Efficient Peptide MRM Transition Prediction via Convolutional Hashing

Ramon Adàlia[1,2][0009−0004−9458−1922]✉, Gemma Sanjuan[1][0000−0002−1946−4345], Tomàs Margalef[1][0000−0001−6384−7389], and Ismael Zamora[2][0000−0002−7700−0354]

[1] Universitat Autònoma de Barcelona, Cerdanyola del Vallès, Spain
`Ramon.Adalia@autonoma.cat`✉, `{Gemma.Sanjuan,Tomas.Margalef}@uab.cat`
[2] Lead Molecular Design, S.L., Sant Cugat del Vallès, Spain
`{ramon.adalia,ismael.zamora}@leadmolecular.com`

Abstract. Measuring multiple reaction monitoring (MRM) transitions for peptides is a critical task in targeted proteomics, enabling the precise identification and quantification of peptides in complex biological samples. This study presents a novel method for MRM transition prediction, leveraging a hash-based representation inspired by convolutional neural networks. Peptide fragments are encoded as sparse count vectors, incorporating local sequence context through efficient hashing. We employ gradient-boosted decision trees (GBDTs) for prediction, capitalizing on their robustness to high-dimensional, sparse input data. Compared with the baseline models, the proposed method significantly improves prediction accuracy, achieving a mean Hits@5 score of 3.4318 for the hash-based model and 3.5405 for a hybrid model that integrates hash-based encoding with target frequency features. These results are statistically significant, demonstrating the efficacy of our approach. Furthermore, the method is computationally efficient, with inference optimized by transpiling trained models into Zig, producing lightweight executables capable of high-speed processing. Our results highlight the potential of this approach for scalable peptide MRM transition prediction, offering a practical solution for high-throughput proteomics applications.

Keywords: MRM transitions · Peptide quantification · Edge computing

1 Introduction

Multiple Reaction Monitoring (MRM) is a cornerstone analytical technique for peptide quantification and is critical in applications ranging from biomarker discovery to pharmaceutical development. A key step involves determining Multiple Reaction Monitoring (MRM) transitions, which rely on accurately identifying the mass-to-charge (m/z) ratios of peptide fragments. These ratios are essential for identifying and quantifying peptides in complex samples. Traditionally, this process requires experimental determination using pure samples, which is time-intensive and resource-heavy [10,9]. Predicting MRM transitions computationally can significantly increase throughput and reduce resource demands.

Although predictive models for small molecules exist [1], they are unsuitable for peptides due to fundamental differences in structure and fragmentation patterns. Despite their widespread usage [3,4], simplistic representations, such as amino acid composition [2], fail to capture critical positional information and features such as charge states or fragmentation sites. These limitations necessitate advanced representations tailored to peptide MRM prediction.

While neural network models, such as Evolutionary Scale Modeling (ESM) [8] and Diffusion Protein Language Models (DPLM) [11], capture sequence features effectively, their computational complexity is prohibitive for high-throughput settings, such as pharmaceutical pipelines. These applications demand not only high accuracy but also rapid inference to handle large data volumes.

To address these challenges, this study proposes a novel method tailored to peptide MRM prediction. Inspired by 1D convolutional neural networks (CNNs) [6], we replace convolutions with hashing to encode peptide fragments efficiently. This approach captures local sequence features in a sparse matrix format, resulting in low computational overhead. The model, trained using LightGBM [5], achieves rapid inference by transpiling decision trees into Zig, enabling high-speed, resource-efficient processing. This combination of efficient representation and lightweight implementation addresses the scalability and accuracy requirements of real-world proteomics workflows.

2 Background

This section provides an overview of the foundational concepts relevant to this work, including Multiple Reaction Monitoring (MRM) and peptides. While a deep understanding of these topics is not needed, this background aims to offer sufficient context for the subsequent sections.

2.1 Multiple Reaction Monitoring

Mass spectrometry (MS) is a technique for identifying and analyzing sample components by measuring the mass-to-charge ratio (m/z) of molecules. It involves ionizing the sample, separating the resulting charged particles based on their m/z, and detecting them to determine the sample's composition. Molecules with different m/z values travel distinct paths under electric or magnetic fields, allowing the mass spectrometer to differentiate between them.

Multiple Reaction Monitoring (MRM) is a mass spectrometry technique that quantifies specific compounds in complex mixtures with high specificity. The process occurs across three chambers:

1. In the first chamber, an electric or magnetic field filters ions with the desired m/z. However, compounds with similar m/z values may also pass through.
2. The second chamber fragments these ions via collisions with a noble gas.
3. In the third chamber, a second m/z filter selectively targets a known fragment of the compound of interest, achieving highly specific detection.

Determining the appropriate m/z values for MRM requires prior knowledge of the target compound. Typically, this is achieved by analyzing a pure compound sample with the second filter deactivated. The most abundant m/z values are then used for MRM analysis. However, this experimental setup is time-consuming, often takes minutes per compound, and limits throughput. Predictive models could streamline this process by estimating the most abundant m/z values, reducing the reliance on pure compound experiments and saving significant time and resources.

2.2 Amino Acids and Peptides

Amino acids are the fundamental building blocks of proteins. Each amino acid comprises the following:

– An **amino group** (-NH_2),
– A **carboxyl group** (-COOH),
– A **hydrogen atom**,
– A variable **side chain** (R group) that determines its chemical properties, such as size, charge, and hydrophobicity.

The 20 standard amino acids, encoded by the genetic code, are represented using single-letter abbreviations: A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, and V.

Peptides are short chains of amino acids linked by **peptide bonds**. These bonds form when the amino group of one amino acid reacts with the carboxyl group of another, releasing a water molecule. Peptides typically consist of fewer than 50 amino acids; chains longer than this are referred to as **proteins**. Peptides play diverse biological roles, including signaling and enzymatic functions. Their sequences, or **peptide sequences**, are commonly represented using the one-letter amino acid codes. For example, the peptide sequence CYIQNCPLG corresponds to the amino acids C, Y, I, Q, N, C, P, L, and G, where C and G represent the N- and C-termini, respectively.

b and y ions are key fragment ions observed in peptide mass spectrometry:

- **b ions** originate from cleavage of the peptide bond at the N-terminal side, resulting in a positively charged N-terminal fragment.
- **y ions** result from cleavage at the C-terminal side, producing a positively charged C-terminal fragment.

The subscripts of b and y ions indicate the number of amino acids in the fragment. For example, in the peptide CYIQNCPLG:

- b_3 corresponds to CYI, the first three amino acids.
- y_2 corresponds to LG, the last two amino acids.

Figure 1 illustrates the notation used for peptide fragmentation.

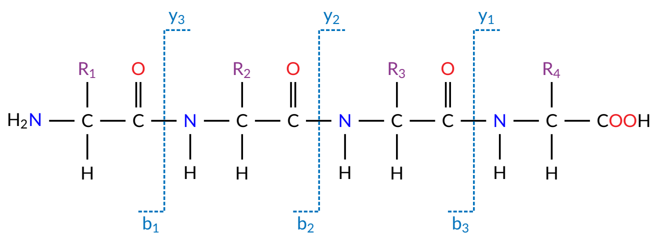

Fig. 1. Peptide fragmentation notation for a peptide with four amino acids. The N- and C-termini are on the left and right, respectively.

3 Methodology

3.1 Data

The dataset used in this study was obtained from the Supplementary Data of the open-access article *Multiple Reaction Monitoring Assays for Large-Scale Quantitation of Proteins from 20 Mouse Organs and Tissues* [7], licensed under the Creative Commons Attribution 4.0 International License. It consists of 2,965 unique peptide sequences, each ranging from 6 to 25 amino acids in length (median length: 11), accompanied by their optimized MRM transitions.

Among the 2,965 peptides, the majority (2,922) are associated with five optimized transitions. The remaining peptides exhibit varying numbers of transitions: 16 have six transitions, 9 have seven, 9 have four, 8 have three, and 4 have nine transitions. In total, the dataset contains 14,881 MRM transitions. Each transition is represented by either a b or y ion, paired with its charge state. For example, the notation b9++ indicates the 9th b ion with a charge of +2.

The charge states of the transitions are distributed as follows: 11,592 transitions have a charge of +1, 3,177 transitions have a charge of +2, 111 transitions have a charge of +3, and one transition has a charge of 0. The single transition with a charge of 0 was excluded from the analysis because it is invalid. No additional invalid transitions were identified in the dataset.

Figure 2 provides an overview of key dataset statistics, including the distribution of peptide sequence lengths and the counts of b and y ions. A significant observation is the predominance of smaller ions, as those with more than 10 amino acids occur infrequently. This skew towards shorter ions aligns with typical peptide fragmentation patterns, where shorter fragments dominate.

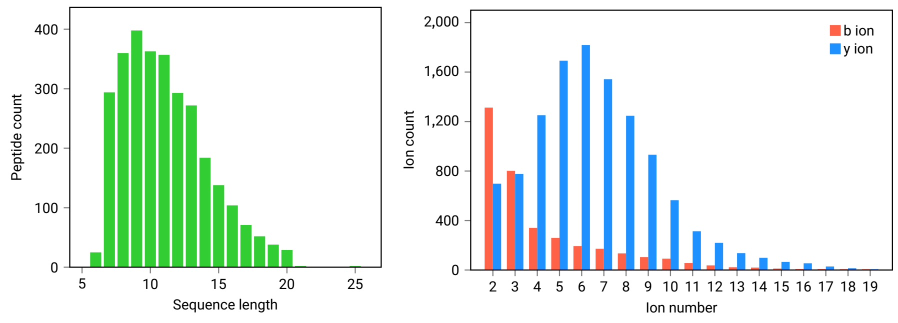

Fig. 2. Overview of the dataset's composition, including the distribution of sequence lengths and the counts of b and y ions.

3.2 Peptide Fragment Representation

Algorithm 1 outlines the process of generating peptide fragment representations. Inspired by 1-dimensional convolutional neural networks (1D CNNs), each amino acid in a peptide sequence is first mapped to an integer vector encoding its properties. These properties include an ordinal encoding of the amino acid (e.g., 1 for A, 2 for C, etc.) and a binary indicator: 1 if the amino acid is part of the fragment, and 0 otherwise. For instance, in the fragment b_2 of the peptide ACQA:

- The first amino acid (A) is encoded as $[1, 1]$, indicating its inclusion in the fragment.

Algorithm 1 Peptide fragment sequence representation with neighborhood hashing

```
Require: Input sequence $s$ of length $n \geq 3$, radius $R \geq 0$
 1: Initialize an empty array $H$ for storing hashes
 2: for $j \leftarrow 1$ to $n$ do
 3:     $v_j^0 \leftarrow [0] \parallel description(s_j)$   ▷ Concatenate step 0 with amino acid description
 4:     $h_j^0 \leftarrow hash(v_j^0)$   ▷ Compute the initial hash
 5:     Append $h_j^0$ to $H$
 6: end for
 7: for $i \leftarrow 1$ to $R$ do
 8:     $v_1^i \leftarrow [i, h_1^{i-1}, h_2^{i-1}]$   ▷ Update for first amino acid
 9:     $h_1^i \leftarrow hash(v_1^i)$
10:     Append $h_1^i$ to $H$
11:     for $j \leftarrow 2$ to $n-1$ do
12:         $v_j^i \leftarrow [i, h_{j-1}^{i-1}, h_j^{i-1}, h_{j+1}^{i-1}]$   ▷ Update for middle amino acids
13:         $h_j^i \leftarrow hash(v_j^i)$
14:         Append $h_j^i$ to $H$
15:     end for
16:     $v_n^i \leftarrow [i, h_{n-1}^{i-1}, h_n^{i-1}]$   ▷ Update for last amino acid
17:     $h_n^i \leftarrow hash(v_n^i)$
18:     Append $h_n^i$ to $H$
19: end for
20: return $H$   ▷ Return the list of hashes
```

– The last amino acid (A) is encoded as $[1, 0]$, reflecting its exclusion from the fragment.

Neighborhood context is captured using a hashing mechanism analogous to convolutional layers in 1D CNNs. Rather than learnable filters, a hash function aggregates the local context by processing the ordered set of neighboring amino acids. Given a sequence $c_1, c_2, \ldots, c_n$, the hash function employs a variation of a linear congruential generator (LCG), iteratively computed as:

$$X_{i+1} = (a \cdot X_i + c_i) \mod m, \quad \text{for } i \in [1, n-1],$$

where a and m are constants with $a < m$. We use $m = 2^{32}$ for efficient computation via modular arithmetic. The generator is initialized with $X_1 = 1013904223$, and $a = 1664525$.

Hashes are computed iteratively for a fixed radius R, with each step prepending an integer indicating the step number to the amino acid descriptions. The final list H is transformed into a sparse count vector of size m, which serves as the peptide fragment's fixed-size representation.

To incorporate charge states, we extend each fragment representation by appending an additional integer feature denoting the charge. This ensures that the model distinguishes identical fragments with varying charges.

3.3 Model Training

We utilize Gradient Boosting Decision Trees (GBDTs) for our predictive modeling. GBDTs are particularly effective for high-dimensional problems, such as those involving large, sparse count vectors. Our choice of GBDTs is driven by several advantages inherent to decision trees:

1. **Invariance to monotonic transformations:** GBDTs do not require input normalization, making them resilient to scale variations in the data.
2. **Robustness to correlated features:** Decision trees effectively handle multicollinearity without significant performance degradation.
3. **Interpretability:** GBDT inference is straightforward and relies on a series of conditional statements and additions. We leverage this interpretability in Section 3.5.

For training, we use the LightGBM library, which is known for its fast and efficient implementation of GBDTs. We train the models with LightGBM's default hyperparameters, which include 100 decision trees, each with up to 31 leaves. A ranking objective is employed: specifically, a LambdaMART implementation with a Normalized Discounted Cumulative Gain (NDCG) objective. This configuration is well-suited for tasks involving ordered outputs or priorities. We fix the radius to $R = 2$ during training.

To improve computational efficiency and address the limitations of LightGBM , we preprocess the input sparse matrix by removing columns with constant values (i.e., columns where all entries are identical). This step significantly reduces the feature count, with the extent of the reduction depending on the radius R. For instance, when $R = 0$, at most 41 features remain. These include:

1. One feature representing the charge state.
2. Twenty features corresponding to amino acids present in the peptide fragment.
3. Twenty features corresponding to amino acids absent from the peptide fragment.

This dimensionality reduction enhances computational efficiency. However, it introduces a key consideration for inference: columns with no variation in the training data may exhibit variation in the inference dataset. Maintaining consistency in feature selection between training and inference is crucial for ensuring accurate predictions.

3.4 Model Evaluation

To assess model performance, we employ a 5-fold cross-validation approach. The dataset, consisting of 2,965 peptides, is randomly partitioned into five equal-sized subsets, with each fold containing 593 peptides. This random partitioning ensures that the subsets are representative of the overall dataset.

In each iteration of the cross-validation, one fold is designated as the test set (held-out fold), whereas the remaining four folds serve as the training set. The model is trained on the training set and used to rank the candidate transitions for peptides in the held-out fold, generating out-of-fold predictions.

To evaluate the model's ranking capability, we compute the Hits@5 metric, which measures the number of correct transitions appearing within the top five ranked predictions. Rather than averaging performance across individual folds, we consolidate the out-of-fold predictions for all 2,965 peptides and assess the model's performance as a whole. This unified approach provides a single, comprehensive evaluation metric that reflects the model's effectiveness across the entire dataset, offering a more robust and interpretable assessment.

3.5 Model Inference

LightGBM offers a plain text output format that contains all the necessary information to execute a trained model. Given the simplicity of inference with GBDTs, which involves a series of additions and conditional statements, we leverage this output to transpile the model into a function written in the Zig programming language.

The transpiled function accepts a hashmap as input, where keys are column indices from the sparse count vector of the peptide fragment, and values represent the corresponding counts. The output is the numerical score assigned by the model.

This approach enables us to embed model inference within a standalone binary executable. By using Zig, we achieve the following:

1. **Compiler optimizations:** We leverage Zig's compiler optimizations for highly efficient execution.
2. **Reduced dependencies:** We eliminate the need for the entire LightGBM library, resulting in fewer dependencies.
3. **Compact executables:** Zig generates small and fast executables that are easy to deploy.

Moreover, Zig's static compilation capabilities ensure maximum compatibility with ABI-compliant hardware. It also supports seamless cross-compilation to various architectures and operating systems, simplifying the deployment of models across diverse environments. This ensures that model inference remains efficient and accessible, irrespective of the target platform.

4 Results and Discussion

This section presents an evaluation of baseline methods and the proposed models, focusing on their ability to predict MRM transitions for peptides. We begin by analyzing the performance of naive models, including a random sampling approach and a frequency-based target model. These serve as reference points for assessing the effectiveness of our hash-based model. The performance is quantified using the Hits@5 metric.

4.1 Baseline Methods

Given the relatively small size of peptides, one might question how well naive models, such as random models, could perform. Rather than relying on simulations, we compute the exact distribution of Hits@5 directly from the data.

For a peptide of length n, the total number of possible transitions is $8(n-1)$, which is calculated as $4 \cdot 2 \cdot (n-1)$. This is because there are $n-1$ possible b-ions, $n-1$ possible y-ions, and 4 possible charge states. When 5 transitions are randomly sampled without replacement from the $8(n-1)$ possible transitions, the number of correct transitions follows a hypergeometric distribution:

$$\text{Hypergeometric}(8(n-1), T, 5),$$

where T represents the number of experimentally identified transitions for that peptide, typically 5. The probability mass function of this distribution is as follows:

$$P(X=k) = \frac{\binom{T}{k}\binom{8(n-1)-T}{5-k}}{\binom{8(n-1)}{5}},$$

where $\binom{a}{b}$ denotes the binomial coefficient.

To determine the overall distribution across the dataset, we average the distributions of all 2,965 peptides. Specifically, the probability of a randomly chosen peptide having k correct transitions in the top 5 predictions is given by the average of the probabilities across all peptides. The expected value of the resulting distribution is 0.3396, which corresponds to the average Hits@5 that one can expect from a random model.

A more effective naive model ranks the candidate transitions according to their overall frequency in the dataset. We refer to this model as the *target* model, as it uses target encoding as a feature. This model always predicts the transitions $[y6+, y5+, b2+, y7+, y4+]$, except when the peptide is not large enough for some of these fragments to be possible. Importantly, this evaluation slightly overfits the training data, as the model uses information about the labels as a feature. Nonetheless, this model performs significantly better than the random model does, achieving an average Hits@5 of 2.2405.

4.2 Model Performance

Using the cross-validation procedure outlined in Section 3.4, we assessed the performance of the random model, the target model, and the hash-based model. To explore further improvements in predictive capability, we also developed a combined model that integrates target encoding with the convolutional hashing component, referred to as the *hash + target* model. Unlike the standalone target model, the combined model incorporates only the frequency of each feature within the training set of the corresponding cross-validation fold, reducing the risk of overfitting. Figure 3 illustrates the models' performance, evaluated using Hits@5.

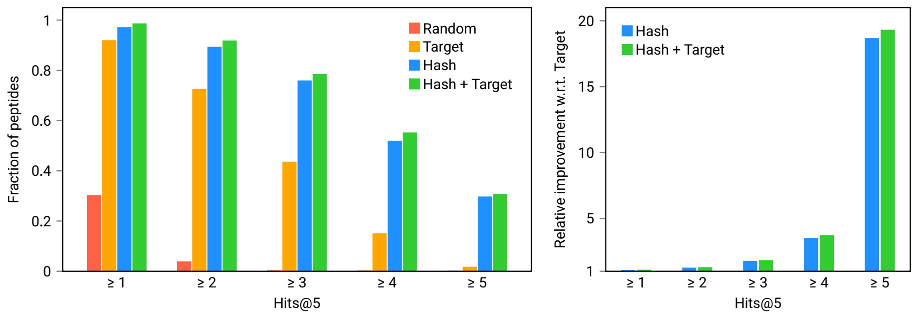

Fig. 3. Distribution of Hits@5 scores for each model.

The results indicate a clear improvement in performance from the baseline models to the hash-based model. The hash-based model achieves an average Hits@5 score of 3.4318, a notable increase compared with the target model's score of 2.2405, representing a 58% improvement. Statistical analysis using a one-sided paired permutation test confirms that this difference is highly significant (p-value $< 1e{-5}$). The combined *hash + target* model further improves upon the hash-based model, achieving an average Hits@5 score of 3.5405, with the improvement being statistically significant (p-value $= 0.00381$).

The distributions of Hits@5 scores reveal that while all the models achieve similar rates of at least one correct prediction within the top 5, the hash-based models exhibit a higher frequency of multiple correct predictions. This finding supports the effectiveness of the proposed approaches in generating more accurate predictions. Notably, the *hash + target* model achieves 19.28 times more perfect predictions than the target model does, highlighting the significant advantage of integrating convolutional hashing. Therefore, the proposed approach should be preferred over naive models whenever maximizing the number of correctly predicted MRM transitions is a priority, such as in the context of accurate peptide quantification.

5 Ablation Study and the Effect of Radius (*R*)

Our ablation study investigates the impact of the convolutional component, parameterized by the radius R, on model performance. The core components of our approach are convolutional hashing and peptide fragment descriptors. Removing any of the descriptors is impractical, as it renders the peptide fragments indistinguishable. Therefore, our analysis focuses on the effect of varying R, with $R = 0$ representing the scenario where the convolution is entirely removed.

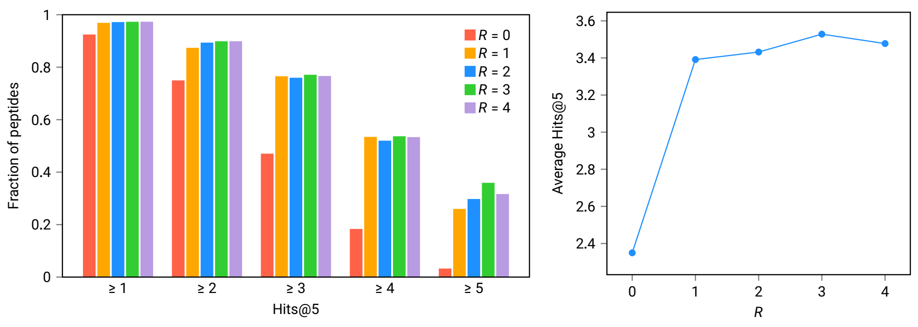

Fig. 4. Effect of the radius R on the distribution of Hits@5 scores.

Figure 4 illustrates the distribution of Hits@5 scores for various values of R. The results reveal a substantial improvement in performance when R increases from 0 to 1, emphasizing the significance of the convolutional component in capturing contextual information. However, performance gains plateau beyond $R = 1$, suggesting that additional contextual information from larger neighborhoods contributes only marginally to model accuracy. This trend underscores the value of the convolutional hashing component in our approach.

5.1 Training and Inference Performance

The radius parameter R also significantly influences the training time because of its effect on the feature matrix size. As R increases, the number of non-constant columns in the feature matrix grows, leading to longer computation times. Table 1 summarizes the training times and feature counts for different values of R, measured using the `time` command on a 12-core Intel® Core™ i7-8750H CPU @ 2.20GHz.

Table 1. Training times and feature counts for different radius values R.

Radius (R)	**Features (Median)**	**User Time (s)**	**System Time (s)**	**Total Time**
0	39	87.13	0.91	17.68 s
1	12,750	434.23	1.28	1:23.89 min
2	50,147	2280.08	3.05	6:45.87 min
3	82,040	3708.35	4.81	11:05.74 min
4	106,735	4862.43	7.51	14:32.50 min

As shown in Table 1, the feature count increases substantially with R. For instance, at $R = 4$, the feature count exceeds 2,700 times that of at $R = 0$. Consequently, the total training time increases from just under 18 s at $R = 0$ to over 14 min at $R = 4$. This highlights the trade-off between incorporating more complex neighborhood information and computational efficiency, a crucial consideration when selecting R for practical applications, particularly in resource-constrained environments.

We further evaluated the inference performance of our model by benchmarking the execution time and memory usage for peptide prediction tasks. Predictions were performed on a newline-separated file containing 2,965 peptide sequences, which were processed sequentially without parallelization. Benchmarking was conducted using `hyperfine` (10 runs per configuration) in conjunction with `time -v` for detailed memory and resource metrics. All tests were run on a Redmi Note 11 Pro 5G Android phone with an 8-core Snapdragon® 695 CPU @ 2.2 GHz, to assess both cross-compilation and performance on a lightweight device.

The evaluation included five configurations of the radius parameter R, which governs the neighborhood size considered during feature hashing. The key performance metrics recorded were as follows:

- **Maximum Resident Set Size (Memory Usage):** The peak memory allocated by the executable during runtime.
- **Execution Time:** Average runtime per prediction, including standard deviation.

– **Prediction Throughput:** The number of peptides processed per second, calculated as the reciprocal of the execution time per peptide.

Table 2. Inference performance for different radius settings (R).

Radius (R)	Maximum Resident Set Size (kB)	Time (s)	Peptides per Second
0	1180	6.564 ± 0.074	451.71
1	1180	6.789 ± 0.012	436.74
2	1180	6.989 ± 0.016	424.24
3	1180	7.205 ± 0.034	411.52
4	1180	7.632 ± 0.078	388.50

Table 2 summarizes the inference performance across different radius settings (R). Notably, the maximum resident set size remains consistent at 1180 kB for all configurations, indicating that memory usage is not substantially affected by changes in R. However, as R increases, the inference time and the number of peptides processed per second show a predictable trade-off. Larger values of R result in slightly slower predictions due to the increased computational complexity of processing extended neighborhoods. For example, at $R = 0$, the model achieves a throughput of 451.71 peptides per second, whereas for $R = 4$, the throughput decreases to 388.50 peptides per second, representing a 14% decrease. These results highlight the model's scalability and ability to maintain a small memory footprint while processing peptides.

6 Conclusion

In this study, we introduced a novel approach for predicting MRM transitions in peptides, leveraging a hash-based representation inspired by convolutional neural networks. This method efficiently encodes peptide fragment information using a sparse representation, which is both lightweight and conducive to high-speed inference. By combining this representation with gradient boosting decision trees, we achieved significant improvements in Hits@5 scores over baseline models, as demonstrated by rigorous statistical evaluation.

Our ablation study highlighted the importance of the convolutional hashing component, with the largest performance gain observed when the radius was increased from $R = 0$ to $R = 1$. This finding emphasizes the value of incorporating local sequence neighborhood information into the representation. Furthermore, the combination of hash-based encoding with target frequency features resulted in the best overall performance, highlighting the complementarity of these approaches.

We also demonstrated the scalability and efficiency of our method in both training and inference. The use of a compact representation enables training on large datasets, whereas the transpilation of LightGBM models into Zig functions ensures minimal computational overhead during inference. These attributes make our approach highly practical for real-world deployment, where computational resources may be limited.

Future work could extend this method to other biomolecule types or incorporate domain-specific knowledge to further increase the predictive accuracy. Additionally, integrating this approach into end-to-end proteomics workflows holds the potential to streamline experimental pipelines and improve peptide identification and quantification accuracy.

Overall, our work represents a significant step forward in developing efficient and accurate models for peptide transition prediction, offering a powerful tool for advancing proteomics research.

Acknowledgments. This study was conducted with the support of Pla de Doctorats Industrials del Departament de Recerca i Universitats de la Generalitat de Catalunya (grant number 2023-DI-00006).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Adàlia, R., Patel, S., Paiva, A., Kaufman, T., Zamora, I., Cai, X., Sanjuan, G., Shou, W.Z.: Development of a predictive multiple reaction monitoring (MRM) model for high-throughput ADME analyses using learning-to-rank (LTR) techniques. Journal of the American Society for Mass Spectrometry **35**(1), 131–139 (nov 2023). https://doi.org/10.1021/jasms.3c00363, https://doi.org/10.1021/jasms.3c00363
2. Chou, K.C.: A novel approach to predicting protein structural classes in a (20-1)-d amino acid composition space. Proteins **21**(4), 319–344 (Apr 1995). https://doi.org/10.1002/prot.340210406
3. Du, Q.S., Jiang, Z.Q., He, W.Z., Li, D.P., Chou, K.C.: Amino acid principal component analysis (aapca) and its applications in protein structural class prediction. Journal of Biomolecular Structure and Dynamics **23**(6), 635–640 (2006). https://doi.org/10.1080/07391102.2006.10507088, https://doi.org/10.1080/07391102.2006.10507088, pMID: 16615809
4. Jahandideh, S., Abdolmaleki, P., Jahandideh, M., Asadabadi, E.B.: Novel two-stage hybrid neural discriminant model for predicting proteins structural classes. Biophysical Chemistry **128**(1), 87–93 (2007). https://doi.org/https://doi.org/10.1016/j.bpc.2007.03.006, https://www.sciencedirect.com/science/article/pii/S0301462207000749

5. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf
6. Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., Inman, D.J.: 1d convolutional neural networks and applications: A survey. Mechanical Systems and Signal Processing **151**, 107398 (2021). https://doi.org/https://doi.org/10.1016/j.ymssp.2020.107398, https://www.sciencedirect.com/science/article/pii/S0888327020307846
7. Michaud, S.A., Pětrošová, H., Sinclair, N.J., Kinnear, A.L., Jackson, A.M., McGuire, J.C., Hardie, D.B., Bhowmick, P., Ganguly, M., Flenniken, A.M., Nutter, L.M.J., McKerlie, C., Smith, D., Mohammed, Y., Schibli, D., Sickmann, A., Borchers, C.H.: Multiple reaction monitoring assays for large-scale quantitation of proteins from 20 mouse organs and tissues. Communications Biology **7**(1), 6 (2024). https://doi.org/10.1038/s42003-023-05687-0, https://doi.org/10.1038/s42003-023-05687-0
8. Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C.L., Ma, J., et al.: Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Proceedings of the National Academy of Sciences **118**(15), e2016239118 (2021). https://doi.org/10.1073/pnas.2016239118, https://www.pnas.org/doi/full/10.1073/pnas.2016239118, bioRxiv 10.1101/622803
9. Shou, W.Z., Zhang, J.: Recent development in high-throughput bioanalytical support for in vitro admet profiling. Expert Opinion on Drug Metabolism & Toxicology **6**(3), 321–336 (2010). https://doi.org/10.1517/17425250903547829, https://doi.org/10.1517/17425250903547829, pMID: 20163321
10. Smalley, J., Xin, B., Olah, T.V.: Increasing high-throughput discovery bioanalysis using automated selected reaction monitoring compound optimization, ultra-high-pressure liquid chromatography, and single-step sample preparation workflows. Rapid Communications in Mass Spectrometry **23**(21), 3457–3464 (Nov 2009). https://doi.org/10.1002/rcm.4264
11. Wang, X., Zheng, Z., Ye, F., Xue, D., Huang, S., Gu, Q.: Diffusion language models are versatile protein learners. In: International Conference on Machine Learning (2024)