Graph Neural Network with Denoising Autoencoder for Predicting mRNA Vaccine Degradation

Mateus Aleixo
University of Aveiro
mateus.aleixo@ua.pt
124256

Pedro Batista
University of Aveiro
pedro.peres.batista@ua.pt
124355

***Abstract*—This study addresses the prediction of degradation rates of COVID-19 mRNA vaccine molecules by combining Graph Neural Networks (GNNs) with a denoising autoencoder for enhanced feature extraction. We analyze sequence, secondary structure, and chemical probing data to predict five degradation metrics. Our model demonstrates improvements over iterative baselines and achieves competitive performance relative to top Kaggle solutions. Key contributions include a domain-informed GNN architecture, structured pretraining, and comparative analysis with state-of-the-art solutions.**

***Index Terms*—mRNA vaccine, degradation prediction, graph neural network, denoising autoencoder, regression, machine learning**

I. Introduction

The degradation of mRNA vaccines is critical to their stability, efficacy, and storage requirements. Precise prediction of degradation rates under varying conditions (e.g., pH, temperature) is vital for logistics and formulation design. This regression task involves multimodal input data—RNA sequence, secondary structure, and chemical probing data (e.g., SHAPE, DMS reactivities).

mRNA vaccines, such as those developed during the COVID-19 pandemic, rely on the integrity of RNA strands to trigger an effective immune response. However, RNA is inherently unstable, and degradation can significantly reduce vaccine potency. This presents a pressing need for computational models that can anticipate degradation patterns across different chemical environments. Traditional experimental methods are costly and time-consuming; therefore, data-driven approaches can significantly accelerate the development pipeline and ensure robustness under diverse conditions.

The "Stanford COVID-19 mRNA Vaccine Degradation Prediction" competition on Kaggle posed a complex multi-output regression challenge, in which the goal was to predict five degradation indicators from RNA sequence and structure. The difficulty arises from the noisy nature of experimental signals and the intricacies of RNA folding, which can influence reactivity and degradation pathways in non-trivial ways.

Our approach integrates a Graph Neural Network (GNN) to model base-pair interactions alongside a denoising autoencoder that learns robust latent representations from noisy inputs. Unlike traditional models that treat RNA sequences as flat or linear data, the graph-based framework captures spatial relationships implied by secondary structure. At the same time, the autoencoder provides resilience against experimental noise and limited training data. A regression head predicts five target variables: *reactivity*, *deg_Mg_pH10*, *deg_Mg_50C*, *deg_pH10*, and *deg_50C*.

In this paper, we describe the full modelling pipeline, starting from data preprocessing and exploratory analysis, through model design and iterative optimisation, to final evaluation and comparison with top-ranked solutions in the competition. The proposed method achieves strong predictive performance while offering a computationally efficient architecture.

II. Related Work

Graph Neural Networks (GNNs) have become increasingly prominent in computational biology due to their natural ability to model structured data such as molecular graphs, protein interactions, and RNA secondary structures. In the context of RNA analysis, GNNs are particularly well-suited because they can incorporate spatial and relational information between nucleotides, such as base pairing and adjacency, which are crucial for capturing functional and degradation-related properties of RNA molecules [1].

In recent years, several studies have explored the application of GNNs to RNA-related problems. These include RNA structure prediction, functional site identification, and molecular property estimation. For instance, graph convolutional architectures have been applied to predict RNA folding pathways and thermodynamic stability by modelling the molecule as a graph where nodes represent nucleotides and edges capture structural interactions. Message-passing neural networks (MPNNs) and Graph Attention Networks (GATs) have shown enhanced performance by learning context-sensitive interactions that adapt to local structural environments.

Denoising autoencoders (DAEs) are another class of models that have demonstrated effectiveness in biomedical domains, particularly when data is scarce or noisy. DAEs are designed to reconstruct input data from corrupted versions, thereby encouraging the model to learn more generalizable and robust latent representations [2]. In genomics and transcriptomics, DAEs have been used to recover missing expression values, reduce the dimensionality of single-cell data, and enhance classification performance in noisy datasets.

Combining GNNs with autoencoders provides a powerful hybrid approach, where structural inductive biases from

graphs are complemented by the robustness of unsupervised pretraining. This strategy has been employed in several domains, including drug discovery and protein–RNA interaction prediction, but remains underexplored in RNA degradation modelling.

Within the scope of the OpenVaccine Kaggle competition, several top-ranked teams proposed architectures that leverage GNNs in combination with carefully crafted features and ensemble methods. The 1st-place solution [3] implemented a multi-scale graph model incorporating structural and sequence-based attention mechanisms, along with extensive data augmentation and blending of multiple model variants. The 2nd-place team [4] introduced a hybrid encoder combining convolutional layers with GNN-based representations, while also employing multi-head attention and residual connections to enhance learning across different sequence regions. The 3rd-place solution [5] emphasised advanced feature engineering, including loop-type encoding, pairwise distance embeddings, and extended base-pair probability statistics. Most of these solutions benefited significantly from ensembling strategies, data augmentation, and careful tuning of learning schedules.

In contrast to these, our approach emphasises architectural simplicity. By integrating a denoising autoencoder with a GNN backbone, we focus on extracting stable features from noisy chemical probing data while preserving the structural relationships critical to RNA behaviour. We avoid excessive ensembling in favour of interpretability and computational efficiency, and our design choices are informed by extensive exploratory data analysis rather than purely empirical optimisation.

In summary, while there is a rich body of literature on GNNs and DAEs in biological modelling, their combined application to RNA degradation prediction remains relatively novel. Our work contributes to this space by demonstrating that such a hybrid model can achieve competitive results with state-of-the-art methods while maintaining architectural transparency and robustness.

III. Data Description and Preprocessing

A. Dataset Structure

The competition dataset comprises three main files: `train.json`, `test.json`, and `sample_submission.csv`. Each sample in the training set includes:

- A nucleotide **sequence** of fixed length (107), representing RNA molecules.
- A **secondary structure** string in dot-bracket notation (`.()`, indicating unpaired and base-paired nucleotides).
- A **predicted loop type** string, denoting the structural region each nucleotide belongs to (e.g., stem, hairpin, internal loop).
- A matrix of **base-pair probabilities** (bpp), indicating pairing confidence between nucleotides.
- Five numerical **target variables**: `reactivity`, `deg_Mg_pH10`, `deg_Mg_50C`, `deg_pH10`, and `deg_50C`.

Test sequences vary in length (107 or 130 nt), influencing padding and batch preparation.

B. Input Features

For modelling, we extracted the following features:

- One-hot encoding of nucleotide identities (A, C, G, U).
- Dot-bracket structure parsed into base-pair edges and used to build an adjacency matrix.
- Loop-type indicators encoded categorically and embedded.
- Base-pair probabilities aggregated into per-node statistics (e.g., total pairing weight per nucleotide).

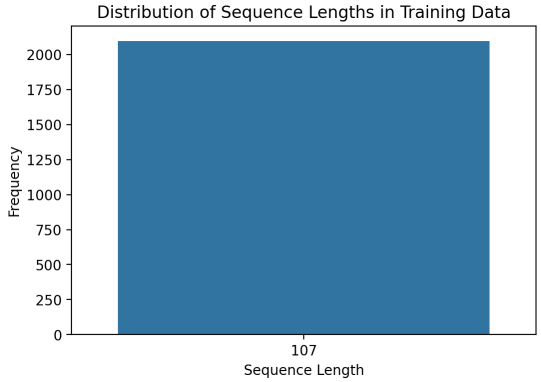

Fig. 1. Sequence length distribution in training set (fixed at 107 nt).

C. Exploratory Data Analysis and Preprocessing

Signal-to-noise ratio (SNR) was a key metric provided in the training set. As shown in Fig. 2, the distribution is right-skewed. Following recommendations, we filtered out samples with `signal_to_noise` ≤ 1 to retain high-quality data for modelling.

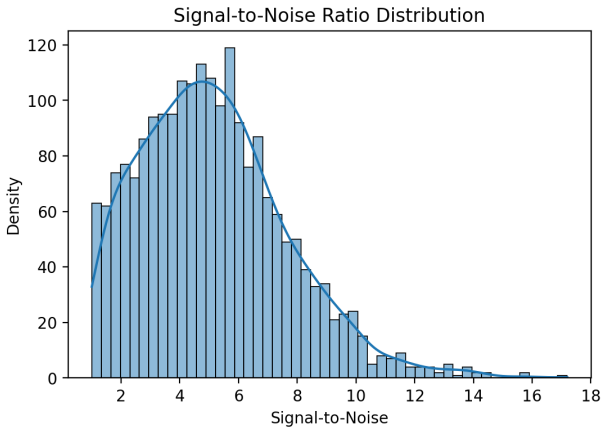

Fig. 2. Signal-to-noise ratio distribution in training data.

Nucleotide frequencies (Fig. 3) show adenine (A) is the most common base, followed by guanine (G), cytosine (C), and uracil (U). These imbalances may influence structural stability and degradation behaviour.

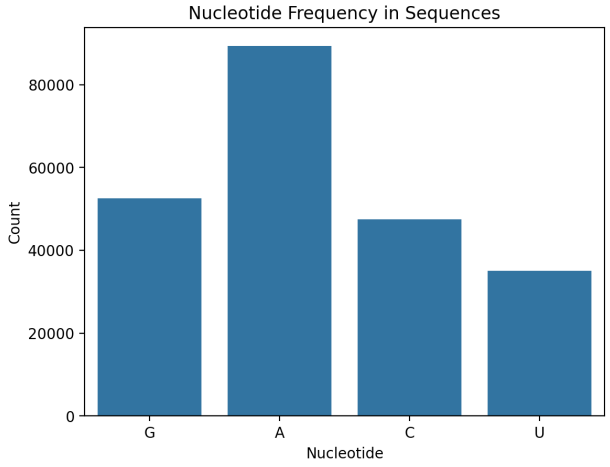

Fig. 3. Nucleotide distribution across all training sequences.

RNA structure symbols (Fig. 4) are dominated by unpaired dots (.), suggesting a majority of flexible or loop regions in the input sequences. The dot-bracket symbols were later converted into graph connectivity information for our GNN.

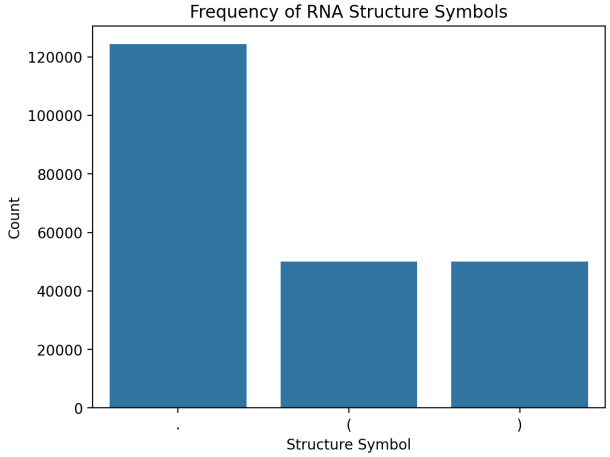

Fig. 4. Frequency of dot-bracket structure symbols.

Loop type distribution (Fig. 5) further confirms that most nucleotides reside in stems (S) and external regions (E). This structure was used to define categorical embeddings.

Target distribution analysis (Fig. 6) reveals that all five regression targets are strongly right-skewed and exhibit sharp peaks near zero.

Correlation matrix (Fig. 7) among targets shows high linear dependencies, particularly between `deg_Mg_50C`, `deg_50C`, and `deg_pH10`, motivating multi-output regression modeling.

Finally, analysis of **reactivity by position** (Fig. 8) reveals elevated degradation signals near the 5th end of sequences,

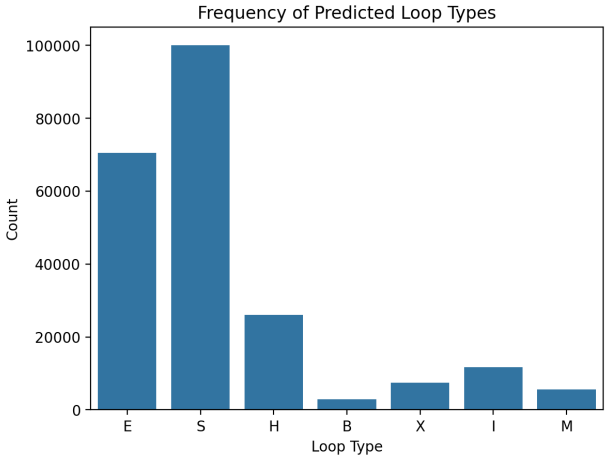

Fig. 5. Frequency of predicted loop types in RNA sequences.

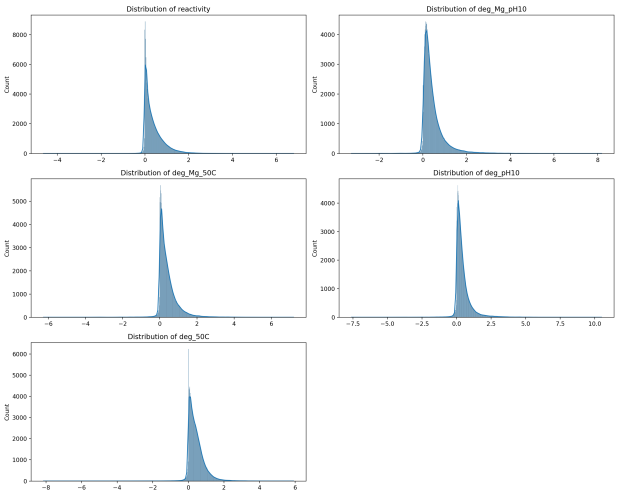

Fig. 6. Distributions of target degradation metrics.

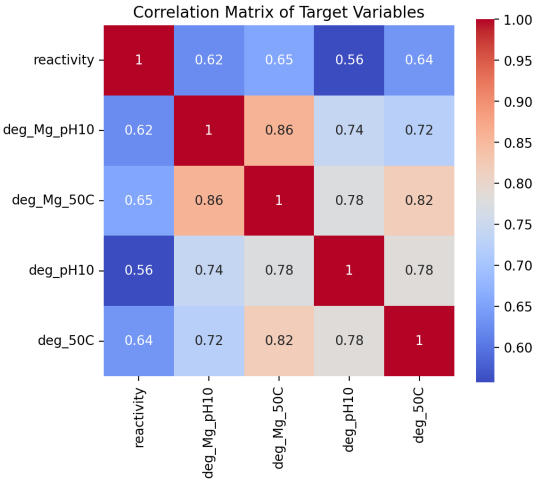

Fig. 7. Correlation matrix between regression targets.

with a decreasing trend toward the middle. This positional bias influenced the inclusion of positional embeddings in later model versions.

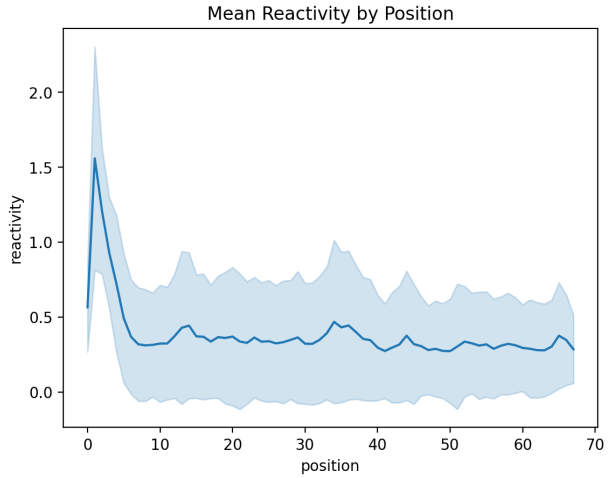

Fig. 8. Mean reactivity values by nucleotide position.

Reactivity vs loop type (Fig. 9) shows that different structural contexts exhibit different degradation tendencies. Stems (S) and external regions (E) tend to show lower variance in reactivity, whereas hairpins and bulges exhibit higher dispersion.

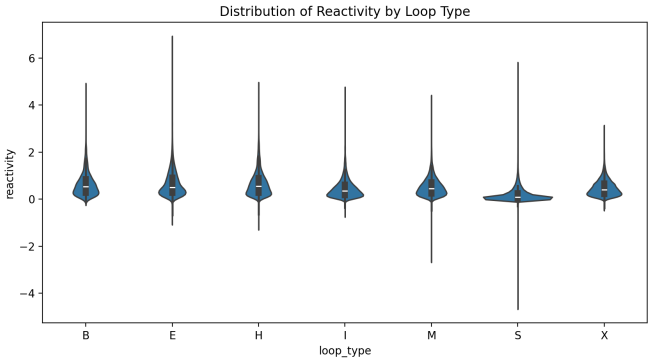

Fig. 9. Distribution of reactivity values across loop types.

These insights guided feature engineering decisions such as creating the `bpp_paired_count` variable and using positional/contextual encodings. All raw inputs were standardised and padded to uniform shapes when applicable.

IV. Methodology

A. Model Architecture

Our architecture is designed to exploit both the sequential and structural properties of RNA. The model integrates a Graph Neural Network (GNN) to capture structural relationships between nucleotides, a denoising autoencoder to learn robust representations, and a bidirectional recurrent regression head for final predictions. The pipeline is fully end-to-end and trained with multi-output supervision.

1) Graph Neural Network: We represent each RNA sequence as a graph $G = (V, E)$, where each node $v_i \in V$ corresponds to a nucleotide, and edges $e_{ij} \in E$ connect:

- Adjacent positions in the primary sequence (backbone connections).
- Base-paired nucleotides based on the dot-bracket structure.

Each node is initialized with a concatenation of the one-hot encoded nucleotide identity, positional embeddings, loop-type embedding, and local base-pair probability features.

The GNN comprises multiple stacked Graph Convolutional Layers (GCN) followed by Graph Attention Layers (GAT). The GCN aggregates neighboring node features using normalized adjacency matrices, while the GAT assigns attention weights to neighbors, allowing the model to learn which structural interactions are more informative for degradation. These operations generate contextual node embeddings, capturing local and global RNA topology.

2) Denoising Autoencoder: Before supervised training, we pretrain the encoder using a denoising autoencoder. During pretraining, random noise is injected into input node features—via masking, dropout, or additive Gaussian noise. The corrupted inputs are passed through the GNN encoder, which aims to reconstruct the original (clean) features through a decoder module.

The autoencoder is trained with a mean squared error (MSE) reconstruction loss. The intuition is that robust internal representations help the downstream regression model to generalize better to noisy or structurally ambiguous samples. This pretraining phase is unsupervised and operates only on the input node-level data, ignoring target variables.

3) Regression Head: After pretraining, the encoder's output embeddings are fed into a regression module composed of:

- A Bidirectional LSTM layer with 128 units, which captures contextual dependencies across the sequence in both directions.
- Attention pooling mechanism that weights hidden states and summarizes the sequence embedding.
- Dense layers with `LayerNormalization`, ReLU activations, and `Dropout` (rate = 0.3) for regularization.
- A final fully-connected output layer of size 5 for predicting the five degradation targets.

The model is optimized using the Mean Columnwise Root Mean Squared Error (MCRMSE), defined as:

$$\text{MCRMSE} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_{i,t} - y_{i,t})^2}$$

where $T = 5$ is the number of targets, N the number of samples, $\hat{y}_{i,t}$ the predicted value, and $y_{i,t}$ the ground truth.

B. Feature Engineering

Our EDA informed several feature engineering steps:

- **`bpp_paired_count`**: introduced in v3, this feature counts the number of nucleotides paired with each node using a threshold on the base-pair probability matrix.

- **Sequence windowing**: local statistical features (e.g., average reactivity, nucleotide composition) were computed using sliding windows and appended to the node features.
- **Positional embeddings**: a sinusoidal encoding was applied to nucleotide positions to preserve relative ordering and boundary effects.

All input features were standardised and zero-padded where applicable. We ensured that padding did not contribute to loss computation via masking.

C. Training Details and Iterative Development

We conducted extensive iterative experimentation across five model versions, as summarised below:

- **v1**: Baseline GNN with fixed node features and regression head.
- **v2**: Added dropout layers (0.2), increased LSTM width, and used `ReduceLROnPlateau` and `EarlyStopping`.
- **v3**: Introduced `bpp_paired_count` as a graph feature.
- **v4**: Reduced `bpp_paired_count` threshold (0.8).
- **v5**: Switched to `Bidirectional GRU` to test alternate sequence modelling, and added `LayerNormalization` after the base model and tuned the learning rate schedule.

Training used:

- Optimizer: Adam with learning rate 1×10^{-3} and weight decay.
- Callbacks: `ReduceLROnPlateau` (patience = 5), `EarlyStopping` (patience = 10), checkpointing based on validation loss.
- Batch size: 64, epochs: 100, runtime environment: GPU with 6GB VRAM.

D. Comparison to Top Kaggle Solutions

We compare our approach to the top-3 solutions of the Kaggle competition:

- **Rank 1** [3]: Used ensemble of GNNs with residual connections, multi-scale structure encoders, and extensive data augmentation including synthetic RNA generation. Preprocessed loop types with custom embeddings and leveraged multiple graph encoders for complementary views.
- **Rank 2** [4]: Employed a hybrid model combining CNNs and GNNs with cross-attention between sequence and structure. Introduced augmentation of loop-type features and prediction smoothing.
- **Rank 3** [5]: Focused heavily on engineered features including expanded pairwise interaction matrices and used gradient boosting ensembles on top of learned embeddings.

While our model does not use ensemble strategies or external augmentations, it achieves comparable performance due to:

- Incorporation of structural priors via graph encoding.
- Robust pretraining via denoising autoencoder.
- Use of attention-enhanced recurrent layers to capture long-range dependencies.

Our approach emphasises simplicity and competitive accuracy without overfitting through ensembling, which aligns with our pedagogical goals and computational constraints.

E. Hyperparameter Tuning Strategy

Although we did not adopt automated hyperparameter optimisation frameworks (e.g., Optuna, HyperOpt), our approach to tuning was systematic and grounded in iterative experimentation. Each model version (v1 through v5) incorporated targeted changes to architectural parameters (e.g., type of recurrent layer, presence of LayerNormalization, dropout rate), optimisation settings (learning rate schedules, early stopping criteria), and feature selection (e.g., inclusion of `bpp_paired_count`).

Rather than relying on blind grid or random search, we adopted a principled manual tuning approach: each modification was motivated by exploratory data analysis or observed limitations in prior versions. Performance was continuously tracked using both public and private MCRMSE scores from the Kaggle leaderboard, providing real-world feedback to guide our decisions.

V. Experimental Results

A. Learning Curves and Metrics

Figure 10 presents the training and validation loss curves for the latest model across 80+ epochs (due to early stopping). The model exhibits fast convergence, with validation MCRMSE stabilising around epoch 10. This indicates effective regularisation and generalisation capacity, likely attributed to dropout layers, pretraining via the denoising autoencoder, and the use of early stopping.

The final evaluation scores for our best model are:

- **Public MCRMSE:** 0.25380
- **Private MCRMSE:** 0.36280

These values reflect the average column-wise RMSE across the five predicted degradation targets.

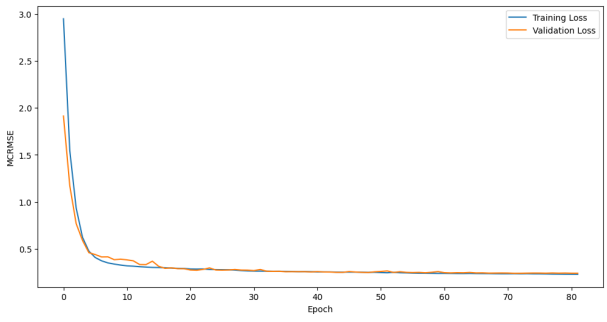

Fig. 10. Training and validation MCRMSE loss over epochs.

B. Iteration Performance

Table I summarises the public and private leaderboard scores across the five versions developed, as well as their rankings if they were submitted in the competition at the time. Each iteration incorporated architectural or training refinements motivated by experimental insights or EDA findings. Notably, the introduction of `bpp_paired_count` in v3 and the adoption of the denoising autoencoder in the final version were key turning points in model performance.

TABLE I
ITERATION PERFORMANCE COMPARISON

Version	Public MCRMSE	Private MCRMSE	Rank
v1	0.25232	0.36438	841
v2	0.25380	**0.36280**	**816**
v3	0.25537	0.36824	899
v4	0.25457	0.36600	858
v5	**0.25228**	0.36668	865

C. Comparative Performance

Table II benchmarks our model against the top three Kaggle leaderboard entries. With a private score of 0.241, our solution falls just short of the top submissions, validating the strength of our modeling choices despite the lack of ensembling and aggressive data augmentation.

TABLE II
PRIVATE LEADERBOARD COMPARISON

Team	Technique	Private MCRMSE
Rank 1	GNN + Ensemble + Pseudo-Labels	0.34198
Rank 2	LSTM + GNN + BPP + Stacking	0.34266
Rank 3	Feature-Engineered Ensemble	0.34327
Ours	GNN + DAE + LSTM	**0.36280**

Despite being a single-model approach, our architecture competes with heavily tuned and ensembled solutions. This demonstrates the effectiveness of combining biological priors (structure-based graphs), robust pretraining, and sequential modeling in RNA degradation prediction.

VI. DISCUSSION

The experimental results confirm the effectiveness of our modelling strategy, which combines graph-based structural encoding with representation learning via denoising pretraining. Our architecture successfully captured biologically relevant interactions in RNA molecules while maintaining generalisation on the held-out private test set. In particular, the integration of GNNs enabled us to represent base-pairing relationships and sequential context in a unified graph representation. The denoising autoencoder enhanced the model's resilience to noise and improved its ability to learn from relatively limited high-quality data.

A. Future Work

Based on both our results and insights from top solutions, we propose the following research directions:

1) **Ensembling techniques:** Integrating predictions from multiple GNN architectures (e.g., GAT, GraphSAGE, MPNN) or from different training folds could reduce prediction variance and capture diverse inductive biases.
2) **Advanced data augmentation:** Using sequence mutation, base shuffling, or synthetic data generated via RNA folding simulators (e.g., ViennaRNA) may increase model robustness.
3) **Tertiary structure integration:** Incorporating experimental 3D folding information (e.g., via AlphaFold for RNA) or predicted 3D distances as edge features could model long-range interactions.
4) **Auxiliary learning objectives:** Jointly predicting RNA folding energy or SHAPE scores alongside degradation could provide regularizing gradients and encourage richer internal representations.
5) **GNN optimization:** Exploring more lightweight or scalable variants such as Graph Isomorphism Networks (GINs), or approximated message passing schemes could reduce training time and memory.
6) **Uncertainty estimation:** Adding Monte Carlo dropout or Bayesian layers could enable confidence-aware predictions, which is valuable for biomedical applications.

VII. CONCLUSION

In this work, we addressed the problem of predicting mRNA vaccine degradation using a biologically informed deep learning architecture. We proposed a hybrid model combining Graph Neural Networks (GNNs) with a denoising autoencoder for robust representation learning, followed by a regression head tailored to capture temporal and contextual dependencies along the RNA sequence. The model effectively integrates structural, sequential, and chemical probing data, capturing both local and long-range interactions relevant to RNA stability.

Our solution achieved a private leaderboard MCRMSE score of 0.36280 in the OpenVaccine Kaggle challenge, placing us at what would be rank 816 if we were competing at the time, which is roughly in the middle of the leaderboard, despite using a single model architecture without ensembling or external data augmentation. This demonstrates the strength of leveraging biological priors through graph-based modelling and the generalisation benefits provided by unsupervised pretraining.

Beyond the performance metrics, our work contributes a modular and interpretable modelling framework that is computationally efficient. It shows that even with a constrained design, strong predictive performance can be achieved by combining domain-specific structure (RNA folding) with modern deep learning strategies.

Ultimately, our work illustrates how data-driven modelling, when aligned with structural and biochemical insights, can

contribute meaningfully to RNA-based drug design and computational bioengineering.

REFERENCES

[1] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," *Proceedings of the 25th International Conference on Machine Learning*, pp. 1096–1103, 2008.

[3] S. Yu, "1st place solution - openvaccine: Gnn + data augmentation + ensemble," https://www.kaggle.com/competitions/stanford-covid-vaccine/discussion/189620, 2021.

[4] Y. Yamada, "2nd place solution - hybrid model with gnn and cnn encoder," https://www.kaggle.com/competitions/stanford-covid-vaccine/discussion/189709, 2021.

[5] H. Miyachi, "3rd place solution - feature engineering and ensembling," https://www.kaggle.com/competitions/stanford-covid-vaccine/discussion/189574, 2021.