns that influence translation initiation and apply them during sequence optimization.

A GNN encoder was selected for its ability to model variable-length sequences and represent RNA structural interactions within a graph. The encoder performed competitively with an established CNN model on synthetic 5′ UTRs and maintained stable accuracy on variable-length human-derived sequences, whereas the CNN baseline showed a clear decline in performance as sequence length increased. However, the secondary structure contributed little to the final predictions in the synthetic dataset, likely because random sequences do not form reliable or biologically meaningful folds. Another explanation is that many regulatory signals in 5′ UTRs are encoded primarily at the sequence level, making sequential features more informative than structural ones. Overall, the encoder's predictive power and its consistent performance on longer human 5′ UTRs are encouraging for extending this approach to other RNA modeling tasks involving heterogeneous sequences. In addition, the attribution patterns revealed by the graph explainers were biologically meaningful and offer useful guidance for sequence design and motif discovery.

The decoder was implemented as an autoregressive LSTM that generates each nucleotide based on the preceding context, an approach well suited for modeling local dependencies in RNA and for producing sequences of arbitrary length without padding. The autoregressive structure is also essential for reinforcement learning–based optimization, since policy-gradient methods such as REINFORCE and DAP-regularization operate by sampling sequences step-by-step and updating the probability of each token according to the reward. In addition, LSTMs train reliably at the available dataset scale and maintain stable performance on sequential prediction tasks.

The multitask autoencoder reconstructed 5′ UTR sequences with near-perfect accuracy

and maintained the predictive performance of the standalone encoder. Building on this foundation, two reinforcement learning–based strategies, REINFORCE and DAP-regularization, were applied to fine-tune the decoder weights toward generating sequences with higher predicted MRL. Both methods successfully increased predicted translation efficiency while preserving overall sequence structure, though REINFORCE tended to introduce more aggressive edits and DAP produced more conservative modifications. A curriculum training schedule, which first exposed the DAP-regularized model to lower MRL sequences, further increased the fraction of generated sequences with improved translation efficiency without causing reward collapse. These results highlight the usefulness of policy-gradient optimization for controlled sequence design.

Compared with existing generative approaches, the autoencoder combined with RL fine-tuning offers a more controllable and scalable path for $5'$ UTR design. Genetic algorithms require many cycles of mutation and selection, which makes them slow and difficult to steer toward specific editing patterns. GAN-based models cannot refine a given input sequence, depend on large datasets, and struggle with discrete token generation, often producing repetitive or unstable outputs. Latent-space perturbation approaches can create plausible sequences but provide little control over the extent of edits or how strongly the MRL is altered. In contrast, the RL-fine-tuned decoder performs targeted, reward-driven editing, with REINFORCE enabling more aggressive exploration and DAP-regularization supporting more conservative modifications. Combined with the millisecond-level runtime per sequence, this approach provides a practical framework for high-throughput and controllable $5'$ UTR optimization.

Following thesis submission, additional experiments were conducted to support subsequent manuscript preparation. The GNN encoder was replaced with a CNN encoder based on the Optimus 5-Prime architecture [14] to evaluate the robustness of the framework to encoder choice. The resulting autoencoder was first trained for sequence reconstruction and MRL prediction using the same data splits described in Section 5.6. The model achieved a sequence reconstruction accuracy of 0.99 and an MRL prediction performance of $R = 0.94$ and $R^2 = 0.88$. However, accurate reconstruction required a substantially larger number of training epochs. The decoder was then fine-tuned using identical data splits (same random seed), training set size, and number of fine-tuning epochs described in Sections 6.6.2 and 6.6.3. The resulting performance is summarized in Table 8.1.

Metric	Original	DAP	Curriculum DAP	REINFORCE
MRL-improved sequences (%)	–	57.32%	90.28%	89.29%
Mean Hamming distance	–	7.12	9.30	15.59
Mean predicted MRL	6.45	6.90	7.53	7.78
Mean GC content	0.50	0.51	0.38	0.44
Mean total entropy	1.94	1.94	1.88	1.92
Mean sliding-window entropy	1.74	1.74	1.67	1.72

Table 8.1: CNN–LSTM model performance under DAP, Curriculum DAP, and REINFORCE fine-tuning.

Overall, both encoder architectures exhibit consistent trends under reinforcement learning optimization, with DAP-regularization producing more conservative edits and REINFORCE resulting in larger sequence modifications. However, the GNN-based model achieves higher fractions of improved sequences and slightly greater gains in predicted MRL. Differences in the extent of sequence edits are also observed, as the GNN model introduces

fewer modifications under DAP-based training and larger changes under REINFORCE. This pattern may reflect more targeted updates under regularization and increased exploration when constraints are relaxed. At the same time, the comparable behavior of the CNN-based model suggests that the RL objective, rather than the encoder architecture, is the primary driver of these improvements.

Furthermore, the GNN–LSTM model was found to exhibit generalizable performance under cross-dataset and zero-shot evaluation settings. Improvements in predicted MRL were observed under zero-shot transfer and training from scratch followed by fine-tuning when applied to a new dataset [15, 92]. This dataset featured sequences with shorter 5′ UTR lengths, additional cell types, and distinct experimental protocols (data not shown).

A few limitations should be noted. The models were trained primarily on synthetic random 5′ UTRs, which reduces the practical value of secondary structure modeling. The optimization reward was based on predicted rather than experimentally measured MRL, meaning that improvements reflect model confidence rather than validated biological gain. In addition, the approach was evaluated on sequences of fixed length and did not impose biological constraints on generated designs. Evaluating the model on natural 5′ UTRs and experimental measurements would provide additional evidence for the method's practical utility.

8.3 Conclusion

The combination of a GNN encoder, autoregressive decoder, and policy gradient fine-tuning represents a novel design framework for RNA generative modeling. Prior approaches have engineered 5′ UTRs through heuristics, adversarial models, or latent-space manipulations. By comparison, the use of graph representations and direct decoder-weight adaptation sets this method apart from existing work.

Beyond 5′ UTR optimization, the framework offers a general strategy for learning structure-informed representations of RNA and refining sequences through controllable, reward-driven edits. This makes it adaptable to other problems in therapeutic mRNA engineering, synthetic biology, and regulatory element design.

Together, the contributions introduced in this work advances computational RNA design and point toward several promising directions for continued development.

8.4 Future Work

Building on this work, several directions remain open for further research and development:

- **Modeling improvements**

  Future work could explore replacing the LSTM decoder with more expressive architectures, including transformers or diffusion models, to better capture long-range

dependencies. Incorporating additional structural features, such as predicted tertiary contacts or local accessibility profiles, may also improve optimization behavior. Using richer node representations, such as contextual embeddings from pretrained RNA language models, would allow each node to incorporate evolutionary and structural patterns that are not captured by one-hot encodings alone.

- **Experimental validation**

  Experimental evaluation of optimized 5′ UTRs in multiple cell types and *in vitro* measurements of translation efficiency would confirm that the predicted gains are biologically meaningful. Additional checks could assess sequence stability or compare predicted versus experimentally observed pairing patterns.

- **Generalization to natural 5′ UTRs**

  Evaluating the framework on endogenous 5′ UTRs would clarify whether optimized editing patterns and MRL gains transfer to biologically evolved regulatory elements. A recent study on natural transcripts demonstrates that deep learning models achieve high MRL predictive accuracy on endogenous 5′ UTRs [93]. Benchmarking against such endogenous datasets would validate the framework's performance against native regulatory logic.

- **Application to broader RNA design tasks**

  Applying the framework to broader RNA constructs such as 3′ UTRs, IRES segments, riboswitches, or therapeutic mRNA transcripts would extend the optimization strategy to to additional sequences whose function depends on structure.

- **Biological constraints and multi-objective optimization**

  Incorporating biological constraints such as avoiding immunogenic features [10, 94, 95] and extending the reinforcement learning setup to optimize multiple objectives including reduced immunogenicity, translation efficiency, and mRNA stability informed by available half-life datasets [96] would increase the practical relevance of the framework. Specifically, user-defined constraints such as preserving known regulatory motifs could be enforced during sequence generation by penalizing constraint violations (e.g., motif disruption) through reward shaping or by masking prohibited mutations during decoding [97].

- **Tissue-specific optimization and cellular context**

  Recent evidence indicates that $5'$ UTR regulatory function is influenced by cellular context. For instance, translational activity measured in HEK293T cell lines does not correlate with that observed in specialized cells such as neurons [98]. Future work could incorporate cell-type-specific features into predictor and RL objectives. Learning these rules would support the design of tissue-specific UTRs optimized for specific therapeutic targets.

These extensions would support applying the framework in a wider range of biological and clinical contexts.

References

[1] Andreas Thess. "Artificial nucleic acid molecules comprising a 5' TOP UTR". U.S. pat. US10080809B2 (United States). CureVac AG. Sept. 25, 2018. URL: https://patents.google.com/patent/US10080809B2/en.

[2] Thomas David Reed. "Synthetic 5'UTRs, expression vectors, and methods for increasing transgene expression". U.S. pat. US8835621B2 (United States). Intrexon Corp. Sept. 16, 2014. URL: https://patents.google.com/patent/US8835621B2/en.

[3] Jason P. Schrum et al. "Modified nucleosides, nucleotides, and nucleic acids, and uses thereof". U.S. pat. US9334328B2 (United States). ModernaTX, Inc. May 10, 2016. URL: https://patents.google.com/patent/US9334328B2/en.

[4] Xiaohan Luan et al. "Innate immune responses to RNA: sensing and signaling". In: *Frontiers in Immunology* 15 (2024), p. 1287940. DOI: 10.3389/fimmu.2024.1287940.

[5] M. Ceppi et al. "Double-stranded secondary structures on mRNA induce type I interferon (IFN α/β) production and maturation of mRNA-transfected monocyte-derived dendritic cells". In: *The Journal of Gene Medicine* 7.4 (2005), pp. 452–465. DOI: 10.1002/jgm.685.

[6] T. G. Johnstone, A. A. Bazzini, and A. J. Giraldez. "Upstream ORFs are prevalent translational repressors in vertebrates". In: *The EMBO Journal* 35.7 (Apr. 2016), pp. 706–723. DOI: 10.15252/embj.201592759.

[7] Natalia Ryczek, Aneta Łyś, and Izabela Makałowska. "The Functional Meaning of 5' UTR in Protein-Coding Genes". In: *International Journal of Molecular Sciences* 24.3 (2023). ISSN: 1422-0067. DOI: 10.3390/ijms24032976.

[8] Elizaveta Razumova et al. "Structural Features of 5' Untranslated Region in Translational Control of Eukaryotes". In: *International Journal of Molecular Sciences* 26.5 (2025). ISSN: 1422-0067. DOI: 10.3390/ijms26051979.

[9] Longfei Jia et al. "Decoding mRNA translatability and stability from the 5′ UTR". In: *Nature Structural & Molecular Biology* 27.9 (2020), pp. 814–821. DOI: 10.1038/s41594-020-0465-x.

[10] Katalin Karikó et al. "Suppression of RNA recognition by Toll-like receptors: The impact of nucleoside modification and the evolutionary origin of RNA". In: *Immunity* 23.2 (2005), pp. 165–175. DOI: 10.1016/j.immuni.2005.06.008.

[11] Silvia Zucchelli et al. "Engineering translation in mammalian cell factories to increase protein yield: The unexpected use of long non-coding SINEUP RNAs". In: *Computational and Structural Biotechnology Journal* 14 (2016), pp. 404–410. DOI: 10.1016/j.csbj.2016.10.004.

[12] Peter Eisenhut et al. "Systematic use of synthetic 5'-UTR RNA structures to tune protein translation improves yield and quality of complex proteins in mammalian cell factories". In: *Nucleic Acids Research* 48.20 (Nov. 2020), e119. DOI: 10.1093/nar/gkaa847.

[13] Jicong Cao et al. "High-throughput 5′ UTR engineering for enhanced protein production in non-viral gene therapies". In: *Nature Communications* 12, 4138 (July 6, 2021). DOI: 10.1038/s41467-021-24436-7.

[14] Paul J. Sample et al. "Human 5' UTR design and variant effect prediction from a massively parallel translation assay". In: *Nature Biotechnology* 37.7 (2019), pp. 803–809. DOI: 10.1038/s41587-019-0164-5.

[15] Sebastian Castillo-Hair et al. "Optimizing 5'UTRs for mRNA-delivered gene editing using deep learning". In: *Nature Communications* 15 (2024), p. 5284. DOI: 10.1038/s41467-024-49508-2.

[16] S. Yoon et al. "Designing 5' UTR sequences improves the capacity of mRNA therapeutics in preclinical models of aging and obesity". In: *Molecular Therapy* (2026). DOI: 10.1016/j.ymthe.2025.12.060.

[17] Logan S. Richards et al. "Targeting α-synuclein translation: Novel PROTEIMERs as 5' UTR directed inhibitors". In: *Journal of Alzheimer's Disease* 106 (2025), pp. 1187–1197. DOI: 10.1177/13872877251351305.

[18] Hannes H Loeffler et al. "Reinvent 4: modern AI–driven generative molecule design". In: *Journal of Cheminformatics* 16.1 (2024), p. 20. DOI: 10.1186/s13321-024-00812-5.

[19] Gökçe Geylan et al. "PepINVENT: generative peptide design beyond natural amino acids". In: *Chemical Science* 16.20 (2025), pp. 8682–8696. DOI: 10.1039/D4SC07642G.

[20] Mary Ann Clark, Matthew Douglas, and Jung Choi. *Biology 2e*. 2nd ed. Accessed: 2025-08-29. Houston, TX: OpenStax, Rice University, 2020. Chap. 3.5 Nucleic Acids. URL: https://openstax.org/books/biology-2e/pages/3-5-nucleic-acids.

[21] Marilyn Kozak. "Point mutations define a sequence flanking the AUG initiator codon that modulates translation by eukaryotic ribosomes". In: *Cell* 44.2 (1986), pp. 283–292. URL: https://www.sciencedirect.com/science/article/pii/0092867486907622.

[22] Marilyn Kozak. "Influences of mRNA secondary structure on initiation by eukaryotic ribosomes". In: *Proceedings of the National Academy of Sciences* 83.9 (1986), 2850–2854. DOI: 10.1073/pnas.83.9.2850.

[23] X. Q. Wang et al. "5′-Untranslated regions with multiple upstream AUGs can support low-level translation via leaky scanning and reinitiation". In: *Nucleic Acids Research* 32 (2004), pp. 1382–1392. DOI: 10.1093/nar/gkh305.

[24] Nicola Whiffin et al. "Characterising the loss-of-function impact of 5′ untranslated region variants in human disease". In: *Nature Communications* 11 (2020), pp. 1–9. DOI: 10.1038/s41467-019-10717-9.

[25] R. F. Harvey et al. "Trans-acting translational regulatory RNA binding proteins". In: *Wiley Interdisciplinary Reviews: RNA* 9.3 (2018), e1465. DOI: 10.1002/wrna.1465.

[26] Akira Fukao, Takumi Tomohiro, and Toshinobu Fujiwara. "Translation initiation regulated by RNA-binding proteins in mammals: the modulation of translation initiation complex by trans-acting factors". In: *Cells* 10 (2021), p. 1711. DOI: 10.3390/cells10071711.

[27] Marilyn Kozak. "Structural features in eukaryotic mRNAs that modulate the initiation of translation". In: *The Journal of Biological Chemistry* 266.30 (1991), pp. 19867–19870.

[28] Alan G. Hinnebusch. "The scanning mechanism of eukaryotic translation initiation". In: *Annual Review of Biochemistry* 85.1 (2016), pp. 35–70. DOI: 10.1146/annurev-biochem-060713-035802.

[29] Kathrin Leppek, Rhiju Das, and Maria Barna. "Functional 5′ UTR mRNA structures in eukaryotic translation regulation and how to find them". In: *Nature Reviews Molecular Cell Biology* 19.3 (2018), pp. 158–174. DOI: 10.1038/nrm.2017.103.

[30] Julian L. Huppert. "Four-stranded nucleic acids: structure, function and targeting of G-quadruplexes". In: *Chemical Society Reviews* 37.7 (2008), pp. 1375–1384. DOI: 10.1039/B702491F.

[31] Ronny Lorenz et al. "ViennaRNA Package 2.0". In: *Algorithms for molecular biology* 6.1 (2011), p. 26. DOI: 10.1186/1748-7188-6-26.

[32] ViennaRNA Package Developers. *ViennaRNA Package*. Accessed: August 2025. 2025. URL: https://www.tbi.univie.ac.at/RNA/.

[33] Michael Zuker and Patrick Stiegler. "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information". In: *Nucleic acids research* 9.1 (1981), pp. 133–148. DOI: 10.1093/nar/9.1.133.

[34] Ivo L Hofacker et al. "Fast folding and comparison of RNA secondary structures". In: *Monatshefte für Chemie/Chemical Monthly* 125.2 (1994), pp. 167–188. DOI: 10.1007/BF00818163.

[35] John S. McCaskill. "The equilibrium partition function and base pair binding probabilities for RNA secondary structure". In: *Biopolymers* 29.6-7 (1990), pp. 1105–1119. DOI: 10.1002/bip.360290621.

[36] Bhavana Khemani et al. "A review of graph neural networks: concepts, architectures and applications". In: *Journal of Big Data* 11.1 (2024), p. 23. DOI: 10.1186/s40537-023-00876-4.

[37] David Buterez et al. "Transfer learning with graph neural networks for improved molecular property prediction in the multi-fidelity setting". In: *Nature Communications* 15 (2024), p. 1517. DOI: 10.1038/s41467-024-45566-8.

[38] Haoquan Liu et al. "RNA–protein interaction prediction using network-guided deep learning". In: *Communications Biology* 8 (2025). DOI: 10.1038/s42003-025-07694-9.

[39] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[40] PyTorch Contributors. *LSTM — PyTorch Stable Documentation*. Accessed: 2025-08-27. 2025. URL: https://docs.pytorch.org/docs/stable/generated/torch.nn.LSTM.html.

[41] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[42] Daniel Quang and Xiaohui Xie. "DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences". In: *Nucleic Acids Research* 44.11 (2016), e107. DOI: 10.1093/nar/gkw226.

[43] Xiaoyong Pan et al. "Prediction of RNA–protein sequence and structure binding preferences using deep convolutional and recurrent neural networks". In: *BMC Genomics* 19 (2018), p. 511. DOI: 10.1186/s12864-018-4889-1.

[44] Dipan Shaw et al. "DeepLPI: a multimodal deep learning method for predicting the interactions between lncRNAs and protein isoforms". In: *BMC Bioinformatics* 22 (2021), p. 24. DOI: 10.1186/s12859-020-03914-7.

[45] Siwen Wu and Jun-tao Guo. "Improved prediction of DNA and RNA binding proteins with deep learning models". In: *Briefings in Bioinformatics* 25.4 (June 2024), bbae285. ISSN: 1477-4054. DOI: 10.1093/bib/bbae285.

[46] Jinho Im, Byungkyu Park, and Kyungsook Han. "A generative model for constructing nucleic acid sequences binding to a protein". In: *BMC Genomics* 20.Suppl 13 (2019), p. 967. DOI: 10.1186/s12864-019-6299-4.

[47] Byungkyu Park and Kyungsook Han. "Discovering protein-binding RNA motifs with a generative model of RNA sequences". In: *Computational Biology and Chemistry* 84 (2020), p. 107171. DOI: 10.1016/j.compbiolchem.2019.107171.

[48] Vishal Agarwal, N. Jayanth Kumar Reddy, and Ashish Anand. "Unsupervised Representation Learning of DNA Sequences". In: *arXiv preprint arXiv:1906.03087* (2019). DOI: 10.48550/arXiv.1906.03087.

[49] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[50] Ronald J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8.3-4 (1992), pp. 229–256. DOI: 10.1007/BF00992696.

[51] Yoshua Bengio et al. "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. 2009, pp. 41–48. DOI: 10.1145/1553374.1553380.

[52] Carlos Florensa et al. "Reverse Curriculum Generation for Reinforcement Learning". In: *Proceedings of the 1st Conference on Robot Learning*. 2017, pp. 482–495. URL: https://proceedings.mlr.press/v78/florensa17a.html.

[53] Claude E. Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27 (1948), pp. 379–423, 623–656. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[54] Thomas D. Schneider and R. Michael Stephens. "Sequence logos: a new way to display consensus sequences". In: *Nucleic Acids Research* 18.20 (1990), pp. 6097–6100. DOI: 10.1093/nar/18.20.6097.

[55] Keyao Pan and Michael W Deem. "Quantifying selection and diversity in viruses by entropy methods, with application to the haemagglutinin of H3N2 influenza". In: *Journal of the Royal Society Interface* 8.64 (2011), pp. 1644–1653. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3177615/.

[56] Marilyn Kozak. "The scanning model for translation: an update". In: *Journal of Cell Biology* 108.2 (1989), pp. 229–241. DOI: 10.1083/jcb.108.2.229.

[57] Grzegorz Kudla et al. "Coding-sequence determinants of gene expression in Escherichia coli". In: *Science* 324.5924 (2009), pp. 255–258. DOI: 10.1126/science.1170160.

[58] Tamir Tuller et al. "An evolutionarily conserved mechanism for controlling the efficiency of protein translation". In: *Cell* 141.2 (2010), pp. 344–354. DOI: 10.1016/j.cell.2010.03.031.

[59] Tamir Tuller et al. "Composite effects of gene determinants on the translation speed and density of ribosomes". In: *Genome Biology* 12.11 (2011), R110. DOI: 10.1186/gb-2011-12-11-r110.

[60] David E. Weinberg et al. "Improved ribosome-footprint and mRNA measurements provide insights into dynamics and regulation of yeast translation". In: *Cell Reports* 14.7 (2016), pp. 1787–1799. DOI: 10.1016/j.celrep.2016.01.043.

[61] Alexander Karollus, Žiga Avsec, and Julien Gagneur. "Predicting mean ribosome load for 5′ UTR of any length using deep learning". In: *PLoS Computational Biology* 17.5 (2021), e1008982. DOI: 10.1371/journal.pcbi.1008982.

[62] Frederick Korbel, Ekaterina Eroshok, and Uwe Ohler. "Interpreting deep neural networks for the prediction of translation rates". In: *BMC Genomics* 25.1 (2024), p. 1061. DOI: 10.1186/s12864-024-10925-8.

[63] Yanyi Chu et al. "A 5' UTR language model for decoding untranslated regions of mRNA and function predictions". In: *Nature Machine Intelligence* 6.4 (2024), pp. 449–460. DOI: 10.1038/s42256-024-00823-9.

[64] Jiayang Chen et al. "Interpretable RNA Foundation Model from Unannotated Data for Highly Accurate RNA Structure and Function Predictions". In: *arXiv* abs/2204.00300 (2022). preprint. DOI: 10.48550/arXiv.2204.00300.

[65] Manato Akiyama and Yasubumi Sakakibara. "Informative RNA base embedding for RNA structural alignment and clustering by deep representation learning". In: *NAR Genomics and Bioinformatics* 4.1 (2022), lqac012. DOI: 10.1093/nargab/lqac012.

[66] The RNAcentral Consortium. "RNAcentral: a hub of information for non-coding RNA sequences". In: *Nucleic Acids Research* 47.D1 (2019), pp. D221–D229. DOI: 10.1093/nar/gky1034.

[67] Rafael Josip Penić et al. "RiNALMo: General-purpose RNA language models can generalize well on structure prediction tasks". In: *Nature Communications* 16.1 (2025), p. 5671. DOI: 10.1038/s41467-025-60872-5.

[68] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998. ISBN: 9780262631853. DOI: 10.7551/mitpress/3927.001.0001.

[69] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 27. Curran Associates, Inc., 2014, pp. 2672–2680. URL: https://papers.nips.cc/paper/5423-generative-adversarial-nets.

[70] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=rkE3y85ee.

[71] Sina Barazandeh et al. "UTRGAN: learning to generate 5' UTR sequences for optimized translation efficiency and gene expression". In: *Bioinformatics Advances* 5.1, vbaf134 (2025). DOI: 10.1093/bioadv/vbaf134.

[72] Aidan T Riley et al. "Generative and predictive neural networks for the design of functional RNA molecules". In: *Nature Communications* 16.1 (2025), p. 4155. DOI: 10.1038/s41467-025-59389-8.

[73] Xiaoshan Tang et al. "A novel deep generative model for mRNA vaccine development: Designing 5′ UTRs with N^1-methyl-pseudouridine modification". In: *Acta Pharmaceutica Sinica B* 14.4 (2024), pp. 1814–1826. DOI: 10.1016/j.apsb.2023.11.003.

[74] Danqi Liao et al. "To Optimize, Not to Invent: RNAGenScape for mRNA Sequence Generation and Optimization Without de novo Design". In: *ICML 2025 Generative AI and Biology (GenBio) Workshop*. 2025. URL: https://openreview.net/forum?id=9g59bzMy1G.

[75] Keisuke Yamada et al. "Multi-objective computational optimization of human 5' UTR sequences". In: *Briefings in Bioinformatics* 26.3 (2025). DOI: 10.1093/bib/bbaf225.

[76] Shaked Brody, Uri Alon, and Eran Yahav. "How Attentive are Graph Attention Networks?" In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=F72ximsx7C1.

[77] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. "Order Matters: Sequence to sequence for sets". In: *CoRR* abs/1511.06391 (2015). URL: https://api.semanticscholar.org/CorpusID:260429228.

[78] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference (SciPy 2008)*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA, USA, 2008, pp. 11–15. DOI: 10.25080/TCWV9851.

[79] Petar Veličković et al. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.

[80] PyTorch Geometric Developers. *PyTorch Geometric: Set2Set Aggregation*. https://pytorch-geometric.readthedocs.io/en/stable/generated/torch_geometric.nn.aggr.Set2Set.html. Accessed: 2025-08-15. 2025.

[81] Paul J. Sample, Bichlien Wang, and Georg Seelig. *GSE114002: Human 5' UTR design and variant effect prediction from a massively parallel translation assay*. NCBI Gene Expression Omnibus. GEO Series accession: GSE114002. Available at: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE114002. 2018.

[82] M. Hasegawa, H. Kishino, and T. Yano. "Dating of the human-ape splitting by a molecular clock of mitochondrial DNA". In: *Journal of Molecular Evolution* 22.2 (1985), pp. 160–174. DOI: 10.1007/bf02101694.

[83] S. R. Eddy and R. Durbin. "RNA sequence analysis using covariance models". In: *Nucleic Acids Research* 22.11 (1994), pp. 2079–2088. DOI: 10.1093/nar/22.11.2079.

[84] Carine Barreau, Laurent Paillard, and Hazel B. Osborne. "AU-rich elements and associated factors: Are there unifying principles?" In: *Nucleic Acids Research* 33.22 (2005), pp. 7138–7150. DOI: 10.1093/nar/gki1012.

[85] Debashish Ray et al. "A compendium of RNA-binding motifs for decoding gene regulation". In: *Nature* 499.7457 (2013), pp. 172–177. DOI: 10.1038/nature12311.

[86] Jesper Tholstrup, Lene B. Oddershede, and Michael A. Sørensen. "mRNA pseudoknot structures can act as ribosomal roadblocks". In: *Nucleic Acids Research* 40.1 (2012), pp. 303–313. DOI: 10.1093/nar/gkr686.

[87] Olivier Namy et al. "A mechanical explanation of RNA pseudoknot function in programmed ribosomal frameshifting". In: *Nature* 441.7090 (2006), pp. 244–247. DOI: 10.1038/nature04735.

[88] Chen Bao et al. "Specific length and structure rather than high thermodynamic stability enable regulatory mRNA stem-loops to pause translation". In: *Nature Communications* 13.1 (2022), p. 988. DOI: 10.1038/s41467-022-28600-5.

[89] Kengo Sato et al. "IPknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming". In: *Bioinformatics* 27.13 (2011), pp. i85–i93. DOI: 10.1093/bioinformatics/btr215.

[90] Peter Kerpedjiev, Stefan Hammer, and Ivo L. Hofacker. "Forna (force-directed RNA): Simple and effective online RNA secondary structure diagrams". In: *Bioinformatics* 31.20 (2015), pp. 3377–3379. DOI: 10.1093/bioinformatics/btv372.

[91] Zhitao Ying et al. "GNNExplainer: Generating Explanations for Graph Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019, pp. 9244–9255. URL: https://proceedings.neurips.cc/paper/2019/hash/d80b7040b773199015de6d3b4293c8ff-Abstract.html.

[92] Sebastian Castillo-Hair et al. *GSE232927: Optimizing 5' UTRs for mRNA-delivered gene editing using deep learning*. NCBI Gene Expression Omnibus. GEO Series accession: GSE232927. Available at: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE232927. 2023.

[93] S. Pan et al. "UTR-Insight: integrating deep learning for efficient 5' UTR discovery and design". In: *BMC Genomics* 26.1 (Jan. 2025), p. 107. DOI: 10.1186/s12864-025-11269-7.

[94] Markus Schlee. "Master sensors of pathogenic RNA – RIG-I like receptors". In: *Immunobiology* 218.11 (2013), pp. 1322–1335. DOI: 10.1016/j.imbio.2013.06.007.

[95] Norbert Pardi et al. "mRNA vaccines — a new era in vaccinology". In: *Nature Reviews Drug Discovery* 17.4 (2018), pp. 261–279. DOI: 10.1038/nrd.2017.243.

[96] Hidenori Tani et al. "Genome-wide determination of RNA stability reveals hundreds of short-lived noncoding transcripts in mammals". In: *Genome Research* 22.5 (2012), pp. 947–956. DOI: 10.1101/gr.130559.111.

[97] Xingyu Chen et al. "Ctrl-DNA: Controllable Cell-Type-Specific Regulatory DNA Design via Constrained RL". In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025. URL: https://openreview.net/forum?id=JeXkIy0JyM.

[98] Stephen P. Plassmeyer et al. "Approaches for identification of 5' UTR mutations impacting translation and protein production from neurodevelopmental disorder genes". In: *Cell Reports Methods* 5.12 (Dec. 2025), p. 101247. DOI: 10.1016/j.crmeth.2025.101247.