Peptide Retention Time Prediction from Sequence Using Transformer Neural Networks

Maksim Makaranka
Faculty of Electronics and Information Technology,
Warsaw University of Technology, Warsaw, Poland

***Abstract*—The prediction of peptide retention times from amino acid sequences is essential for bioinformatics workflows such as LC-MS analysis. This study implements and evaluates transformer-based neural networks, comparing encoder-only and decoder-only architectures. Models are trained on datasets of peptide sequences and measured retention times. Results indicate that transformer models achieve high predictive accuracy, with encoder-only architectures performing best. The findings demonstrate the effectiveness of self-implemented transformer networks for peptide retention time prediction.**

***Index Terms*—Neural network, transformer, peptide, retention time, regression, sequence prediction, bioinformatics, amino acid, attention, encoder, decoder.**

I. Introduction

Prediction of peptide retention time based solely on amino acid sequence is a fundamental task in bioinformatics and proteomics, with direct applications in liquid chromatography-mass spectrometry (LC-MS) workflows. Accurate retention time prediction enables higher confidence in peptide identification, improves the design of acquisition experiments, and allows for comparison and alignment across datasets and platforms. Traditional approaches, such as retention coefficient-based models or classical machine learning, have provided moderate success but are limited by their inability to capture the full complexity of peptide behavior, particularly sequence context and nonlinear dependencies.

Peptide retention in chromatographic systems is dictated by multiple, intricate factors arising from the physicochemical properties and the sequential arrangement of amino acids, rendering the underlying mapping from sequence to retention time highly complex and nonlinear. Analytical or rule-based models lack the flexibility to account for these effects, especially as peptide sequence space expands and modification types increase in modern proteomics data.

Data-driven models, and in particular deep neural networks, have emerged as the de facto approach for this task due to their expressiveness and capacity to automatically capture complex sequence patterns. Among these, transformer architectures are especially well-suited, as they natively encode sequence information, learn long-range dependencies between amino acid positions, and outperform traditional recurrent or convolutional architectures in a range of sequence modeling tasks. In retention time prediction, transformers enable direct modeling of arbitrary-length peptides and can generalize across a wide diversity of sequences with fewer engineered features.

Given the lack of simple analytical solutions and the demonstrated limits of prior statistical or shallow learning approaches, deep neural networks - especially transformer models - represent the most effective and robust methodology for sequence-based peptide retention time prediction in modern LC-MS proteomics.

II. Related Work and Background

Accurate prediction of peptide retention time (RT) is a key task in computational proteomics, with a long history of method development. Early models used regression on amino acid indices or hydrophobicity scales, but were limited in their ability to handle larger or modified peptides [1].

A significant advance came with artificial neural networks (ANNs), such as the approach by Petritis et al., where RT was predicted using only the counts of each amino acid in the peptide, without considering their order in the sequence [2]. While more flexible than regression, these early ANNs could not capture sequence-specific or long-range effects.

With the availability of large datasets, deep learning models have replaced shallow ANNs. Modern methods, such as DeepRT, use deep neural architectures to automatically learn complex features from peptide sequences, yielding high predictive accuracy for various chromatography types [3].

A limitation of many such models is the need for explicit training data on each peptide modification. DeepLC addressed this by representing peptides using atomic composition, enabling reliable RT prediction even for previously unseen modifications [4].

As reviewed by Moruz and Käll, current trends move toward flexible, data-driven models capable of learning sequence dependencies [1], and transformer-based architectures are a promising future direction for this task.

III. Methodology Overview

This section formalises the prediction task and outlines the sequence-to-value models evaluated in this study.

A. Problem statement

A peptide is uniquely described by its amino acid sequence $s = (a_1, a_2, \ldots, a_L)$, where each a_j denotes an amino acid and L is the length of the sequence. The available data consist of a set $\mathcal{D} = \{(s_i, y_i)\}_{i=1}^{N}$, where each $s_i \in \mathcal{S}$ is a peptide sequence and $y_i \in \mathbb{R}$ is its experimentally measured retention time (RT). The objective is to learn a regression function

$f : \mathcal{S} \to \mathbb{R}$ that accurately predicts retention time for any given peptide sequence, by minimising the empirical loss on the training set and achieving good generalization performance on previously unseen peptides.

B. Input representation

- **Tokenization** Each amino acid is mapped to an integer using a fixed vocabulary ($|V| = 27$): 20 canonical residues, five uncommon residues (B, J, O, U, Z), plus `<pad>` and `<unk>`.
- **Vectorization** Tokens are embedded via a learnable matrix $E \in \mathbb{R}^{|V| \times d_{\text{model}}}$, where $|V|$ is the vocabulary size and d_{model} is a model hyperparameter. A sequence of L tokens is thus represented as $E(s) \in \mathbb{R}^{L \times d_{\text{model}}}$.
- **Positional encoding** For each input sequence, a fixed sinusoidal positional encoding matrix $P \in \mathbb{R}^{L \times d_{\text{model}}}$ is added element-wise to the token embeddings, forming the final input representations $X = E(s) + P$. This enables the model to distinguish token positions and capture sequence order [5].
- **Padding mask** Mini-batches are padded to equal length; a boolean mask marks the pad positions for subsequent attention masking.

C. Encoder-Only Transformer

This architecture applies a standard Transformer encoder stack [5] to the embedded peptide sequence.

- **Input**: Peptide sequences, tokenized and embedded as $X \in \mathbb{R}^{L \times d_{model}}$. A sinusoidal positional encoding is added to incorporate order information.
- **Encoder stack**: The input X is passed through n_{layers} stacked Transformer encoder layers [5]. Each layer consists of multi-head self-attention, capturing relationships between every pair of sequence positions, followed by a position-wise feedforward network with hidden dimension $4 \times d_{model}$. The output is a matrix $H \in \mathbb{R}^{L \times d_{model}}$, where each row H_t represents the t-th peptide position (amino acid) after contextualization.
- **Attention pooling**: The sequence output H is summarized to a single vector using attention pooling, a learnable global pooling technique:
  - Each position t is assigned a score s_t by passing its feature vector H_t through a linear layer.
  - All s_t scores are run through a softmax function to obtain normalized weights α_t (i.e., higher α_t means the position is considered more important).
  - The final peptide representation h is computed as a weighted average of all position vectors:

$$h = \sum_{t=1}^{L} \alpha_t H_t$$

    where $\sum_{t=1}^{L} \alpha_t = 1$.
  - *Remark:* Attention pooling enables the model to automatically select and emphasize the most informative amino acids or positions for retention time prediction, potentially improving accuracy compared to simple mean pooling, which treats all positions equally. This is particularly helpful if specific sequence subregions contribute disproportionately to the retention characteristics of peptides.
- **Regression head**: A small two-layer neural network (MLP) (with GELU activation) maps the pooled vector to a single retention time value.

D. Decoder-Only Transformer with Learnable Queries

In this variant, inspired by Perceiver-style models [6], a small set of trainable "query" vectors is used to extract information from the peptide sequence.

- **Input**: The peptide is encoded as before (embedding + positional encoding), forming "memory" of shape $L \times d_{\text{model}}$.
- **Learnable queries**: A set of q vectors (e.g., $q = 4$), initialized randomly and learned during training, are used as the decoder input. These queries act as information bottlenecks that selectively gather features from the peptide.
- **Decoder stack**: Each query attends to the embedded peptide (cross-attention) and is refined via n_{layers} stacked Transformer decoder layers [5].
- **Attention pooling**: The model computes scores for each output vector produced by the queries, normalizes these scores (softmax), and then computes a weighted sum of the output vectors based on the normalized scores. This mechanism allows the model to aggregate information from all q query channels into a single summary vector, effectively learning how to combine relevant information from the entire set of outputs.
- **Regression head**: As above, the pooled vector is processed by an MLP to produce the final retention time prediction.

E. Training Procedure

The models are trained to minimize the Smooth L1 (Huber) loss, which is robust to outliers and balances sensitivity and stability during regression:

$$\mathcal{L}(\hat{y}, y) = \begin{cases} 0.5\,(\hat{y} - y)^2/\beta & \text{if } |\hat{y} - y| < \beta \\ |\hat{y} - y| - 0.5 * \beta & \text{otherwise} \end{cases}$$

with $\beta = 1$. The Adam optimizer is used during training.

To assess predictive performance, a comprehensive set of regression and correlation metrics is reported, including mean absolute error (MAE), root mean squared error (RMSE), coefficient of determination (R^2), and both Pearson and Spearman correlation coefficients (see VI).

F. Hyperparameter Selection

Hyperparameters are selected empirically based on extensive experimentation. The main parameters shared across datasets are:

- embedding/model dimension ($d_{model} = 64$)

- number of attention heads ($n_{heads} = 4$)
- batch size (64)
- number of encoder/decoder layers ($n_{layers} = 5$)
- hidden dimension of the feedforward sublayer ($d_{ff} = 4 * d_{model}$)
- number of queries in decoder ($n_{queries} = 4$)
- dropout rate (0.1), which helps prevent overfitting by randomly disabling 10% of neurons (activations) during training

The number of training epochs and learning rate are dataset-dependent. Larger datasets (e.g., `misc_dia`) use more training epochs and a higher learning rate (> 300 epochs, 3×10^{-4}), while smaller datasets (e.g., `cysty`) use fewer epochs and a lower learning rate (120-150 epochs, 1×10^{-4}) to avoid overfitting.

Model size, especially the number of layers, was initially adjusted according to dataset size; however, subsequent experiments focused on larger datasets, and the final configuration used this larger model size throughout. With these settings, the decoder-only architecture contains 340,034 parameters, while the encoder-based model contains 255,938 parameters.

Hyperparameter values thus reflect a balance between dataset characteristics and observed empirical performance.

IV. Implementation

The full implementation is provided in modular Python files using the PyTorch library [7]. The main classes are:

- Model classes `PeptideRTEncoderModel` and `PeptideRTDecoderModel` (`model.py`): implement encoder- and decoder-based architectures.
- `AATokenizer` (`tokenizer.py`): a utility for converting amino acid sequences to and from integer token representations.
- `PeptideRTDataset` (`dataset.py`): a dataset class for loading peptide/retention time pairs from text files, and a collate function for batch processing.

Training logic, including the command-line interface for running experiments and saving models, is provided in `rt_transformer.py`. Supporting utilities such as metrics, batching, and random data splitting are found in `utils.py`.

Although all reported experiments and tests were performed in interactive Python notebooks, full training can also be launched from the command line as follows:

```
python rt_transformer.py --data <DATAPATH>
--arch encoder --output <MODELPATH>
```

All key model and training hyperparameters may be set via command-line arguments.

The code organization facilitates reproducibility and easy adaptation to other datasets or peptide prediction tasks.

V. Experimental Setup

A. Hardware and Software Environment

All experiments were conducted on a local workstation equipped with an NVIDIA RTX 2050 GPU, running CUDA for accelerated training and inference. Model development and evaluation were carried out using separate Jupyter notebooks for each architecture (encoder and decoder). The implementation relies on PyTorch and standard Python scientific libraries.

B. Datasets

Models are evaluated on two datasets with distinct characteristics. The `cysty` dataset, provided by the course instructor, contains 11,041 peptide entries and consists of normalized retention time values ranging from 0 to 1. The `misc_dia` dataset, independently collected from an online source [8], contains 146,588 entries; in this dataset, retention time values are not normalized and typically span a much larger range (several thousands). This dual-dataset evaluation allows validation of the models' ability to generalize across data sources, as well as across different retention time scales and distribution types.

C. Training and Evaluation Workflow

The experimental workflow is organized into four Jupyter notebooks:

- Notebooks `PeptideRT_Encoder_Train.ipynb` and `PeptideRT_Decoder_Train.ipynb` conduct the training process for the encoder- and decoder-based models, respectively. Each notebook includes per-epoch reporting of training and validation loss, as well as visualization of the learning curves to monitor convergence and detect overfitting.
- Notebooks `PeptideRT_Encoder_Eval.ipynb` and `PeptideRT_Decoder_Eval.ipynb` perform the evaluation of the trained models. For each model, a scatter plot is generated showing predicted versus experimental (true) retention times, providing a visual assessment of model accuracy and systematic bias. All implemented metrics described in VI (such as MAE, RMSE, R^2, Pearson, and Spearman) are computed and reported. Outlier analysis is included to highlight mispredicted cases. Example code snippets for both single predictions and batch inference are also provided.

D. Reproducibility

All experimental notebooks are self-contained and organized to allow for reproducibility. To ensure consistent results across runs, a fixed random seed (`SEED = 42`) is set for all relevant libraries (`torch`, `random`, and `numpy`). While training and evaluation can also be launched from the command line, all reported results correspond to notebook-based experiments, where key outputs (such as figures and metrics) are generated for analysis and reporting.

VI. Results and Discussion

This section first introduces the evaluation metrics that are reported throughout the paper, then reviews the observed learning behaviour, presents qualitative error analyses, compares the four trained models quantitatively, and finally relates the obtained performance to earlier work.

A. Evaluation Metrics

The study employs a comprehensive set of regression and correlation metrics to assess model performance, following established practice in the peptide RT prediction literature. The main reported metrics are:

- **MAE (Mean Absolute Error):** Measures the average magnitude of prediction errors, regardless of direction. Particularly suitable for comparing models across datasets with differing RT scales.
- **RMSE (Root Mean Squared Error):** Penalizes large prediction errors more strongly than MAE and reflects the spread of the error distribution.
- **Coefficient of Determination (R^2):** Indicates the proportion of observed variance in RT explained by the model ($R^2 = 1$ for perfect predictions).
- **Pearson Correlation:** Quantifies linear association between predicted and true RT, robust against absolute shifts in scale/offset.
- **Number of Outliers:** Peptides for which absolute prediction error exceeds two times the median absolute error, summarizing extreme prediction failures.

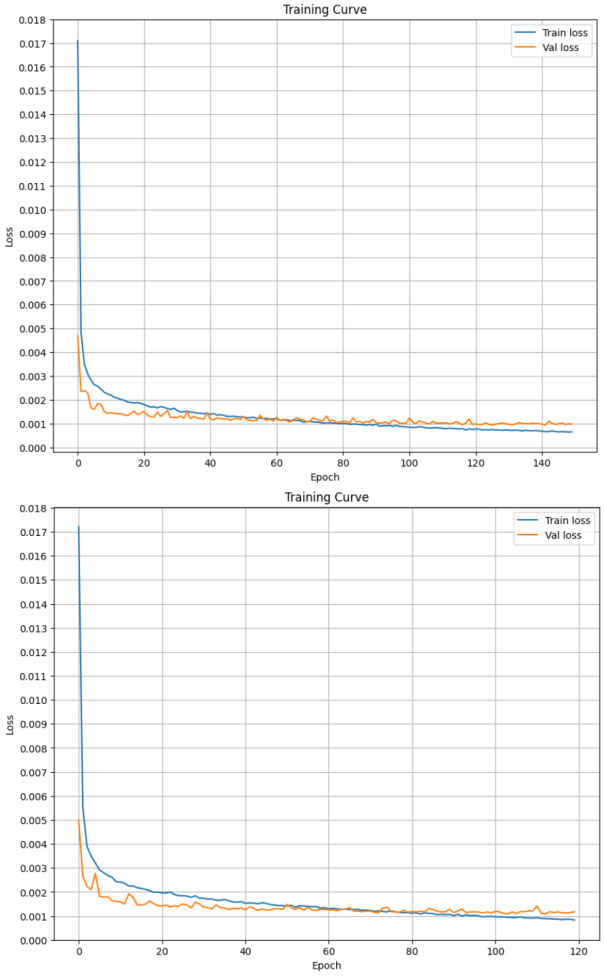

Fig. 1: Learning curves for encoder and decoder on the cysty dataset.

B. Training Behaviour and Learning Curves

Figures 1 and 2 show the learning curves for both encoder and decoder models on each dataset. For the smaller, normalized `cysty` dataset, both architectures reach convergence within 90–110 epochs, after which the training and validation losses stabilize (see Fig. 1). On the much larger `misc_dia` dataset, the loss for the encoder-only model continues to decrease even after 600 epochs. Due to hardware limitations (approximately 11 hours for 600 epochs), encoder-only training was stopped at this point, although further improvement may still be possible with extended training and hyperparameter optimization. The decoder-based model was terminated after 300 epochs, as it clearly underperformed compared to the encoder model, making further training unjustified. With a more advanced computational environment and additional training time, the results for the encoder-only model could potentially be improved even further.

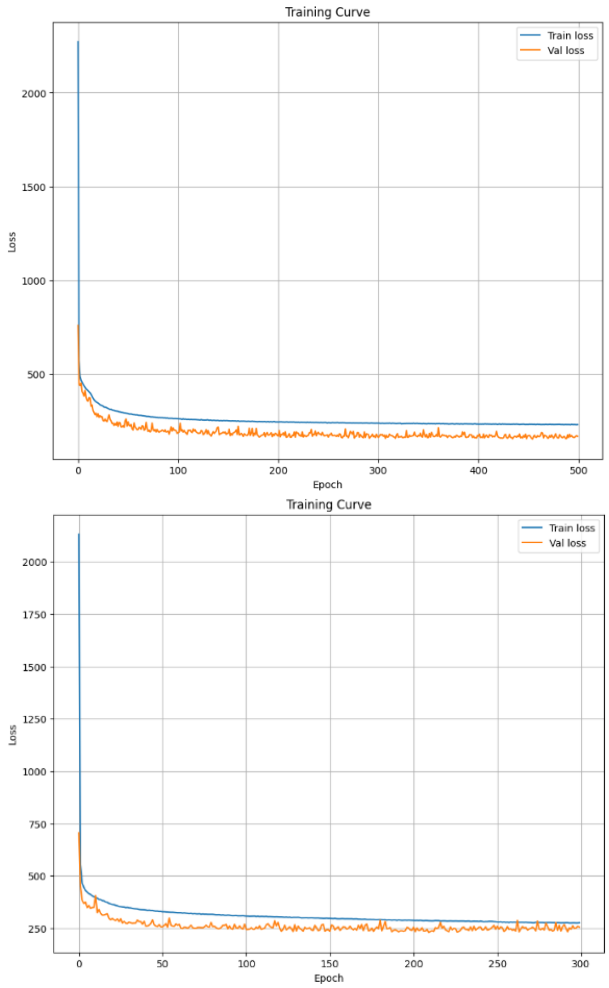

Fig. 2: Learning curves for encoder (left) and decoder (right) on the misc_dia dataset.

C. Predicted vs Experimental RT and Outliers

Figures 3 and 4 present scatter plots comparing predicted and true retention times for all models and datasets. Ideally,

all points would cluster along the diagonal. On `cysty`, the encoder-only model produces visibly tighter clustering, with fewer and less extreme outliers, even though the outliers tend to correspond to the same peptides in both models.

It is also apparent that, when comparing the same number of data points from the validation splits, the scatter plot for `misc_dia` exhibits a denser clustering of points around the diagonal, indicating a higher correlation between predictions and experimental values. This suggests that training on a larger dataset enables the model to achieve better predictive performance, a trend that is further confirmed by the evaluation metrics.

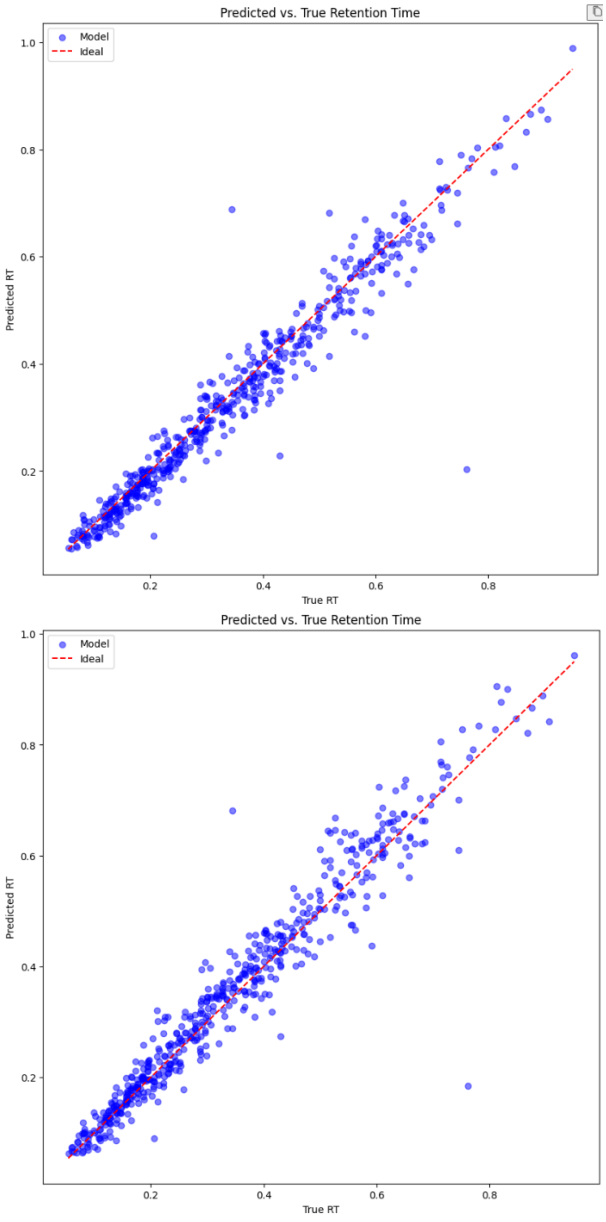

Fig. 3: Predicted vs. true retention times for encoder (top) and decoder (bottom) on `cysty` dataset (552 data points).

Outlier analysis revealed 9 outliers for the encoder and 14 for the decoder on the `cysty` validation set out of 552 records (about 1.6–2.5%), and 322 (encoder) and 345 (decoder) outliers among 7329 validation examples for `misc_dia` (about 4.4%). Here, an outlier is defined as a peptide with an absolute prediction error exceeding two standard deviations of the error distribution, which is a stricter threshold than the commonly used rule of three standard deviations. Using this stricter threshold increases the number of outliers, making it easier to spot moderate mispredictions and compare how well different models handle such cases.

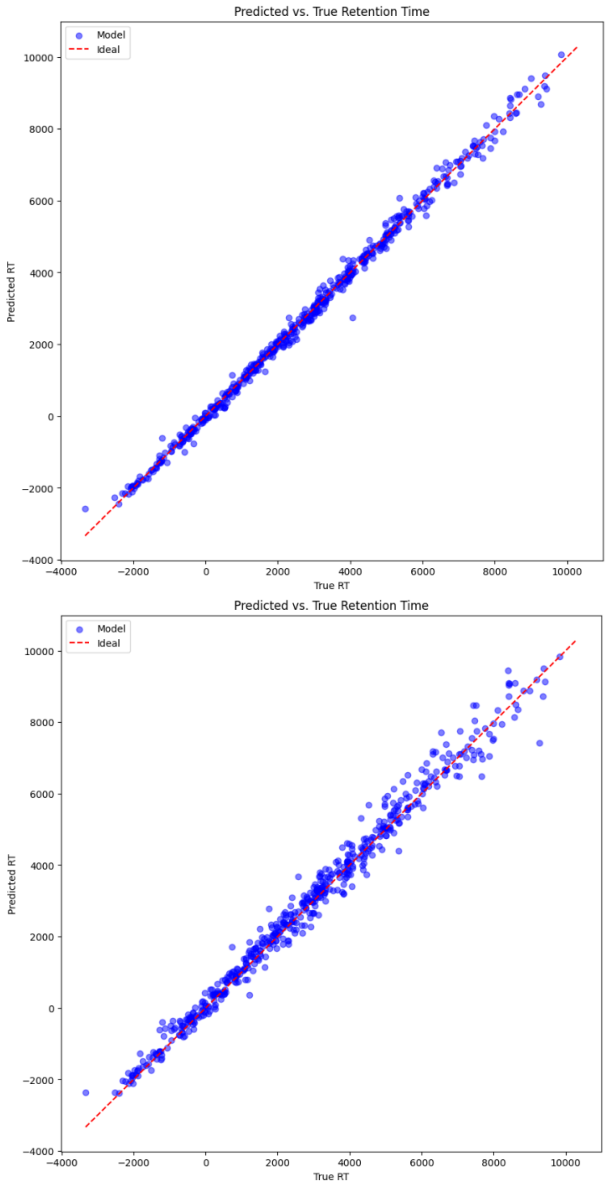

Fig. 4: Predicted vs. true retention times for encoder (top) and decoder (bottom) on `misc_dia` dataset (552 data points).

Many outliers correspond to peptides with highly unusual sequences or physicochemical properties, suggesting rare chromatographic behaviours or possible dataset labelling issues. The concentration of error in a small subset of peptides implies

that further improvements could be realized by targeted outlier mitigation (such as data curation), or by expanding the training dataset.

D. Quantitative Comparison across Models and Datasets

Table I summarizes performance for both model architectures and both datasets, highlighting best results per metric. The encoder-only outperforms the decoder-only on every metric, achieving lower error and higher correlation, with relative advantages most pronounced on the challenging `misc_dia` dataset.

TABLE I: Key validation metrics (best in bold). Outl.: percentage of predictions identified as outliers.

Dataset	Architecture	MAE	RMSE	R^2	Pearson	Outl.
cysty	encoder	**0.027**	**0.044**	**0.945**	**0.974**	**1.63%**
	decoder	0.030	0.048	0.934	0.970	2.54%
misc_dia	encoder	**124.7**	**174.9**	**0.996**	**0.998**	**4.39%**
	decoder	256.2	357.6	0.983	0.992	4.71%

In addition, the encoder's consistently higher R^2 and Pearson correlation on both normalized and non-normalized data confirm its robustness to different RT scales.

E. Comparison with Literature

The obtained results are benchmarked against three major classes of peptide retention time prediction models from literature: classical MLP neural networks [2], deep capsule networks (DeepRT) [3], and atomic composition deep learning models (DeepLC) [4].

On normalized RT data (such as the `cysty` dataset), the transformer encoder demonstrates a visibly better fit than a shallow neural network, like the one-layer MLP. Most notably, on the large-scale `misc_dia` dataset, the encoder-only transformer achieves outstanding validation metrics: $R^2 = 0.996$, Pearson correlation $= 0.998$, and Spearman correlation $= 0.998$, with a median absolute error (MAE) of 124.7 and an RMSE of 174.9. These scores are as good as or better than those reported for DeepRT ($R^2 = 0.994$) and DeepLC (Pearson > 0.99), both of which rely on more specialized architectures.

It is important to note that, unlike DeepRT and DeepLC, the current study was conducted in a computationally restricted environment, without hyperparameter optimization or access to extensive computational resources. Nevertheless, the encoder-only transformer achieves state-of-the-art generalization (as shown by R^2 and correlation statistics) with a compact architecture (five transformer layers, under 0.3M parameters).

Overall, these results highlight the strength of transformer-based models for peptide retention time prediction. The encoder-only approach not only outperforms traditional neural network models, but also matches or exceeds more specialized deep learning methods in predictive accuracy and robustness, even when trained under limited computational constraints.

In summary, the findings confirm that a moderately sized transformer encoder can surpass earlier neural network approaches and achieve accuracy competitive with current deep learning state-of-the-art models. Further improvements are likely feasible with additional training time, hyperparameter optimization, or enhanced data preprocessing and outlier handling.

REFERENCES

[1] L. Moruz and L. Käll, "Peptide retention time prediction," 2016. [Online]. Available: https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/10.1002/mas.21488

[2] K. Petritis, L. J. Kangas, P. L. Ferguson, G. A. Anderson, L. Pasá-Tolic, M. S. Lipton, K. J. Auberry, E. F. Strittmatter, Y. Shen, R. Zhao, and R. D. Smith, "Use of artificial neural networks for the accurate prediction of peptide liquid chromatography elution times in proteome analyses," 2003. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/12641221/

[3] C. Ma, Y. Ren, J. Yang, Z. Ren, H. Yang, and S. Liu, "Improved peptide retention time prediction in liquid chromatography through deep learning," 2018. [Online]. Available: https://pubs.acs.org/doi/10.1021/acs.analchem.8b02386

[4] R. Bouwmeester, R. Gabriels, N. Hulstaert, L. Martens, and S. Degroeve, "DeepLC can predict retention times for peptides that carry as-yet unseen modifications," 2021. [Online]. Available: https://www.biorxiv.org/content/10.1101/2020.03.28.013003v2.full

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762v7

[6] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General perception with iterative attention," 2021. [Online]. Available: https://arxiv.org/abs/2103.03206

[7] M. Makaranka, "Transformer-peptide-rt-prediction," 2025, GitHub repository. [Online]. Available: https://github.com/maksanm/Transformer-peptide-RT-prediction

[8] O. B. C. Garzon Otero, D.; Akbari, "Pepmnet: A hybrid deep learning model for predicting peptide properties using hierarchical graph representations," misc_dia dataset retrieved from https://github.com/danielgarzonotero/PepMNet. [Online]. Available: https://doi.org/10.1039/D4ME00172A