universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Benchmarking and Optimizing Deep Learning Architectures for Protein-to-mRNA Ratio Prediction

verfasst von | submitted by

Felix Elias Krause BSc (WU)

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 645
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Data Science
Betreut von | Supervisor:	Assoz. Prof. Dipl.-Ing. Dr.techn. Sebastian Tschiatschek BSc

Contents

Contents i

Acknowledgements iv

Abstract v

Zusammenfassung vi

List of Tables vii

List of Figures vii

1 Introduction 1
- 1.1 Setting the Scene . . . 1
- 1.2 Research Questions . . . 1
- 1.3 Challenges . . . 2
- 1.4 Novelties & Contributions . . . 2
- 1.5 Outline . . . 3

2 Related Work 4
- 2.1 PTR Ratio Prediction . . . 4
- 2.2 Deep Learning Architectures for Genomic Sequences . . . 5

3 Background 7
- 3.1 Genomics . . . 7
  - 3.1.1 Transcription . . . 7
  - 3.1.2 Translation . . . 8
  - 3.1.3 Codons . . . 8
  - 3.1.4 RNA Structure . . . 8
  - 3.1.5 Regulatory Mechanisms of Gene Expression . . . 9
- 3.2 Foundations of Machine Learning . . . 11
  - 3.2.1 Supervised Learning . . . 11
  - 3.2.2 Classification . . . 12
  - 3.2.3 Model Generalization & Evaluation . . . 12
  - 3.2.4 Over- and Underfitting . . . 12
  - 3.2.5 Random Forest Classifier . . . 13
- 3.3 Deep Learning . . . 13
  - 3.3.1 Cross Entropy Loss Function . . . 14
  - 3.3.2 Optimization . . . 14
  - 3.3.3 Pretraining with Denoising Autoencoders . . . 15
  - 3.3.4 Evaluation: Area Under the Curve . . . 15
- 3.4 Deep Learning Architectures . . . 16
  - 3.4.1 Dense Feedforward Neural Networks . . . 16
  - 3.4.2 Convolutional Neural Networks . . . 17
  - 3.4.3 Recurrent Neural Networks . . . 20
  - 3.4.4 Transformers . . . 23
  - 3.4.5 Selective Structured State Space Models . . . 24
- 3.5 Encoding Biological Sequences . . . 26
  - 3.5.1 Encoding RNA Sequences . . . 26

3.5.2 Encoding RNA Structure . . . 28

4 Methodology **29**
4.1 Data . . . 29
4.1.1 Data Sources . . . 29
4.1.2 Target Variable . . . 29
4.1.3 Data Pre-Processing . . . 30
4.1.4 Train, Validation, and Test Split . . . 32
4.2 Benchmarking Models . . . 32
4.2.1 Training Pipeline . . . 33
4.2.2 Backbone Model Implementations . . . 34
4.2.3 Codon Frequency Baseline Models . . . 36
4.2.4 Hyperparameter Tuning . . . 36
4.2.5 Evaluation . . . 36
4.3 Model Optimization: Custom PTRnet Model . . . 37
4.3.1 Training & Tuning . . . 37
4.3.2 Evaluation . . . 38

5 Results **39**
5.1 Benchmarking Results . . . 39
5.1.1 Hyperparameter Tuning Results . . . 39
5.1.2 Analysis of Hyperparameter Tuning . . . 39
5.1.3 Benchmarking Results . . . 41
5.1.4 Computational Efficiency and Model Complexity . . . 42
5.2 PTRnet Optimization Results . . . 43
5.2.1 PTRnet Configuration Tuning . . . 43
5.2.2 Final PTRnet Architecture . . . 45
5.2.3 Ablation Study Results . . . 45
5.3 Overall Test Set Results . . . 47
5.4 PTRnet Predictions Analysis . . . 48
5.4.1 Prediction Distribution . . . 48
5.4.2 Predictions per Tissue . . . 49

6 Discussion **52**
6.1 Discussion of Results . . . 52
6.2 Limitations . . . 53
6.3 Future Work . . . 54

7 Conclusion **56**

References **58**

Acronyms **65**

Appendix **66**

A Detailed Dataset Composition **66**

B Sequence Length Distributions **68**

C Benchmarking Hyperparameter Tuning Analysis **69**

D Hardware and Software Configuration of Experimental Servers **74**

E Benchmarking Test Set Metrics **75**

F Benchmarking Train and Test Set Training Curves **76**

G Hyperparameter Tuning of PTRnet **77**

H PTRnet Ablation Study Train and Test Set Training Curves **78**

Acknowledgements

If someone had asked me during the first weeks of this program whether I would make it through the first semester, I probably would have said no. With intense courses and a heavy workload, the thought of dropping out crossed my mind more than once. But thanks to my incredible colleagues who encouraged me to keep going and supported me in every way they could, I pulled through. Without them, I genuinely do not believe I would have learned and understood fast enough to keep the studies up. They not only helped me endure that challenging time but also made it one of the most memorable and rewarding periods of my life. I am deeply grateful to my fellow students – especially Damian Bednarz, Alina Behrens, Michael Oster, and Jonas Quenzer – for their unwavering support and friendship.

My heartfelt thanks also go to my family, who have always stood by me in both my academic journey and life more broadly. And to my second family – my closest friends – thank you for lifting me up in moments of doubt and helping me stay motivated.

I am also grateful to the University of Vienna for the high-quality education I received. Beyond teaching me about data science and AI, it allowed me to grow personally and professionally. Even more, it helped me discover what I can now truly call my passion. I especially appreciate the university's provision of the computational resources that made this thesis possible.

Finally, I would like to sincerely thank my supervisor, Sebastian Tschiatschek, for his kind support, guidance, and patience throughout the thesis process. His critical and constructive feedback was invaluable. Thanks also to Gabriele Martino for engaging discussions and valuable input along the way.

Though this thesis bears only my name, it is the result of the support and encouragement of many people and institutions that enabled me to come this far – and especially during my master's studies. I am deeply thankful to all of them.

Abstract

Designing optimal mRNA sequences to enhance protein synthesis in human cells is crucial for the advancement of gene therapies and mRNA vaccines. In this context, the Protein-to-mRNA (PTR) ratio serves as a useful proxy for translation efficiency. This master's thesis systematically benchmarks nine state-of-the-art deep learning architectures, along with simple baselines, to classify whether an mRNA sequence results in a low or high PTR ratio across 29 human tissues. All models are trained on 11,000 mRNA sequences using a 70/15/15 train/validation/test split, with performance measured by the area under the receiver operating characteristic curve (AUC). Building upon the best-performing backbone, we introduce PTRnet, an enhanced version of the CNN-based RiboNN architecture. PTRnet integrates secondary structure features, domain-specific training strategies, and unsupervised pretraining. While several sequence-based models show reasonable generalization – e.g., RiboNN achieving a 69.1% test AUC and PTRnet 68.6% – they are limited by early overfitting during training. Notably, a simple codon frequency-based Multilayer Perceptron (MLP) achieves a 72.2% test AUC, substantially outperforming both a Random Forest baseline (66.8%) and the more complex pretrained PTRnet. These results suggest that the deep models evaluated struggle to capture subtle patterns beyond what is already encoded in codon frequencies, which appear to carry much of the signal necessary to distinguish between low and high protein expression.

Source code: github.com/f-krause/master-thesis

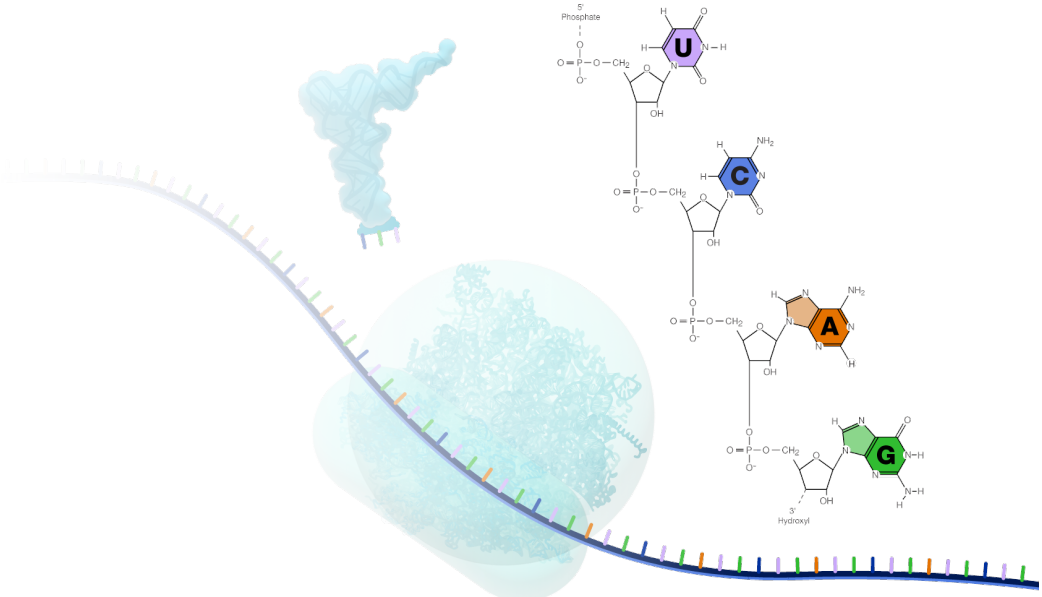


Illustration of National Human Genome Research Institute (2025) in public domain.

Zusammenfassung

In der Entwicklung von Gentherapien und mRNA-basierten Impfstoffen ist die gezielte Optimierung von mRNA-Sequenzen zur Steigerung der Proteinsynthese in menschlichen Zellen von zentraler Bedeutung. In diesem Kontext dient das Verhältnis von Protein zu mRNA (PTR-Ratio) als nützlicher Proxy zur Abschätzung der Translationseffizienz. Die vorliegende Masterarbeit untersucht systematisch die Leistungsfähigkeit von neun modernen Deep-Learning-Architekturen sowie einfachen Baselines bei der Klassifikation, ob eine gegebene mRNA-Sequenz in 29 verschiedenen menschlichen Geweben zu einer niedrigen oder hohen PTR-Ratio führt. Alle Modelle wurden auf einem Datensatz von 11.000 mRNA-Sequenzen mit einem 70/15/15-Split für Training, Validierung und Test trainiert bzw. evaluiert. Die Modellleistung wird anhand der Fläche unter der Receiver-Operating-Characteristic-Kurve (AUC) bewertet. Aufbauend auf dem leistungsstärksten Backbone-Modell wird PTRnet eingeführt, eine erweiterte Version der CNN-basierten RiboNN-Architektur. PTRnet integriert zusätzlich Informationen über die Sekundärstruktur der mRNA, domänenspezifische Trainingsstrategien sowie Unsupervised Pretraining. Während mehrere sequenzbasierte Modelle eine gewisse Generalisierungsfähigkeit zeigen – etwa RiboNN mit einer Test-AUC von 69,1% und PTRnet mit 68,6% – ist ihre Qualität durch frühes Overfitting im Trainingsprozess limitiert. Ein einfaches Multilayer-Perceptron (MLP), das lediglich auf Codonhäufigkeiten basiert, erreicht allerdings eine deutlich höhere Test-AUC von 72,2% und übertrifft damit sowohl die Random-Forest-Baseline (66,8%) als auch das komplexere, pretrained PTRnet. Diese Ergebnisse deuten darauf hin, dass die untersuchten Deep-Learning-Modelle Schwierigkeiten haben, subtilere Muster zu erfassen, die über die in den Codonfrequenzen enthaltenen Informationen hinausgehen. Letztere enthalten offenbar bereits den Großteil der Information, die zur Unterscheidung zwischen niedriger und hoher Proteinexpression erforderlich ist.

Quellcode: github.com/f-krause/master-thesis

List of Tables

1 Features of codon- and nucleotide-encoded dataset . . . 31
2 Number of samples in codon- and nucleotide-encoded datasets . . . 33
3 Best parameters of hyperparameter tuning . . . 40
4 Search spaces and best configuration of PTRnet . . . 44

List of Figures

1 The central dogma of molecular biology . . . 7
2 Correspondence between codons and amino acids . . . 9
3 Secondary structure types identified by the bpRNA algorithm . . . 10
4 Under- and overfitting depending on model complexity . . . 13
5 Illustration of the Receiver Operating Characteristic (ROC) and its associated Area Under the Curve (AUC) . . . 16
6 Dense feedforward neural network . . . 17
7 Simple CNN network . . . 18
8 LegNet architecture . . . 19
9 RNN illustration . . . 20
10 LSTM unit . . . 21
11 GRU unit . . . 22
12 xLSTM architecture . . . 23
13 Transformer architecture . . . 25
14 Selective State Space Model architecture . . . 26
15 Mamba block architecture . . . 27
16 Encodings for RNA sequences and RNA structure . . . 27
17 Classification procedure for one mRNA sequence's targets . . . 30
18 Demonstration of nucleotide-level data structure with secondary structure and loop type predictions . . . 31
19 Training pipeline for model benchmarking . . . 34
20 Test and validation set AUC scores across models . . . 41
21 Overview of benchmarking results . . . 42
22 Relative hyperparameter importance of PTRnet tuning . . . 44
23 PTRnet architecture and pretraining . . . 46
24 Ablation study results of PTRnet . . . 46
25 Overview performance scores of major models . . . 48
26 Confusion matrix and number of correctly classified targets per sequence frequency of PTRnet and MLP (freq) . . . 49
27 Model AUC scores per tissue . . . 51

1 Introduction

1.1 Setting the Scene

Recent advancements in biotechnology and molecular biology have significantly accelerated research and clinical applications in gene therapy and vaccine development, as seen in the fast COVID-19 vaccine development (Hernandez-Alias et al., 2023; Kim et al., 2024; Shen et al., 2024; Xue et al., 2023). At the forefront of these innovations is messenger RNA (mRNA), which plays a pivotal role as an intermediary between genetic information encoded in DNA and protein synthesis.

Being able to predict the amount of protein produced from a given mRNA sequence is crucial to better understand the underlying mechanisms of gene expression, the process by which genetic information in DNA is converted into functional proteins. This knowledge can help optimize mRNA vaccine design, enhance gene therapy efficacy, and reduce side-effects of such treatments (Eraslan et al., 2019; Gould et al., 2014; Hwang et al., 2024; Kim et al., 2024; Shen et al., 2024). Current studies indicate considerable opportunities for effective therapies for rare diseases (Shen et al., 2024) and cancer therapy (Kong et al., 2023) based on mRNA technology.

The rate at which proteins are synthesized from mRNA sequences in human cells can be estimated by the Protein-to-mRNA (PTR) ratio. This ratio serves as a proxy measure for translational efficiency and is critical for understanding how genes regulate protein production independently of mRNA levels in the cell (Franks et al., 2017; Hernandez-Alias et al., 2023). Although previous research extensively studied gene expression and protein synthesis factors (Eraslan et al., 2019; Franks et al., 2017; Guimaraes et al., 2014; Hernandez-Alias et al., 2023; Vogel et al., 2010), prediction methods for PTR ratios using Deep Learning (DL) remain underexplored.

Employing DL techniques to predict PTR ratios, however, presents promising opportunities. DL models excel in capturing complex, non-linear relationships and nuanced interactions within biological sequences and large datasets (Hwang et al., 2024; Zrimec et al., 2021). The advent of high-throughput RNA sequencing has generated vast RNA biology datasets (Fagerberg et al., 2014; Frankish et al., 2023), which have been successfully utilized by DL models across various biological applications (Hwang et al., 2024). These precedents demonstrate that sequence-based neural networks can extract transferable features from nucleotide data and therefore hold strong potential for advancing PTR ratio prediction as well.

Drawing parallels to Natural Language Processing (NLP), the nucleotides of an mRNA sequence can be compared to words in sentences. Just as linguistic context influences the meaning of words, biological context and structural information impact gene expression (Gould et al., 2014). Consequently, established NLP techniques and models could be applied to this setting. DL sequence models such as Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997), Transformer (He et al., 2023; Vaswani et al., 2017), and the novel State Space Model (SSM), Mamba (Gu & Dao, 2023), could be well suited to model the complex biological dependencies required for effective PTR ratio prediction.

Despite the availability of large-scale, high-quality transcriptomic and proteomic datasets, as used, for example, in Eraslan et al. (2019), until May 2025 no peer-reviewed and published deep learning-based study on PTR ratio prediction using raw mRNA sequence data has been found. Identified previous work has primarily relied on shallow models and custom engineered features (Eraslan et al., 2019; Hernandez-Alias et al., 2023; Vogel et al., 2010). This thesis aims to close this gap by applying deep learning directly to raw sequences in the context of PTR ratio classification.

1.2 Research Questions

To systematically investigate the potential of DL for PTR ratio classification, the following research questions will be addressed:

1. How accurately can mRNA sequences be classified into low- and high-PTR ratio outcome across different human tissues using state-of-the-art deep learning models?

2. Which additional features, particularly secondary structure predictions, enhance the predictive accuracy of PTR ratio classification?

3. Can the most promising DL architecture be further improved through extensive hyperparameter tuning, architectural enhancements, and advanced training strategies such as unsupervised pretraining?

1.3 Challenges

The application of DL to biological sequence data, particularly for predicting PTR ratios, presents several challenges:

- **Variable Sequence Lengths:** mRNA sequences vary significantly in length, from a few hundred to thousands of bases, complicating model training and inference.
- **Computational Complexity:** Processing long biological sequences using advanced DL architectures requires substantial computational resources, regarding storage, memory, and compute.
- **High Feature Dimensionality:** The high dimensionality of the input data, particularly when incorporating secondary structure features, poses challenges for model training, and generalization.
- **Generalizability:** The risk of overfitting due to the complexity of DL models and the limited dataset size necessitates careful regularization and validation strategies. Models trained on a specific dataset may not generalize well to unseen real-world data.
- **Large Search Space:** The multitude of possible data encodings, pre-processing steps, architectures, model configurations, and hyperparameters results in a high-dimensional search space that needs to be navigated.

1.4 Novelties & Contributions

This thesis aims to contribute to the field of bioinformatics and deep learning in several novel ways:

- **Integrated Tissue Modeling:** Unlike previous works that trained separate models for each tissue (Eraslan et al., 2019; Hernandez-Alias et al., 2023), we develop unified models capable of predicting PTR ratios across multiple tissues simultaneously.
- **Application of State-of-the-Art DL Models:** Early application and comprehensive benchmarking of DL models in the domain of PTR ratio prediction on raw sequences, including the novel Mamba (Gu & Dao, 2023), xLSTM (Beck et al., 2024), and mRNA tailored RiboNN architecture (Zheng et al., 2025).
- **Incorporation of mRNA Structural Information:** By integrating secondary structure and loop type predictions, this thesis evaluates the impact of structural information on translational efficiency predictions.
- **Targeted Model Optimization:** Focused optimization of the most promising DL architecture – including hyperparameter search and unsupervised pretraining – to quantify how far deep sequence models can be pushed for PTR ratio classification under realistic computational budgets.

1.5 Outline

This thesis follows a structured, two-step approach: starting with simple baseline models to establish performance benchmarks and then progressively increasing complexity by utilizing more advanced training procedures, enhancing the model architecture, and incorporating additional sequence information, such as secondary structure features. This incremental approach not only facilitates a robust comparison of model performance but also ensures computational feasibility despite the challenges posed by variable sequence lengths and high dimensionality.

Results indicate that while DL models can accurately classify PTR ratios across tissues using simple codon frequency-based features, more complex architectures suffer from overfitting, likely preventing them from capturing additional relevant information from raw sequences. Initial benchmarking of several widely used DL architectures demonstrated the feasibility of applying DL directly to raw sequences, ultimately leading to the identification of a lightweight CNN-based model with promising performance. However, further optimization, additional features, and pretraining of this architecture resulted in only minor, statistically non-significant improvements, suggesting that codon frequencies already encapsulate much of the relevant information required for PTR ratio classification and more complex models fail to capture more fine-grained distinctions. This observation is consistent with previous studies emphasizing the role of codon usage in translational efficiency (Guimaraes et al., 2014; Hernandez-Alias et al., 2023; Vogel et al., 2010). Notably, a frequency-based Multilayer Perceptron (MLP) substantially outperforms both sophisticated sequence-based models and shallow custom feature-based approaches from prior work, underscoring the potential of deep learning when codon frequencies are used as input features.

The remainder of this thesis is structured as follows: Chapter 2 provides an overview of related work in DL applications for gene expression prediction and mechanisms behind PTR ratio outcomes. Chapter 3 introduces foundational biological concepts and DL methodologies relevant to PTR ratio classification. Chapter 4 details the experimental setup, data processing, model architectures, and training procedures that underpin the empirical analysis. Chapter 5 presents benchmarking and ablation studies that progressively incorporate architectural refinements, structural features, and optimization strategies. Chapter 6 interprets the findings in light of biological constraints and methodological limitations, highlighting promising avenues for future research. Finally, Chapter 7 summarizes the key contributions and concludes the thesis.

2 Related Work

This chapter reviews the state of research relevant to predicting protein abundance from mRNA sequence data, with a particular focus on the PTR ratio. It first summarizes key studies that have investigated the determinants of protein levels and PTR ratio variation, highlighting both classical statistical approaches and more recent machine learning methods. Then the application of deep learning architectures to genomic sequence analysis is discussed, outlining advances in model design and their potential for improving predictive performance in this domain.

2.1 PTR Ratio Prediction

The amount of protein synthesized in a cell is largely determined by mRNA levels, as detailed in Section 3.1.5. However, additional variation in protein levels is captured by the protein-to-mRNA (PTR) ratio and understanding its determinants remains an active area of research.

Vogel et al. (2010) were among the first to investigate the relationship between mRNA and protein levels. Using multivariate adaptive regression splines (MARS), a shallow nonlinear regression method, they analyzed absolute protein and mRNA levels for over 1,000 genes from the human Daoy medulloblastoma cell line. Their model explained approximately two-thirds of the variance in protein abundance, attributing 31% to translation elongation factors and 27% to mRNA concentration. Additional contributing features included translation and degradation-related elements such as mRNA sequence length, amino acid properties, upstream open reading frames (uORFs), and 5' UTR secondary structures.

Similarly, Guimaraes et al. (2014) studied around 800 genes in Escherichia coli using partial least squares (PLS) regression with over 100 engineered mRNA sequence features. Their model also explained two-thirds of the protein abundance variation, with 53% attributed to mRNA levels and 12% to translation elongation factors, such as codon usage bias and specific amino acid compositions.

Franks et al. (2017) analyzed across-tissue PTR ratio variation using datasets of twelve human tissues. They found that this variation is substantially smaller than across-gene protein level differences and is primarily driven by post-transcriptional regulation or measurement noise, rather than by sequence-intrinsic properties. This underscores the challenge of predicting tissue-specific protein output from sequence data alone.

Lahtvee et al. (2017) applied MARS with mRNA sequence-based features of yeast and also found mRNA abundance (61%) and translation elongation features (15%) to be significant predictors of protein levels.

Expanding on these findings, Li et al. (2019) demonstrated that mRNA sequence features exert a stronger influence on translation rates than previously assumed. Analyzing several bacterial species and the human genome, they showed that these features could explain between 42% and 81% of the variance in translation rates.

Unlike earlier studies that primarily focused on predicting protein levels using mRNA sequence features, including mRNA levels, Eraslan et al. (2019) directly modeled the PTR ratio to better isolate the influence of purely sequence-based factors. They trained multivariate linear regression models on over 180 engineered features derived from 11,575 mRNA sequences across 29 human tissues. These features included codon usage frequency, secondary structure, GC content, and others. On average, their models explained 58% of the tissue-specific variance in PTR ratios. The authors concluded that mRNA sequence features alone account for a substantial portion of the variability in protein abundance across tissues, emphasizing that the genetic code influences both the synthesis and stability of proteins and mRNAs. However, their results did not support the existence of tissue-specific optimal codons, which are triplets of mRNA bases (see Section 3.1.5 for background information).

More recently, Hernandez-Alias et al. (2023) explored codon usage effects on protein synthesis across 36 human tissues. They trained tissue-specific random forest classifiers to distinguish between low- and high-PTR genes using codon frequency features, achieving an average Area Under the Curve

(AUC; explained in Section 3.3.4) of 63%. Their results indicate that codon preferences differ across tissues and can be exploited for tissue-specific gene design. They also observed that GC-ending codons are more prevalent in the human genome and are associated with higher mRNA and protein expression levels across genes, while across tissues some favor AT-ending codons. The present work builds upon the ideas and methodologies introduced in this study.

In their preprint[1], Zheng et al. (2025) developed a deep learning model based on Convolutional Neural Networks (CNNs; see Section 3.4.2), termed RiboNN, to predict translation efficiency – approximated by normalized ribosome density – from full mRNA sequences. The model was trained on a compendium of 2,277 ribosome profiling samples, achieving an R^2 of approximately 62% in predicting mean translation efficiency across 78 human and 68 mouse cell types. The authors benchmarked RiboNN against a classical machine learning approach by training a light gradient-boosting machine (LGBM) on hand-crafted features, similar to those used in prior studies and observed nearly identical performance for the human dataset. These features included regional and total sequence lengths, codon frequencies, nucleotide frequencies in untranslated regions (UTRs), amino acid frequencies, and secondary structure characteristics. Notably, secondary structure features did not significantly contribute to translation efficiency prediction.

2.2 Deep Learning Architectures for Genomic Sequences

To date, most approaches for predicting protein levels from mRNA have relied on shallow models trained on yeast, Escherichia coli, and human data, using engineered features rather than raw sequence data. DL methods specifically targeting human PTR ratio prediction are not yet established. Nevertheless, DL architectures are increasingly applied to biological sequence tasks and show strong potential for this domain (Hwang et al., 2024; Zrimec et al., 2021), although there are still issues regarding generalization of DL models for mRNA translation prediction to unseen datasets (Schlusser et al., 2024).

While Convolutional Neural Networks (CNNs; see Section 3.4.2) initially dominated DL-based approaches in genomics (Hwang et al., 2024; Zrimec et al., 2021), more advanced architectures have recently emerged that might be better equipped to capture the complex dependencies inherent in genomic sequences. Several of these models could also be promising for PTR ratio prediction.

DNABERT. DNABERT (Ji et al., 2021) is a BERT-based (Devlin et al., 2019) Transformer architecture pretrained on DNA sequences, accepting sequences of length up to 512 bases. Pretraining significantly boosts performance across various tasks, including promoter detection, splice site prediction, and identification of transcription factor binding sites. DNABERT-2, an improved version, promises better sequence encoding and training efficiency (Zhou et al., 2024).

RNAdegformer. RNAdegformer (He et al., 2023) combines Transformers (see Section 3.4.4) with convolutional layers to predict degradation rates of COVID-19 vaccine mRNA sequences. Developed for a 2020 Kaggle competition (Das et al., 2020), it was trained on sequences of up to 130 nucleotides enriched with secondary structure information. The model uses unsupervised pretraining, followed by full fine-tuning, and semi-supervised fine-tuning with pseudo-labels. It achieved high accuracy, placing 7th out of over 1,500 teams (Das et al., 2020; He et al., 2023; Wayment-Steele et al., 2022).

LegNet. LegNet (Penzar et al., 2023, see Section 3.4.2), inspired by EfficientNetV2 (Tan & Le, 2021), is a deep CNN trained on 6.7 million promoter sequences of length 80. It achieved the best performance in the 2022 DREAM challenge on expression prediction in yeast. The top-performing teams employed CNNs, Transformer variants, or hybrid architectures. Challenge results analysis showed that advanced

[1] Paper was not peer-reviewed at the time of consideration, 01.05.2025

training strategies and model optimization are similarly important as the architecture itself (Rafi et al., 2024).

HyenaDNA. HyenaDNA (Nguyen et al., 2023) is a long-range sequence model that addresses scaling limitations of Transformers through implicit convolutions. Pretrained on the human genome, it handles sequences up to one million nucleotides and performs well on various downstream genomics benchmarks, including species classification and biotype embeddings.

Mamba. Mamba (Gu & Dao, 2023, see Section 3.4.5) is a state space model (SSM) designed for efficient processing of long sequences, offering linear scaling, fewer parameters, and strong performance. It outperforms HyenaDNA on several long-sequence benchmarks. Caduceus (Schiff et al., 2024), a recent extension, enhances Mamba with bidirectionality and other refinements, achieving even better results after pretraining on the human genome and task-specific fine-tuning.

Bio-xLSTM. The xLSTM architecture (Beck et al., 2024, see Section 3.4.3) improves upon standard LSTMs (Hochreiter & Schmidhuber, 1997, see Section 3.4.3) by addressing issues such as scalability, parallelizability, and long-sequence performance. Its DNA-specific implementation, DNA-xLSTM, performed on par with or surpassed Transformers and the Mamba-based Caduceus on several genomic benchmark tasks (Schmidinger et al., 2024).

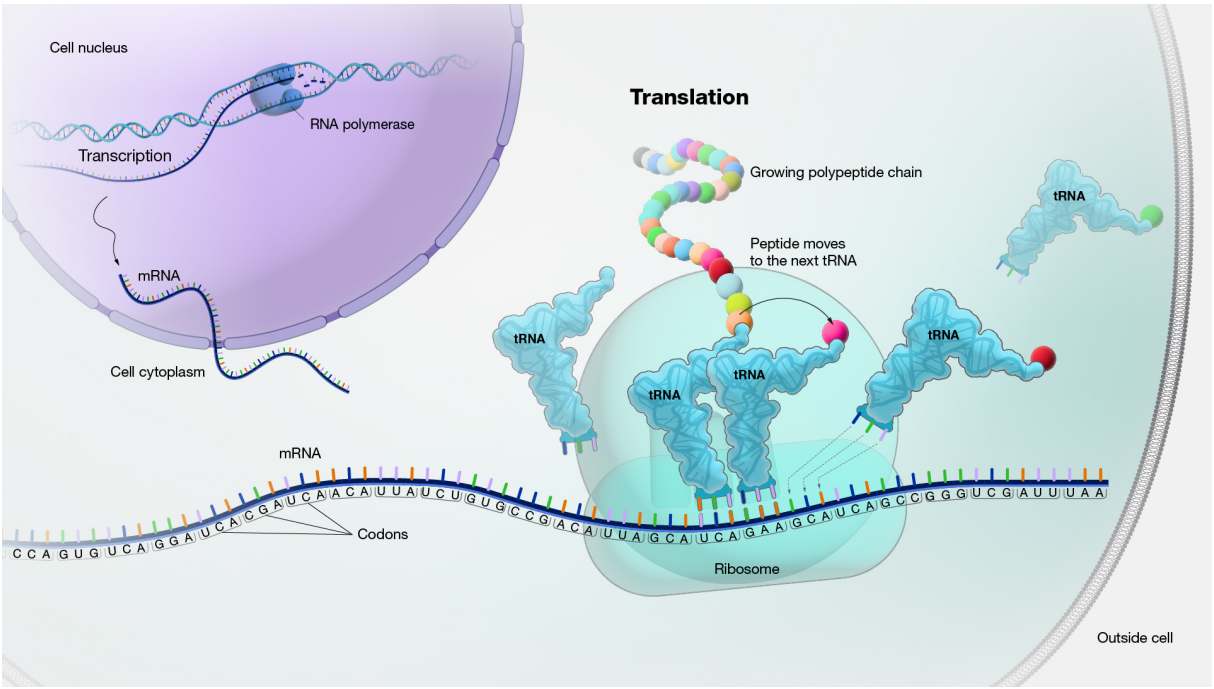


Figure 1: The central dogma of molecular biology. Illustration of National Human Genome Research Institute (2025) in public domain.[2]

3 Background

Understanding the relationship between genetic information and protein synthesis is fundamental to molecular biology and underpins many advances in computational biology. This chapter tries to provide a sufficient foundation to understand the key concepts in genomics, including gene expression and the regulatory mechanisms that influence protein abundance. For detailed information on molecular biology, the reader can refer to the works of Alberts et al. (2022). After a genomics overview, the chapter outlines foundational principles of Machine learning (ML) and Deep Learning (DL), with a focus on architectures and techniques relevant to biological sequence processing. Finally, the chapter discusses methods for encoding biological sequences and structures, which are essential for applying modern computational models to genomics data. Together, these sections establish the theoretical context for the research presented in subsequent chapters.

3.1 Genomics

All living organisms on earth are composed of cells, which store their hereditary information in the form of large, complex molecules known as DNA. This genetic code is organized into genes – specific segments of DNA that encode proteins essential for the cell's structure, function, and regulation. Gene expression (see Figure 1) occurs through two fundamental processes: transcription and translation (Alberts et al., 2022, pp. 1-5; Jia et al., 2024).

3.1.1 Transcription

The first step of gene expression, transcription, takes place in the cell nucleus of eukaryotic cells, where DNA is located. During transcription, a specific DNA segment is transcribed into mRNA by the

[2] License: https://www.genome.gov/about-nhgri/Policies-Guidance/Copyright; downloaded from: https://www.genome.gov/genetics-glossary/Translation

enzyme RNA polymerase. This mRNA serves as a template for protein synthesis and is composed of four nucleotides: adenine (A), uracil (U), cytosine (C), and guanine (G), each complementary to a corresponding DNA nucleotide. Note that mRNA uses uracil (U) instead of thymine (T), used in DNA.

After transcription, the mRNA undergoes several modifications before it leaves the nucleus to the cytoplasm. These modifications include the addition of a 5' cap, a poly-A tail, and the removal of non-coding sequences (introns) through a process called splicing. The remaining coding regions (exons) are then joined to form a continuous sequence.

The mature mRNA molecule comprises three main components: the 5' Untranslated Region (5' UTR), the Coding DNA sequence (CDS), and the 3' Untranslated Region (3' UTR) (Alberts et al., 2022, pp. 323-358).

3.1.2 Translation

The second step of gene expression, translation, occurs in the cytoplasm, where ribosomes read the mRNA sequence to synthesize the corresponding protein. Translation consists of three phases: initiation, elongation, and termination. During initiation, guided by initiation factors, the small ribosomal subunit binds to the mRNA and scans for the start codon ("AUG"; see Figure 2). Once the start codon is located, the large ribosomal subunit binds to the mRNA and elongation begins.

As the ribosomes move along the mRNA in the 5' to 3' direction, transfer RNA (tRNA) molecules deliver the appropriate amino acids, which are linked together to form a polypeptide chain. When a stop codon is encountered, release factors bind to the ribosome, terminating translation and releasing the completed polypeptide (Jia et al., 2024; Alberts et al., 2022, pp. 358-389).

3.1.3 Codons

The CDS of an mRNA encodes the amino acid sequence of the protein and is read in triplets of nucleotides, known as codons. Each codon corresponds to a specific amino acid or a stop signal during the translation process (see Figure 2). There are 64 codons in total, including the start codon, three termination codons, and codons encoding 20 distinct amino acids. Multiple codons can encode the same amino acid, known as redundancy or degeneracy of the genetic code. This adds flexibility to the coding sequence, as codon changes in the DNA sequence may not alter the resulting protein (Alberts et al., 2022, pp. 358-360) but can influence gene expression, which is known as codon usage bias (Hanson & Coller, 2018).

3.1.4 RNA Structure

RNA nucleotides can interact through base stacking, which contributes to mRNA folding, or by forming stable hydrogen bonds between complementary bases. These interactions are not limited to adjacent nucleotides in the sequence, allowing distant regions to pair and giving rise to secondary structures.

These interactions result in a variety of base-pairing geometries and characteristic secondary structure types (see Figure 3). Common examples include stems, formed by base-paired regions of double-stranded RNA, and loops, which consist of unpaired nucleotides. Other typical secondary structures include hairpin loops, bulges, internal loops, and multiloops. Beyond these two-dimensional configurations, RNA can fold into complex three-dimensional conformations, referred to as tertiary structures (Leontis & Westhof, 2001, pp. 1 f., Danaee et al., 2018).

[3] Accessed from https://commons.wikimedia.org/wiki/File:Aminoacids_table.svg

[4] License: https://creativecommons.org/licenses/by/4.0/; downloaded from: https://academic.oup.com/view-large/figure/466870504/NAR_46_11_5381_f2.jpeg

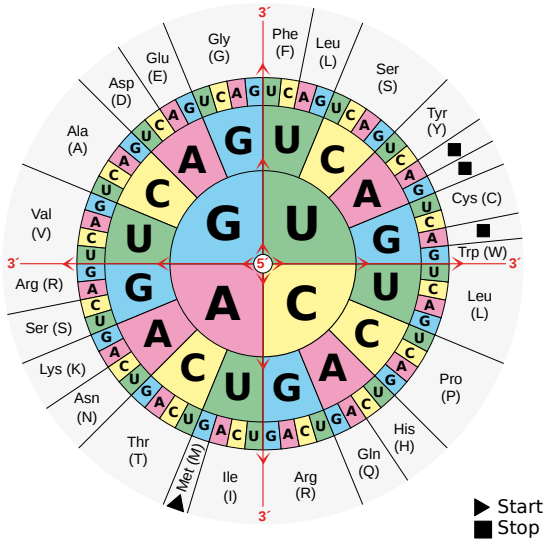


Figure 2: Correspondence between codons and amino acids, read from 5' UTR (center) to 3'UTR. Image in public domain.[3]

3.1.5 Regulatory Mechanisms of Gene Expression

The level of protein produced in a cell, referred to as protein abundance, is strongly correlated with mRNA abundance, the amount of mRNA present at a given time. Prior studies estimate that around 66-85% of protein expression can be explained by mRNA levels, which are governed by transcriptional regulation and mRNA degradation rates (Zrimec et al., 2021, pp. 18-20). In other words, the amount of mRNA transcribed from a gene is a key determinant of how much protein is ultimately synthesized. To focus solely on mRNA sequence related factors, translational efficiency quantifies how many protein molecules are produced per mRNA molecule and can be approximated by the PTR ratio.

Substantial research has focused on identifying factors that influence mRNA translation (see Section 2). However, the completeness of the catalog of such features as well as their quantitative contribution to the PTR ratio remains a subject of debate (Zrimec et al., 2021, pp. 18-20). Some of the most relevant and widely discussed factors include:

Codon Usage Bias. Due to the redundancy of the genetic code, multiple codons can encode the same amino acid. The preferential use of certain synonymous codons over others, known as codon usage bias, seems to vary across tissues and can impact both protein folding and translational efficiency. Translation depends on the availability of codon matching tRNAs in the cell. Rarely used codons tend to correspond to less abundant tRNAs, while frequently used codons match more abundant tRNAs (Gould et al., 2014; Hernandez-Alias et al., 2023). However, the extent to which codon usage bias affects protein synthesis in human cells remains debated (Hanson & Coller, 2018, pp. 21 f., Eraslan et al., 2019, pp. 1 f., Guimaraes et al., 2014).

Additionally, the context of neighboring codons, especially those neighboring the start and stop codons, can influence translational elongation dynamics (Gould et al., 2014, p. 2; Eraslan et al., 2019, pp. 1 f.).

GC Content. Sequences with a high proportion of codons ending in guanine (G) or cytosine (C) tend to exhibit elevated mRNA and protein abundance. Notably, codon preferences seem to vary between human tissues. In some tissues, genes rich in adenine- or thymine-ending codons may exhibit higher expression levels on average (Hernandez-Alias et al., 2023; Kudla et al., 2006).

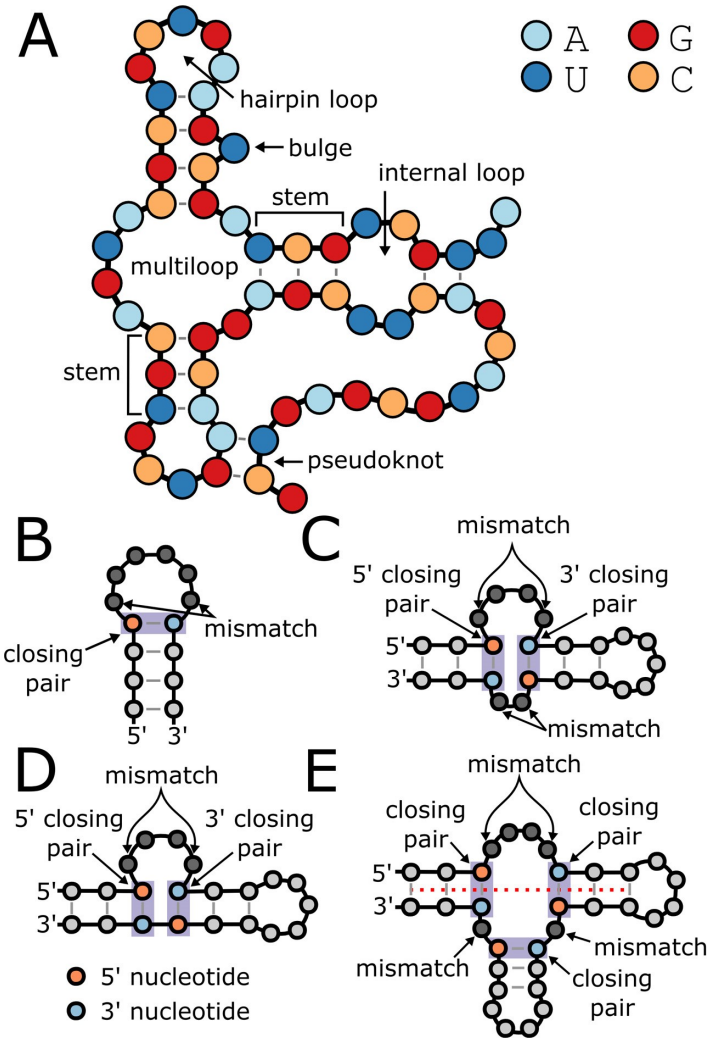


Figure 3: Secondary structure types identified by the bpRNA algorithm (Danaee et al., 2018). A) Schematic overview of RNA structure types. B) Hairpin. C) Internal loop. D) Bulge. E) Multiloop. Illustration published under CC BY 4.0 license.[4]

mRNA Secondary Structure. The secondary structure of mRNA can affect ribosomal transit speed and influence the stability and lifespan of the transcript. Both of these factors contribute to the overall translational efficiency of the mRNA molecule. In particular the secondary structure in the 5' UTR could be relevant (Gould et al., 2014, pp. 2 f., Eraslan et al., 2019; Zrimec et al., 2021; Li et al., 2019; Vogel et al., 2010).

Kozak Sequence. The Kozak sequence is a specific nucleotide sequence that plays a crucial role in the initiation of translation. It is typically located in the 5' UTR, just upstream of the start codon (AUG). The presence of a Kozak sequence can enhance the efficiency of translation initiation by facilitating ribosome binding and positioning over the start codon (Zrimec et al., 2021, pp. 14 f., Eraslan et al., 2019).

Additional Factors. Regulatory molecules such as microRNAs (miRNAs) and RNA-binding proteins (RBPs) can interact with mRNA to modulate translation (Eraslan et al., 2019, p. 2). Specific sequence motifs, which are characteristic subsequences of mRNA, can influence protein output through single strand or structure-recognition mechanisms (Zrimec et al., 2021, pp. 1 f.). Furthermore, post-transcriptional regulation such as protein degradation also impacts the PTR ratio by altering the intracellular concentration of proteins (Eraslan et al., 2019, pp. 1 f., Franks et al., 2017). Also the length of the mRNA sequence, in particular the coding region, upstream open reading frames (uORFs), and ORF length can impact translational efficiency (Zrimec et al., 2021).

3.2 Foundations of Machine Learning

3.2.1 Supervised Learning

In many real-world scenarios, we aim to predict a target variable y, which is often difficult to obtain, based on a set of readily available features $\boldsymbol{x} = (x_1, x_2, ..., x_d)$. For example, while it is relatively easy to observe an mRNA sequence (features), defining and determining the corresponding protein outcome (target) is considerably more complex (Eraslan et al., 2019; D. Wang et al., 2019). This setting is commonly referred to as supervised learning, where each sample consists of a set of features and a corresponding target variable (James et al., 2023, pp. 17-19).

In such cases, we assume that the data generating process follows the form:

$$y = f(\boldsymbol{x}) + \epsilon \tag{1}$$

where the target variable y is determined by an unknown but fixed function f of the features $\boldsymbol{x}$, along with a random additive error term ϵ. This error is assumed to be independent of the features and to have zero mean. To estimate the target variable $\hat{y}$ based on observed data $\boldsymbol{x}$, we estimate a function $\hat{f}$, referred to as the model:

$$\hat{y} = \hat{f}(\boldsymbol{x}) \tag{2}$$

The accuracy of our prediction $\hat{y}$ depends on two factors: The reducible error introduced by approximating the true function $f(\boldsymbol{x})$ with the estimate $\hat{y}$. And the irreducible error introduced by ϵ, which can arise from unmeasurable variability (James et al., 2023, pp. 15-17).

This relationship can be expressed mathematically as:

$$\begin{aligned} \mathbb{E}[(y - \hat{y})^2] &= \mathbb{E}\big[\big(f(\boldsymbol{x}) + \epsilon - \hat{f}(\boldsymbol{x})\big)^2\big] \\ &= \underbrace{\big(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\big)^2}_{\text{Reducible}} + \underbrace{\mathrm{Var}(\epsilon)}_{\text{Irreducible}} \end{aligned} \tag{3}$$

The goal in Machine learning (ML) is to find a model $\hat{f}(\boldsymbol{x}, \boldsymbol{\theta})$, with parameters $\boldsymbol{\theta}$ that closely approximates the true but unknown data generating function f, ensuring that $y \approx \hat{f}(\boldsymbol{x}, \boldsymbol{\theta})$ for any given sample $(\boldsymbol{x}, y)$ (James et al., 2023, p. 30).

3.2.2 Classification

If the target variable y is not numerical but instead belongs to a set of categories k, the problem is referred to as a classification task. In this case, the learning algorithm must construct a function $\hat{f} : \mathbb{R}^d \to \{1, ..., k\}$ that either directly assigns a numeric label to each category or predicts a probability distribution over the possible classes (Goodfellow et al., 2016, p. 100). When there are only two possible categories, the problem is specifically known as binary classification.

3.2.3 Model Generalization & Evaluation

To assess the capabilities of a trained model to accurately predict targets for unseen observations, it is essential to evaluate its performance on a held-out test set. This test set should be drawn from the same data generating process as the training data, ensuring that samples are independently and identically distributed (i.i.d.). With such a setup the expected error on the test set provides an upper-bound of the expected error on the train set and hence a generalization error estimate (Goodfellow et al., 2016, 110 f.).

Similarly, when setting hyperparameters – parameters to configure a models learning process – an additional validation set of previously unseen samples should be used. This prevents bias in estimating the generalization loss and avoids overfitting to the training set. After tuning hyperparameters on the validation set, the final generalization error is estimated using the test set (Goodfellow et al., 2016, p. 121).

3.2.4 Over- and Underfitting

Continuous evaluation on a dedicated validation set allows for monitoring of the model's generalization error during the training process. In particular, tracking both training and validation error helps identify underfitting and overfitting:

A model that exhibits a high training error and fails to learn meaningful patterns from the data is underfitting (Goodfellow et al., 2016, 110 f.). This is illustrated by the yellow linear model in Figure 4, which is too simplistic to adequately capture the true data distribution.

Conversely, a model that is excessively tailored to the training set, capturing random noise rather than meaningful patterns, will have a very low training error but a high validation error. This phenomenon, known as overfitting, leads to poor generalization performance (James et al., 2023, 30 f.). The dark orange, dotted model in Figure 4 exemplifies this issue.

In the simplified setting of Figure 4, the balanced model with a polynomial degree of 4 achieves the best trade-off between model complexity and generalization. It effectively captures the underlying data distribution without being overly sensitive to irreducible noise caused by omitted variables and inherent randomness (James et al., 2023, pp. 17-19).

This balance is commonly referred to as the bias-variance trade-off, which involves managing the trade-off between two sources of error: High variance, where an overly complex model is too sensitive to small fluctuations in the training data, leading to poor generalization. High bias, where an overly simplistic model fails to capture the underlying patterns in the data, leading to systematic errors. An optimal model finds a middle ground, minimizing both bias and variance to achieve strong generalization performance (James et al., 2023, pp. 31-33).

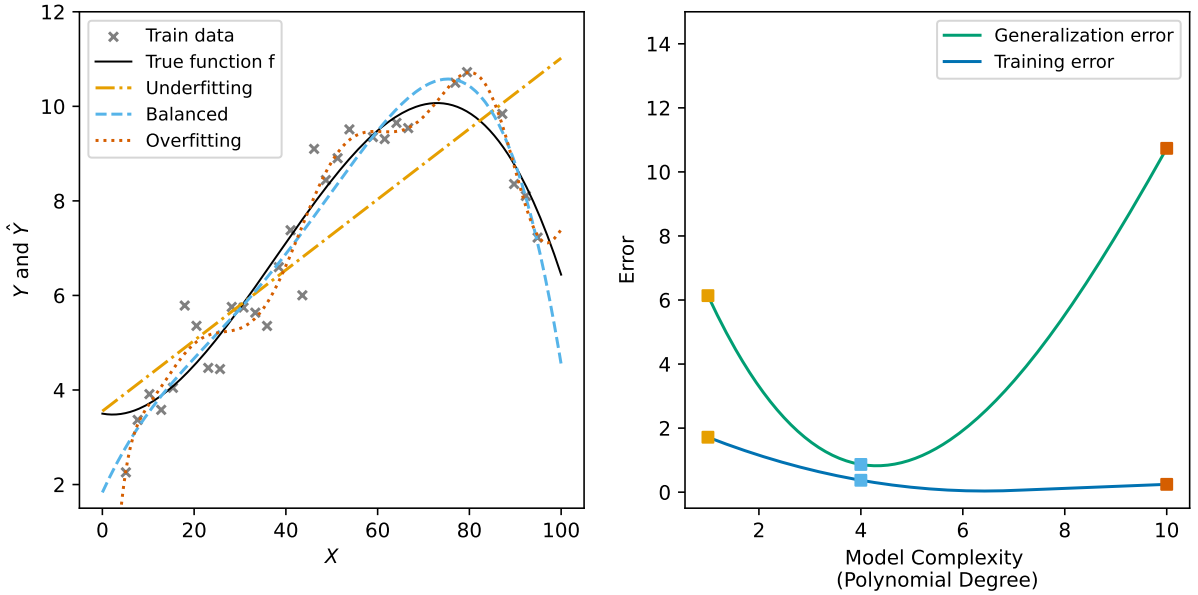


Figure 4: Under- and overfitting depending on model complexity. Inspired by James et al. (2023, p. 29).

3.2.5 Random Forest Classifier

The Random Forest is a well-established and widely utilized machine learning algorithm (Breiman, 2001). As an ensemble method built upon decision trees, it supports both regression and classification tasks. A decision tree is constructed by recursively partitioning the training data based on feature values, using an objective function – such as minimizing node impurity – to identify splits that best separate the classes, in classification scenarios. A Random Forest Classifier (RFC) generates multiple such trees, each trained on a random subset of the data and using a randomly selected subset of features at each split. Final predictions are obtained by aggregating the individual tree outputs: each tree provides a class prediction based on the majority class in its terminal node and the overall prediction is determined by majority voting across all trees. This ensemble strategy effectively mitigates the overfitting tendency of individual decision trees, reducing variance compared to a single tree and enhancing predictive performance (James et al., 2023, pp. 343-347).

3.3 Deep Learning

When estimating the data generating function f, as described in Section 3.2.1, plenty of supervised learning classification algorithms are available. These range from simple methods such as logistic regression, decision trees, random forests, and the Support Vector Machine (SVM) to more advanced approaches like Artificial Neural Networks (ANNs). The latter form the core of DL.

An ANN typically consists of multiple interconnected layers, each performing a specific transformation on the input data. The layers situated between the input and output layers of such a function $\hat{f}$ are referred to as hidden layers (Hwang et al., 2024; James et al., 2023). These layers apply a series of linear transformations followed by non-linear activation functions to learn complex representations of the data. In simplified terms, the universal approximation theorem states that an ANN with enough hidden units and a non-linear activation function can approximate any continuous function (Goodfellow et al., 2016, pp. 198 f., Hornik et al., 1989). While this holds theoretically, basic neural networks may require an impractically large number of parameters to solve complex problems or may fail altogether. More sophisticated architectures (see Section 3.4) help mitigate these challenges and have proven effective in various domains, including mRNA applications (see Section 2), provided the training dataset is sufficiently large (Hwang et al., 2024; James et al., 2023).

Training a deep learning model typically follows a standard procedure: preparing a clean dataset, defining the model, specifying a differentiable loss function and selecting an optimization strategy (Goodfellow et al., 2016, 153 f.).

3.3.1 Cross Entropy Loss Function

To train classification models, the Cross-Entropy (CE) loss is a commonly used objective function. For a single data point $\boldsymbol{x}$ with ground-truth label $\boldsymbol{y}$ and a classifier $\hat{f}$ that outputs predicted class probabilities $\hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{x}; \boldsymbol{\theta})$, the loss is defined as

$$\mathcal{L}_{\text{CE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{j=1}^{k} y^{(j)} \log \hat{y}^{(j)}, \quad \text{where } \hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{x}; \boldsymbol{\theta}) \tag{4}$$

Given k classes, $\boldsymbol{y} = (y^{(1)}, \ldots, y^{(k)})$ is the one-hot encoded ground-truth vector, where $y^{(j)} = 1$ if class j is the true class and 0 otherwise. The vector $\hat{\boldsymbol{y}} = (\hat{y}^{(1)}, \ldots, \hat{y}^{(k)})$ denotes the predicted class probabilities. The loss decreases as the predicted probability of the true class approaches 1, indicating increasing model confidence in its correct prediction (Goodfellow et al., 2016, pp. 153–155; Leskovec et al., 2019, pp. 537 f.).

3.3.2 Optimization

To minimize the loss, an optimization strategy is required to adjust the model parameters $\boldsymbol{\theta}$. In DL, large datasets are often necessary to achieve strong generalization capabilities. However, directly optimizing the loss function over an entire dataset is typically computationally infeasible. Instead, the negative gradient – indicating the direction in which $\boldsymbol{\theta}$ should be updated to reduce the loss the most – can be estimated using a small set of m randomly drawn samples (Goodfellow et al., 2016, pp. 151-153). The parameter m is commonly referred to as the batch size. The gradient estimate for a batch of samples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$ is given by:

$$\boldsymbol{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{CE}}\big(\boldsymbol{y}_i, \hat{f}(\boldsymbol{x}_i; \boldsymbol{\theta})\big) \tag{5}$$

To efficiently compute the gradients, the backpropagation algorithm is used (Rumelhart et al., 1986). Neural networks are nested functions of the form:

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = f^{(L)}\big(f^{(L-1)}(\ldots f^{(1)}(\boldsymbol{x}; \boldsymbol{\theta}^{(1)}) \ldots; \boldsymbol{\theta}^{(L-1)}); \boldsymbol{\theta}^{(L)}\big) \tag{6}$$

where L denotes the number of layers and $\boldsymbol{\theta}^{(l)}$ the parameters of layer l. By repeatedly applying the chain rule of calculus, backpropagation efficiently computes the gradient of the loss with respect to every parameter in the network. This is achieved by propagating the error backward through the network, layer by layer, starting from the output layer and moving towards the input layer. Thereby improving the model's predictions based on the training set or a random batch thereof (James et al., 2023, 428 f.). Once the gradient is computed, the parameters $\boldsymbol{\theta}$ are updated in the direction of steepest descent of the loss, scaled by the learning rate η:

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \eta \boldsymbol{g} \tag{7}$$

This iterative update method, estimating the gradient based on a small subset of the training set, is commonly known as mini-batch gradient descent (Goodfellow et al., 2016, pp. 151-153). Once every sample in the training set has been used in a gradient update step, an epoch is completed. Training typically continues for multiple epochs until the model converges to a local minimum of the loss function or overfitting appears.

Since neural networks involve non-convex optimization, the loss landscape typically contains numerous local minima rather than a single global minimum (Goodfellow et al., 2016, pp. 283-285). If the learning rate η is too large, convergence may be prevented because of oscillations in the parameters, while a too small value may slow down convergence or lead to suboptimal solutions (Leskovec et al., 2019, 545 f.). Hence, the learning rate is another hyperparameter to be carefully specified.

To prevent the optimization from becoming trapped in poor local minima and to accelerate convergence, advanced optimization techniques are often employed. For example, Kingma and Ba (2015) introduced one of the most widely used optimizers, called Adam, which incorporates momentum-based parameter updates to enhance parameter adjustment. Additionally, adaptive learning rates with scheduled warm restarts (Loshchilov & Hutter, 2017) and regularization via weight decay to promote model generalization (Loshchilov & Hutter, 2019) are commonly applied.

3.3.3 Pretraining with Denoising Autoencoders

In certain DL settings, it can be beneficial to pretrain a model on a larger unlabeled dataset before fine-tuning it on a smaller labeled one. This strategy is part of unsupervised learning, as only the observations $\boldsymbol{x}$ are provided, without associated targets y. Unsupervised pretraining can enhance model performance by enabling the learning of meaningful representations from unlabeled data, which can then be transferred to a supervised task through fine-tuning on labeled data (Goodfellow et al., 2016, pp. 526-528). In particular, this technique has been shown to improve model initialization and reduce the likelihood of convergence to suboptimal local minima (Vincent et al., 2008).

To pretrain DL models for mRNA data, autoencoders can be employed. An autoencoder consists of an encoder that compresses the input sequence into a lower-dimensional latent representation and a decoder that reconstructs the original input from this representation. A denoising autoencoder builds upon this concept by corrupting the input, typically by removing parts of the input vector, to encourage the learning of more robust latent representations. The model is then trained to reconstruct the original, uncorrupted input from the corrupted version (Vincent et al., 2008). This approach has been successfully applied to biological sequences, such as single guide RNA (sgRNA), to learn latent representations that capture structural and functional relationships within RNA sequences (Chuai et al., 2018).

To introduce noise into mRNA sequences, the method proposed by N. Wang et al. (2024) can be adopted. They employ a sophisticated masking strategy that randomly selects from three masking algorithms: masking random individual bases, masking random contiguous subsequences, or masking predefined motifs – subsequences known to be relevant for gene expression. This strategy enables the model to learn biologically meaningful representations from mRNA sequences, particularly by emphasizing the importance of motifs.

3.3.4 Evaluation: Area Under the Curve

In a binary classification setting, a model typically predicts the probability that a given sample belongs to the positive class. Given a threshold τ, the model classifies a sample as positive if the predicted probability exceeds τ. The resulting predictions fall into one of four categories: true positives (TP), true negatives (TN), false positives (FP), or false negatives (FN). Based on these counts, various performance metrics can be computed. For instance, the true positive rate (TPR):

$$\mathrm{TPR}_\tau = \frac{\mathrm{TP}_\tau}{\mathrm{TP}_\tau + \mathrm{FN}_\tau} \tag{8}$$

measures the proportion of correctly classified positive samples, whereas the false positive rate (FPR):

$$\mathrm{FPR}_\tau = \frac{\mathrm{FP}_\tau}{\mathrm{FP}_\tau + \mathrm{TN}_\tau} \tag{9}$$

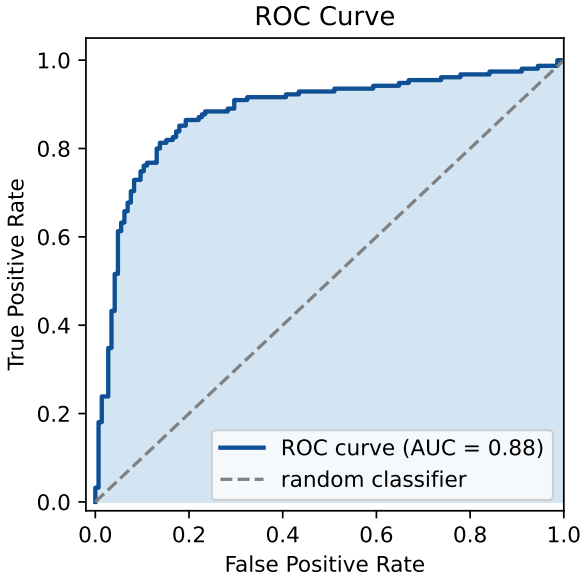


Figure 5: Illustration of the Receiver Operating Characteristic (ROC) curve (dark blue), along with its associated Area Under the Curve (AUC), represented by the light blue shaded region.

quantifies the proportion of negative samples incorrectly classified as positive (Kevin P. Murphy, 2022, pp. 171-173).

The Area Under the Curve (AUC) is a widely used metric that measures the area under the Receiver Operating Characteristic (ROC) curve, which displays the trade-off between the true positive rate and the false positive rate across different classification thresholds τ. The AUC ranges from 0.5 for a random classifier to 1 for a perfect classifier. Ideally, the ROC curve should be as close as possible to the top-left corner of Figure 5, indicating a high true positive rate and a low false positive rate across all thresholds. The AUC is a useful metric for evaluating the performance of a binary classifier, as it is threshold-independent and provides a single scalar value to compare different models (James et al., 2023, pp. 154-156).

3.4 Deep Learning Architectures

Learning complex patterns becomes increasingly more difficult with increasing dimensionality of the data due to the curse of dimensionality. The development of DL architectures was motivated by the need to address this challenge, where each architecture comes with distinct advantages and limitations (Goodfellow et al., 2016, pp. 155-157). In the context of mRNA data, the following architectures are particularly relevant and will be assessed in this work.

3.4.1 Dense Feedforward Neural Networks

As introduced in Section 3.3, feedforward neural networks (see Figure 6) form the foundation of DL and often serve as building blocks for more complex architectures. An architecture consisting of multiple dense layers, where each neuron is linked to all neurons in the subsequent layer, is called Multilayer Perceptron (MLP). The output of a neuron is computed by applying a non-linear activation function ϕ to a weighted sum of inputs, either the features $\boldsymbol{x}$ or the output from the preceding layer f^{l-1}:

$$f^{(l)} = \phi(\boldsymbol{W}^{(l)} f^{(l-1)} + \boldsymbol{b}^{(l)}) \tag{10}$$

where $\boldsymbol{W}^{(l)}$ is the learnable weight matrix, $\boldsymbol{b}^{(l)}$ the learnable bias vector and ϕ the activation function. The final network output is obtained as a composition of all L layers:

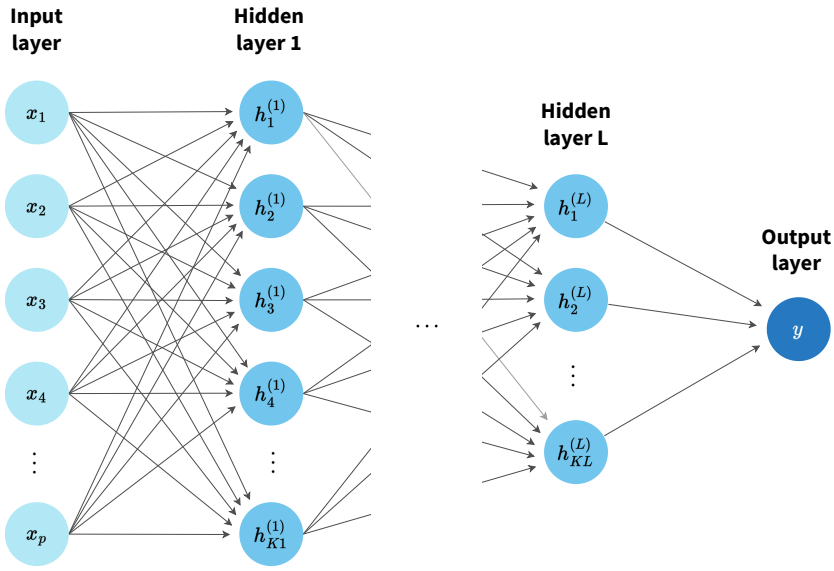


Figure 6: Dense feedforward neural network. Data flows from the input nodes in the left through all nodes of the network to generate a final output. Bias terms and activations are left out for simplicity. Inspired by James et al. (2023, p. 404).

$$\hat{y} = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}(\boldsymbol{x}) \tag{11}$$

The network depth is determined by the number of hidden layers, while width refers to the neurons per layer (Goodfellow et al., 2016, pp. 168-170). Designing such networks involves carefully selecting hyperparameters, including layer count, neuron count, activation function, loss function, optimizer and learning rate (Leskovec et al., 2019, p. 529; Goodfellow et al., 2016). Given the numerous design choices involved – each of which can significantly affect model performance – constructing effective neural networks is often considered more of an art than a science (Leskovec et al., 2019, p. 555).

The activation function ϕ enables modeling non-linear relationships, essential for capturing complex patterns (Goodfellow et al., 2016, pp. 168-170). To ensure stable training and prevent vanishing or exploding gradients, activation functions must satisfy specific properties (Leskovec et al., 2019, p. 531). Common choices include Rectified Linear Unit (ReLU), sigmoid, hyperbolic tangent, and more advanced functions like Swish and GELU (Leskovec et al., 2019; Ramachandran et al., 2017). The ReLU function (Nair & Hinton, 2010) is defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{12}$$

3.4.2 Convolutional Neural Networks

For high-dimensional input data, feedforward neural networks require a large number of parameters to capture complex patterns. This can lead to overfitting and demands extensive computational resources. Convolutional Neural Networks (CNN) (LeCun et al., 1989) address this issue by leveraging the spatial structure of the input data, making them particularly well-suited for image and sequence data. Its design makes CNNs parallelizable which enables training speedup. Also compared to dense networks, they reduce both computational costs and memory requirements as they need less parameters due to parameter sharing (Goodfellow et al., 2016, pp. 335-337).

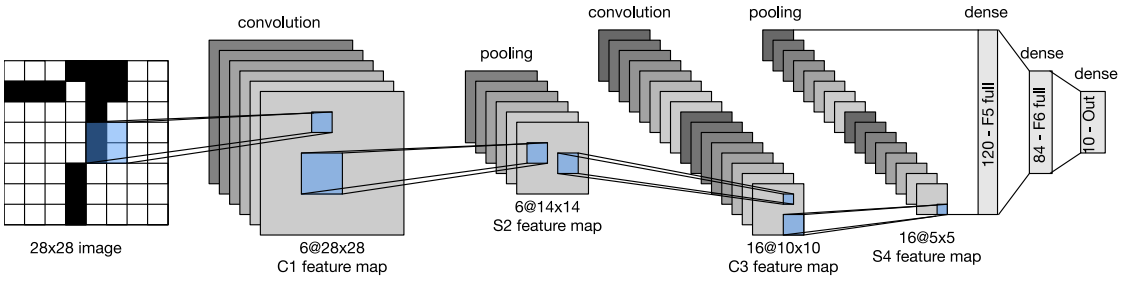


Figure 7: Simple CNN network with convolution and pooling layers as well as dense output network. Illustration of Zhang et al. (2023), published under CC BY 4.0 license[5].

As can be seen in Figure 7, a typical CNN consists of alternating convolutional and pooling layers, often followed by a fully connected layer (Leskovec et al., 2019, p. 553). The convolutional layer serves as the core of a CNN, where convolution kernels slide across input features, computing the sum of the elementwise product between the kernel and the features. Through iterative processing with multiple kernels and layers, the model learns to detect task-relevant patterns (Hwang et al., 2024, 1299 f.). The pooling layer then condenses the convolutional output into a smaller representation, reducing dimensionality while preserving essential features (James et al., 2023, p. 410). This process helps the model achieve approximate invariance to translations of the input data (Goodfellow et al., 2016, p. 342).

CNNs first extract abstract low-level features and patterns via convolution kernels, which are then combined to form higher-level representations. The presence or absence of these high-level features contributes to the final prediction (James et al., 2023, p. 407). In the context of mRNA sequences, convolution kernels can act as automatically learned motif detectors, identifying relevant local sequence motifs that influence gene expression (Hwang et al., 2024, 1299 f.). Since such motifs play a crucial role in gene expression, CNNs have become a widely used and effective architecture for predicting gene expression based on RNA and DNA data (Hwang et al., 2024, p. 1305; Rafi et al., 2024; Zrimec et al., 2021).

LegNet. LegNet (Penzar et al., 2023) is a sophisticated CNN-based architecture (see Figure 8) specifically designed for predicting gene expression from DNA sequences with fixed length of 150 bases. The model achieved first place in the DREAM challenge (Rafi et al., 2024), outperforming existing models at the time of publication. Inspired by EfficientNetV2 (Tan & Le, 2021), LegNet features a carefully designed structure comprising custom convolutional blocks and squeeze-and-excitation (SE) blocks with skip connections, which concatenate the block's output with the untransformed input features (Penzar et al., 2023).

RiboNN. The RiboNN network developed by Zheng et al. (2025) is another CNN-based architecture designed to predict the translational efficiency of full mRNA sequences, closely aligning with the objective of the present study. The model consists of ten stacked convolutional blocks, each comprising

[5]License: https://creativecommons.org/licenses/by/4.0/; downloaded from: https://github.com/d2l-ai/d2l-en/blob/master/img/lenet.svg
[6]License: https://creativecommons.org/licenses/by/4.0/; downloaded from supplementary data.

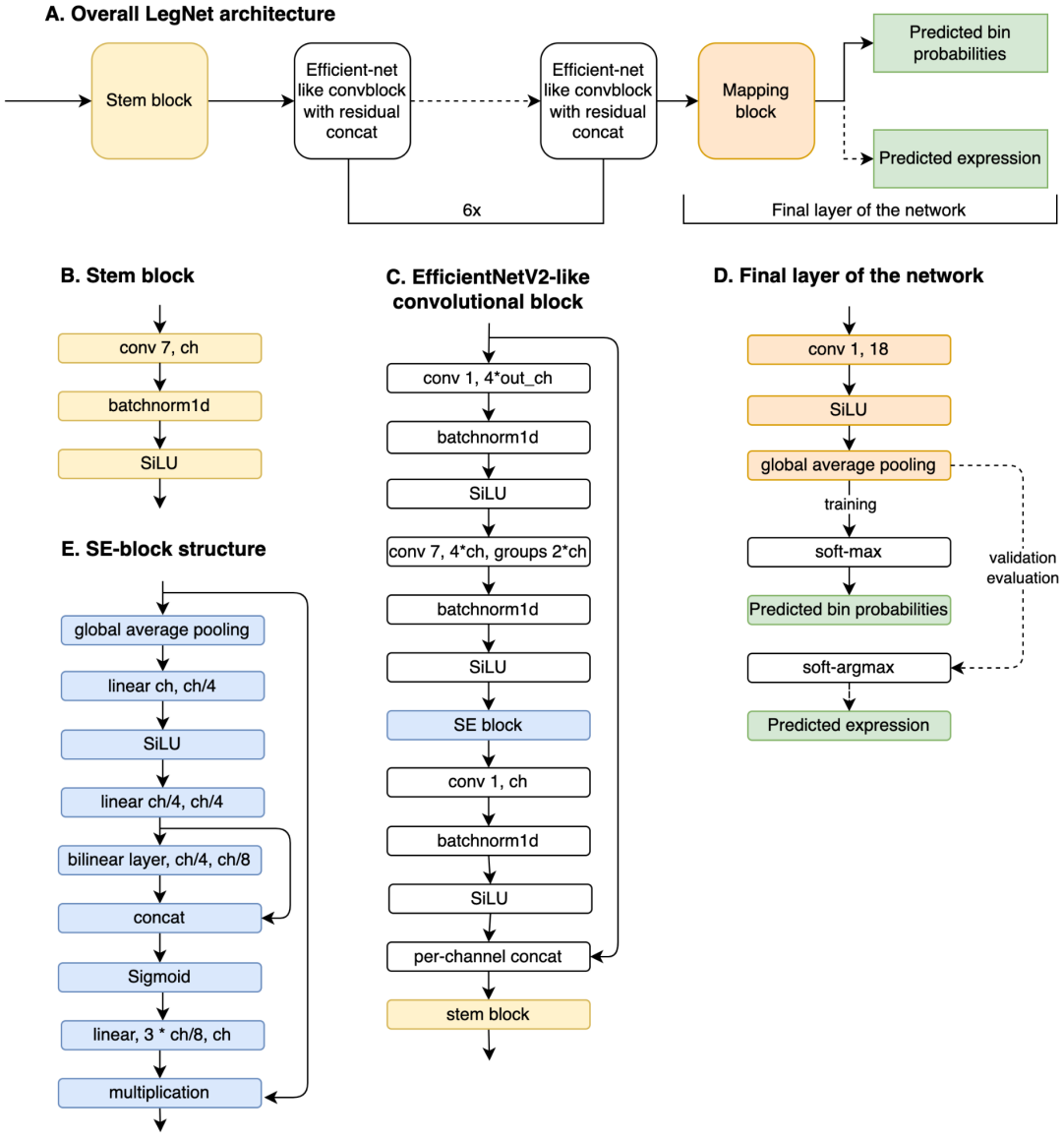


Figure 8: LegNet architecture. Illustration copied from supplementary data of Penzar et al. (2023), published under CC BY 4.0[6].

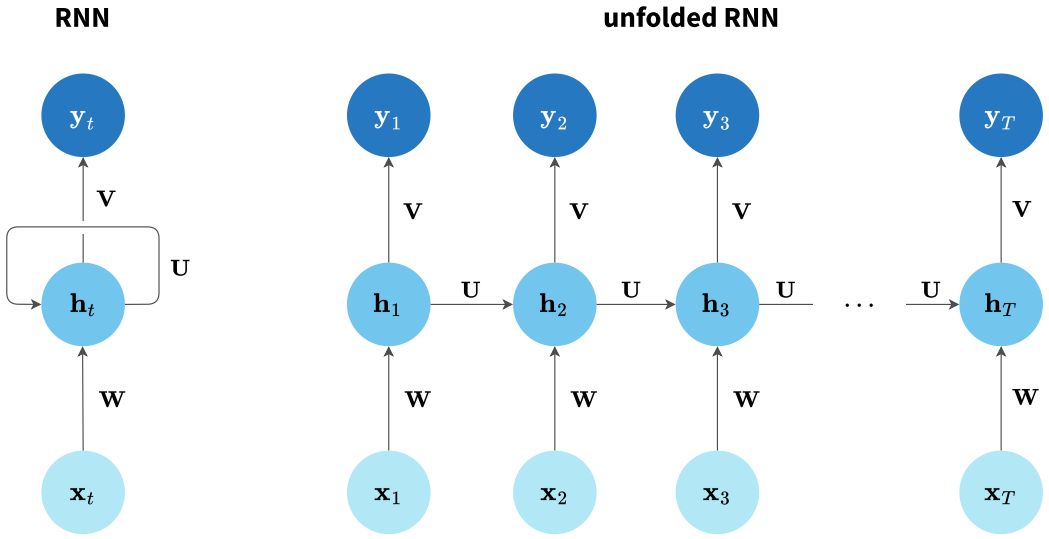


Figure 9: RNN, classic view (left) and unfolded (right). Inspired by James et al. (2023, p. 417).

a layer normalization, a ReLU activation, a one-dimensional convolution, a dropout layer and a max-pooling layer. Initially, RiboNN was pretrained in a multitask setting to predict translational efficiency across multiple cell types. It was subsequently fine-tuned to a specific cell type to improve its predictive performance.

3.4.3 Recurrent Neural Networks

While CNNs are specifically designed for spatial data such as images, Recurrent Neural Networks (RNN) are tailored for sequential data, including time series, text and biological sequences like RNA. RNNs process sequences recursively, where each recurrent unit receives some hidden state from the preceding step, processes it and passes it to the next unit while also generating an output. This recursive structure enables RNNs to capture dependencies between distant elements within a sequence (Hwang et al., 2024, p. 1300) and parameter sharing, as each unit applies the same operation with a fixed set of weights to produce its output (Goodfellow et al., 2016, p. 374). The basic RNN unit is defined as:

$$\boldsymbol{h}_t = \phi(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(h)}) \tag{13}$$

where $\boldsymbol{h}_{t-1}$ is the hidden state at step $t-1$, $\boldsymbol{W}$ and $\boldsymbol{U}$ the weight matrices, $\boldsymbol{b}^{(h)}$ the bias vector and ϕ the activation function (see Figure 9). The hidden state $\boldsymbol{h}_t$ serves as memory, retaining information from previous inputs. The unit's output is given by:

$$\boldsymbol{y}_t = \phi(\boldsymbol{V}\boldsymbol{h}_t + \boldsymbol{b}^{(y)}) \tag{14}$$

where $\boldsymbol{V}$ and $\boldsymbol{b}^{(y)}$ denote another weight matrix and bias vector, respectively. Depending on the application, either all unit outputs or only the last unit's output is used, often passed to a fully connected layer (Leskovec et al., 2019, pp. 557-559).

Long Short-Term Memory (LSTM). Vanilla RNNs struggle with learning long-term dependencies, often either overaccumulating or prematurely forgetting relevant information. They are also prone to vanishing or exploding gradient issues (Leskovec et al., 2019, pp. 561 f., Hwang et al., 2024). To mitigate these challenges, advanced RNN variants such as Long Short-Term Memory (LSTM) (Hochreiter &

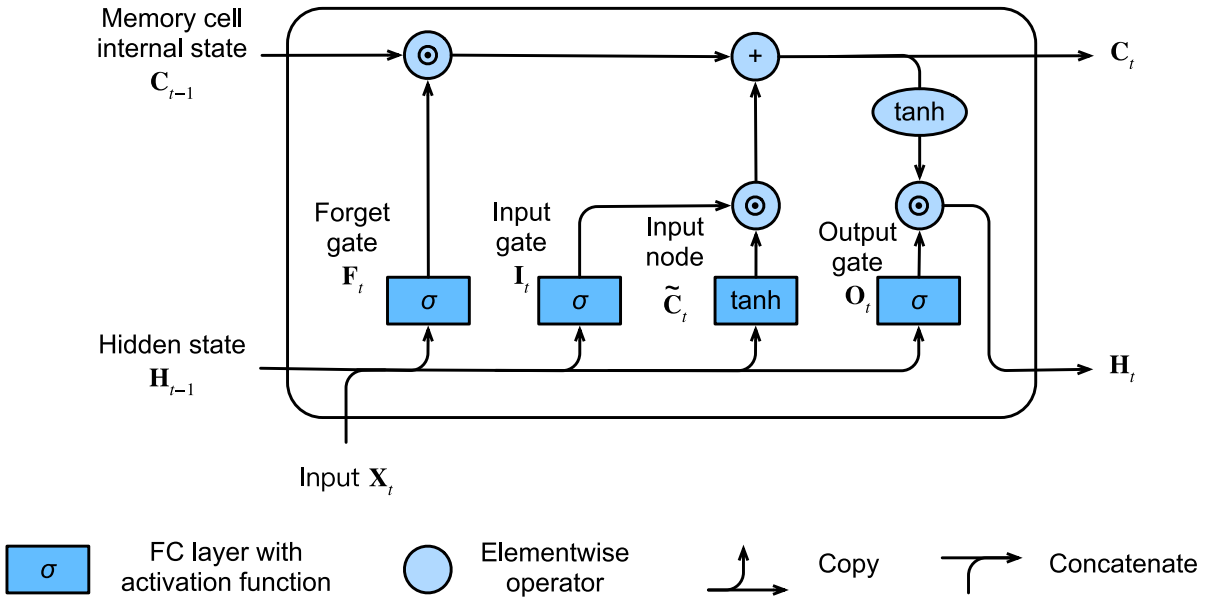


Figure 10: LSTM unit. Illustration of Zhang et al. (2023), published under CC BY 4.0 license[7].

Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) introduce gating mechanisms within the RNN unit that enable to selectively remember and forget information.

LSTM units comprise three gates (see Figure 10): the input gate, forget gate, and output gate. The input and forget gates are defined as:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{U}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}) \tag{15}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}) \tag{16}$$

Here, $\boldsymbol{W}$ and $\boldsymbol{U}$ are weight matrices of the input gate i and forget gate f (denoted by the superscripts), $\boldsymbol{b}$ represents bias vectors and σ denotes the sigmoid activation function, which outputs values between 0 and 1. The forget gate $\boldsymbol{f}_t$ controls the extent to which information is discarded from the long-term memory, while the input gate $\boldsymbol{i}_t$ determines how much new information from the current input $\boldsymbol{x}_t$ should be stored. The cell state $\boldsymbol{c}_t$ (i.e., the long-term memory) is updated as:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \tag{17}$$

where $\odot$ denotes element-wise multiplication and $\tilde{\boldsymbol{c}}_t$, the candidate cell state, is computed as:

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}^{(c)}\boldsymbol{x}_t + \boldsymbol{U}^{(c)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(c)}) \tag{18}$$

The output gate regulates the extent to which the updated cell state contributes to the output (i.e., the new hidden state $\boldsymbol{h}_t$) and is given by:

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{U}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}) \tag{19}$$

The resulting output, which serves as the next hidden state (short-term memory), is then:

$$\boldsymbol{y}_t = \boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{20}$$

For downstream tasks, either the last output $\boldsymbol{y}_T$ or the concatenation of all outputs across all steps may be passed to a fully connected layer (Leskovec et al., 2019, p. 559).

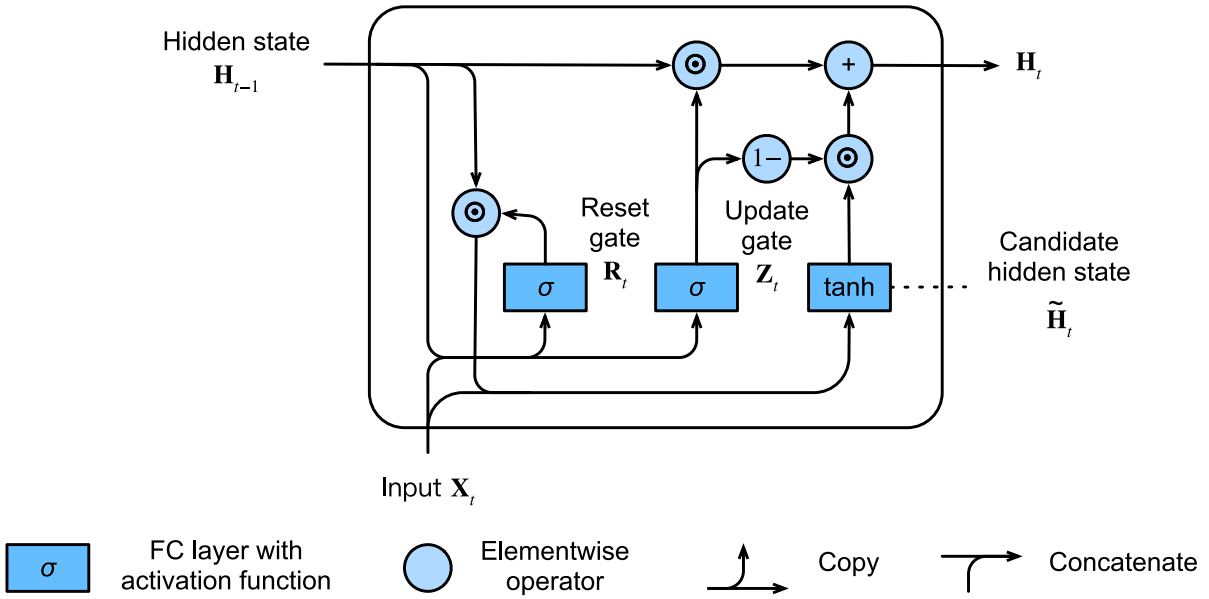


Figure 11: GRU unit. Illustration of Zhang et al. (2023), published under CC BY 4.0 license[8].

These architectural improvements enhance the model's ability to capture long-term dependencies and mitigate gradient instability, making LSTMs well-suited for processing long sequences and ensuring more stable training dynamics.

Gated Recurrent Unit (GRU). The Gated Recurrent Unit (GRU) (Cho et al., 2014) is a streamlined alternative to the LSTM that consolidates the forget and input gates into a single update gate (see Figure 11). It also merges the cell state and hidden state (long-term and short-term memory) into a unified representation, thereby reducing both the parameter count and computational overhead (Leskovec et al., 2019, p. 564; Cho et al., 2014).

xLSTM. A further development of the LSTM architecture is the extended LSTM (xLSTM) (Beck et al., 2024), which addresses several limitations of the standard LSTM. These include restricted revision capabilities of storage decisions due to the sigmoid activation function, limited memory capacity and poor parallelizability. Moreover, xLSTM aims to improve both computational efficiency and predictive accuracy. Unlike single LSTM units (see Section 3.4.3), xLSTM units are composed of stacked scalar LSTM (sLSTM) and matrix LSTM (mLSTM) modules. An xLSTM architecture comprises multiple such stacked xLSTM units (see Figure 12).

sLSTM introduces exponential gating in place of the sigmoid activation used in standard LSTMs. By applying exponential activation functions in the input and forget gates, the model gains the ability to make more pronounced updates to the long-term memory (i.e., cell state) when new information is presented – enabling both rapid forgetting and memory amplification. A dedicated normalization state is introduced to maintain balance between the input and forget gates with exponential gating.

mLSTM also adopts exponential gating, but additionally introduces a matrix-based cell state. Each memory unit in the cell state is now represented as a matrix, allowing the model to store and manipulate a richer set of features at each step. This structure facilitates the capture of more complex and

[7] License: https://creativecommons.org/licenses/by/4.0/; downloaded from: https://github.com/d2l-ai/d2l-en/blob/master/img/lstm-3.svg

[8] License: https://creativecommons.org/licenses/by/4.0/; downloaded from: https://github.com/d2l-ai/d2l-en/blob/master/img/gru-3.svg

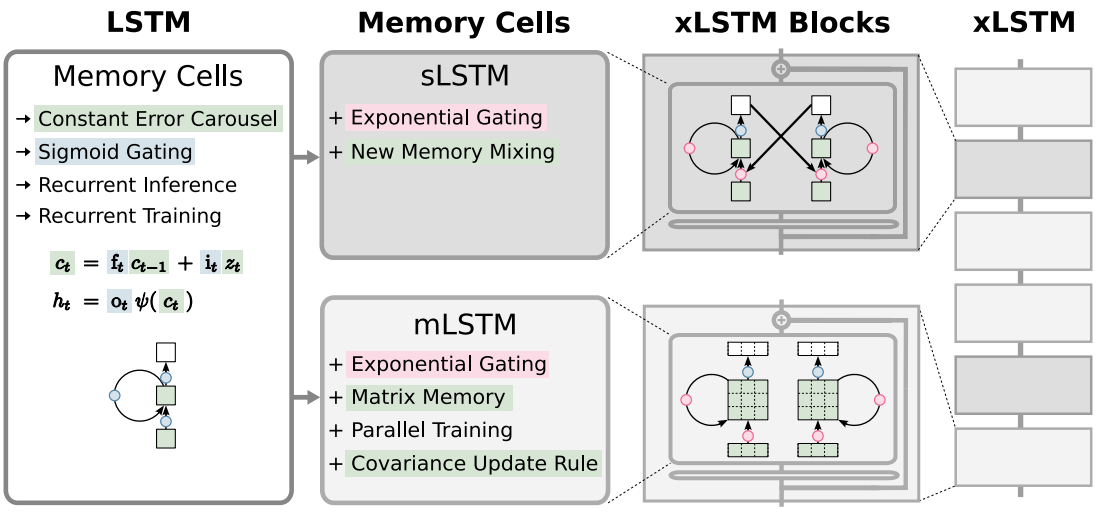


Figure 12: xLSTM architecture. Illustration of Beck et al. (2024), published under Apache 2.0 license[9].

longer-range dependencies. Crucially, mLSTM units are designed to operate on the full input sequence in parallel, enhancing computational throughput.

The xLSTM architecture combines these two components: parallelizable mLSTM units, which process the entire sequence simultaneously and sequential sLSTM units, which account for the inherently sequential structure of the data. According to the authors, this structure offers great performance and scalability, offering linear computational complexity and constant memory complexity with respect to sequence length (Beck et al., 2024; Ahmed, 2024). The usage of xLSTM for biological tasks, among others on DNA data, has been showcased by Schmidinger et al. (2024).

3.4.4 Transformers

The transformer architecture (Vaswani et al., 2017) represents a groundbreaking advancement in deep learning, particularly for sequence to sequence tasks as well as generative AI. Unlike traditional models such as CNNs and RNNs, which rely on convolutional or recurrent layers to process data sequentially, transformers leverage the self-attention mechanism to directly capture long-range dependencies within sequences. This mechanism simultaneously compares the relevance of every token with every other token in the sequence, assigning attention to relevant token pairs, allowing for a high degree of parallelizability (Vaswani et al., 2017; Choi & Lee, 2023, pp. 3 f., Hwang et al., 2024, pp. 1300 f.).

Self-attention. The self-attention mechanism is traditionally implemented as scaled dot-product attention (see right side of Figure 13). It computes a weighted sum of values $\boldsymbol{V}$, where the weights are determined by the similarity between a query $\boldsymbol{Q}$ and keys $\boldsymbol{K}$:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{21}$$

Here, $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are the queries, keys, and values, respectively. The term d_k is the dimensionality of the keys and queries and the softmax function ensures that the attention weights sum to 1 across each row of the output. The attention mechanism allows the model to focus on different parts of the input sequence when generating each output, enabling it to capture long-range dependencies effectively (Vaswani et al., 2017).

[9] License: https://www.apache.org/licenses/LICENSE-2.0; downloaded from https://github.com/NX-AI/xlstm/blob/main/res/desc_xlstm_overview.svg

Multi-head Attention. The multi-head attention mechanism (see middle part of Figure 13) extends the self-attention mechanism by allowing the model to jointly attend to information from different representation subspaces. This is achieved by applying the self-attention mechanism multiple times in parallel, each with its own set of learned weight matrices. The outputs of these attention heads are then concatenated and linearly transformed. Mathematically, this is expressed as:

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)\boldsymbol{W}^{(O)} \tag{22}$$

where each attention head is computed as:

$$\text{head}_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^{(Q)}, \boldsymbol{K}\boldsymbol{W}_i^{(K)}, \boldsymbol{V}\boldsymbol{W}_i^{(V)}) \tag{23}$$

Here, $\boldsymbol{W}_i^{(Q)}$, $\boldsymbol{W}_i^{(K)}$, and $\boldsymbol{W}_i^{(V)}$ are the learned projection matrices for the i-th head and $\boldsymbol{W}^{(O)}$ is the output projection matrix. The use of multiple attention heads allows the model to capture diverse patterns and relationships in the input data, e.g. by specializing on a specific pattern per head. Due to this design, the attention mechanism scales quadratically with the sequence length, making it computationally expensive for long sequences (Vaswani et al., 2017).

Positional Encoding. The Transformer architecture employs positional encoding to preserve the sequential structure of input data. Since the self-attention mechanism is permutation-invariant and thus unable to inherently capture token order, positional encodings are added to the input embeddings (Kevin P. Murphy, 2022, 528 f. Vaswani et al., 2017).

Architecture. The transformer architecture consists of an encoder and a decoder (see left side of Figure 13), both built using stacked layers of multi-head attention and feedforward neural networks. The encoder processes the input sequence, generating a sequence of hidden representations, while the decoder uses these representations along with the target sequence to generate the output. In contrast to the encoder, the decoder can only consider previous tokens in the target sequence during training, ensuring that predictions are made in an autoregressive manner (Vaswani et al., 2017; Islam et al., 2024, pp. 3-5).

Applications. Transformers have been successfully applied in various domains, including natural language processing, computer vision as well as audio, speech, and signal processing (Islam et al., 2024). In the context of biological data, transformers have shown promise in many tasks, among others in gene expression prediction and RNA secondary structure prediction (Choi & Lee, 2023; Hwang et al., 2024, pp. 1300 & 1309; Rafi et al., 2024).

3.4.5 Selective Structured State Space Models

While RNNs offer linear scaling during inference due to their sequential nature, they lack parallelizability and struggle with modeling long-range dependencies. Transformers, in contrast, are highly parallelizable and excel at capturing long-range dependencies, but incur a quadratic computational cost due to the self-attention mechanism. The Selective Structured State Space Model, Mamba (Gu & Dao, 2023), aims to unify the advantages of both architectures while mitigating their respective limitations.

State Space Models. The models discussed so far receive a discrete sequence $\boldsymbol{x}_i$ as input. In contrast, State Space Models (SSMs), see Figure 14, represent a continuous-time process $\boldsymbol{x}(t)$ via a latent state $\boldsymbol{h}(t)$, modeling system dynamics through equations that describe temporal evolution. SSMs can be viewed as a hybrid between RNNs and CNNs: they enable efficient convolution during training and linear-time recurrence during inference. The model is typically defined by the following equations:

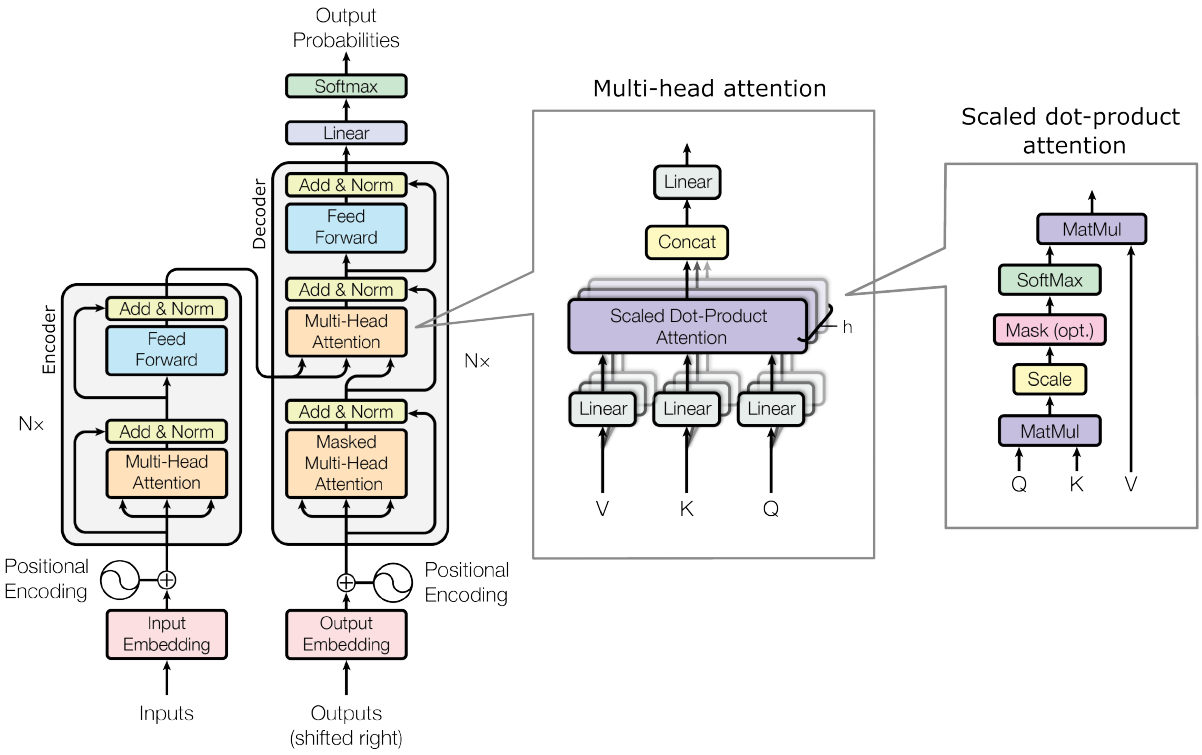


Figure 13: Original Transformer architecture based on Vaswani et al. (2017, 3 f.), usage allowed for scholarly works. Illustration inspired by Weng (2018).

$$\boldsymbol{h}'(t) = \boldsymbol{A}\boldsymbol{h}(t) + \boldsymbol{B}\boldsymbol{x}(t) \tag{24}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{h}(t) \tag{25}$$

Here, $\boldsymbol{h}'(t)$ is the state equation, describing how the hidden state $\boldsymbol{h}(t)$ evolves over time (via matrix $\boldsymbol{A}$) and how the input $\boldsymbol{x}(t)$ influences the system (via matrix $\boldsymbol{B}$). The output equation, $\boldsymbol{y}(t)$, transforms the latent state into the output through matrix $\boldsymbol{C}$. This continuous-time formulation models the system's evolution based on observed input data. In practice, these equations are discretized to accommodate the discrete nature of real-world data such as text or mRNA sequences (Gu & Dao, 2023; Grootendorst, 2025).

SSMs can be expressed in mathematically equivalent but computationally distinct forms with different advantages: the recurrent representation enables efficient inference due to constant-time scaling with sequence length, while the convolutional view is well-suited for training due to their parallelizable structure (Grootendorst, 2025).

Structured SSMs. Structured State Space Models (S4) are a subclass of SSMs that introduce structure into the state matrix to optimize computational efficiency. For instance, Mamba sets the matrix $\boldsymbol{A}$ to a diagonal form to simplify computations (Gu & Dao, 2023).

Mamba. Mamba extends S4s by integrating a selective attention mechanism, which enhances the model's ability to focus on relevant parts of the input sequence – much like self-attention in trans-

[10] License: https://www.apache.org/licenses/LICENSE-2.0; downloaded from https://github.com/state-spaces/mamba/blob/main/assets/selection.png

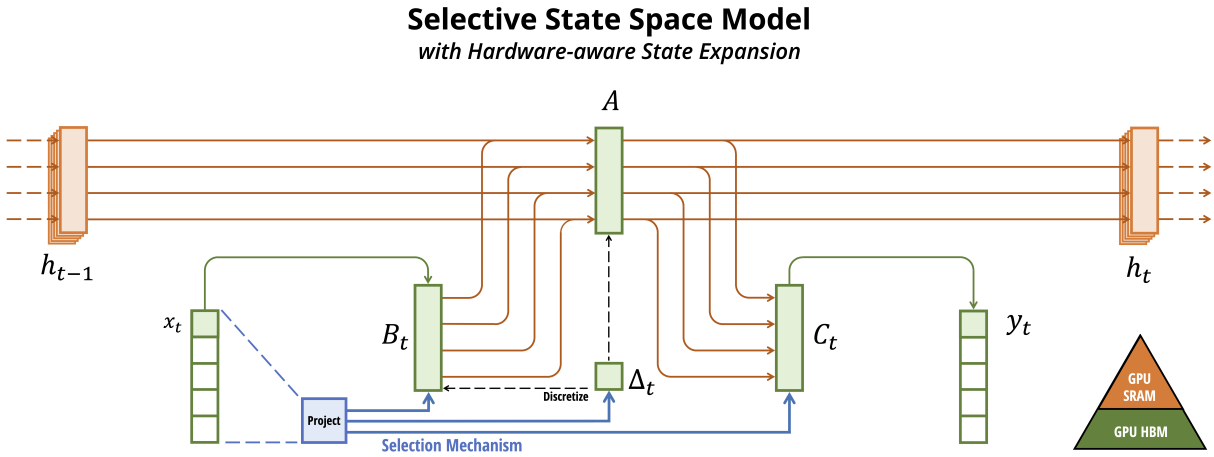


Figure 14: Selective State Space Model architecture in discrete time. Copied from Gu and Dao (2023), published under Apache-2.0 license[10].

formers – while maintaining the efficient scaling properties of SSMs. This design allows Mamba to retain long-range modeling capabilities without incurring the quadratic costs of attention.

As shown in Figure 15, a Mamba block begins with a linear projection that doubles the input dimensionality. This is followed by a convolutional layer, a selective S4 block and a gated linear projection that restores the original dimensionality. The SiLU (also called Swish) activation function is employed throughout. Several Mamba blocks can be stacked to form a deeper architecture (Gu & Dao, 2023).

Due to Mamba's combination of favorable scaling properties and its promised performance, it has been successfully applied to DNA sequence modeling (Gu & Dao, 2023; Schiff et al., 2024).

3.5 Encoding Biological Sequences

The described DL models require input data in the form of numerical, multidimensional arrays. However, mRNA sequences and secondary structure information are typically represented as character strings, which cannot be directly processed by these models. Consequently, it is essential to transform these sequences into suitable numerical representations that capture their generalizable properties. This encoding step is critical for the effective application of DL models, as it enables the extraction of meaningful patterns and relationships within the data. A variety of encoding methods exist, each with distinct advantages and limitations (Hwang et al., 2024, p. 1298).

3.5.1 Encoding RNA Sequences

One-hot Encoding. To date, one-hot encoding (see upper left of Figure 16) has been the most commonly used method for representing biological sequences (Hwang et al., 2024; Rafi et al., 2024). In this approach, each vector element corresponds to a distinct class – specifically, one of the four nucleotides (A, C, G, U) or a specific codon. The vector position representing the present nucleotide or codon is then set to 1, while all others are set to 0. This method allows the model to learn suitable feature representations of nucleotides during training (Hwang et al., 2024, p. 1298).

Embeddings. Biological sequences bear similarities to linguistic data in NLP, where codons or nucleotides can be treated analogously to words in a sentence. Accordingly, NLP techniques such as

[11] License: https://creativecommons.org/licenses/by/4.0/; downloaded from: https://www.nature.com/articles/s12276-024-01243-w/figures/1

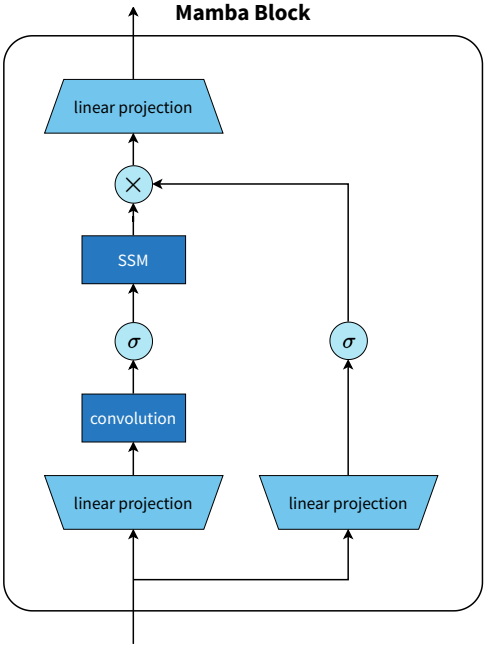


Figure 15: Mamba block architecture. Several Mamba blocks might be stacked on top of each other in a deep architecture. Inspired by Gu and Dao (2023, p. 8).

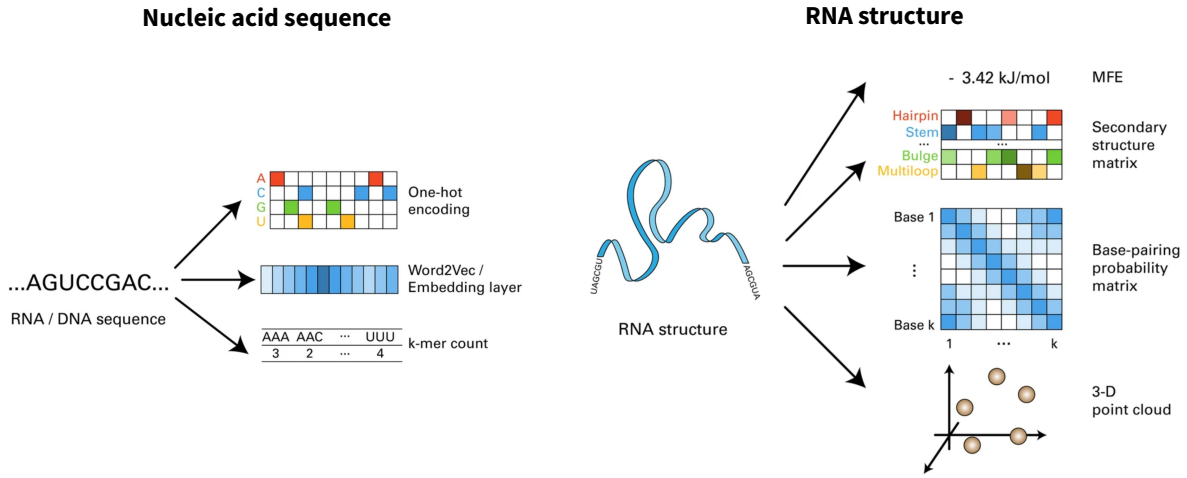


Figure 16: Different encodings for RNA sequences (left) and RNA structure (right). Illustration of Hwang et al. (2024, p. 1299), published under CC BY 4.0 license[11].

word2vec can be employed to generate dense numerical embeddings (middle left of Figure 16) (Mikolov, Chen, et al., 2013; Mikolov, Sutskever, et al., 2013). These embeddings may either be pretrained or learned in task-specific models via a dedicated embedding layer. This allows the model to capture task-relevant nucleotide or codon representations (Hwang et al., 2024, p. 1298). Prior studies suggest that embeddings generally outperform one-hot encodings in DL applications involving DNA and RNA sequences (Trabelsi et al., 2019).

k-mer Encoding. Alternatively, sequences can be encoded by counting the frequency of k-mers, subsequences of length k, within the sequence (bottom left of Figure 16). In the context of RNA, this could involve computing the frequency of codons or other short motifs (Hwang et al., 2024, p. 1298).

3.5.2 Encoding RNA Structure

RNA structure carries crucial information about the molecule's function, interactions and stability (Hwang et al., 2024, p. 1298) as well as influences translation (Gould et al., 2014).

A common representation is the dot-bracket notation, where unpaired nucleotides are denoted by dots and base pairs by opening or closing brackets (Huang et al., 2019). From this notation, more complex structural features such as stems, loops, and bulges can be inferred (see Section 3.1.4 and upper right of Figure 16) (Danaee et al., 2018). These categorical features can be encoded with the above-mentioned techniques, for example using one-hot vectors or embeddings (Hwang et al., 2024, p. 1298).

Additionally, the structure can be represented through distance matrices, encoding pairwise distances between nucleotides, or through binding probability matrices that reflect the likelihood of base-pair interactions (middle right of Figure 16). These matrix-based encodings capture spatial relationships and provide structural context to the model. Alternatively, full 3D structural representations – such as atomic or base-level coordinates – can be numerically encoded for even more detailed tertial structural modeling (Hwang et al., 2024, p. 1298).

4 Methodology

This chapter details the methodological approach used to address the research questions outlined in Section 1.2 which is structured in two main phases:

1. A systematic benchmarking of various DL architectures for predicting low- and high-PTR ratio outcomes from mRNA sequences across different tissues (Section 4.2).
2. An in-depth optimization of the most promising architecture, termed PTRnet, involving targeted adjustments to the input data, model design, hyperparameters, and training strategies to maximize predictive performance (Section 4.3).

4.1 Data

4.1.1 Data Sources

The data used in this thesis originates from two major data sources: the Human Protein Atlas (HPA) (Fagerberg et al., 2014) and the GENCODE project (Frankish et al., 2023).

Human Protein Atlas. The HPA provides gene and protein abundance data for 29 human tissues[12] (Fagerberg et al., 2014). The raw transcriptomics data was processed by Eraslan et al. (2019) to obtain PTR ratios per mRNA sequence for the available tissues and published by Hernandez-Alias et al. (2023).

GENCODE Project. The GENCODE project provides comprehensive gene annotations for coding and non-coding mRNA regions (Frankish et al., 2023). Processed data from this source was obtained from Wolfinger (2023). By integrating the processed gene annotation data from GENCODE with the processed transcriptomics data from the HPA, a combined dataset of PTR ratios across multiple tissues per unique mRNA sequence was constructed.

Bias. Tissue samples were sourced from Uppsala's Biobank (Fagerberg et al., 2014). However, donor information, including demographic diversity, is not publicly available.[13] This limitation likely introduces a bias toward (northern) European populations, reducing the dataset's ability to capture global genetic diversity. Consequently, models trained on this data may have limited generalizability (Sirugo et al., 2019; Hwang et al., 2024, pp. 1315 f.). Addressing such biases is crucial for ensuring the real-world applicability of these models.

4.1.2 Target Variable

This thesis focuses on a binary classification task based on the PTR ratio across tissues for individual mRNA sequences, rather than directly predicting the numeric PTR ratio. For this simplified setting, we adopt the classification scheme proposed by Hernandez-Alias et al. (2023), which determines whether a given gene-tissue pair exhibits exceptionally low or high protein expression.

Only genes with PTR ratio measurements available in at least three tissues are considered. High-PTR combinations are defined by two criteria: (1) the PTR ratio is at least twice the gene's average across tissues and (2) it represents the highest PTR value for that mRNA sequence. Conversely, low-PTR combinations satisfy: (1) the PTR ratio is less than half the average and (2) it is the lowest observed value for the sequence. These stringent thresholds exclude genes with relatively uniform PTR ratios,

[12] The included tissues are: Adrenal, Appendices, Brain, Colon, Duodenum, Uterus, Esophagus, Fallopian Tube, Fat, Gallbladder, Heart, Kidney, Liver, Lung, Lymph Node, Ovary, Pancreas, Placenta, Prostate, Rectum, Salivary Gland, Small Intestine, Smooth Muscle, Spleen, Stomach, Testis, Thyroid, Tonsil, and Urinary Bladder.

[13] See www.proteinatlas.org for some details on sample sources.

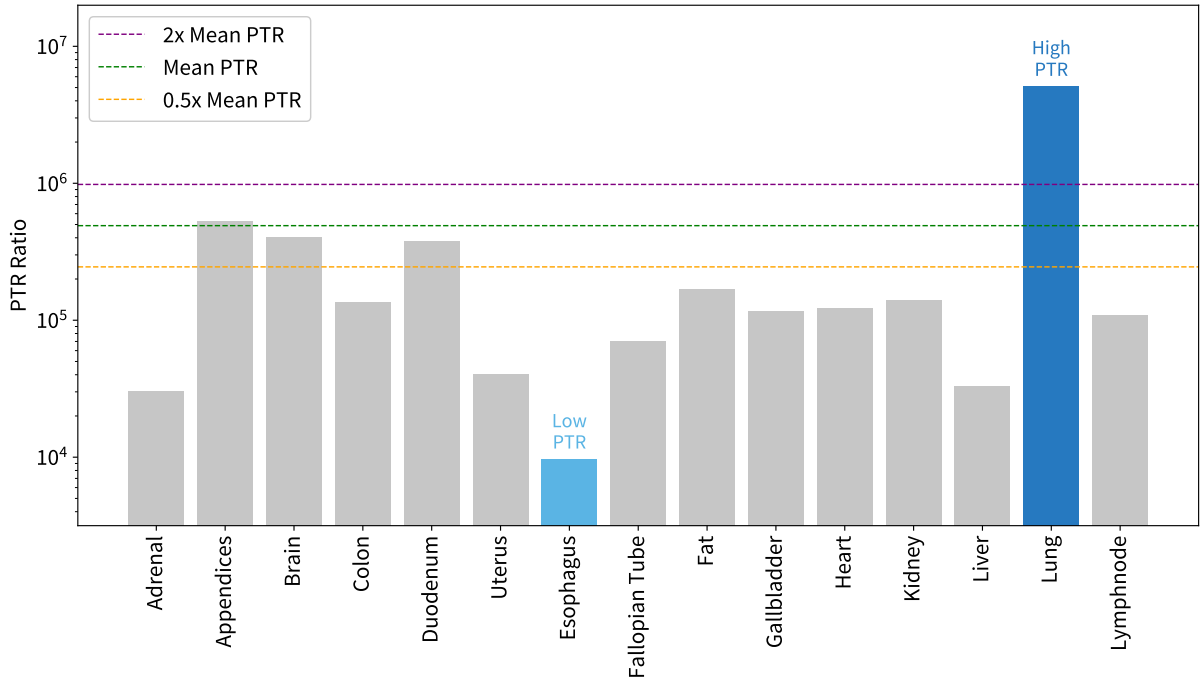


Figure 17: Classification procedure for one mRNA sequence's targets (mock up data with fewer tissues for illustration).

resulting in some genes appearing in the training data with only a low- or high-PTR label – or being excluded entirely. The classification procedure is illustrated in Figure 17.

Following this mapping, the dataset comprises approximately twice as many training samples as there are unique mRNA sequences (see Section 4.1.4 for details). The resulting class distribution is balanced between high- and low-PTR combinations, ensuring that the classification model is not biased towards either class. Nonetheless, class imbalance exists at the tissue level due to the specified classification procedure (see Appendix A).

4.1.3 Data Pre-Processing

As outlined in Section 4.1.1, we assemble datasets by integrating processed data from the HPA and GENCODE projects and extend them with secondary structure features. They comprise 11,279 unique mRNA sequences along with their corresponding PTR ratios for up to 29 tissues, with an average coverage of slightly above 20 tissues per sequence. We prepare two training datasets with different sequence encodings: one codon-encoded and one nucleotide-encoded (see Table 1).

Codon-Encoded Dataset. The codon-encoded dataset contains only the coding regions (CDS) of mRNAs, along with tissue identifiers and binary classification labels. Only sequences up to 8,100 nucleotides (2,700 codons) were retained, resulting in the exclusion of fewer than 2% of sequences (see the "dropped" column in Table 2). Since some mRNA sequences can exceed 100,000 nucleotides, this constraint enables the use of models with fixed maximum input lengths, reducing complexity and accelerating training (see Appendix B for sequence length distributions).

Nucleotide-Encoded Dataset. The nucleotide-encoded dataset enables a more detailed representation by incorporating structure-related features. Each entry includes the complete mRNA sequence at the nucleotide level, encompassing both the 5' and 3' untranslated regions (UTR), which play essential roles in the post-transcriptional regulation of protein abundance (see Section 3.1.5; Eraslan et al.

Table 1: Features of codon- and nucleotide-encoded dataset.

Dataset Encoding	Feature	Description	Max Sequence Length
Codon	Codon-encoded sequence	Coding region only (CDS)	8,100 nucleotides
	Tissue ID	Tissue sample identifier	
Nucleotide	Nucleotide-encoded sequence	Full mRNA sequence (5' UTR, CDS & 3' UTR)	9,000 nucleotides
	Coding area	encoding 5' UTR, 3' UTR ends, or nucleotide index in codon	
	Secondary structure	LinearFold-based secondary structure prediction (dot-bracket notation)	
	Loop Type	bpRNA-based loop type prediction	
	Tissue ID	Tissue sample identifier	

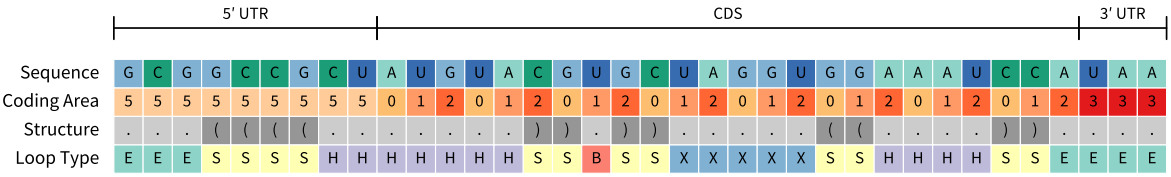


Figure 18: Demonstration of nucleotide-level data structure with secondary structure and loop type predictions. Illustration inspired by He et al. (2023, p. 6).

(2019) and Zrimec et al. (2021)). These regions increase the overall sequence length compared to the codon-encoded dataset, which contains only the CDS. To ensure the inclusion of as many sequences from the codon-encoded dataset as possible – despite the longer lengths – a higher maximum sequence length was required. Accordingly, a threshold of 9,000 nucleotides was imposed, balancing the goal of maximizing dataset overlap with the need to exclude excessively long sequences. This limit also helps maintain manageable computational demands and reasonable training times.

We also incorporate structural features (see Figure 18), given their relevance for predicting the PTR ratio (see Section 3.1.5). These features include base pairing, represented in dot-bracket notation and predicted by "LinearFold" (Huang et al., 2019), as well as loop types identified by "bpRNA" (Danaee et al., 2018). Additionally, an index is included to indicate whether a given nucleotide resides in the 5' or 3' UTR, or – if within the CDS – its position within the codon. As in the codon-encoded dataset, tissue identifiers and classification labels are also provided.

Conventional RNA secondary structure prediction tools typically scale with $O(n^3)$, making them unusable for long sequences of several thousand bases. In contrast, "LinearFold" operates in linear time, enabling rapid predictions, even for the longest sequences in the dataset, within one minute, while achieving accuracy comparable to state-of-the-art tools such as "mfold" and "RNAfold" in secondary structure prediction (Binet et al., 2023).

4.1.4 Train, Validation, and Test Split

As discussed in Section 3.2, separate validation and test sets comprising unseen samples from the same distribution as the training set are essential for unbiased estimation of the generalization error, which is necessary for hyperparameter tuning and final performance evaluation (Goodfellow et al., 2016). To preserve the original data distribution and composition across splits, stratified sampling is applied based on the length of the mRNA coding region. This strategy minimizes sampling bias, particularly for the relatively rare long sequences (see sequence length distributions in Appendix B). The validation and test sets are each assigned 15% of the total data, balancing representativeness with sufficient training data.

Stratified Sampling. Each mRNA sequence is usually associated with a low- and high-PTR label for different tissues[14], resulting in several gene-tissue pairs per sequence. Preliminary testing showed that models tend to overfit to mRNA sequences seen during training, as variation across tissues is generally smaller than across sequences (Hernandez-Alias et al., 2023, p. 9). To mitigate this, the splitting strategy ensures that no mRNA sequence appears in more than one dataset split.

To achieve this, sequences are first grouped into ten bins based on their coding region lengths. Within each bin, sequences are randomly selected one at a time and all associated tissue-target pairs are allocated to the training set until its target size (70%) is reached. The same procedure is then applied to form the validation set, with remaining sequences assigned to the test set. Due to this sequential sampling approach, the validation and test set proportions (15% each) may only be approximately met. However, the overall dataset size allows these targets are closely achieved in practice (see Table 2). This method maintains mutual exclusivity of sequences across splits and preserves the training distribution, including representation of rare long sequences, through stratified selection.

Evaluation Comparability across Datasets. Due to differences in the included regions (CDS only vs. full transcript with UTRs) and sequence length constraints, the codon- and nucleotide-encoded datasets contain slightly different sets of mRNA sequences.

To enable fair model comparisons, the test set is filtered to include only genes common to both datasets. This requires adherence to both sequence length constraints: a maximum coding region length of 8,100 nucleotides and a maximum full sequence length (including UTRs) of 9,000 nucleotides. Enforcing these constraints reduces the test set by 72 sequences.

Because the training and validation sets are used for model development and hyperparameter tuning rather than final evaluation, slight discrepancies in their composition across datasets are expected to only introduce slight bias and hence considered acceptable. Prioritizing data volume over strict overlap results in minor differences in set sizes, as detailed in Table 2.

4.2 Benchmarking Models

The ability of various deep learning architectures to predict PTR ratios from mRNA sequences is evaluated using the codon-encoded dataset (see Section 4.1.3). Specifically, the task is framed as a binary classification problem: determining whether a given sequence-tissue pair exhibits low or high protein abundance, as defined in Section 4.1.2.

To ensure a fair and consistent comparison, a standardized pipeline is used to train and evaluate a set of backbone architectures commonly employed for sequence-based prediction tasks. This pipeline encompasses feature processing, model training, and performance evaluation, as detailed in the following sections.

[14] as mentioned in Section 5.4, there are few edge cases where only one label is assigned for a sequence or two tissues have identical PTR values, resulting in two low- or high-PTR labels for two tissues of a sequence.

Table 2: Number of samples in codon- and nucleotide-encoded datasets. The "Dropped length" column indicates the number of sequences excluded due to exceeding maximum sequence length. The "Dropped set alignment" column indicates the number of sequences additionally removed to create identical test datasets across both dataset encodings.

Dataset encoding	Category	Train	Valid.	Test	Total	Dropped length	Dropped set align.
Codon	Unique mRNA seq.	7,782	1,688	1,599	11,069	210	-
	Binary classification	13,596	2,951	2,773	19,320	-	-
Nucleotide	Unique mRNA seq.	7,489	1,628	1,599	10,716	491	72
	Binary classification	13,055	2,843	2,773	18,671	-	-

4.2.1 Training Pipeline

The training data comprises the codon-encoded CDS region of each mRNA sequence (up to 8,100 nucleotides, which equals 2,700 codons) alongside the associated tissue ID and classification label. Validation data is used throughout training to monitor progress and detect issues such as overfitting or model misconfiguration. How different hyperparameters of this training pipeline are tuned will be described in Section 4.2.4.

Training Procedure. Model training is implemented using the PyTorch deep learning framework (Paszke et al., 2019), following best practices in the field (Rafi et al., 2024; Wayment-Steele et al., 2022). A binary cross entropy loss is used (PyTorch Contributors, 2023b). Optimization is performed using mini-batch stochastic gradient descent (SGD) (Goodfellow et al., 2016, pp. 151-153) in conjunction with the Adam optimizer (Kingma & Ba, 2015) and weight decay (PyTorch Contributors, 2023a). As proposed by Loshchilov and Hutter (2017) a cosine annealing learning rate schedule with warm restarts after a specific number of epochs is utilized (PyTorch Contributors, 2024b).

Early Stopping. To prevent overfitting and ensure generalization, early stopping is employed as a regularization technique, inspired by the implementation suggestion of isle_of_gods (2022). Training is halted if the validation AUC (see Section 3.3.4) stays below the highest achieved AUC of that training run minus 0.01 for eight epochs. The model checkpoint with the highest validation AUC is retained for evaluation (Goodfellow et al., 2016, pp. 246-248).

Sequence Reversal. Inspired by the use of bidirectional LSTMs in RNA modeling, random sequence reversal is used as a form of data augmentation. This technique leverages the bidirectionality of biological sequences, encouraging the model to learn directionally robust representations of biological interactions without altering the underlying architecture (Goodfellow et al., 2016, pp. 240 f.). This approach has been successfully applied in prior competitions, including the OpenVaccine (Das et al., 2020; Wayment-Steele et al., 2022) and DREAM challenges (Rafi et al., 2024).

Sequence Encoding. As discussed in Section 3.5, biological sequences can be encoded using various methods. One-hot encoding, while commonly used, results in high-dimensional and sparse representations – especially in the present codon setting, with 64 possible codons. An alternative is to use dense, low-dimensional representations inspired by NLP word embeddings (Mikolov, Sutskever, et al., 2013). In this approach, each codon is mapped to a trainable embedding vector via a fully connected embedding layer (PyTorch Contributors, 2023f), allowing the representation to adapt during backpropagation (Hwang et al., 2024, p. 1298). The embedding dimension is treated as a tunable hyperparameter.

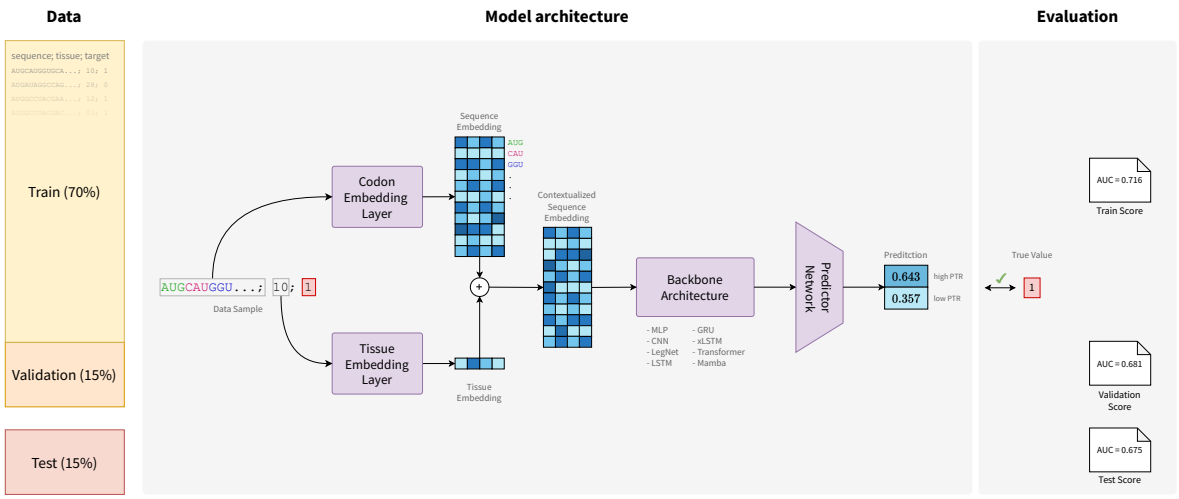


Figure 19: Training pipeline for model benchmarking. The codon-encoded input sequence is first embedded and contextualized with tissue information, then passed to a backbone model and finally to a dense predictor network.

The sequence of codon embeddings is element-wise summed with a fixed tissue embedding of the same dimensionality, forming contextualized embeddings that incorporate tissue-specific information at each codon position (see Figure 19). This strategy, inspired by the addition of positional encodings in Transformer models (Vaswani et al., 2017), outperformed simple concatenation of embeddings in later experiments (Section 5.2.1).

Predictor Network. The outputs of the backbone models are passed to a predictor network with one hidden layer. This network reduces dimensionality and produces a binary classification probability through a final sigmoid function. Dropout is applied to hidden layers to improve generalization and prevent overfitting (Srivastava et al., 2014; James et al., 2023, p. 431). The popular ReLU activation function (Nair & Hinton, 2010) is used in the hidden layers, shown to provide solid performance across multiple domains (Ramachandran et al., 2017).

Automatic Mixed Precision. Automatic mixed precision (AMP) is used to accelerate training and reduce memory usage by utilizing lower-precision floating-point numbers where possible (Micikevicius et al., 2018). This technique is particularly beneficial for large models, as it allows for faster training times without sacrificing model performance. The PyTorch framework provides built-in support for AMP, making it easy to integrate into the training pipeline (PyTorch Contributors, 2024a).

4.2.2 Backbone Model Implementations

With a standardized training pipeline in place, the selected models need to be implemented, which entails lots of possible design choices. The theoretical foundations of these architectures are discussed in Section 3.4. The concrete design choices made for each model are described in the following paragraphs while the chosen hyperparameters can be found in Table 3.

Multilayer Perceptron. A basic MLP is implemented as a baseline model. The first hidden layer's size is a tunable hyperparameter, while the second hidden layer contains half as many neurons. The output layer consists of a single neuron followed by a sigmoid to produce a binary class probability. ReLU activations are used in the hidden layers. Dropout is applied for regularization and its rate is treated as a tunable hyperparameter. Because input sequences vary in length, they are padded with

zeros to a fixed maximum of 2,700 codons. The input to the MLP is a flattened vector of the contextualized codon embeddings, with input dimensionality defined by the maximum sequence length of 2,700 codons times the embedding dimension.

Convolutional Neural Network. Given their success in gene expression prediction tasks (Hwang et al., 2024, p. 1309; Zrimec et al., 2021), a simple CNN architecture is implemented. Following padding, embedding, and tissue contextualization, the sequence is passed through two convolutional blocks, each followed by ReLU activations (Nair & Hinton, 2010) and max pooling (Goodfellow et al., 2016, pp. 339-341). The resulting output is then passed to the predictor network (see Section 4.2.1).

LegNet. The LEGnet architecture (Penzar et al., 2023) is integrated using its publicly available implementation.[15] Although originally designed for short sequences of only 150 bases, its CNN-based architecture enables seamless integration as a backbone model within the standardized training pipeline employed here. Unlike the original implementation, which employs one-hot encoding, the input here consists of tissue-contextualized codon embeddings.

RiboNN. As RiboNN is not yet open-sourced – the corresponding paper is currently a preprint – the architecture is implemented based on the descriptions and accompanying supplementary materials of Zheng et al. (2025). To incorporate the model as a backbone within the standardized training pipeline, slight modifications were necessary, primarily involving the use of the predictor network as the final model head. Similar to LegNet, the original RiboNN implementation employed one-hot encoding, representing another point of divergence from the present approach.

Recurrent Neural Networks. For the recurrent models LSTM and GRU input sequences are padded and then packed using PyTorch utilities (PyTorch Contributors, 2023g). This enables the RNNs to handle variable-length sequences efficiently by ignoring padded elements. The packed sequences are processed using PyTorch's native LSTM (PyTorch Contributors, 2023d) and GRU (PyTorch Contributors, 2023c) implementations. The final hidden state of the last recurrent unit is used as input to the predictor network.

xLSTM. The same procedure is applied to the currently developed xLSTM architecture (Beck et al., 2024)[16], with the exception that sequence packing is omitted, as it was not supported at the time of experimentation.

Transformer. An encoder-only Transformer architecture is implemented, which enables full attention over the entire sequence, as opposed to a decoder-only Transformer which can only attend previous elements of the sequence. This setup, similar to the BERT model (Devlin et al., 2019), allows the model to access global sequence context for improved prediction. PyTorch's Transformer encoder module (PyTorch Contributors, 2023e) is used, with sinusoidal positional encodings added to the tissue-contextualized codon embeddings, as proposed by Vaswani et al. (2017). The output corresponding to the final codon position is passed to the predictor network.

Mamba. The recently proposed Mamba architecture (Gu & Dao, 2023)[17] is also evaluated. The processed, tissue-contextualized input is passed through stacked Mamba layers – which is a tunable hyperparameter – and the final hidden state is fed into the predictor network to produce the classification output.

[15] Source code: https://github.com/autosome-ru/LegNet/blob/main/legnet/model.py
[16] Source code: https://github.com/NX-AI/xlstm, version 2.0.0
[17] Source code: https://github.com/state-spaces/mamba, version 2.2.2

4.2.3 Codon Frequency Baseline Models

Codon frequency features have demonstrated strong predictive power in related studies (Eraslan et al., 2019; Hernandez-Alias et al., 2023), making them a natural choice for baseline models in this work. These features represent the relative frequencies of the 64 possible codons within the coding sequence (CDS), capturing compositional patterns potentially associated with PTR ratios. Two baseline models are implemented to evaluate the predictive capacity of codon frequency features.

The first model is a simple MLP with a single hidden layer. This layer is followed by a ReLU activation and dropout for regularization. The input to the model is a concatenation of the codon frequency vector and the corresponding tissue embedding vector. This design enables the model to incorporate tissue-specific context alongside sequence composition in its predictions.

The second model is a basic RFC, similar to the approach of Hernandez-Alias et al. (2023). The existing implementation of the scikit-learn library (scikit-learn Developers, 2023) was utilized with default parameters: 100 trees, no maximum tree depth, and the Gini impurity criterion for evaluating splits. The input to this model is – as for the MLP – the concatenation of codon frequency vector with one-hot-encoded tissue identifier.

Compared to the sequence-based deep learning models, both baselines operate on significantly smaller feature spaces – fewer than 100 input features versus several thousand – making them compact and highly efficient. These models establish a reference point for evaluating the benefits of using more complex architectures in low and high PTR ratio classification.

4.2.4 Hyperparameter Tuning

The implemented models and used training pipeline involve many hyperparameters, such as learning rate, dropout probability, embedding dimensionality, and architecture specific parameters, that can substantially impact performance. However, identifying optimal values is non-trivial, as the complexity of the search grows exponentially with the number of parameters. Consequently, an efficient and scalable strategy for hyperparameter optimization is essential.

For this purpose, we employ Optuna (Akiba et al., 2019). A scalable and highly performant framework for automated hyperparameter tuning. Optuna provides tools for defining the search space and leverages advanced optimization algorithms to sample hyperparameters effectively. In this work, the tree-structured Parzen estimator (TPE) algorithm (Bergstra et al., 2011; Optuna Contributors, 2018c) is used to guide the search. TPE constructs a probabilistic model of the objective function and prioritizes exploration of the most promising regions in the parameter space. To further reduce computational cost, a median-based pruning strategy (Optuna Contributors, 2018b) is applied, which terminates underperforming trials early based on intermediate results. Each trial is evaluated on the dedicated validation set, ensuring that the selected hyperparameter configurations generalize well to unseen data. Optuna's dynamic sampling and pruning capabilities make it particularly well-suited for navigating the high-dimensional search spaces of the deep learning models described in this thesis by balancing a fast tuning process and model quality.

4.2.5 Evaluation

To obtain a final unbiased estimate of performance across models, while accounting for the use of the validation set during hyperparameter optimization, the models are trained on both the train and validation set and evaluated on the test set (see Section 4.1.4). Performance metrics and further training details are provided in Section 5.1.

4.3 Model Optimization: Custom PTRnet Model

Building on the architecture benchmarking of Section 4.2, a custom model – referred to as PTRnet – is developed to further enhance the predictive performance of low and high PTR ratio classification. PTRnet is based on the most promising architecture identified during benchmarking and integrates advanced training strategies along with additional input features to improve overall accuracy.

The emphasis on sophisticated training strategies arises from the recognition that, in deep learning, training procedures can be just as crucial to model performance as the architecture itself (Rafi et al., 2024; He et al., 2023; Zheng et al., 2025).

In contrast to the benchmark setup, which relied on the codon-encoded sequence of the coding region only, the PTRnet model utilizes the full nucleotide-encoded mRNA sequence, including the 5' and 3' UTR regions as well as structure features (see Section 4.1.3). This comprehensive input could enable the model to capture the full sequence context and structural attributes, which have been shown to influence translation efficiency, as discussed in Section 3.1.5.

4.3.1 Training & Tuning

To develop a high-performing custom model for PTR ratio classification, the training procedure described in Section 4.2.1 is extended and optimized by systematically exploring alternative training approaches, architectural modifications, and hyperparameter configurations. For this purpose, the Optuna framework (Akiba et al., 2019) is employed on the validation set to efficiently search the large space of possible parameter, architecture, and training strategy combinations. Based on the optimized PTRnet model, an ablation study is conducted to assess the contribution of each adaptation to model performance, enabling a comprehensive understanding of their respective impacts.

The following strategies, architectural changes, and hyperparameters are explored:

Aligning AUGs. To enhance the model's capacity to learn critical features near the start codon, such as the Kozak sequence (see Section 3.1.5), input sequences are aligned at their respective start codon. To facilitate this alignment, the 5' UTR is padded with zeros. This technique has shown effectiveness in a CNN-based architecture, as proposed in the preprint by Zheng et al. (2025). Depending on the architecture type, this alignment may be more or less beneficial. For instance, RNNs can inherently capture long-range dependencies, potentially reducing the need for explicit alignment, whereas CNNs may benefit more from such preprocessing due to their localized receptive fields.

Concatenating Tissue Embeddings with Sequence Embeddings. In the existing training pipeline (Section 4.2.1), tissue information is incorporated by element-wise addition of sequence and tissue embeddings, inspired by the approach of Vaswani et al. (2017) to incorporate positional information to word embeddings of a sentence. As an alternative, concatenating the tissue embedding with the sequence embedding is explored as a different strategy for integrating tissue information.

Inputting Frequency Features to Predictor Network. Given that codon frequency-based models have demonstrated strong predictive performance (Eraslan et al., 2019; Hernandez-Alias et al., 2023), incorporating codon frequency features into the predictor network is evaluated as a potential method to enhance performance.

One-Hot Encoding for Sequence Encoding. As discussed in Section 3.5.1, one-hot encoding is a widely used method for representing biological sequences. Although it produces high-dimensional and sparse vectors, it remains a valid alternative to dense embeddings and is therefore considered in the model variations.

Using Sequence Only. PTRnet is designed to operate on nucleotide-level mRNA sequences enriched with annotations such as coding region indicators (codon affinity in the codon region, 5' and 3' UTRs), secondary structure, and loop type features (see Figure 18). To evaluate the importance of these enrichments, the model is also tested using sequence information alone.

Extended Hyperparameter Tuning. Beyond the core hyperparameters optimized during benchmarking (see Section 4.2.4 and results in Section 5.1.1), additional parameters are included in the tuning process. These include settings of the cosine annealing learning rate scheduler (PyTorch Contributors, 2024b), the weight decay parameter in the AdamW optimizer (PyTorch Contributors, 2023a), as well as the hidden size and dropout rate of the predictor network.

Motif-Aware Pretraining Strategy. As discussed in Section 3.3.3, unsupervised pretraining using a denoising autoencoder can significantly improve downstream model performance by providing better weight initialization (Vincent et al., 2008; Chuai et al., 2018). In this work, PTRnet is pretrained on the complete dataset (train, validation, and test sets) using a denoising autoencoder with motif-aware noise, following the strategy described by N. Wang et al. (2024). The exact implementation details can be found in Section 5.2.2.

4.3.2 Evaluation

Following training and tuning, the PTRnet model is evaluated on the test set (see Section 4.1.4), which comprises previously unseen mRNA sequences – identical to those used in the codon-based benchmarking described in Section 4.2. This evaluation serves to provide an unbiased estimate of the model's generalization performance. Results from the tuning process, ablation study and test set evaluation are presented in Section 5.2.

5 Results

This chapter presents the empirical findings of the methodological approach described in Section 4, addressing the research questions outlined in Section 1.2. We begin by benchmarking a range of state-of-the-art DL models on the codon-encoded dataset (Section 4.1.3), with detailed benchmarking results provided in Section 5.1. Building on these results, we introduce and optimize a custom model, PTRnet, which integrates additional sequence features and advanced training strategies. The outcomes of this optimization are discussed in Section 5.2. Section 5.3 summarizes the overall test set performance of the key models. Finally, Section 5.4 provides an in-depth analysis of PTRnet's predictions, offering further insights into model behavior and limitations.

5.1 Benchmarking Results

Based on the codon-encoded dataset (see Section 4.1.3), we tune, train and evaluate the different backbone architectures according to the pipeline described in Section 4.2.1. The results and model predictions will be presented and analyzed in more detail in the following sections and address the first research question stated in Section 1.2: How accurately can mRNA sequences be classified into low- and high-PTR ratio outcome across different human tissues using state-of-the-art deep learning models?

5.1.1 Hyperparameter Tuning Results

The dataset is split into a training set (70%), a validation set (15%), and a test set (15%). The validation set is used to evaluate models trained on the training data during hyperparameter tuning, which is performed using the Optuna framework (Akiba et al., 2019) for efficient parameter search, as outlined in Section 4.2.4. The search space comprises values for the different hyperparameters, informed by prior work as well as our own experience and experiments. To constrain the search space and maintain a practical tuning duration, we try to focus only on the most influential hyperparameters for each model, as detailed in Appendix C.

All models are trained on two servers, each equipped with two A100 GPUs (see software and hardware configuration in Appendix D), following the procedure described in Section 4.2.1. As the hardware is shared with other users, training times may be affected by concurrent workloads.

We use a batch size of 64 samples for all experiments, except for xLSTM, where GPU memory limitations lead us to reduce the batch size to 16. All sequence-based models are trained for up to 100 epochs with early stopping. However, the codon frequency MLP model ("MLP (freq)") is trained for 150 epochs, as it requires more iterations to converge. The learning rate for the AdamW optimizer (PyTorch Contributors, 2023a) is treated as a tunable hyperparameter. In contrast, weight decay and the parameters of the cosine annealing scheduler with warm restarts (PyTorch Contributors, 2024b) remain fixed to reduce the size of the search space. The tuning process spans a maximum of 150 trials over a 24-hour period, including 10 warm-up trials with randomly selected parameters and employs the median pruner to terminate unpromising trials early.

The best-performing hyperparameters identified during tuning are summarized in Table 3 and are used to evaluate model performance on the test set in Section 5.1.3.

5.1.2 Analysis of Hyperparameter Tuning

As shown in Table 3, the models exhibit substantial differences in training speed, which limits the number of hyperparameter tuning trials for slower models. This constraint is particularly problem-

[18] This value corresponds to the learning rate used for the final test evaluation. The value obtained via Optuna tuning on the training set alone suggested a tenfold larger value, which led to unstable training when applied to the full training dataset (i.e., including both training and validation sets).

Table 3: Best parameters of hyperparameter tuning. For documentation the runtime in hours (capped at 24 hours) and number of trials (capped at 150) of the Optuna runs is given. Included hyperparameters are codon embedding dimension (`embed_dim`), predictor network hidden dimension (`pred_dim`), learning rate (`lr`), random sequence reversal flag (`seq_rev`), and model-specific hyperparameters.

Model	Runtime (h)	Trials	`embed_dim`	`pred_hid`	`lr`	`seq_rev`	Model-Specific Params
MLP (baseline)	1.55	150	128	-	3.21e-4	True	`hidden_size`: 512 `dropout`: 0.4
MLP (freq)	16.53	150	64	-	6.02e-4	True	`hidden_size`: 512 `dropout`: 0.0
CNN	2.05	150	128	128	9.88e-4	True	`num_kernels_conv1`: 64 `num_kernels_conv2`: 32 `kernel_size_conv1`: 6 `kernel_size_conv2`: 9 `max_pool1`: 20 `max_pool2`: 5
LegNet	24.00	80	128	32	2.30e-3	False	`kernel_size`: 11 `resize_factor`: 6 `se_reduction`: 8 `filter_per_group`: 2
RiboNN	6.40	150	128	32	1.80e-3	True	`num_layers`: 4 `grad_clip_norm`: 0.27 `dropout`: 0.1
LSTM	24.00	48	128	32	1.56e-4	False	`num_layers`: 2 `rnn_hidden_size`: 512 `bidirectional`: True `dropout`: 0.1
GRU	24.00	18	128	64	1.32e-5	True	`num_layers`: 2 `rnn_hidden_size`: 256 `bidirectional`: True `dropout`: 0.1
xLSTM	24.00	9	64	64	2.96e-4	False	`num_blocks`: 5 `conv1d_kernel_size`: 9 `m_qkv_proj_blocksize`: 8 `num_heads`: 8 `proj_factor`: 1 `dropout`: 0.4
Transformer	24.00	11	32	128	1.09e-4[18]	True	`num_layers`: 12 `dim_feedforward`: 128 `num_heads`: 4 `dropout`: 0.2
Mamba	5.28	150	128	128	2.16e-4	True	`num_layers`: 2 `d_state`: 16 `d_conv`: 2

atic for models with large search spaces, as the randomly initialized configurations in the early trials may strongly influence the tuning outcome. This issue is especially pronounced for xLSTM, which completes only nine trials despite its expansive search space, but it also affects GRU and Transformer models. Consequently, the optimal hyperparameter combination may not be identified within the allotted time.

Most architectures tend to prefer larger embedding dimensions (`embed_dim`) under the current training pipeline. Dropout is commonly selected as a regularization strategy, though its extent varies across models. Random sequence reversal, used as a form of data augmentation, appears beneficial for seven out of ten models. However, the reversals overall influence on model performance is consistently among the least significant hyperparameters (see Appendix C). More broadly, embedding dimension stands out as a consistently important hyperparameter across models, whereas the

predictor network hidden dimension (`pred_hid`) tends to be less impactful on average. Notably, hyperparameter importance varies considerably across models. For instance, learning rate and dropout show high sensitivity in some architectures but limited relevance in others.

Training dynamics, as illustrated in Appendix C, further reveal that certain models, such as the baseline MLP and Mamba backbone, exhibit instability and are highly sensitive to changes in hyperparameters. In contrast, models like the codon frequency-based MLP, LegNet, and Transformer backbones demonstrate greater stability and robustness to hyperparameter variations.

5.1.3 Benchmarking Results

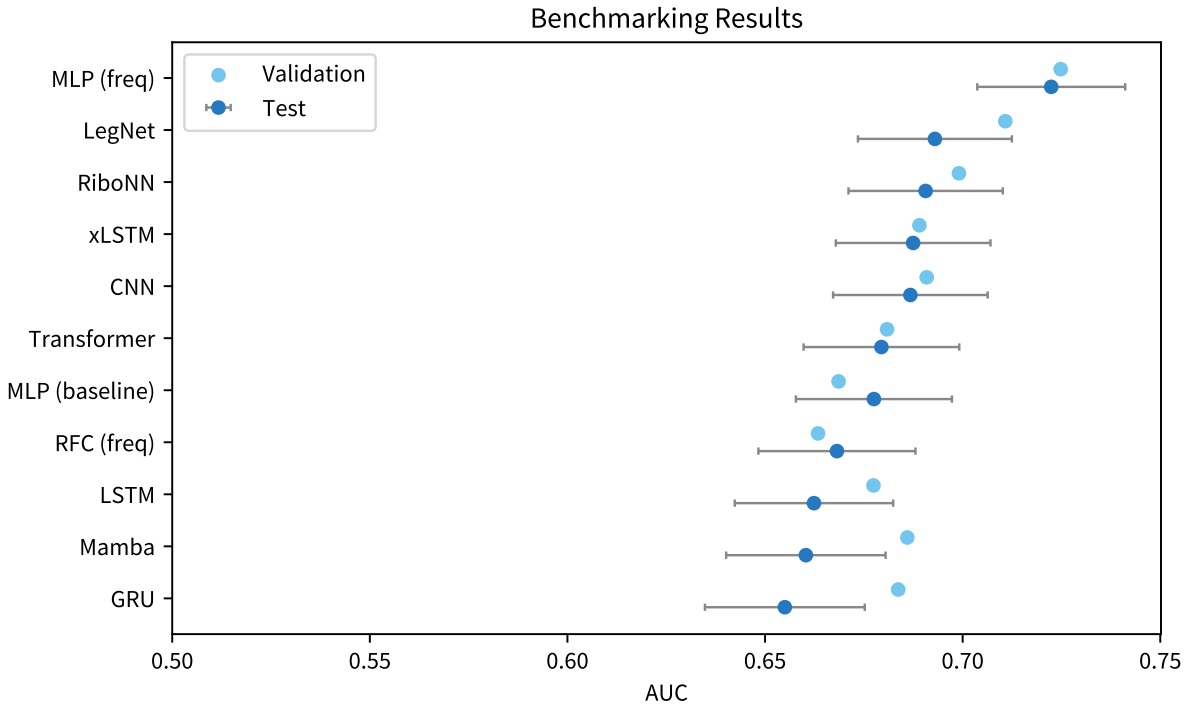


Figure 20: Test set AUC scores with 95% confidence intervals (Gildenblat, 2023) across models (trained on train and validation set) and best validation set score of hyperparameter tuning with Optuna (on train set).

Model Performance Overview. After training, AUC scores on the test and validation set are collected for each model in Figure 20. We report 95% confidence intervals computed using the Python package `confidenceinterval` (Gildenblat, 2023), which employs a fast implementation of DeLong's algorithm for analytical interval estimation (Sun & Xu, 2014). The simple codon frequency-based MLP baseline ("MLP (freq)") outperforms all more complex models, reaching 72% AUC on the test set. The shallow RFC ("RFC (freq)") achieves only 67% AUC, performing approximately on par with many deep learning models. This is in line with previous findings by Hernandez-Alias et al. (2023) and Zheng et al. (2025), which show that simple models leveraging hand-crafted features can be highly effective for gene expression prediction. However, it is important to interpret the results with caution, as the relatively wide confidence intervals suggest that the performance differences between models may not be statistically significant.

Among the sequence-based models, LegNet performs best with 69% test AUC, though still three percentage points below the frequency-based MLP. CNN-based backbones (LegNet, RiboNN, and CNN) generally perform better than other architectures, providing support for the frequent use of CNNs in

previous work in the field (Hwang et al., 2024; Zheng et al., 2025; Zrimec et al., 2021). Also the novel xLSTM model achieves favorable results. In contrast, GRU, LSTM, Transformer, and the novel Mamba model yield lower performance, with AUC scores around 67-68%. Across all sequence-based models, performance variation is modest, ranging from 65% to 69% AUC. These results slightly exceed those of Hernandez-Alias et al. (2023), who report an average AUC of 63% across tissue-specific models.

Validation-Test Score Discrepancies. Validation scores obtained through Optuna-based hyperparameter tuning generally align with test scores, but notable lower test scores occur for GRU, Mamba, LSTM, and LegNet – which suggests overfitting to the validation set. The Transformer model initially showed a large gap between validation and test AUC. Examination of loss curves revealed that the tuned learning rate is too high when training on the combined train and validation sets, causing instability and oscillating validation losses. Reducing the learning rate by a factor of ten stabilized training and yields more reliable benchmarking results. This intervention seemed reasonable, as the Transformer architecture was not able to learn at all which prevented a fair comparison. Similar issues of suboptimal learning rates for the larger training set size may have affected GRU, LSTM and Mamba, where tuning results do not fully transfer to the larger training setup.

5.1.4 Computational Efficiency and Model Complexity

	test AUC	val AUC (tuning)	train AUC	train time (min.)	avg epoch time (min.)	# parameters	# FLOPS	best epoch
MLP (freq)	0.7224	0.7248	0.7773	18.1244	0.1285	68417	66048	141.0000
LegNet	0.6930	0.7108	0.7304	99.0532	1.5477	5178971	59553722	64.0000
RiboNN	0.6906	0.6991	0.7081	5.9546	0.1751	1137089	636920096	34.0000
xLSTM	0.6875	0.6891	0.7044	243.5825	5.5360	97793	215165520	44.0000
CNN	0.6868	0.6909	0.7176	1.8047	0.1203	192289	115040	15.0000
Transformer	0.6794	0.6809	0.6885	316.8248	3.5598	159937	408935584	89.0000
MLP (baseline)	0.6775	0.6686	0.8744	8.0000	0.5333	177091329	177078528	15.0000
RFC (freq)	0.6682	0.6634		0.3148				
LSTM	0.6624	0.6774	0.6767	43.2947	1.7318	8976193	37920	25.0000
Mamba	0.6603	0.6860	0.7399	14.4450	0.2889	260865	353911552	50.0000
GRU	0.6550	0.6837	0.6604	130.8890	1.3634	1821569	35392	96.0000

Figure 21: Overview of benchmarking results. Including test and train AUC (trained on train and validation set), training time, average epoch duration, best epoch used for evaluation, number of model parameters and FLOPs for one forward pass. For comparison, the best validation set score from hyperparameter tuning (on train set) is also included. The darker the green color, the better the performance (AUC). The darker the red colors the higher the resource requirement (time, parameters, FLOPs, epochs). Since the RFC baseline is different in architecture and training procedure, some values are not reported.

Training Efficiency and Resource Requirements. Figure 21 provides a detailed comparison of models with respect to performance, training procedure and model complexity. Alongside test, validation, and train AUC scores, the figure includes total training time, average epoch duration, and the best epoch selected for evaluation. Model complexity is captured through parameter counts and FLOPs (floating point operations) for a single forward pass. While parameter counts are retrieved using the PyTorch model summary, FLOPs are estimated via the `fvcore` library (Facebook Research, 2022), giving a best guess on FLOP counts. Notably, some components, especially within novel architectures

and the predictor network, could not be included in the FLOP estimation, so reported values might represent lower bounds.

Model Trade-offs: Performance vs. Efficiency. Mamba and xLSTM stand out with low parameter counts but relatively high FLOP counts, indicating computationally intensive operations despite compact architectures. In contrast, the MLP baseline exhibits a high number of parameters due to its large fully connected layers. Surprisingly, xLSTM is the slowest to train among all models despite its efficient parameter usage. LegNet is both parameter-heavy and relatively slow but offers top-tier predictive performance. RiboNN achieves competitive results while training is an order of magnitude faster than LegNet, suggesting a favorable trade-off between performance and efficiency. The slim and simple CNN backbone stands out with rather few parameters and FLOPs while still achieving high performance. The shallow RFC remains an outlier, offering solid performance despite its simplicity and exceptionally fast training.

Overfitting Behavior. Analysis of the train AUC scores reveals that several models – especially the frequency- and sequence-based MLPs, Mamba, and LegNet – exhibit signs of overfitting, with noticeably higher training AUC compared to test AUC in the best epoch. The train and test set loss curves can be found in Appendix F.

5.2 PTRnet Optimization Results

As outlined in Section 5.1, the simple codon frequency-based MLP baseline outperformed all other models, achieving 72% AUC on the test set. However, this model does not utilize sequence information beyond codon frequencies, potentially limiting its capacity to capture more complex and nuanced patterns in the data. To overcome this limitation, we introduced a novel architecture in Section 4.3, PTRnet, which extends the performant and efficient RiboNN architecture (Zheng et al., 2025).

This section presents the results of optimizing PTRnet. The optimization process includes hyperparameter tuning, architectural modifications, and unsupervised pretraining. An ablation study is conducted to evaluate the contribution of these components on unseen data. The findings are discussed with respect to the second and third research questions defined in Section 1.2:

- Which additional features, particularly secondary structure predictions, enhance the predictive accuracy of PTR ratio classification?
- Can the most promising DL architecture be further improved through extensive hyperparameter tuning, architectural enhancements, and advanced training strategies such as unsupervised pretraining?

5.2.1 PTRnet Configuration Tuning

As described in Section 4.3.1, we extend the RiboNN architecture by incorporating additional sequence information, modifying its architecture, and applying advanced training techniques. The optimization process was executed over 42 hours using two A100 GPUs in parallel, leveraging Optuna for configuration search (Akiba et al., 2019). The search space and identified best configuration, based on highest validation AUC, are summarized in Table 4.

[19] The importance of embedding dimension related features (`dim_embedding` and `dim_embedding_tissue`) is omitted in the dashboard, but was provably part of tuning. Since `dim_embedding` depends on the concatenation or summing of sequence and tissue embeddings (i.e. if summed, `dim_embedding` = `tissue_dim_embedding`), if `concat_tissue_feature` is True, `tissue_dim_embedding` is considered a parameter. Hence, the importance of these parameters might not be shown in the dashboard due to the conditional nature of their search space.

Table 4: Search space, best identified configuration and description of each parameter used in tuning the PTRnet model.

Parameter	Search Space	Best Value	Explanation
`seq_encoding`	{"embedding", "ohe"}	embedding	Input encoding type (`embedding` vs. `one-hot`)
`dim_embedding`	{16, 32, 64, 128}	32	Sequence feature embedding dimensionality
`dim_embedding_tissue`	{4, 8, 16, 32}	4	Tissue embedding dimensionality
`lr`	[1e-5, 1e-2]	4.59e-4	Learning rate of AdamW optimizer
`weight_decay`	[1e-5, 1e-3]	6.79e-4	Weight decay coefficient in AdamW optimizer
`grad_clip_norm`	[0.1, 5.0]	0.152	Norm of gradient clipping
`reset_epochs`	{5, 10, 20}	20	Epoch interval for cosine annealing LR scheduler
`T_mult`	{1, 2, 3}	1	Multiplicative factor for cosine annealing cycle length
`num_layers`	{6, 7, 8, 9}	6	Number of convolutional layers in PTRnet
`dropout`	{0.0, 0.1, 0.3}	0.0	Dropout applied between layers in PTRnet
`seq_only`	{True, False}	False	Only sequence, no structure and coding area features
`concat_tissue_feature`	{True, False}	False	Concatenate tissue to nucleotide embeddings
`align_aug`	{True, False}	True	Align AUG codons across sequences by padding
`frequency_features`	{True, False}	True	Input frequency-based features to predictor network
`predictor_hidden`	{32, 64, 128}	32	Hidden dimension of the predictor network
`predictor_dropout`	{0.0, 0.1, 0.3}	0.0	Dropout rate in the predictor network

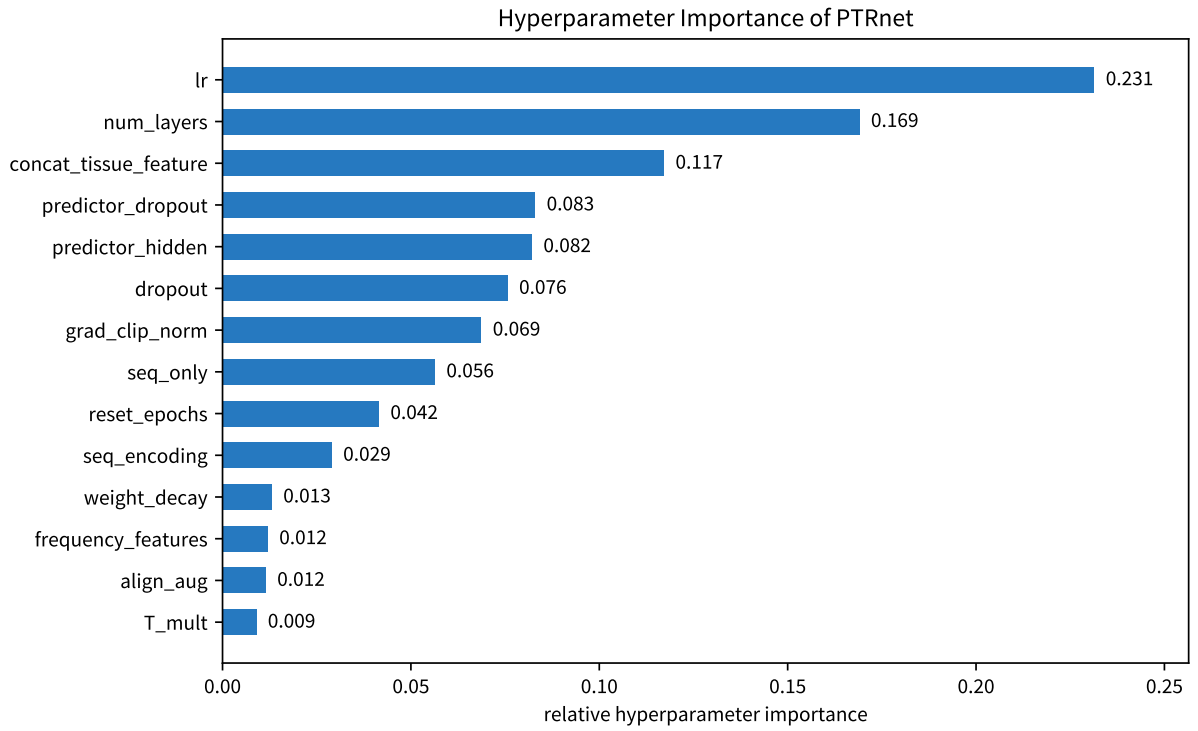


Figure 22: Relative hyperparameter importance of PTRnet based on Optuna's fANOVA importance evaluator (Optuna Contributors, 2018a).[19]

Figure 22 displays the relative importance of each parameter from the tuning process (see Appendix G for training curves, trial timeline and optimization history of the tuning process). The relative importance is based on Optuna's fANOVA importance evaluator (Optuna Contributors, 2018a), which quantifies the impact of each hyperparameter on the validation AUC score by fitting random forest regression models which predict the objective values given the parameter configurations of completed trials (Hutter et al., 2014). The most influential parameters include the learning rate, number of convolutional layers and strategy for contextualizing sequence embeddings – specifically, summing the se-

quence and tissue embeddings rather than concatenating them (`concat_tissue_feature`), as done in the benchmarking pipeline before in Section 4.2.1. Other notable factors, though less critical, include dropout in the convolutional layers, the gradient clipping norm and both the hidden dimension and dropout rate of the predictor network.

5.2.2 Final PTRnet Architecture

The final PTRnet architecture is depicted in Figure 23. It extends the RiboNN model by introducing several key modifications designed to enhance both performance and generalization. These modifications, guided by the configuration tuning results detailed in Section 5.2.1, include:

- **Pretraining:** PTRnet is pretrained using a denoising autoencoder strategy, where the model learns to reconstruct the original sequence from a corrupted input (see also Section 4.3.1). This pretraining phase allows the model to develop robust representations before fine-tuning on the downstream task. The approach involves an encoder that maps the input sequence to a latent representation, from which a decoder reconstructs the original sequence. The encoder comprises six stacked convolutional layers applied to the sum of tissue embeddings and nucleotide embeddings (with secondary structure and coding region features). The decoder mirrors the encoder architecture and consists of six transposed convolutional layers. The model is trained to minimize reconstruction loss, computed as the cross-entropy between the original and reconstructed sequences. After pretraining, the decoder is discarded, and the encoder's weights are used to initialize the PTRnet model for subsequent supervised training on the PTR classification task. Pretraining is performed using the same hyperparameters as the baseline PTRnet model (see Parameters in Table 4).
- **AUG Alignment:** Start codons (AUG) are aligned across sequences by padding the sequence beginnings, ensuring consistent positional context.
- **Encoding:** Categorical input features are encoded using a learned embedding layer, instead of the commonly utilized one-hot encoding.
- **Secondary Structure Features:** Predicted secondary structure and coding region indicators are integrated as additional input features. These are summed with nucleotide embeddings to augment the sequence representation.
- **Tissue Information:** The model backbone receives a tissue-contextualized sequence embedding, computed as the element-wise sum of the augmented sequence representation and tissue embeddings, rather than their concatenation.
- **Convolutional Layers:** The number of convolutional layers is reduced from 10 (as used in the benchmarking setup) to 6, based on tuning outcomes. The hyperparameters used for the original RiboNN configuration are provided in Table 3 and can be compared with the ones of PTRnet in Table 4.
- **Codon Frequency Features:** The input to the predictor network is augmented by concatenating codon frequency features with the latent sequence representation produced by the convolutional layers.

5.2.3 Ablation Study Results

Following the final PTRnet architecture of Section 5.2.2, we perform an ablation study to evaluate the contribution of individual configurations and components. This is achieved by systematically removing or altering specific elements and assessing the impact on test set performance. Results are summarized in Figure 24.

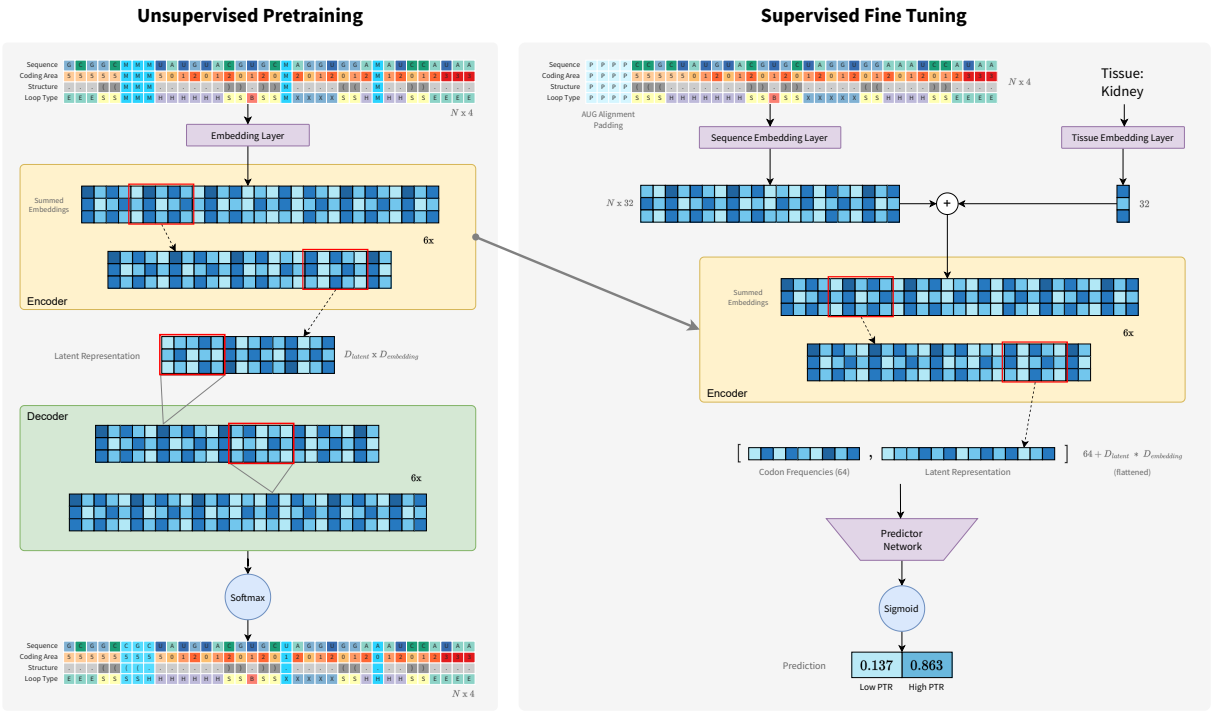


Figure 23: Final PTRnet architecture. The model extends RiboNN by incorporating additional sequence features, including codon frequencies and applying AUG alignment (right). The pretraining strategy employs a denoising autoencoder approach in which the model learns to reconstruct the original sequence from a corrupted input (left). Illustration inspired by Chuai et al. (2018, p. 4).

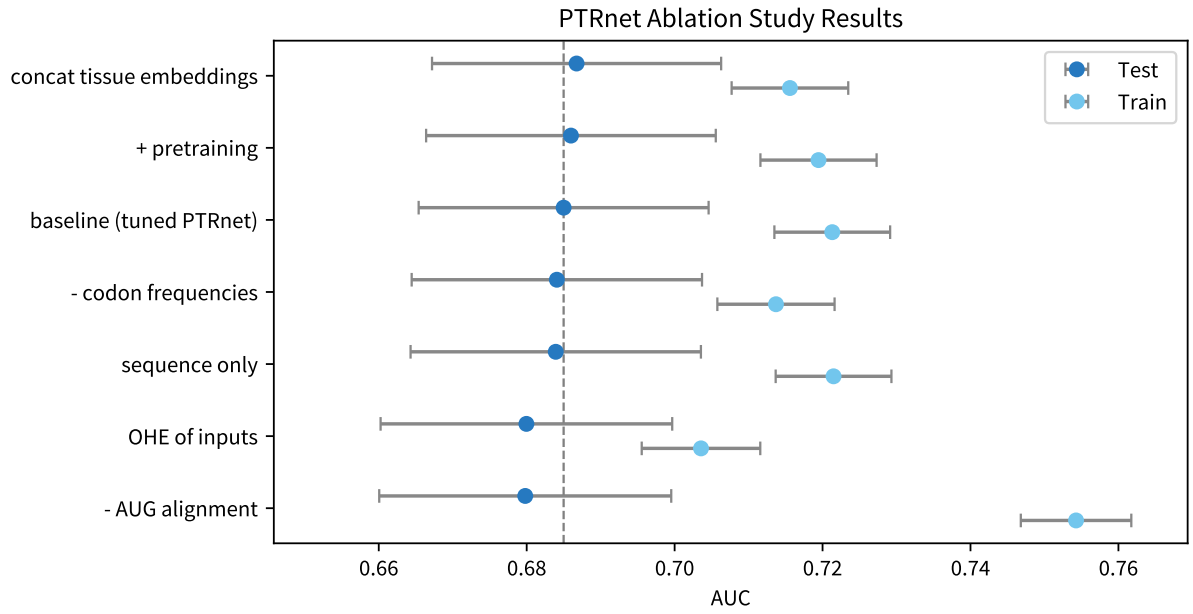


Figure 24: Ablation study results of PTRnet on the test set. Showing AUC scores with 95% confidence intervals (Gildenblat, 2023) for different model configurations. The baseline model is based on the best model configuration obtained from Optuna (Akiba et al., 2019) training, described in Section 5.2.2.

With all test AUC scores lying within 67.98% to 68.67%, model performance varies only marginally and is, overall, statistically indistinguishable across different ablation configurations. The baseline model, optimized through configuration tuning, achieves a test AUC of 68.50%. Pretraining this model yields an AUC of 68.59%, corresponding to eleven additional correctly classified samples out of 2,773, indicating that while pretraining is marginally beneficial, its impact remains limited in the current implementation. However, on the training set, certain ablation variants – particularly those omitting AUG alignment or using one-hot encoded (OHE) inputs – show statistically significant deviations from the baseline, suggesting that these design choices might meaningfully affect model learning capacity.

Interestingly, concatenating tissue and sequence embeddings – contrary to the summation strategy identified as optimal during tuning – results in a slight performance increase to 68.67%. The effects of incorporating codon frequencies into the predictor network or training solely on sequence data appear negligible, suggesting that either the model inherently captures tissue-specific codon usage bias and secondary structure features or it fails to make meaningful use of this additional information.

In contrast, replacing learned embeddings with one-hot encoding and omitting AUG alignment both reduce performance slightly, with AUC scores dropping to approximately 67.98% in each case. Removing AUG alignment, however, led to a strong increase of train AUC, indicating overfitting behavior in this setting.

Training curves for all ablation experiments are provided in Appendix H, where consistent overfitting is observed after approximately 40 epochs. Across all experiments, even with stronger regularization by utilizing dropout, performance appears to plateau below 70% AUC – an upper limit that none of our deep learning models has yet surpassed on the test dataset. This plateau is evident in the final test set runs (Appendices F and H) and in the Optuna-based tuning results on the validation set (Appendices C and G).

5.3 Overall Test Set Results

Figure 25 summarizes the final test set performance of the most relevant models. These models were trained on the combined training and validation sets using the best configurations identified through hyperparameter tuning and the PTRnet ablation study.

For comparison, results from Hernandez-Alias et al. (2023) are included. In their study, tissue-specific Random Forest Classifiers (RFCs) were trained on custom-engineered features derived from either sequence data or codon frequencies. These models were evaluated via 5-fold cross-validation using the same dataset as this work. The reference score shown in Figure 25 represents the average performance across all tissue-specific RFC models.

The lowest performance is observed in the shallow RFC models from Hernandez-Alias et al. (2023), with a mean cross-validation AUC of 63.3%.[20] When applied to our test set using codon frequency features across all tissues, an RFC baseline achieved 66.82% AUC. In contrast, the pretrained PTRnet model, trained on full nucleotide-encoded mRNA sequences enriched with secondary structure features, achieved a higher test AUC of 68.6%. However, this is slightly lower than the 69.1% AUC achieved by the simpler RiboNN model during benchmarking (Section 5.1.3). RiboNN, which serves as the architectural foundation for PTRnet, was trained on codon-encoded representations of the coding regions only. The best overall performance was achieved by the MLP baseline model, which was trained solely on codon frequency data. This simple model outperformed all others with a test AUC of 72.2%, significantly outperforming the RFC baseline models.

[20] Since the result is taken from Hernandez-Alias et al. (2023), no confidence intervals could be estimated.

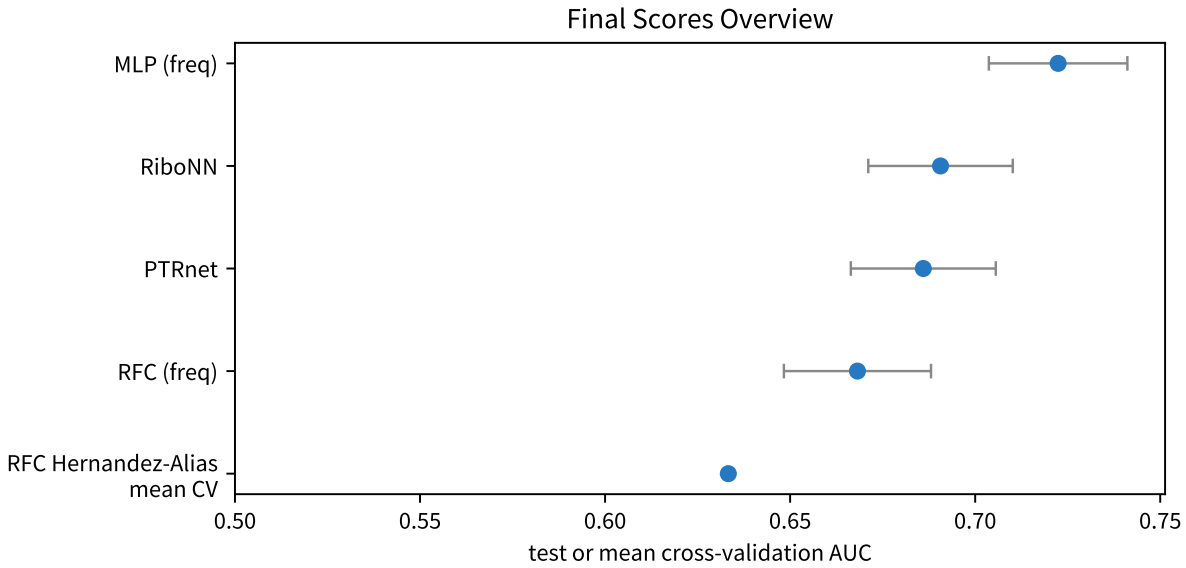


Figure 25: Mean cross-validation AUC of RFCs of Hernandez-Alias et al. (2023) and test set AUC with 95% confidence intervals (Gildenblat, 2023) of major models trained on the combined training and validation sets.

5.4 PTRnet Predictions Analysis

As outlined in Section 5.3, PTRnet struggles to match the performance of simpler models when applied to full nucleotide-encoded mRNA sequences enriched with secondary structure features. To better understand the model's behavior, this section analyzes PTRnet's predictions and compares them with those of the best-performing model – the codon frequency-based MLP ("MLP (freq)").

Recall that the binary classification task was designed so that nearly all mRNA sequences appear twice in the dataset: once associated with a tissue exhibiting a low PTR ratio (lowest among all tissues of a sequence and at least half of the across-tissue PTR ratio average) and once with a tissue exhibiting a high PTR ratio (highest among all tissues and at least double the across tissue average). Only a few edge cases deviate from this pattern: 35 sequences appear only once, due to only a low- or high-PTR tissue meeting the target criteria (see Section 4.1.2) and four sequences appear three times, resulting from tied PTR values that produce two low or two high labels for the same sequence.

5.4.1 Prediction Distribution

Figure 26 presents two visualizations based on the test set for both the PTRnet and MLP (freq) models: a confusion matrix of predicted versus true labels (left) and the distribution of correctly classified samples per sequence frequency (right). The right-hand plots illustrate, for each unique mRNA sequence (y-axis), how many of its occurrences in different tissues were correctly classified (x-axis) as either low- or high-PTR ratio samples.

The confusion matrix shows that PTRnet correctly classifies 1,772 out of 2,773 samples, corresponding to an accuracy of 64%, whereas the codon frequency-based MLP achieves a slightly higher accuracy of nearly 67%.

Given that most mRNA sequences appear twice in the dataset – once paired with a tissue showing a low PTR ratio and once with a tissue showing a high PTR ratio – it is informative to examine how often the model predicts the same class for both occurrences. The middle row of the right-hand plot for PTRnet reveals that this occurs in 1,204 cases (43%). In comparison, the model predicts both samples correctly for 1,132 sequences (41%) and both incorrectly for 390 sequences (17%). The MLP

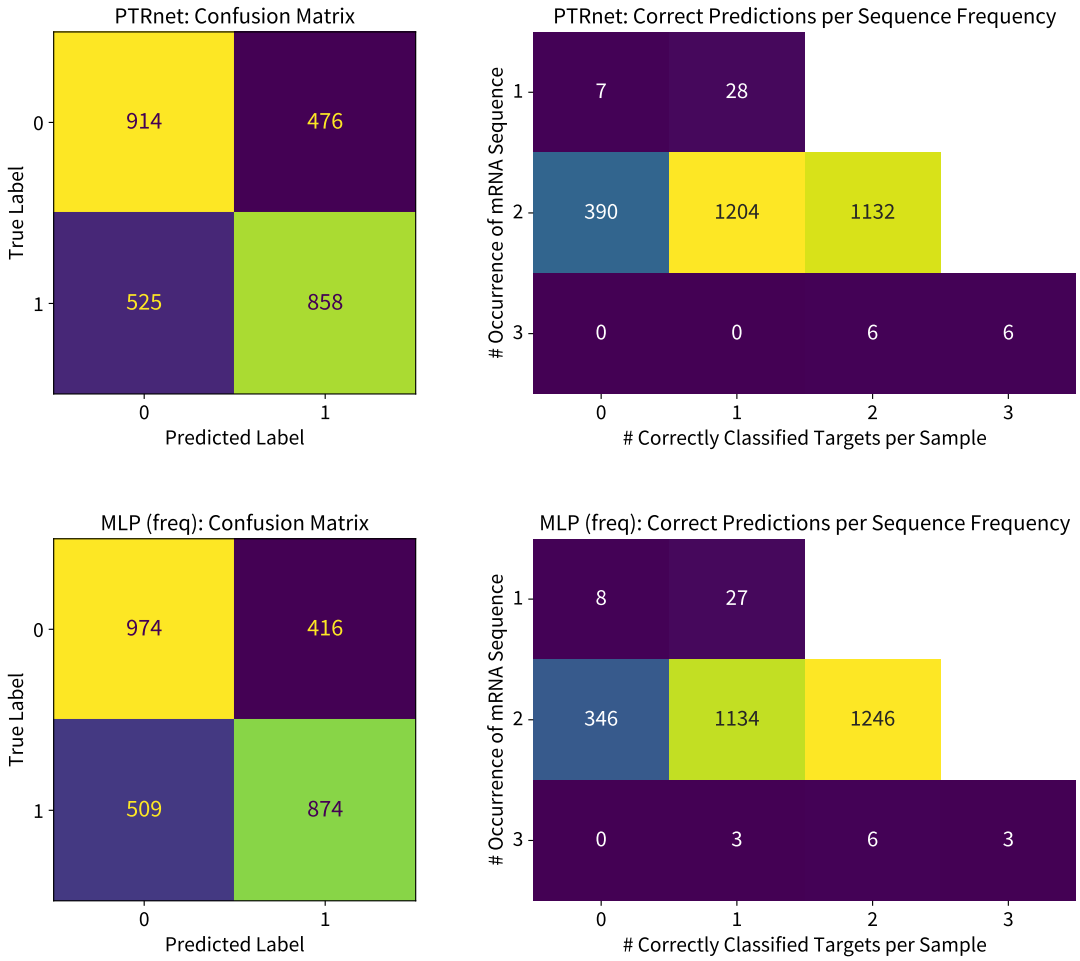


Figure 26: Confusion matrices (left) and number of correctly classified targets per sequence frequency (right) for PTRnet (top) and MLP (freq) (bottom) on the test set. Some combinations are omitted due to infeasibility: the number of correct predictions cannot exceed the total number of occurrences.

baseline performs marginally better across all categories. The upper and lower rows represent the edge cases described in Section 5.4, where sequences appear only once or three times. Overall, these results indicate that for only slightly more than 40% of the sequences, either model successfully captures tissue-specific differences in PTR ratios. While for a similar number of sequences, the model just assigned one of the two categories, to score at least one correct. This suggests that while the models exhibit some sensitivity to tissue context, their predictive capacity remains limited for many sequences. The next section explores how these predictive patterns vary across tissues.

5.4.2 Predictions per Tissue

Figure 27 compares AUC scores across tissues for three different models: pretrained PTRnet, the codon frequency-based MLP baseline ("MLP (freq)"), and the tissue-specific RFCs from Hernandez-Alias et al. (2023). Tissues marked with † indicate that PTRnet produced only majority-class predictions based on in tissue class imbalance, effectively failing to distinguish between low and high PTR classes. This issue did not arise for the MLP baseline, while no such metadata is available for the RFC models.

The performance of PTRnet varies substantially across tissues, with AUC scores ranging from near-random (around 50%) to strong predictive performance exceeding 80%. A similar trend is observable for the RFCs and MLP baseline, though the variation is generally less pronounced.

All models appear to struggle with certain tissues, such as Testis, Esophagus, and Liver, whereas

others like Kidney and Lung consistently yield strong performance. However, there remains notable variation across models, each exhibiting distinct strengths and weaknesses. Importantly, AUC scores may be artificially inflated due to within-tissue class imbalances, as can be seen in Appendix A. This is particularly evident in tissues such as Fat, Brain, Appendices, and Urinary Bladder, where PTRnet achieves high AUC values but resorts to majority-class predictions. In these cases, the model appears to exploit label imbalance rather than learning biologically meaningful patterns.

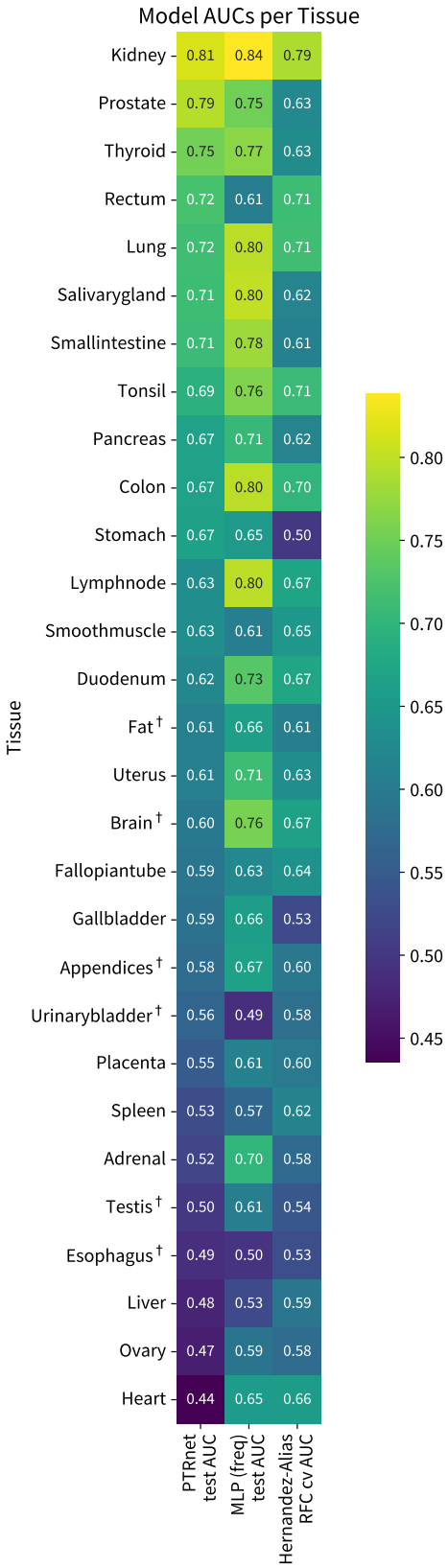


Figure 27: Model AUC scores per tissue for the pretrained PTRnet model, the codon frequency-based MLP and reference scores from Hernandez-Alias et al. (2023). Tissues marked with † indicate that the PTRnet model produced only majority-class predictions for that tissue.

6 Discussion

The results of Section 5 provide insights into the feasibility and challenges of classifying sequence-tissue pairs into low- and high-PTR categories using DL models. Here, we discuss the results in relation to the research questions, highlighting the key findings, limitations, and potential avenues for future research.

6.1 Discussion of Results

Benchmarking Results. The benchmarking experiments presented in Section 5.1 address the first research question outlined in Section 1.2, namely, how accurately mRNA sequences can be classified into low- and high-PTR ratio categories across different human tissues using state-of-the-art DL models. Within the standardized evaluation pipeline, the codon frequency-based MLP achieved the highest test AUC of 72.2%, also outperforming the shallow models used in previous studies (Hernandez-Alias et al., 2023). In contrast, all sequence-based DL architectures plateaued below 70% AUC while incurring substantially higher computational and memory costs (see Table 21). Most began to overfit after approximately 30–40 epochs despite the applied regularization strategies (Appendix F; see also Section 5.3). These observations align with earlier work showing that hand-crafted codon usage features contain considerable discriminative signal for PTR ratio classification (Eraslan et al., 2019; Hernandez-Alias et al., 2023). Biologically, the superior performance of codon frequency representations suggests that tissue-specific synonymous codon biases and tRNA availability remain the dominant determinants of translational efficiency in the current dataset, whereas incorporating secondary structure information did not yield measurable improvements (Hanson & Coller, 2018; Hernandez-Alias et al., 2023).

Achieving comparability across architectures required enforcing identical training data, codon-level encoding, and predictor heads throughout the benchmarking pipeline (Section 4.2.1). While this design isolates architectural effects and limits confounding from hyperparameter choices (Table 3), it may also restrict certain models from reaching their optimal performance, potentially explaining the suboptimal results of well-established architectures such as the Transformer. Consequently, part of the observed performance gap may arise from these harmonization constraints rather than from inherent limitations of the sequence models, and the reported scores should therefore be interpreted within the bounds of the adopted pipeline.

Model Optimization Results. The second and third research questions (see Section 1.2) examine whether additional features – particularly secondary structure features – can enhance the predictive accuracy of PTR ratio classification and whether the most promising DL architecture can be further improved through extensive hyperparameter tuning, architectural modifications, and advanced training strategies.

To address these questions, the RiboNN architecture (Zheng et al., 2025) was selected as the most promising sequence-based backbone architecture from benchmarking, achieving the second-highest test AUC (69.1%) with high training speed. Building on this foundation, the custom PTRnet model was developed as an extension of RiboNN. PTRnet utilizes nucleotide-encoded sequence inputs, integrates additional features such as coding region indicators, secondary structure as well as loop type and employs advanced training strategies, including unsupervised pretraining, AUG alignment, and extensive hyperparameter optimization.

Despite these enhancements, PTRnet did not surpass the codon frequency-based MLP, achieving a lower test AUC of 68.6% compared to 72.2%. As detailed in the ablation study (Section 5.2.3), architectural refinements such as learned embeddings and AUG alignment yielded only minor, statistically non-significant improvements. Similarly, incorporating structural annotations and unsupervised pretraining did not meaningfully affect performance. Despite employing multiple regularization

and training strategies – including pretraining, dropout, weight decay, data augmentation via random sequence reversal, and layer normalization – overfitting could not be sufficiently mitigated (see Appendix H). Consequently, the second and third research questions can be answered in the negative: under the current data regime, neither richer feature sets – including, among others, secondary structure features – nor extensive optimization of the leading sequence architecture is sufficient to narrow the performance gap to codon frequency-based models. This finding further supports prior evidence that codon usage remains the dominant correlate of PTR ratios (Eraslan et al., 2019; Hernandez-Alias et al., 2023).

A persistent limitation across architectures is the difficulty of capturing tissue-specific context. For 43% of sequences, PTRnet predicts the same class for both the low- and high-PTR tissue occurrences, underscoring limited sensitivity to within-gene differences (Section 5.4.1). Tissue-level evaluation further reveals majority-class behaviour in imbalanced tissues such as Fat, Brain, Appendices, or Urinary Bladder, which inflates AUC values without reflecting genuine regulatory insight (Section 5.4.2). Nevertheless, tissues including Kidney and Lung exhibit strong performance, suggesting that the models do learn biologically meaningful patterns when sufficient discriminative signal and balanced labels are present.

6.2 Limitations

In addition to the limitations mentioned in the previous section, the following limitations of this thesis should also be considered:

Target Variable Definition. The current binary classification scheme, based on Hernandez-Alias et al. (2023), emphasizes across-tissue variation by designating one tissue per gene as low-PTR and another as high-PTR (with edge cases as described in Section 5.4). This method yields a balanced dataset and highlights the most pronounced tissue-gene combinations – those exhibiting particularly low or high protein abundance. While this strategy effectively focuses the model on significant across-tissue differences, it substantially limits the available information. Specifically, it discards PTR ratio data from other tissues for each gene, narrows the task's scope and thereby potentially overlooks broader expression patterns and more subtle but biologically relevant tissue-dependent signals.

Training Data Quantity and Quality. The predictive power of all machine learning models is fundamentally limited by the amount and quality of available training data (James et al., 2023; Goodfellow et al., 2016). In this thesis, the dataset is constrained by the number of mRNA sequences with reliable PTR measurements across tissues. Furthermore, measurement errors and noise in the underlying experimental data can propagate through the models, limiting achievable accuracy (Zrimec et al., 2021, p. 20; Franks et al., 2017).

Secondary Structure Prediction Accuracy. The integration of secondary structure features in this thesis relies on the predictions of LinearFold (Huang et al., 2019) and bpRNA (Danaee et al., 2018). While effective and efficient, they are not perfectly accurate and only show one probable secondary structure outcome among many possible ones (Huang et al., 2019; Danaee et al., 2018; Binet et al., 2023). Prediction errors or limitations in modeling complex mRNA structures may introduce noise into the feature set, potentially obscuring true biological signals.

Experimental Scope and Methodology. The experimental design, including the choice, design, and training procedure of models, feature encodings as well as evaluation metrics, represents only a subset of possible approaches. Other architectures, feature representations, or training strategies not explored here may yield different or improved results.

Model Combinations. This thesis evaluates individual models in isolation. Potential benefits from combining different architectures – in an ensemble or hybrid fashion – are not assessed.

6.3 Future Work

Based on the findings and limitations of this thesis, several avenues for future research can be pursued to further enhance the understanding and prediction of mRNA translation efficiency:

Data-Related Considerations

- **Increased Data Availability:** Expanding the dataset to include more mRNA sequence samples across diverse tissues, each paired with corresponding PTR ratio values, could enhance model training. This could be particularly beneficial for more complex architectures – which currently show limited performance – by providing them with the volume and diversity of data needed to exploit their full capacity.
- **Alternative Target Definition:** Revising the current binary classification scheme to incorporate all mRNA-tissue pairs with PTR ratios above or below a defined threshold, rather than restricting the dataset to the top and bottom tissues per gene, could increase data size. Postprocessing could still facilitate obtaining a balanced dataset. This approach would expose the model to a wider range of tissue contexts per gene, capturing more nuanced protein expression patterns and enabling more robust tissue-specific learning.

Architectural Extensions

- **Graph Neural Networks (GNNs):** GNNs present a promising direction, as mRNA molecules exhibit intricate secondary and tertiary structures that can naturally be represented as graphs (Hwang et al., 2024, p. 1301; Das et al., 2020). Leveraging GNNs may enable more effective modeling of structural dependencies and contextual relationships within mRNA sequences.
- **Specialized Transformers:** Given the strong performance of Transformer architectures in genomic tasks (Choi & Lee, 2023), more specialized variants merit exploration. Models such as Longformer (Beltagy et al., 2020), RNAdegformer (He et al., 2023), and DNABERT (Ji et al., 2021) could be adapted to better capture long-range dependencies and structural features in mRNA sequences.
- **HyenaDNA & Bio-xLSTM:** Novel architectures like HyenaDNA (Nguyen et al., 2023) and Bio-xLSTM (Schmidinger et al., 2024) offer innovative alternatives to traditional attention mechanisms and recurrent structures.
- **Ensemble and Hybrid Architectures:** Ensemble methods, such as model averaging or majority voting across multiple seeds or models could improve prediction generalization and accuracy (Goodfellow et al., 2016, pp. 256-258). Similarly, hybrid models that integrate different architectural paradigms – such as combining CNNs with Transformers or incorporating GNN layers – may help harness complementary strengths.

Pretraining

- **Expanded Pretraining Corpus:** Enhancing the unsupervised pretraining dataset by including a broader set of mRNA sequences – along with their predicted secondary structures – could improve the quality of learned latent representations.

- **Alternative Unsupervised Objectives:** Exploring different unsupervised learning objectives, such as variational autoencoders (VAEs) or masked language modeling (MLM), may prove more effective (He et al., 2023; N. Wang et al., 2024).

Training Procedures and Optimization

- **Regularization Techniques:** To counteract the overfitting observed in Appendices F and H, future work could focus on refining regularization strategies. Techniques such as early stopping, dropout at various levels, weight decay, data augmentation, and layer normalization were employed in this thesis. However, further exploration of these methods or the introduction of novel regularization approaches may help improve model generalization.
- **Alternative Encoding Strategies:** Various encoding schemes for mRNA sequences, their associated features, and tissue-specific context could be explored, such as k-mer embeddings, position-aware encodings, or learned latent representations (Hwang et al., 2024).

Alternative Prediction Objectives

- **Classification vs Regression:** Initial approaches of this thesis to predict the numeric PTR ratio in a regression setting did not generalize effectively, achieving only an R^2 of below 15% due to overfitting. This limitation could be addressed with access to larger, high-quality datasets and the application of more advanced modeling techniques.
- **Predicting Deltas Across Tissues:** In a regression framework, predicting the PTR ratio deltas from across-tissue means instead of the individual values for each unique mRNA sequence could shift focus of DL models to tissue-specific variations.
- **Multi-Output Predictions:** Jointly predicting PTR ratios for all tissues in a multi-output setting could eliminate the need to explicitly encode tissue information in the input sequence.
- **Auxiliary tasks:** Integrating auxiliary tasks such as token reconstruction or secondary structure prediction in a multi-task learning framework could improve model convergence and foster the development of more generalizable representations. This setup may also help models implicitly learn biologically meaningful patterns relevant to the primary classification objective.

Interpretability and Explainability

- **Explainable AI Methods:** Applying interpretability techniques (e.g., relevance propagation or saliency maps) could help investigate what features the models focus on and gain insights into why complex models overfit and fail to capture more subtle biological signals.
- **Model and Prediction Analysis:** Performing a more in-depth error analysis – such as examining gradients, activations, or misclassified examples – can provide insights into where and why models fail, thereby revealing potential areas for improvement.

7 Conclusion

This thesis investigates whether Deep Learning (DL) models can distinguish low and high Protein-to-mRNA (PTR) ratios directly from mRNA sequences in a specific tissue. Across eleven thousand mRNA sequences and 29 human tissues, we systematically compared different network architectures, training strategies, and feature augmentations, including secondary-structure predictions. While earlier work has mostly emphasized across-gene variability as the main determinant of translational efficiency, this thesis concentrates on the relatively smaller across-tissue differences that may contribute to the identification of regulatory elements relevant to therapeutic gene design (Hernandez-Alias et al., 2023; Eraslan et al., 2019; Zrimec et al., 2021).

Our findings indicate that sequence-based DL models can classify PTR ratios across tissues but remain outperformed by simpler codon frequency-based approaches. The MLP trained exclusively on codon usage achieved a test AUC of 72.2%, compared to 69.3% for the best-performing codon-encoded backbone (LegNet) and 68.6% for the custom-designed PTRnet, which was pretrained and optimized using secondary structure-enriched, nucleotide-level inputs. The shallow RFC baseline, which reached an average of 63.3% AUC when trained separately per tissue in earlier work (Hernandez-Alias et al., 2023), improved to 66.8% in our unified setting, yet still fell short of the codon frequency-based MLP. These performance gaps underscore how quickly the deep, sequence-based models tend to overfit despite extensive regularization.

The superiority of codon-informed models is consistent with long-standing evidence that synonymous codon usage mirrors tissue-specific tRNA availability and thereby modulates translation efficiency (Gould et al., 2014; Hanson & Coller, 2018; Hernandez-Alias et al., 2023). By aggregating each sequence into codon frequency profiles, the MLP effectively capitalized on these translational signatures without needing to rediscover them from raw nucleotide context. This behaviour echoes reports that GC- versus AT-ending codon preferences partition human tissues into distinct translational regimes (Kudla et al., 2006), reinforcing the interpretation that our classifier succeeds precisely when codon composition harmonizes with the tissue-specific protein synthesis machinery.

Despite these promising results on codon frequency data, considerable variability in prediction performance across tissues persists. Kidney, Lung, and Thyroid achieved AUC values above 75%, while Testis, Esophagus, and Liver remained close to random guessing (Figure 27), in line with previous work (Hernandez-Alias et al., 2023; Eraslan et al., 2019). Moreover, the confusion-matrix analysis revealed that PTRnet predicted the same class for both tissue occurrences of an mRNA sequence in 1,204 out of 2,773 samples (43%; Figure 26), inflating accuracy in class-imbalanced tissues such as Fat, Brain, Appendices, and Urinary Bladder where majority-class predictions yield deceptively high scores (Figure 27).

Another important observation is that training unified models across all tissues consistently improved generalization relative to tissue-specific training. The unified multi-tissue RFC, for instance, surpassed the previously reported per-tissue average (Hernandez-Alias et al., 2023) by more than three percentage points: 66.8% versus 63.3% AUC (Figure 25). These gains suggest the presence of transferable features that encode shared translational control mechanisms across tissues.

Several factors may explain why the sophisticated PTRnet model did not meet expectations. Limitations in data quality or quantity, architectural shortcomings, insufficient regularization, or a lack of learnable sequence-level signals could all contribute. Indeed, it is possible that, as Eraslan et al. (2019, p. 16) hypothesized, there may be no robust, deterministic sequence features that enable reliable prediction of across-tissue variability after all. As outlined in Section 6.3, future efforts could aim to improve data curation, experiment with alternative modeling strategies, and integrate domain-specific biological knowledge. The application of explainable AI techniques may also offer insights into current model limitations and guide the design of more effective predictive tools.

Overall, this thesis demonstrates that applying deep neural networks to codon frequencies currently provides the most reliable option for effective PTR ratio classification. Future priorities include

collecting larger, more balanced across-tissue datasets, along with exploring improved strategies to mitigate overfitting. Incorporating measures of tRNA availability or codon optimality could further clarify whether the learned representations truly reflect tissue-specific translational control and may ultimately reveal actionable levers for therapeutic sequence optimization (Hanson & Coller, 2018; Hernandez-Alias et al., 2023).

References

Ahmed, Z. (2024, May). xLSTM: Enhancing Long Short-Term Memory for Large Language Models. Retrieved April 14, 2025, from https://medium.com/@zahmed333/xlstm-enhancing-long-short-term-memory-for-large-language-models-b9f0f07618aa

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631. https://doi.org/10.1145/3292500.3330701

Alberts, B., Heald, R., Johnson, A., Morgan, D., Raff, M., Roberts, K., & Walter, P. (2022). *Molecular biology of the cell: Seventh edition*. WW Norton & Company.

Arakelyan, G., Soghomonyan, G., & The Aim team. (2020, June). *Aim* (Version 3.9.3). https://doi.org/10.5281/zenodo.6536395

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., & Hochreiter, S. (2024). Xlstm: Extended long short-term memory. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, & C. Zhang (Eds.), *Advances in neural information processing systems 38: Annual conference on neural information processing systems 2024, neurips 2024, vancouver, bc, canada, december 10 - 15, 2024*. Retrieved July 24, 2024, from https://arxiv.org/abs/2405.04517

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *CoRR*, *abs/2004.05150*. Retrieved June 19, 2025, from https://arxiv.org/abs/2004.05150

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems*, *24*. Retrieved January 19, 2025, from https://proceedings.neurips.cc/paper_files/paper/2011/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html

Binet, T., Padiolleau-Lefèvre, S., Octave, S., Avalle, B., & Maffucci, I. (2023). Comparative Study of Single-stranded Oligonucleotides Secondary Structure Prediction Tools. *BMC Bioinformatics*, *24*(1), 422. https://doi.org/10.1186/s12859-023-05532-5

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP 2014, october 25-29, 2014, doha, qatar, A meeting of sigdat, a special interest group of the ACL* (pp. 1724–1734). ACL. https://doi.org/10.3115/V1/D14-1179

Choi, S. R., & Lee, M. (2023). Transformer Architecture and Attention Mechanisms in Genome Data Analysis: A Comprehensive Review. *Biology*, *12*(7), 1033. https://doi.org/10.3390/biology12071033

Chuai, G., Ma, H., Yan, J., Chen, M., Hong, N., Xue, D., Zhou, C., Zhu, C., Chen, K., Duan, B., Gu, F., Qu, S., Huang, D., Wei, J., & Liu, Q. (2018). DeepCRISPR: Optimized CRISPR guide RNA design by deep learning. *Genome Biology*, *19*(1), 80. https://doi.org/10.1186/s13059-018-1459-4

Danaee, P., Rouches, M., Wiley, M., Deng, D., Huang, L., & Hendrix, D. (2018). bpRNA: Large-scale automated annotation and analysis of RNA secondary structure. *Nucleic Acids Research*, *46*(11), 5381–5394. https://doi.org/10.1093/nar/gky285

Das, R., Wayment-Steele, H., Kim, D. S., Choe, C., Tunguz, B., Reade, W., & Demkin, M. (2020). Openvaccine: Covid-19 mrna vaccine degradation prediction [Kaggle]. https://kaggle.com/competitions/stanford-covid-vaccine

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computa-*

tional Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Eraslan, B., Wang, D., Gusic, M., Prokisch, H., Hallström, B. M., Uhlén, M., Asplund, A., Pontén, F., Wieland, T., Hopf, T., Hahne, H., Kuster, B., & Gagneur, J. (2019). Quantification and discovery of sequence determinants of protein-per-mRNA amount in 29 human tissues [Publisher: John Wiley & Sons, Ltd]. *Molecular Systems Biology*, *15*(2), e8513. https://doi.org/10.15252/msb.20188513

Facebook Research. (2022). fvcore package. Retrieved May 5, 2025, from https : / / github . com / facebookresearch/fvcore

Fagerberg, L., Hallström, B. M., Oksvold, P., Kampf, C., Djureinovic, D., Odeberg, J., Habuka, M., Tahmasebpoor, S., Danielsson, A., Edlund, K., Asplund, A., Sjöstedt, E., Lundberg, E., Szigyarto, C. A.-K., Skogs, M., Takanen, J. O., Berling, H., Tegel, H., Mulder, J., … Uhlén, M. (2014). Analysis of the Human Tissue-specific Expression by Genome-wide Integration of Transcriptomics and Antibody-based Proteomics * [Publisher: Elsevier]. *Molecular & Cellular Proteomics*, *13*(2), 397–406. https://doi.org/10.1074/mcp.M113.035600

Frankish, A., Carbonell-Sala, S., Diekhans, M., Jungreis, I., Loveland, J. E., Mudge, J. M., Sisu, C., Wright, J. C., Arnan, C., Barnes, I., Banerjee, A., Bennett, R., Berry, A., Bignell, A., Boix, C., Calvet, F., Cerdán-Vélez, D., Cunningham, F., Davidson, C., … Flicek, P. (2023). GENCODE: Reference annotation for the human and mouse genomes in 2023. *Nucleic Acids Research*, *51*(D1), D942–D949. https://doi.org/10.1093/nar/gkac1071

Franks, A., Airoldi, E., & Slavov, N. (2017). Post-transcriptional regulation across human tissues [Publisher: Public Library of Science]. *PLOS Computational Biology*, *13*(5), e1005535. https://doi.org/10.1371/journal.pcbi.1005535

Gildenblat, J. (2023). A python library for confidence intervals.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. Retrieved October 7, 2024, from http://www.deeplearningbook.org

Gould, N., Hendy, O., & Papamichail, D. (2014). Computational Tools and Algorithms for Designing Customized Synthetic Genes [Publisher: Frontiers]. *Frontiers in Bioengineering and Biotechnology*, *2*. https://doi.org/10.3389/fbioe.2014.00041

Grootendorst, M. (2025). A Visual Guide to Mamba and State Space Models. Retrieved April 15, 2025, from https://www.maartengrootendorst.com/blog/mamba

Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, *abs/2312.00752*. https://doi.org/10.48550/ARXIV.2312.00752

Guimaraes, J. C., Rocha, M., & Arkin, A. P. (2014). Transcript level and sequence determinants of protein abundance and noise in Escherichia coli. *Nucleic Acids Research*, *42*(8), 4791–4799. https://doi.org/10.1093/nar/gku126

Hanson, G., & Coller, J. (2018). Codon optimality, bias and usage in translation and mRNA decay [Publisher: Nature Publishing Group]. *Nature Reviews Molecular Cell Biology*, *19*(1), 20–30. https://doi.org/10.1038/nrm.2017.91

He, S., Gao, B., Sabnis, R., & Sun, Q. (2023). RNAdegformer: Accurate prediction of mRNA degradation at nucleotide resolution with deep learning. *Briefings in Bioinformatics*, *24*(1), bbac581. https://doi.org/10.1093/bib/bbac581

Hernandez-Alias, X., Benisty, H., Radusky, L. G., Serrano, L., & Schaefer, M. H. (2023). Using protein-per-mRNA differences among human tissues in codon optimization. *Genome Biology*, *24*(1), 34. https://doi.org/10.1186/s13059-023-02868-2

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/NECO.1997.9.8.1735

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Huang, L., Zhang, H., Deng, D., Zhao, K., Liu, K., Hendrix, D. A., & Mathews, D. H. (2019). LinearFold: Linear-time approximate RNA folding by 5'-to-3' dynamic programming and beam search. *Bioinformatics*, *35*(14), i295–i304. https://doi.org/10.1093/bioinformatics/btz375

Hutter, F., Hoos, H., & Leyton-Brown, K. (2014, January). An Efficient Approach for Assessing Hyperparameter Importance. In E. P. Xing & T. Jebara (Eds.), *Proceedings of the 31st international conference on machine learning* (pp. 754–762, Vol. 32). PMLR. Retrieved September 17, 2025, from https://proceedings.mlr.press/v32/hutter14.html

Hwang, H., Jeon, H., Yeo, N., & Baek, D. (2024). Big data and deep learning for RNA biology [Publisher: Nature Publishing Group]. *Experimental & Molecular Medicine*, *56*(6), 1293–1321. https://doi.org/10.1038/s12276-024-01243-w

*Islam, S., Elmekki, H., Elsebai, A., Bentahar, J., Drawel, N., Rjoub, G., & Pedrycz, W. (2024). A comprehensive survey on applications of transformers for deep learning tasks. *Expert Systems with Applications*, *241*, 122666. https://doi.org/10.1016/j.eswa.2023.122666

isle_of_gods. (2022, September). *Re: Early stopping in pytorch [answer to question "early stopping in pytorch"]* [Stack Overflow].

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning: With Applications in Python*. Springer International Publishing. https://doi.org/10.1007/978-3-031-38747-0

Ji, Y., Zhou, Z., Liu, H., & Davuluri, R. V. (2021). DNABERT: Pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, *37*(15), 2112–2120. https://doi.org/10.1093/bioinformatics/btab083

Jia, X., He, X., Huang, C., Li, J., Dong, Z., & Liu, K. (2024). Protein translation: Biological processes and therapeutic strategies for human diseases [Publisher: Nature Publishing Group]. *Signal Transduction and Targeted Therapy*, *9*(1), 1–17. https://doi.org/10.1038/s41392-024-01749-9

*Kevin P. Murphy. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press. http://probml.github.io/book1

Kim, Y.-A., Mousavi, K., Yazdi, A., Zwierzyna, M., Cardinali, M., Fox, D., Peel, T., Coller, J., Aggarwal, K., & Maruggi, G. (2024). Computational design of mRNA vaccines. *Vaccine*, *42*(7), 1831–1840. https://doi.org/10.1016/j.vaccine.2023.07.024

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. Retrieved September 18, 2025, from http://arxiv.org/abs/1412.6980

Kong, B., Kim, Y., Kim, E. H., Suk, J. S., & Yang, Y. (2023). mRNA: A promising platform for cancer immunotherapy. *Advanced Drug Delivery Reviews*, *199*, 114993. https://doi.org/10.1016/j.addr.2023.114993

Kudla, G., Lipinski, L., Caffin, F., Helwak, A., & Zylicz, M. (2006). High Guanine and Cytosine Content Increases mRNA Levels in Mammalian Cells [Publisher: Public Library of Science]. *PLOS Biology*, *4*(6), e180. https://doi.org/10.1371/journal.pbio.0040180

Lahtvee, P.-J., Sánchez, B. J., Smialowska, A., Kasvandik, S., Elsemman, I. E., Gatto, F., & Nielsen, J. (2017). Absolute Quantification of Protein and mRNA Abundances Demonstrate Variability in Gene-Specific Translation Efficiency in Yeast [Publisher: Elsevier]. *Cell Systems*, *4*(5), 495–504.e5. https://doi.org/10.1016/j.cels.2017.03.003

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition [Conference Name: Neural Computation]. *Neural Computation*, *1*(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

Leontis, N. B., & Westhof, E. (2001). Geometric nomenclature and classification of RNA base pairs. *RNA*, *7*(4), 499–512. https://doi.org/10.1017/S1355838201002515

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2019). *Mining of Massive Datasets*.

Li, J. J., Chew, G.-L., & Biggin, M. D. (2019). Quantitative principles of cis-translational control by general mRNA sequence features in eukaryotes. *Genome Biology*, *20*(1), 162. https://doi.org/10.1186/s13059-019-1761-9

Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. Retrieved September 18, 2025, from https://openreview.net/forum?id=Skq89Scxx

Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. Retrieved September 18, 2025, from https://openreview.net/forum?id=Bkg6RiCqY7

Micikevicius, P., Narang, S., Alben, J., Diamos, G. F., Elsen, E., García, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., & Wu, H. (2018). Mixed Precision Training. *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. Retrieved September 18, 2025, from https://openreview.net/forum?id=r1gs9JgRZ

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In Y. Bengio & Y. LeCun (Eds.), *1st international conference on learning representations, ICLR 2013, scottsdale, arizona, usa, may 2-4, 2013, workshop track proceedings*. Retrieved September 18, 2025, from http://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 26: 27th annual conference on neural information processing systems 2013. proceedings of a meeting held december 5-8, 2013, lake tahoe, nevada, united states* (pp. 3111–3119). Retrieved September 18, 2025, from https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html

Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

National Human Genome Research Institute. (2025). National human genome research institute website [Courtesy: National Human Genome Research Institute].

Nguyen, E., Poli, M., Faizi, M., Thomas, A. W., Wornow, M., Birch-Sykes, C., Massaroli, S., Patel, A., Rabideau, C. M., Bengio, Y., Ermon, S., Ré, C., & Baccus, S. (2023). Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information processing systems 36: Annual conference on neural information processing systems 2023, neurips 2023, new orleans, la, usa, december 10 - 16, 2023*. Retrieved September 18, 2025, from https://papers.neurips.cc/paper_files/paper/2023/file/86ab6927ee4ae9bde4247793c46797c7-Paper-Conference.pdf

Optuna Contributors. (2018a). Optuna optuna.importance.FanovaImportanceEvaluator documentation. Retrieved September 17, 2025, from https://optuna.readthedocs.io/en/stable/reference/generated/optuna.importance.FanovaImportanceEvaluator.html

Optuna Contributors. (2018b). Optuna optuna.pruners.MedianPruner documentation. Retrieved January 19, 2025, from https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.TPESampler.html

Optuna Contributors. (2018c). Optuna optuna.samplers.TPESampler documentation. Retrieved January 19, 2025, from https://optuna.readthedocs.io/en/stable/reference/generated/optuna.pruners.MedianPruner.html

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, *32*.

Retrieved January 14, 2025, from https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Penzar, D., Nogina, D., Noskova, E., Zinkevich, A., Meshcheryakov, G., Lando, A., Rafi, A. M., de Boer, C., & Kulakovskiy, I. V. (2023). LegNet: A best-in-class deep learning model for short DNA regulatory regions. *Bioinformatics*, *39*(8), btad457. https://doi.org/10.1093/bioinformatics/btad457

PyTorch Contributors. (2023a). AdamW Documentation. Retrieved January 27, 2025, from https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

PyTorch Contributors. (2023b). BCELoss Documentation. Retrieved January 27, 2025, from https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html

PyTorch Contributors. (2023c). PyTorch GRU documentation. Retrieved January 17, 2025, from https://pytorch.org/docs/stable/generated/torch.nn.GRU.html

PyTorch Contributors. (2023d). PyTorch LSTM documentation. Retrieved January 17, 2025, from https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html

PyTorch Contributors. (2023e). PyTorch torch.nn.TransformerEncoder documentation. Retrieved January 17, 2025, from https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html

PyTorch Contributors. (2023f). torch.nn.Embedding Documentation. Retrieved January 13, 2025, from https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html

PyTorch Contributors. (2023g). torch.nn.utils.rnn.pack_padded_sequence Documentation. Retrieved January 17, 2025, from https://pytorch.org/docs/stable/generated/torch.nn.utils.rnn.pack_padded_sequence.html

PyTorch Contributors. (2024a). Automatic Mixed Precision Documentation. Retrieved May 1, 2025, from https://pytorch.org/docs/stable/amp.html

PyTorch Contributors. (2024b). CosineAnnealingWarmRestarts Documentation. Retrieved February 3, 2025, from https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.CosineAnnealingWarmRestarts.html

Rafi, A. M., Nogina, D., Penzar, D., Lee, D., Lee, D., Kim, N., Kim, S., Kim, D., Shin, Y., Kwak, I.-Y., Meshcheryakov, G., Lando, A., Zinkevich, A., Kim, B.-C., Lee, J., Kang, T., Vaishnav, E. D., Yadollahpour, P., Kim, S., … de Boer, C. G. (2024). A community effort to optimize sequence-based deep learning models of gene regulation [Publisher: Nature Publishing Group]. *Nature Biotechnology*, 1–11. https://doi.org/10.1038/s41587-024-02414-w

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *CoRR*, *abs/1710.05941*. Retrieved January 16, 2025, from http://arxiv.org/abs/1710.05941

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors [Publisher: Nature Publishing Group]. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Schiff, Y., Kao, C., Gokaslan, A., Dao, T., Gu, A., & Kuleshov, V. (2024). Caduceus: Bi-directional equivariant long-range DNA sequence modeling. *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. Retrieved September 18, 2025, from https://openreview.net/forum?id=mk3A5IUdn8

Schlusser, N., González, A., Pandey, M., & Zavolan, M. (2024). Current limitations in predicting mRNA translation with deep learning models. *Genome Biology*, *25*(1), 227. https://doi.org/10.1186/s13059-024-03369-6

Schmidinger, N., Schneckenreiter, L., Seidl, P., Schimunek, J., Hoedt, P., Brandstetter, J., Mayr, A., Luukkonen, S., Hochreiter, S., & Klambauer, G. (2024). Bio-xlstm: Generative modeling, representation and in-context learning of biological and chemical sequences. *CoRR*, *abs/2411.04165*. https://doi.org/10.48550/ARXIV.2411.04165

scikit-learn Developers. (2023). scikit-learn sklearn.ensemble.RandomForestClassifier documentation. Retrieved January 18, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Shen, G., Liu, J., Yang, H., Xie, N., & Yang, Y. (2024). mRNA therapies: Pioneering a new era in rare genetic disease treatment. *Journal of Controlled Release*, *369*, 696–721. https://doi.org/10.1016/j.jconrel.2024.03.056

Sirugo, G., Williams, S. M., & Tishkoff, S. A. (2019). The Missing Diversity in Human Genetic Studies. *Cell*, *177*(1), 26–31. https://doi.org/10.1016/j.cell.2019.02.048

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. Retrieved January 16, 2025, from http://jmlr.org/papers/v15/srivastava14a.html

Sun, X., & Xu, W. (2014). Fast Implementation of DeLong's Algorithm for Comparing the Areas Under Correlated Receiver Operating Characteristic Curves. *IEEE Signal Processing Letters*, *21*(11), 1389–1393. https://doi.org/10.1109/LSP.2014.2337313

Tan, M., & Le, Q. V. (2021). Efficientnetv2: Smaller models and faster training. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event* (pp. 10096–10106, Vol. 139). PMLR. Retrieved April 15, 2025, from http://proceedings.mlr.press/v139/tan21a.html

*Trabelsi, A., Chaabane, M., & Ben-Hur, A. (2019). Comprehensive evaluation of deep learning architectures for prediction of DNA/RNA sequence binding specificities. *Bioinformatics*, *35*(14), i269–i277. https://doi.org/10.1093/bioinformatics/btz339

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. u., & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems*, *30*. Retrieved July 24, 2024, from https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th international conference on Machine learning*, 1096–1103. https://doi.org/10.1145/1390156.1390294

Vogel, C., de Sousa Abreu, R., Ko, D., Le, S.-Y., Shapiro, B. A., Burns, S. C., Sandhu, D., Boutz, D. R., Marcotte, E. M., & Penalva, L. O. (2010). Sequence signatures and mRNA concentration can explain two-thirds of protein abundance variation in a human cell line [Publisher: John Wiley & Sons, Ltd]. *Molecular Systems Biology*, *6*(1), 400. https://doi.org/10.1038/msb.2010.59

Wang, D., Eraslan, B., Wieland, T., Hallström, B., Hopf, T., Zolg, D. P., Zecha, J., Asplund, A., Li, L.-h., Meng, C., Frejno, M., Schmidt, T., Schnatbaum, K., Wilhelm, M., Ponten, F., Uhlen, M., Gagneur, J., Hahne, H., & Kuster, B. (2019). A deep proteome and transcriptome abundance atlas of 29 healthy human tissues [Publisher: John Wiley & Sons, Ltd]. *Molecular Systems Biology*, *15*(2), e8503. https://doi.org/10.15252/msb.20188503

Wang, N., Bian, J., Li, Y., Li, X., Mumtaz, S., Kong, L., & Xiong, H. (2024). Multi-purpose RNA language modelling with motif-aware pretraining and type-guided fine-tuning [Publisher: Nature Publishing Group]. *Nature Machine Intelligence*, *6*(5), 548–557. https://doi.org/10.1038/s42256-024-00836-4
Github: https://github.com/CatIIIIIIII/RNAErnie/.

Wayment-Steele, H. K., Kladwang, W., Watkins, A. M., Kim, D. S., Tunguz, B., Reade, W., Demkin, M., Romano, J., Wellington-Oguri, R., Nicol, J. J., Gao, J., Onodera, K., Fujikawa, K., Mao, H., Vandewiele, G., Tinti, M., Steenwinckel, B., Ito, T., Noumi, T., … Das, R. (2022). Deep learning models for predicting RNA degradation via dual crowdsourcing [Publisher: Nature Publishing Group]. *Nature Machine Intelligence*, *4*(12), 1174–1184. https://doi.org/10.1038/s42256-022-00571-8

Weng, L. (2018, June). Attention? Attention! [Section: posts]. Retrieved April 14, 2025, from https://lilianweng.github.io/posts/2018-06-24-attention/

Wolfinger, M. (2023, July). Mtw/GENCODE43: GENCODE v43 annotation data. Retrieved November 19, 2024, from https://github.com/mtw/GENCODE43/tree/main

Xue, W., Li, T., Gu, Y., Li, S., & Xia, N. (2023). Molecular engineering tools for the development of vaccines against infectious diseases: Current status and future directions [Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/14760584.2023.2227699]. *Expert Review of Vaccines*, *22*(1), 563–578. https://doi.org/10.1080/14760584.2023.2227699

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into deep learning* [https://D2L.ai]. Cambridge University Press.

Zheng, D., Persyn, L., Wang, J., Liu, Y., Montoya, F. U., Cenik, C., & Agarwal, V. (2025). Predicting the translation efficiency of messenger RNA in mammalian cells. *bioRxiv*, 2024.08.11.607362. https://doi.org/10.1101/2024.08.11.607362

Zhou, Z., Ji, Y., Li, W., Dutta, P., Davuluri, R. V., & Liu, H. (2024). DNABERT-2: efficient foundation model and benchmark for multi-species genomes. *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. Retrieved September 18, 2025, from https://openreview.net/forum?id=oMLQB4EZE1

*Zrimec, J., Buric, F., Kokina, M., Garcia, V., & Zelezniak, A. (2021). Learning the Regulatory Code of Gene Expression [Publisher: Frontiers]. *Frontiers in Molecular Biosciences*, *8*. https://doi.org/10.3389/fmolb.2021.673363

Acronyms

ANN Artificial Neural Network

AUC Area Under the Curve

CDS Coding DNA sequence

CNN Convolutional Neural Network

DL Deep Learning

GRU Gated Recurrent Unit

HPA Human Protein Atlas

LSTM Long Short-Term Memory

ML Machine learning

MLP Multilayer Perceptron

mRNA messenger RNA

NLP Natural Language Processing

PTR Protein-to-mRNA

ReLU Rectified Linear Unit

RFC Random Forest Classifier

RNN Recurrent Neural Network

S4 Structured State Space Model

SSM State Space Model

UTR Untranslated region

xLSTM extended LSTM

Appendix

A Detailed Dataset Composition

Class distribution per tissue in the codon- and nucleotide-encoded datasets. The test set has been aligned to contain identical sequences to allow for direct comparison of the models.

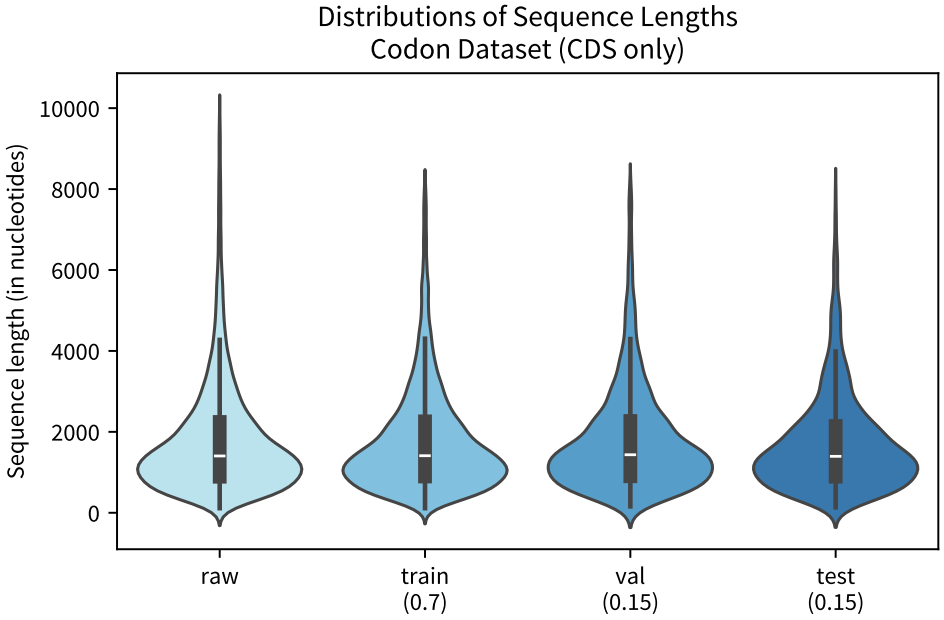
Tissue	PTR Class	Codon-encoded			Nucleotide-encoded		
		Train	Val	Test*	Train	Val	Test*
Adrenal	low	116	32	24	114	31	24
	high	149	44	23	144	41	23
Appendices	low	418	86	89	406	84	89
	high	194	37	43	181	36	43
Brain	low	204	49	38	196	47	38
	high	523	124	81	501	119	81
Colon	low	207	54	44	200	53	44
	high	199	33	41	191	31	41
Duodenum	low	161	32	34	157	32	34
	high	144	30	25	141	27	25
Uterus	low	92	24	24	87	24	24
	high	136	29	29	130	28	29
Esophagus	low	258	70	59	249	67	59
	high	158	29	31	153	29	31
Fallopiantube	low	139	37	27	136	35	27
	high	224	40	62	219	40	62
Fat	low	510	99	108	479	96	108
	high	194	29	38	186	27	38
Gallbladder	low	177	36	44	169	36	44
	high	140	18	25	132	18	25
Heart	low	402	64	72	391	64	72
	high	253	59	51	238	56	51
Kidney	low	274	71	66	259	66	66
	high	295	61	62	293	60	62
Liver	low	324	57	78	305	53	78
	high	206	51	40	194	50	40
Lung	low	303	63	55	292	59	55
	high	552	111	103	521	108	103
Lymphnode	low	186	23	25	176	23	25
	high	177	46	35	174	46	35
Ovary	low	369	82	63	354	78	63
	high	308	72	68	301	70	68
Pancreas	low	484	119	97	463	116	97
	high	346	73	71	333	70	71
Placenta	low	194	32	30	185	32	30
	high	210	59	52	196	55	52
Prostate	low	202	45	51	200	43	51
	high	196	46	31	181	43	31
Rectum	low	114	32	25	109	30	25
	high	166	41	29	163	38	29

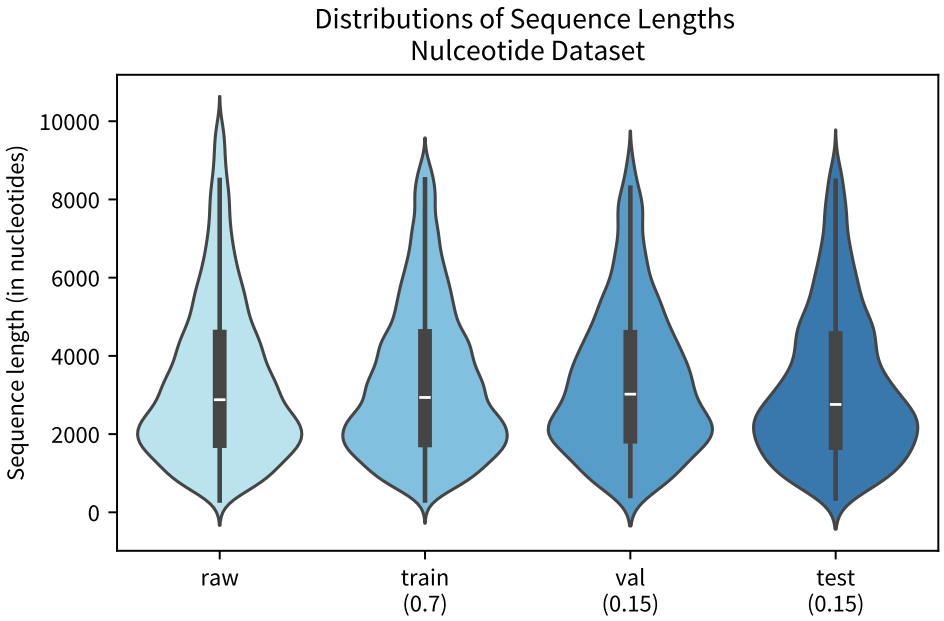
Tissue	PTR Class	Codon-encoded			Nucleotide-encoded		
		Train	Val	Test*	Train	Val	Test*
Salivarygland	low	244	45	33	232	44	33
	high	218	50	45	209	47	45
Smallintestine	low	113	21	29	109	20	29
	high	181	30	28	174	28	28
Smoothmuscle	low	197	42	35	195	39	35
	high	223	50	60	210	49	60
Spleen	low	133	42	37	129	41	37
	high	152	30	28	151	28	28
Stomach	low	105	25	21	103	24	21
	high	135	41	34	126	40	34
Testis	low	233	37	59	223	36	59
	high	383	86	94	373	83	94
Thyroid	low	181	49	40	166	45	40
	high	363	73	73	353	72	73
Tonsil	low	220	61	43	214	58	43
	high	230	45	50	223	45	50
Urinary Bladder	low	248	47	40	237	47	40
	high	133	38	31	129	36	31
Sum	low	6,808	1,476	1,390	6,535	1,423	1,390
	high	6,788	1,475	1,383	6,520	1,420	1,383
Total		**13,596**	**2,951**	**2,773**	**13,055**	**2,843**	**2,773**

B Sequence Length Distributions

The longest mRNA sequence in the dataset consists of 109,224 nucleotides. For better visual comparison of the raw data distributions, we do not show sequences with more than 10,000 nucleotides. 72 sequence-tissue tuples were hence not displayed from the codon encoded dataset and 392 from nucleotide dataset.

The plots illustrate the sequence length distribution across sequence-tissue pairs in the training data. These distributions represent sequence-tissue pairs rather than unique mRNA sequence lengths as this is the final model input and in edge cases, sequences might also appear only once or even three times.





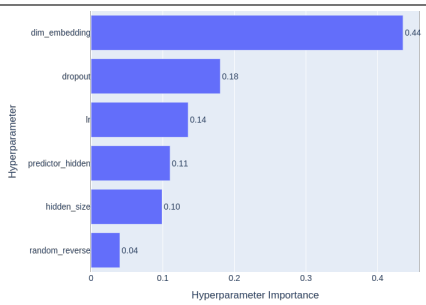
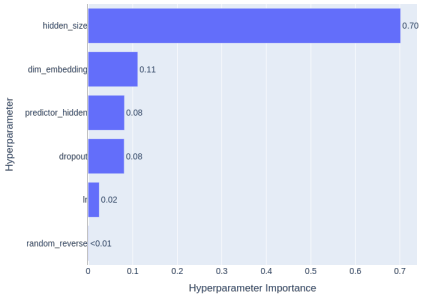
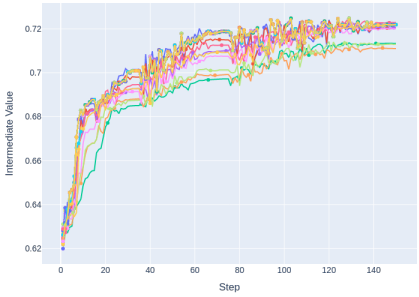
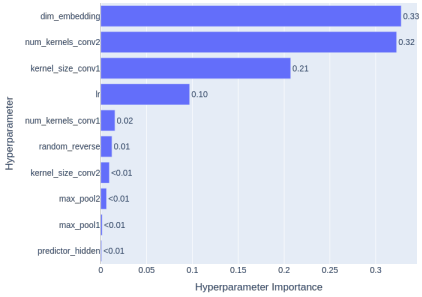
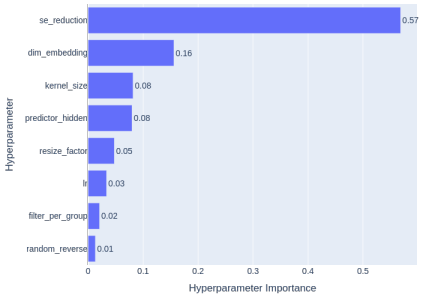
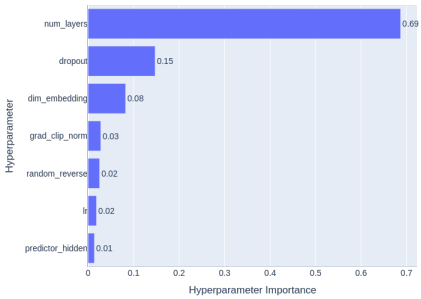
C Benchmarking Hyperparameter Tuning Analysis

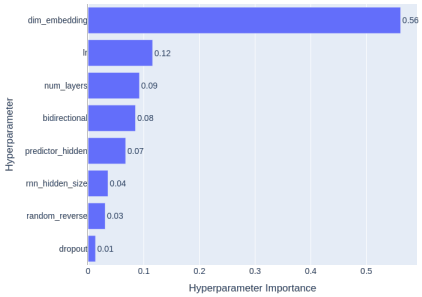
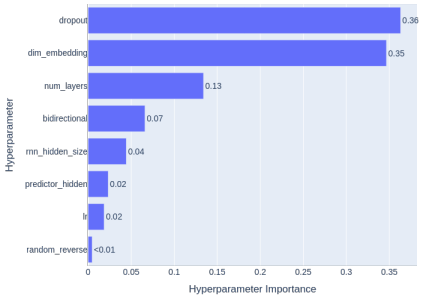
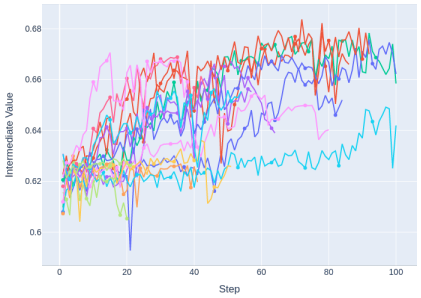
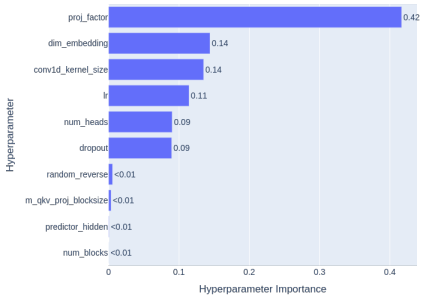
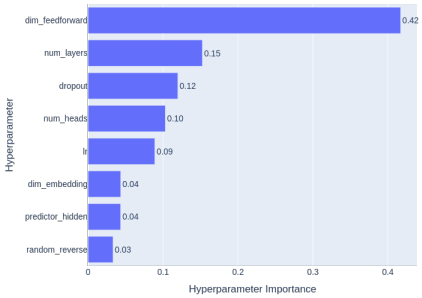
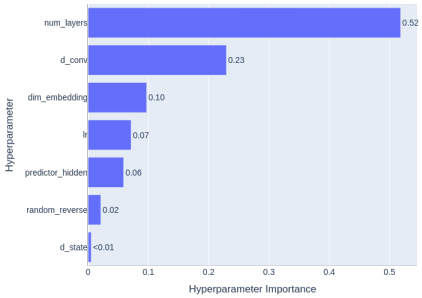
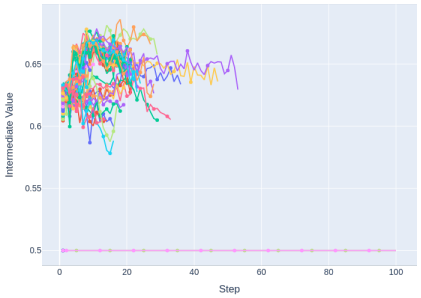
Hyperparameter Search Spaces

Parameter search spaces for all benchmarked models are listed, with the first row containing shared parameters: codon embedding dimension (`embed_dim`), predictor hidden dimension (`pred_dim`), learning rate (`lr`), sequence reversal flag (`seq_rev`), and model-specific hyperparameters. The `lr` and `grad_clip_norm` are defined as ranges optimized using Optuna (Akiba et al., 2019). For the xLSTM architecture, configurations with `num_blocks` set to 5 or 6 use only sLSTM blocks, while a value of 9 uses exclusively mLSTM blocks, in accordance with the DNA-xLSTM variants described in Schmidinger et al. (2024). The optimal parameters for each model are reported in Table 3.

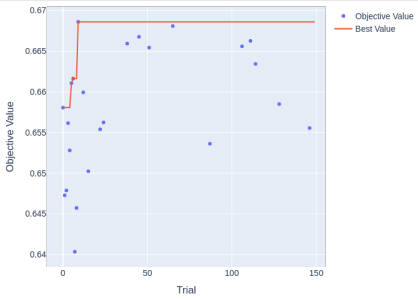
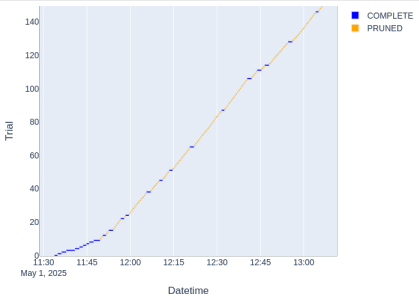
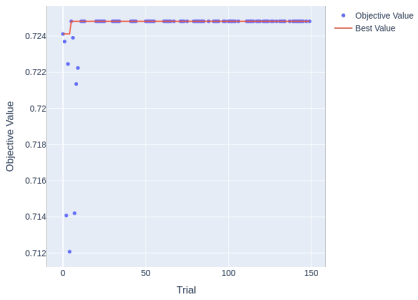
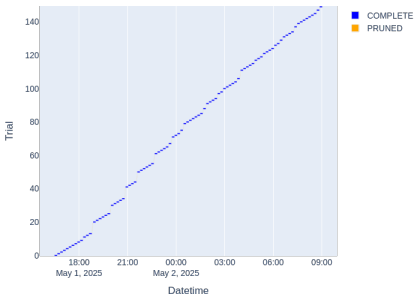
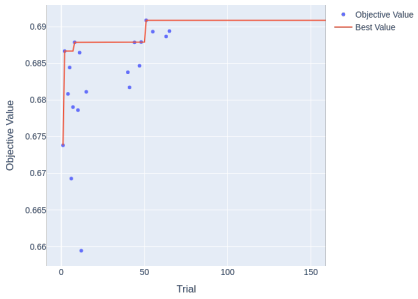
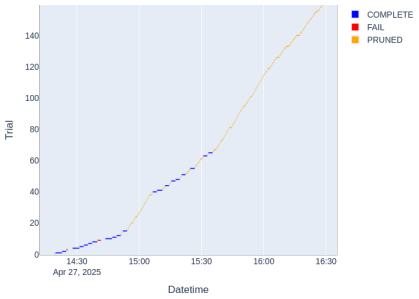
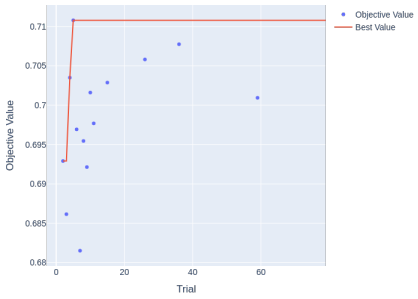
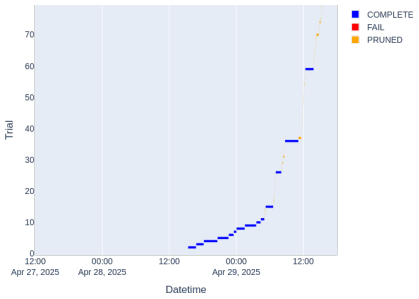
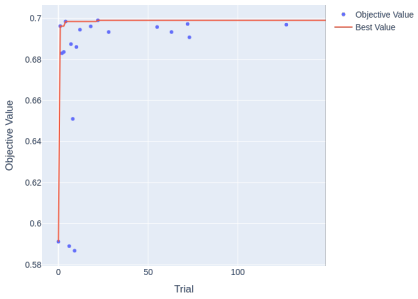
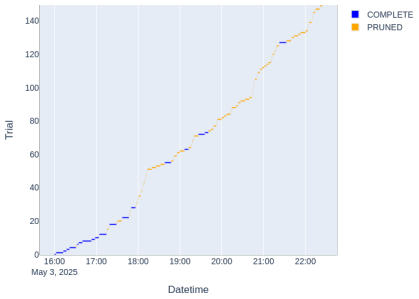
Model	Parameter	Search Range
General Parameters (all models)	`embed_dim`	{16, 32, 64, 128}
	`seq_rev`	{True, False}
	`pred_hid`	{32, 64, 128}
	`lr` (log)	[1e-5, 1e-2]
MLP (baseline & frequency)	`hidden_size`	{64, 128, 256, 512}
	`dropout`	{0.0, 0.1, 0.2, 0.4}
CNN	`num_kernels_conv1`	{32, 64, 128, 256}
	`num_kernels_conv2`	{8, 16, 32, 64}
	`kernel_size_conv1`	{3, 6, 9, 15}
	`kernel_size_conv2`	{3, 9, 15}
	`max_pool1`	{10, 20, 30}
	`max_pool2`	{3, 5, 10}
LegNet	`kernel_size`	{5, 7, 9, 11, 13}
	`resize_factor`	{2, 4, 6, 8}
	`se_reduction`	{2, 4, 8}
	`filter_per_group`	{1, 2, 4}
RiboNN	`num_layers`	{1, 2, 4, 8}
	`grad_clip_norm` (log)	[0.1, 5.0]
	`dropout`	{0.0, 0.1, 0.2, 0.4}
LSTM	`num_layers`	{1, 2, 4}
	`rnn_hidden_size`	{128, 256, 512, 1024}
	`bidirectional`	{True, False}
	`dropout`	{0.0, 0.1, 0.2, 0.4}
GRU	`num_layers`	{1, 2, 4}
	`rnn_hidden_size`	{128, 256, 512, 1024}
	`bidirectional`	{True, False}
	`dropout`	{0.0, 0.1, 0.2, 0.4}
xLSTM	`num_blocks`	{5, 6, 9}
	`conv1d_kernel_size`	{4, 6, 9}
	`m_qkv_proj_blocksize`	{2, 4, 8}
	`num_heads`	{2, 4, 8}
	`proj_factor`	{1, 1.25, 2}
	`dropout`	{0.0, 0.1, 0.2, 0.4}
Transformer	`num_layers`	{4, 6, 8, 12}
	`dim_feedforward`	{128, 256, 512}
	`num_heads`	{2, 4, 8, 16}
	`dropout`	{0.0, 0.1, 0.2, 0.4}
Mamba	`num_layers`	{1, 2, 4, 8}
	`d_state`	{8, 16, 32, 64}
	`d_conv`	{2, 3, 4}

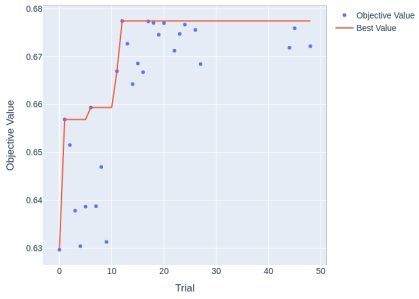
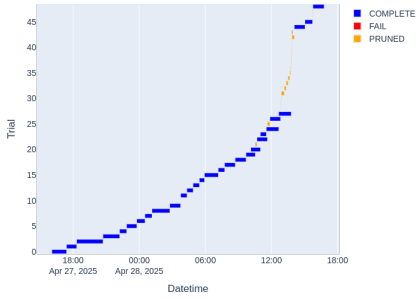
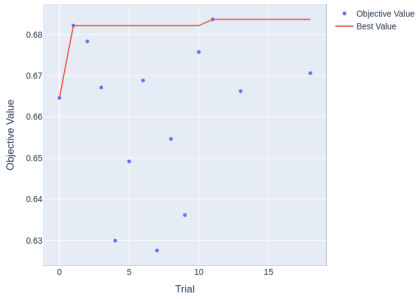
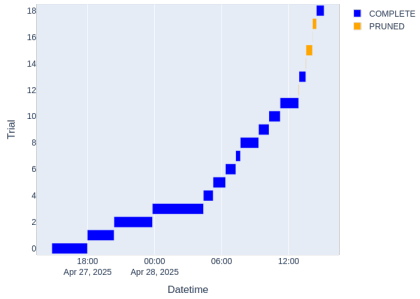
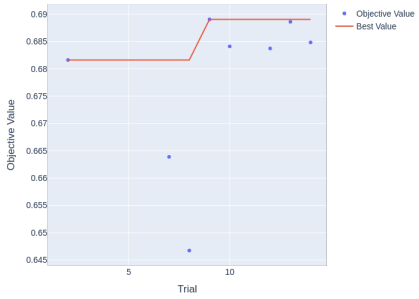
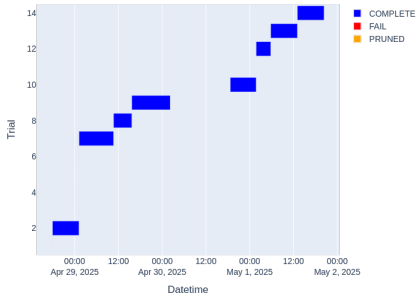
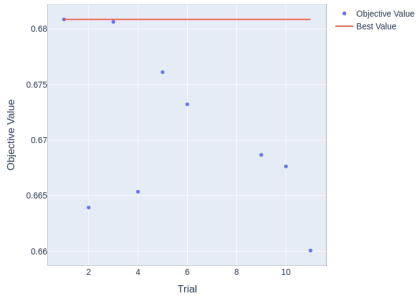
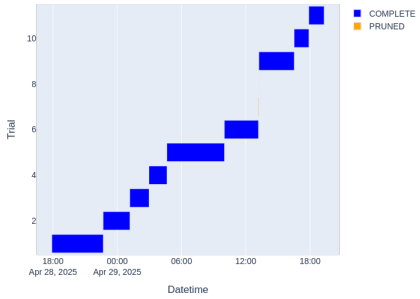
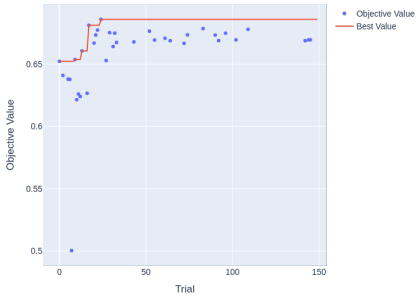
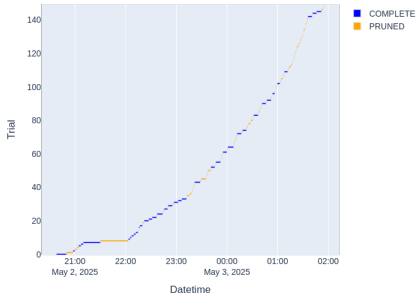
Hyperparameter Importance & Training Curves

Model	Parameter Importance	Training Curves (AUC)
MLP (baseline)		
MLP (freq)		
CNN		
LegNet		
RiboNN		

Model	Parameter Importance	Training Curves (AUC)
LSTM		
GRU		
xLSTM		
Transformer		
Mamba		

Optimization History & Trial Timeline

Model	Optimization History	Trial Timeline
MLP (baseline)		
MLP (freq)		
CNN		
LegNet		
RiboNN		

Model	Optimization History	Trial Timeline
LSTM		
GRU		
xLSTM		
Transformer		
Mamba		

D Hardware and Software Configuration of Experimental Servers

For the exact package versions used in the experiments, please refer to the `environment.yml` file in the GitHub repository (see below). The environment was created using the `mamba` package manager[21], which is a faster alternative to `conda`. The environment file contains all the necessary dependencies and their versions to replicate the setup.

For the experiments, we had access to two identical servers with shared filesystem with the following specifications:

Component	Specification
CPU Model	Intel Xeon Silver 4214R @ 2.40GHz
CPU min MHz	1000 MHz
CPU max MHz	3500 MHz
CPU Cores	24 cores per socket, 2 sockets (48 threads total)
L1 Cache	768 KiB L1d + 768 KiB L1i (per 24 instances)
L2 Cache	24 MiB (24 instances)
L3 Cache	33 MiB (2 instances)
Operating System	Ubuntu Linux 6.8.0-54-generic (x86_64)
GPU Model	2× NVIDIA A100-PCIE-40GB
GPU Memory	40960 MiB per GPU
CUDA Version	12.2
NVIDIA GPU Driver Version	535.216.03
Python Version	3.10.14
PyTorch Version	2.4.1

GitHub source code: https://github.com/f-krause/master-thesis

Environment file: https://github.com/f-krause/master-thesis/blob/main/environment_files/environment_linux_py3.10.yml

[21] https://mamba.readthedocs.io/en/latest/index.html

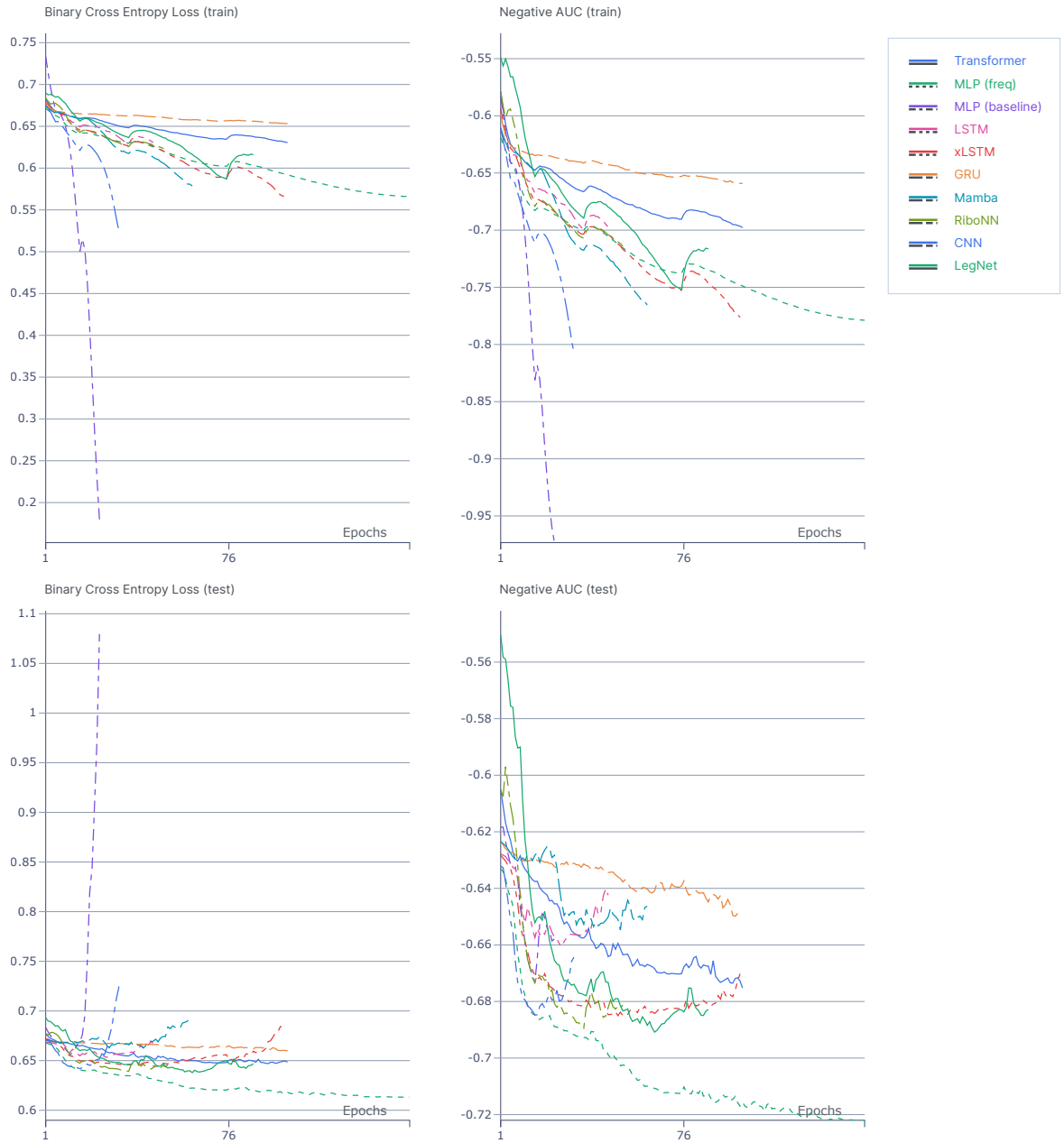
E Benchmarking Test Set Metrics

Below we report diverse metrics for the training, validation, and test sets. The validation metric corresponds to the best AUC score achieved on the validation set during hyperparameter tuning with Optuna. The training and test scores are computed using the model trained with the optimal hyperparameters on the combined training and validation sets and evaluated on the test set. A darker shade of green indicates better performance.

	train AUC	val AUC (tuning)	test AUC	test F1 Score	test Precision	test Recall
MLP (freq)	0.7773	0.7248	0.7224	0.6539	0.6775	0.6320
LegNet	0.7304	0.7108	0.6930	0.6404	0.6454	0.6356
RiboNN	0.7081	0.6991	0.6906	0.6292	0.6391	0.6197
xLSTM	0.7044	0.6891	0.6875	0.6231	0.6398	0.6074
CNN	0.7176	0.6909	0.6868	0.6220	0.6447	0.6009
PTRnet	0.7195		0.6860	0.6316	0.6432	0.6204
Transformer	0.6885	0.6809	0.6794	0.6405	0.6266	0.6551
MLP (baseline)	0.8744	0.6686	0.6775	0.6143	0.6398	0.5907
RFC (freq)		0.6634	0.6682	0.6132	0.6097	0.6168
LSTM	0.6767	0.6774	0.6624	0.6038	0.6288	0.5806
Mamba	0.7399	0.6860	0.6603	0.6143	0.6299	0.5994
GRU	0.6604	0.6837	0.6550	0.6059	0.6242	0.5886

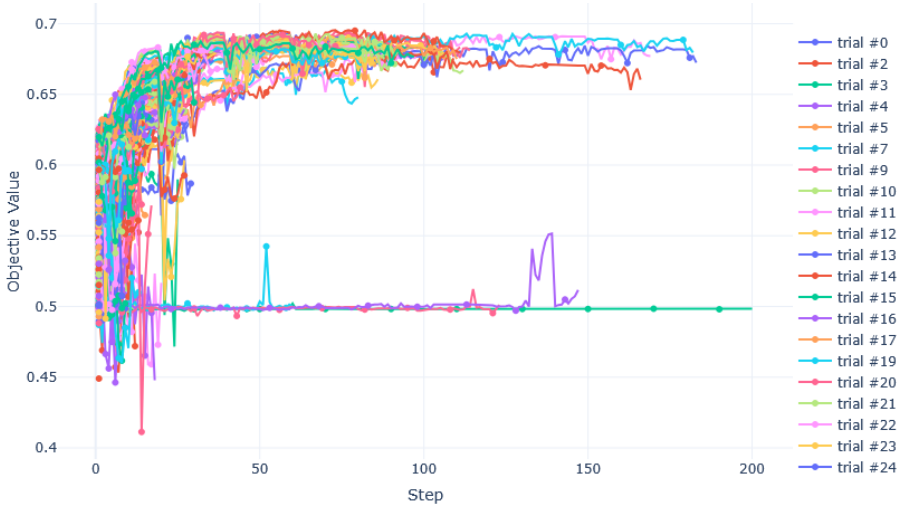
F Benchmarking Train and Test Set Training Curves

The following plots show the training and test loss curves (Binary Cross Entropy Loss and negative AUC) for all benchmarked models when trained on the training and validation set and evaluated on the test set. The visualization is obtained from the tracking software Aim (Arakelyan et al., 2020). The curves are smoothed for better visualization and comparability.

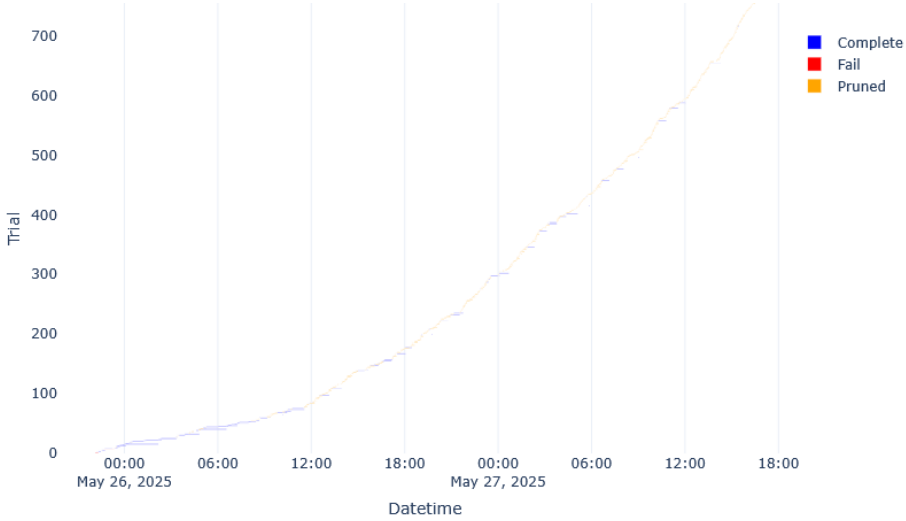


G Hyperparameter Tuning of PTRnet

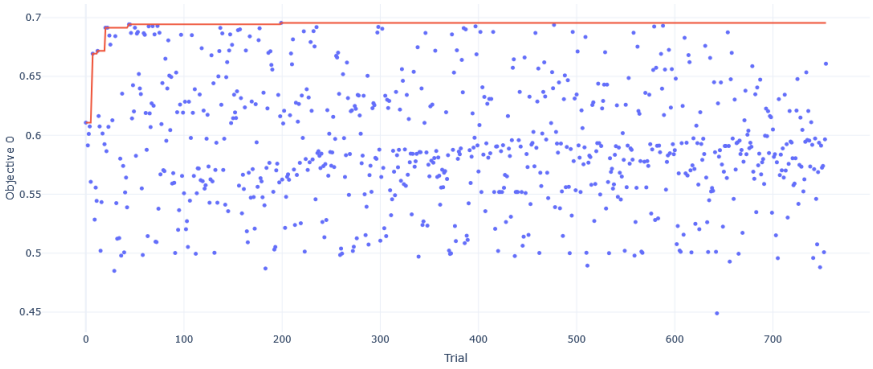
Hyperparameter tuning visualizations for PTRnet.



(a) Training Curves (AUC)



(b) Trial Timeline



(c) Optimization History

H PTRnet Ablation Study Train and Test Set Training Curves

The following plots show the training and test loss curves (Cross Entropy Loss and negative AUC) for the PTRnet ablation study models when trained on the training and validation set and evaluated on the test set. The visualization is obtained from the tracking software Aim (Arakelyan et al., 2020). The curves are smoothed for better visualization and comparability.