UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Erik Otović

BINARY CLASSIFICATION OF PEPTIDES USING DEEP NEURAL NETWORKS AND TRANSFER LEARNING

DOCTORAL DISSERTATION

Rijeka 2024.

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET

Erik Otović

BINARNA KLASIFIKACIJA PEPTIDA KORIŠTENJEM DUBOKIH NEURONSKIH MREŽA I UČENJA PRIJENOSOM ZNANJA

DOKTORSKI RAD

Mentor: izv. prof. dr. sc. Goran Mauša
Komentor: doc. dr. sc. Daniela Kalafatović

Rijeka 2024.

UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Erik Otović

BINARY CLASSIFICATION OF PEPTIDES USING DEEP NEURAL NETWORKS AND TRANSFER LEARNING

DOCTORAL DISSERTATION

Rijeka 2024.

Doctoral dissertation supervisor: Assoc. Prof. Goran Mauša, PhD (Faculty of Engineering, University of Rijeka)
Doctoral dissertation cosupervisor: Assist. Prof. Daniela Kalafatović, PhD (Faculty of Biotechnology and Drug Development, University of Rijeka)

The doctoral dissertation was defended on ______________ at the University of Rijeka, Faculty of Engineering, Croatia, in front of the following Evaluation Committee:

1. Prof. Kristijan Lenac, PhD (Faculty of Engineering, University of Rijeka) - Committee Chair
2. Prof. Marina Ivašić-Kos, PhD (Faculty of Informatics and Digital Technologies, University of Rijeka)
3. Prof. Tell Tuttle, PhD (Department of Pure and Applied Chemistry, University of Strathclyde)

This research was funded by the Croatian Science Foundation/Hrvatska zaklada za znanost projects *UIP-2019-04-7999* (Design of short catalytic peptides and peptide assemblies (DeShPet)) and *DOK-2020-01-4659*.

This research utilized the resources of the supercomputer Bura from the University of Rijeka, Center for Advanced Computing and Modelling.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor Assoc. Prof. Goran Mauša, PhD and my co-supervisor Assist. Prof. Daniela Kalafatović, PhD for their invaluable guidance, encouragement, patience and mentorship throughout my doctoral studies and during the writing of this doctoral dissertation.

A big thanks to my DeShPet lab colleagues for answering all my chemistry questions and for the stimulating discussions we had.

Thanks to all my co-workers for the drinks we shared, and for making my office a lively place with their frequent visits. Thank you Marko for your support and endless discussions. And, of course, thanks to Arian for playing music when I needed it.

Thank you to my friends who have been by my side throughout these years.

Special thanks to my family for their unwavering support, encouragement, and for always believing in me. This would not have been possible without you.

ABSTRACT

Machine learning is increasingly used for high-throughput peptide screening, providing a rapid and efficient method to identify peptides with desired functions in contrast to traditional trial-and-error approaches that are time-consuming and resource-intensive. It streamlines the exploration of the vast peptide space in a data-driven way and accelerates the discovery of novel peptides.

This thesis investigates three dominantly used peptide representation schemes and analyzes them based on the type of information they capture. Considering that machine learning models require input to be in a numerical form, the choice of peptide representation scheme is crucial as it can directly influence model performance. Therefore, a novel *sequential properties* representation scheme is proposed to address gaps identified in existing schemes. Additionally, a manually curated dataset comprising 126 peptides evaluated for the catalysis of ester and phosphoester hydrolysis is presented.

The experimental evaluation of four peptide representation schemes in combination with deep neural networks was conducted using antimicrobial, antiviral and catalytic datasets. Results on the antimicrobial and antiviral datasets were used for statistical tests and to draw reliable conclusions due to their diversity and size. Statistical tests applied across seven evaluation metrics demonstrated that the introduced sequential properties scheme significantly outperformed other representations in 90% of cases. The antimicrobial and antiviral datasets were downsampled to create smaller target datasets to assess the effectiveness of transfer learning. Results showed that knowledge transfer was beneficial only when transferring from the more diverse antimicrobial dataset encompassing multiple subfunctions to less diverse antiviral dataset, improving the ROC-AUC score by 6.9% with statistical significance. Moreover, the results show that the transfer learning model outperforms the baseline model by more than 1% when the target dataset contains fewer than 275 peptides.

Keywords: peptide function prediction, peptide representation schemes, sequential properties, catalytic peptides, transfer learning

PROŠIRENI SAŽETAK

Strojno učenje se sve više koristi za visoko-propusno pregled peptida, pružajući brzu i učinkovitu metodu za identifikaciju peptida sa željenim funkcijama za razliku od tradicionalnih pristupa baziranih na postupku pokušaja i pogreške koji zahtijevaju puno vremena i resursa. Također pojednostavljuje istraživanje velikog prostora peptida metodama strojnog učenja koje su vođene dostupnim podacima te ubrzava otkrivanje novih peptida.

Ova doktorska disertacija razmatra tri dominantno korištene sheme predstavljanja peptida i analizira ih na temelju vrsta informacija koje obuhvaćaju. Uzimajući u obzir da modeli strojnog učenja zahtijevaju ulazne podatke u numeričkom obliku, odabir sheme predstavljanja peptida je ključan odabir jer može imati izravni utjecaj na učinak modela. Stoga, nova shema predstavljanja nazvana *slijedne značajke* je predložena u ovoj disertaciji s ciljem premošćivanja identificiranih nedostatak u postojećim shemama. Dodatno, u disertaciji je predstavljen skup podataka koji se sastoji od 126 ručno prikupljenih peptida ispitanih za katalizu hidrolize estera i fosfoestera.

Provedeno je eksperimentalno ispitivanje četiri sheme predstavljanja peptida u kombinaciji s dubokim neuronskim mrežama korištenjem antimikrobnih, antivirusnih i katalitičkih skupova podataka. Rezultati na antimikrobnom i antivirusnom skupu podataka korišteni su za statističke testove i donošenje pouzdanih zaključaka zbog raznolikosti i veličine tih skupova podataka. Statistički testovi primijenjeni na sedam metrika vrednovanja pokazali su da je predstavljena shema slijednih značajki statistički značajno nadmašila ostale sheme u 90% slučajeva. Smanjene inačice antimikrobnog i antivirusnog skupa podataka korištene su kao odredišni skupovi podataka za ispitivanje učinka učenjem prijenosom znanja. Rezultati pokazuju da je poboljšanje ostvareno samo pri prijenosu znanja iz raznovrsnijeg antimikrobnog skupa podataka, koji obuhvaća nekoliko podfunkcija na antivirusni skup podataka, na manje raznoliki antivirusni skup rezultirajući u statistički

signifikantnom povećanju ROC-AUC metrike za 6.9%. Povrh toga, rezultati su pokazali da model baziran na prijenosu znanja ostvaruje učinak veći od 1% u usporedbi sa modelom koji nije koristio prijenos znanja kada ciljni skup podataka sadrži manje od 275 peptida.

Ključne riječi: predviđanje funkcije peptida, shema predstavljanja peptida, katalitički peptidi, učenje prijenosom znanja

CONTENTS

Acknowledgements **III**

Abstract **V**

Prošireni sažetak **VII**

1. Introduction **1**

1.1. Functional Peptides 2

1.2. Data-Driven Approaches for Peptide Discovery 4

1.3. Scientific Hypotheses and Contributions 9

1.4. Research Methodology 10

1.5. Thesis Structure Overview 12

2. Machine Learning **13**

2.1. Supervised Machine Learning 15

2.2. Artificial Neural Networks 16

2.3. Overfitting Control 22

2.4. Convolutional Neural Networks 23

2.5. Word Embeddings 24

2.6. Recurrent Neural Networks 25

2.7. Evaluation Metrics 28

2.8. Cross-Validation 32

2.9. Leave-One-Cluster-Out 33

2.10. Feature Selection 34

2.11. Transfer Learning 35

2.12. Statistical Tests 40

3. Representation Schemes 43
3.1. Peptide Properties 45
3.2. One-Hot Encoding 48
3.3. Word Embedding 50
3.4. Sequential Properties 52

4. Datasets and Preprocessing 55
4.1. Feature Scaling 55
4.2. Similarity 56
4.3. Catalytic Dataset 60
4.4. Antiviral Dataset 67
4.5. Antimicrobial Dataset 68

5. Representation Schemes Evaluation 75
5.1. Models 75
5.2. Evaluation Procedure 77
5.3. Analysis of Feature Selection 83
5.4. Statistical Analysis 86

6. Transfer Learning 91
6.1. Evaluation Procedure 91
6.2. Statistical Analysis 94
6.3. Optimal Transfer Learning Strategy 98

7. Conclusion 101

Bibliography 105

List of Figures 115

List of Tables 119

List of Code Listings 123

List of Abbreviations 125

Appendix **127**

A. Feature Selection Results 129

B. Optimal Hyperparameters Found During Pretraining 141

Curriculum Vitae **143**

List of Publications **145**

1. Chapter

INTRODUCTION

Rapid hardware development and exponential growth in computational power facilitated the adoption of advanced optimization algorithms to tackle complex problems that were once beyond reach. Such approaches led to the automatic exploration of solutions to problems in various fields, including bioinformatics. Computer-based tools have been used to analyze large amounts of experimental data, recognizing patterns, predicting properties, conducting simulations, and much more. For example, AlphaFold 3 is a recently developed deep learning system capable of predicting the structures of proteins, DNA, RNA and ligands [1]. By formulating the exploration of the chemical space as an optimization task with quantifiable objectives, the computational methods enabled scientists to explore it more expeditiously and systematically. This intractable space is challenging to comprehend and might remain out of reach without the aid of advanced computational techniques. Moreover, the integration of machine learning (ML) systems, robotics and digital technologies is powering self-driving laboratories that operate autonomously [2, 3]. They enabled streamlined and automated exploration of the chemical space by performing synthesis, characterization and testing tasks while saving resources and time. As they are powered by ML, they may discover unconventional solutions that experts might not have considered. Hence, the improvements made in ML amplify the capabilities of self-driving laboratories and expedite research in general.

Peptides are short chains of amino acids that play various roles in biological processes, and the exploration of peptide chemical space presents a challenging combinatorial problem due to the vast number of possible sequences. As an illustration, a sequence with

10 residues limited only to the 20 natural amino acids covers 10^{20} unique peptides. The extensive peptide search space highlights the need for sophisticated methods to navigate it efficiently [4]. Understanding the correlation between a peptide's sequence and its function is crucial for identifying novel functional peptides and effectively navigating the peptide space. Peptide function refers to the ability of a peptide to perform specific biological or chemical functions which is influenced by its amino acid sequence and environmental factors. Consequently, data-driven methods have been employed to model and predict this relationship, guiding the discovery of peptides with desirable properties. For instance, antimicrobial peptides have been thoroughly investigated using these methods, demonstrating their potential in drug discovery and therapeutic applications [5, 6, 7, 8, 9].

1.1. Functional Peptides

Therapeutic peptides encompass a wide range of bioactive peptides and they have been extensively investigated because of their potential in various medical applications [10, 11]. Most notably, it includes a category of antimicrobial peptides that play a crucial role in the host's innate immunity against a broad range of microorganisms. Antimicrobial peptides are further categorized into subcategories according to the type of pathogen they combat, such as antiviral and antifungal peptides [12].

Peptides naturally occur within the human body and play an important role in various biological processes. They act as hormones, growth factors, neurotransmitters, ion channel ligands, and anti-infective agents. The research of peptide therapeutics began by exploiting the potential of natural peptides, such as insulin [10]. However, the scope of peptide therapeutics research has since expanded beyond peptides found in nature, especially when it became possible to synthesize them in the laboratory. Furthermore, understanding the function of the peptide and its interaction with their targets can provide valuable insight, thereby aiding in the design of more efficient drugs.

Even though antibiotic resistance is naturally occurring, it has recently been rising more rapidly due to the overuse of traditional antibiotics. This has led to an interest in therapeutic peptides, which are seen as next-generation drugs that could replace traditional antibiotics. Therapeutic peptides offer several advantages over small molecules. They have a high binding affinity and high target specificity, which enables them to bind strongly to the designated target while reducing off-target binding and interaction with non-target organisms [13]. As a result, they have fewer side effects. They also have a low risk of toxicity and low drug-drug interaction, which means that their effectiveness is unlikely to be influenced by other drugs the patient may be taking [13]. Compared to small molecules, peptides can be easily modified to improve their function or enhance their properties.

Peptides are usually broken down into shorter sequences or individual amino acids by enzymes in human serum, or filtered out by the kidneys and excreted from the body. This effect is usually expressed through a peptide half-life which is defined as the amount of time it takes for a peptide concentration in the body to be reduced by half. For instance, the half-life of insulin, which is used to control blood glucose levels, is between 4 and 6 minutes [14]. For most peptides, the half-life is usually between a few minutes and a few hours [15]. This short half-life limits the discovery of novel peptide therapeutics and reduces their applicability and effectiveness, since peptides have a short time to reach the designated target. Furthermore, this can also require frequent dosing to maintain therapeutic levels in the body. However, such problems are commonly circumvented by introducing non-natural amino acids, chemical modifications, and cyclization which can significantly extend the half-life of therapeutic peptides, enhancing their potential as effective treatments [16].

Empirically, novel candidate sequences can be obtained by modifying the existing ones according to the researcher's experience and intuition. Such discovery and optimization of peptide sequences is resource and time-intensive as it is based on a trial-and-error approach requiring multiple rounds of experimental validation. Furthermore, such an approach also introduces a human bias towards specific patterns and methods as the approach also depends on the researcher's past experiences. Although modification of known peptides may lead to peptides similar to the original ones with more preferable properties, completely new peptides with unique sets of properties might be missed.

While empirical methods rely heavily on researchers' intuition and a time-consuming trial-and-error process, homology search represents a more systematic, albeit rudimentary, extension of these empirical approaches. It leverages sequence alignment and similarity metrics to identify active peptides [17, 18, 19]. This technique is based on the premise that peptides with similar sequences may exhibit similar functions. However, it is limited to identifying peptides that resemble those already present in the database and may not effectively uncover novel peptides with distinct sequence features or functions. Consequently, it does not fully address the challenge of identifying entirely new peptide sequences with unique sequence features.

Molecular dynamics (MD) simulates the physical movements and interactions of molecules within a fixed time frame, providing insight into the dynamics of the system. It has been used to explore the behavior of peptides in the presence of bacterial membranes, which can be used as an indicator of antimicrobial function [20]. MD has also been employed to derive 3D features from the results of structural simulations which can then be used as input for data-driven methods [21]. MD simulations usually require powerful hardware and typically take hours or days to complete. Furthermore, a new simulation has to be set up and run for each compound separately, making these simulations unsuitable for the virtual screening of a large number of compounds.

Although homology search and MD simulations complement the trial-and-error approach in peptide discovery, they do not aim to exploit and generalize the sequence-function relationship to a broad range of novel peptide candidates. As a result, these methods are not capable to efficiently explore the vast peptide space. In contrast, ML methods, being data-driven, offer greater consistency and objectivity, reducing human bias and enabling the discovery of novel peptides from previously unexplored regions. By generalizing observations from databases of characterized peptides, data-driven methods can significantly enhance the search for novel peptides.

1.2. Data-Driven Approaches for Peptide Discovery

The query shown in Code listing 1.1 was used to find publications indexed in Web of Science that employ ML methods for peptide function prediction. It can be seen from Figure 1.1 that the number of such publications has increased over nine times in the 10 years from 2013 to 2023, indicating that ML techniques have attracted interest recently. These methods have the capacity to learn patterns from existing data and apply them to unseen data. They are capable of handling non-linear relationships in the data and deep neural networks (DNNs) can even automatically extract features relevant to the task. This also represents a shift in the discovery paradigm from rational design to data-driven

List of Code Listings 1.1: Query used to find publications that simultaneously mention *machine learning*, *deep learning* or *neural network* and *peptide* in their abstracts or titles.

```
AB = (("machine learning" OR "deep learning" OR "neural network")
        AND "peptide") OR
TI = (("machine learning" OR "deep learning" OR "neural network")
        AND "peptide")
```

design, thus reducing human bias towards specific patterns or approaches.

Traditional ML methods, such as the support vector machine (SVM) and the random forest (RF), have been widely used for peptide function prediction [22, 23, 24]. SVMs are frequently employed due to their simplicity and widespread availability in many machine-learning libraries. As they maximize the margin between classes, the generalization between classes is encouraged and can be used to highlight peptide features relevant for prediction. In comparison, a RF is an ensemble of decision tree models that together produce the final prediction which reduces variation and improves accuracy. Decision trees are generally considered to be explainable and transparent models as they are composed of simple conditions that are used to make a prediction. This makes it possible to trace the sequence of decisions that lead to a final prediction. Nevertheless, expert knowledge in the domain of application is necessary to engineer the informative features that these models require. More complex models based on DNNs have started to emerge recently due to their ability to capture complex interactions between high-level input features and automatically learn the necessary intermediate representations, which removes the need for manual feature engineering [25]. However, such an increase in complexity is followed by an increase in computational resources and they are less interpretable. These models typically require large quantities of labeled data containing both active and inactive peptides for training. Collecting large amounts of experimentally verified peptides can be challenging due to the lengthy process and financial costs associated with their experimental validation. Furthermore, researchers often publish only active peptides, leading to a scarcity of inactive peptides reported in the literature. Having positive and negative instances in a dataset is crucial for training an accurate ML model, and class balance helps prevent bias towards one class during model training.

As ML models require their input data to be numerical, it is necessary to use a representation scheme to transform peptides into a numerical representation. Various

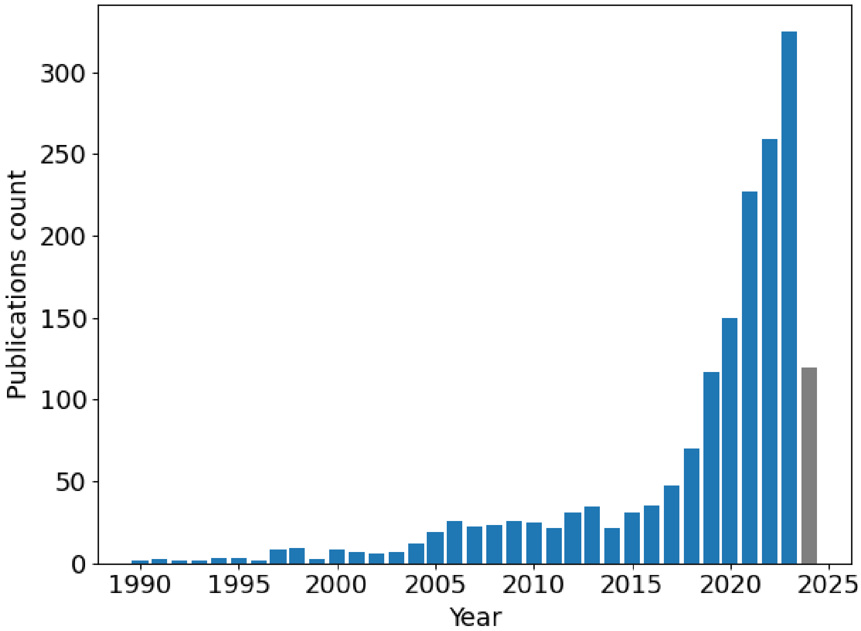
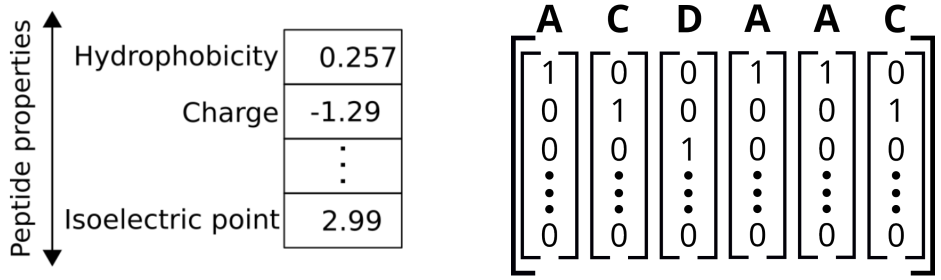


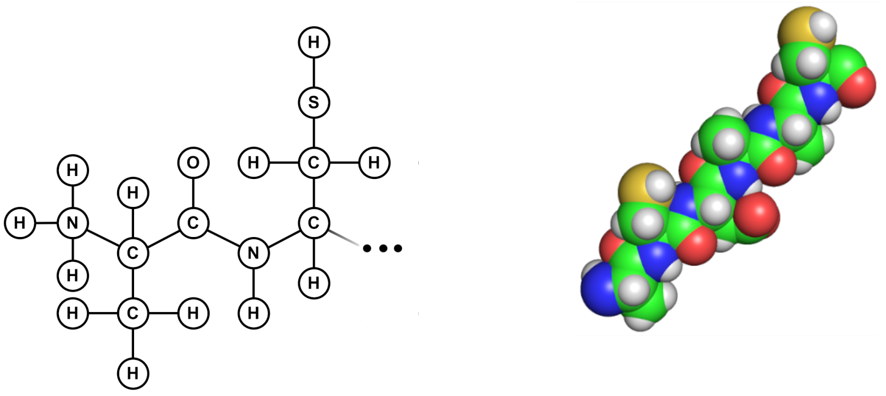
Figure 1.1: Number of publications per year indexed in Web of Science that satisfy query shown in Code listing 1.1. The 2024 (gray) data is incomplete as it was retrieved on July 11$^{\text{th}}$, 2024.

representation schemes have been proposed, and the model's performance depends on the chosen scheme's informativeness [4, 26, 27, 28]. The broad categorization of representation schemes according to how they structure and encode data is shown in Figure 1.2. Representations based on peptide properties (Figure 1.2a) encode a peptide by a set of properties that usually encompass physico-chemical and compositional properties. Such representations are well-suited for traditional models like RF, SVM, but also for multilayer perceptron (MLP) neural networks. Sequence-based representations (Figure 1.2b) encode a sequence of amino acids that allow a model to learn patterns directly from the sequence. This makes them particularly suitable for sequence-modeling tasks using recurrent neural networks (RNNs). Graph-based representations (Figure 1.2c) encode peptides as graphs based on their molecular structure, where atoms are represented as nodes and bonds are represented with edges. Such representations can capture spatial and topological information and graph neural networks are a logical choice for this type of representation. Image- and model-based representations (Figure 1.2d) capture the structural and spatial properties of peptides. Whereas 2D images provide a view of the peptide from a single perspective, 3D models offer a more informative representation that



(a) Peptide properties-based representations.

(b) Sequence-based representations.



(c) Graph-based representations.

(d) Image- or model-based representations.

Figure 1.2: Broad categorization of peptide representation schemes according to how they encode data, illustrated using the peptide sequence ACDAAC.

includes occluded parts of the molecule and is independent of viewpoint. For example, such representations can be obtained by exporting 2D images or 3D models with or without a time component from MD simulations and are typically used in combination with convolutional neural networks (CNNs).

There are two common ML strategies for exploring peptide space. The first involves building a screening system that requires a human-provided list of peptides in order to identify promising peptides, while the second approach is oriented towards building a system that generates promising peptide candidates with a desired function. Once an ML model is trained, it is capable of making predictions relatively quickly, enabling it to scan large numbers of peptides for specific functions. This makes ML a viable option for high-throughput virtual screening systems, which can accelerate the identification of promising peptides for experimental verification, reducing time and financial costs associated with the discovery of novel peptides.

Generative models such as generative adversarial networks and variational autoencoders have been applied recently to generate candidate peptide that have a high chance of being active. The former approach involves two competing neural networks, where one generates novel peptides while the other tries to discriminate real from the generated ones. As they compete, the generating network becomes better at generating peptides that have properties similar to the ones in the dataset. In the latter approach, an encoder network compresses the peptide representation into a lower-dimensional representation (latent space) capturing features essential for peptide activity. Simultansously, a decoder network is trained to reconstruct the original peptides from the latent space. Active peptides tend to cluster together in latent space which can be exploited to generate new peptide sequences that are likely active by sampling points from those regions. The decoder is then used to translate these sampled points into peptide sequences. Furthermore, some studies have employed global search algorithms, such as the genetic algorithm, in combination with a trained ML model [29]. The search algorithm improves its set of solutions through generations by using the ML model to measure the fitness of each solution. In this hybrid approach, a ML model, which is not capable of generating novel peptide candidates by itself, is used to guide the search algorithm towards active candidates.

Although predictive and generative ML models perform different tasks, they have many components and challenges in common. Both types of models share the most of architecture design principles and neural network modules. Both approaches also face the same challenges in terms of data preprocessing and require the choice of an adequate representation scheme. Therefore, by improving one of the approaches, other approaches benefit as well. Furthermore, existing models can be used to mitigate these challenges, and their configurations may provide starting points for building more advanced models.

1.3. Scientific Hypotheses and Contributions

Multiple peptide representation schemes have been explored and utilized for ML-based peptide function prediction [4, 24]. The most dominant schemes used in deep learning are analyzed in terms of the information they capture and their predictive performance. This thesis introduces a novel representation scheme named *sequential properties*, designed to bridge an information gap identified in existing representation schemes. The introduced scheme enhances peptide representation and consequently improves the accuracy of antimicrobial, antiviral and catalytic peptides prediction. The effectiveness of the introduced scheme is confirmed on antiviral and antimicrobial datasets that contain 599 and 4877 instances, respectively. Furthermore, the new representation scheme is applied in the transfer learning (TL) setting to mitigate the identified challenge of ML application to peptide function prediction with a scarce amount of data available. The approach is evaluated on small datasets of various sizes that are sampled from two previously mentioned datasets.

A manually curated dataset containing experimentally verified catalytic peptides is presented in this thesis. Being the first dataset of its kind, it will allow for the application of ML methods to the research of catalytic peptides and facilitate their discovery. The previously devised representation scheme and TL methodology are applied to the small catalytic dataset comprised of 86 sequences to create a neural network-based model for the prediction of the catalytic function.

Based on this, the following two hypotheses have been defined:

1. **The integration of a deep learning model with a proposed peptide representation scheme improves the classification performance of active and inactive peptides on antimicrobial and antiviral datasets.**

2. **TL enables the transfer of knowledge gained on a large dataset to a small one in order to achieve higher predictive performance.**

This thesis makes three contributions to the field of ML-based peptide function prediction, namely:

1. A novel hybrid representation scheme that combines physico-chemical properties of individual amino acids with the information on the ordering of amino acids in a sequence and overcomes the identified gap in the literature.

2. A manually curated dataset of catalytic peptides that have been experimentally verified for ester hydrolysis.

3. A RNN-based model for the classification of catalytic peptides for ester hydrolysis.

1.4. Research Methodology

The research methodology of this thesis is designed to test the given hypotheses and achieve defined contributions to the field. It consists of four main phases, as described below.

In the first phase, relevant sources of peptides for antiviral and antimicrobial functions were identified. The peptide sequences from these data sources were extracted to compile an antiviral and an antimicrobial dataset. Furthermore, a dataset of catalytic peptides was created by manually collecting sequences from the literature. For all three cases, each peptide had to satisfy three conditions to be included in the final dataset for ML: (i) it had to be experimentally verified to have or not have a certain function, (ii) peptide had to have a maximum length of 50, and (iii) it had to contain only natural amino acids. The sequence similarity of the peptides within each dataset was analyzed, as high similarities could bias the model. The subsequent handling of each dataset depended on the computed similarity. A detailed analysis of a newly created catalytic dataset was performed in terms of its statistical, compositional and physico-chemical properties and SMILES notation was provided for peptides containing only natural amino acids.

In the second phase, commonly used representation schemes and their corresponding models were identified. Each scheme was analyzed in the context of the information it captures, as well as its advantages and disadvantages compared to the other schemes. For peptide properties-based representation schemes, MLP neural networks are commonly employed, while RNNs and CNNs are often used for sequence-based representations. The schemes are discussed in the context of an information gap identified among them, and a novel representation scheme named sequential properties that combines the physicochemical properties with the order of amino acids in which they appear in the sequence was introduced to bridge this gap.

In the third phase, the binary classification task for peptide function prediction was introduced and a methodology for comparing the introduced representation schemes was devised. Depending on the dataset size, 10 times repeated stratified 10-fold cross-validation or 20 times repeated leave-one-cluster-out cross-validation was used for evaluation. DNNs for the prediction of peptide function were developed and this included the identification of the optimal feature set for each dataset and representation scheme separately, and the optimization of the model architecture. The introduced representation schemes were compared in the context of model performance using multiple evaluation metrics for binary classification and their sensitivity to the classification threshold. Furthermore, tests of statistical significance were used to identify the best-performing model.

In the fourth phase, a TL methodology was devised and the benefits and problems associated with TL were explored. This methodology focused on comparing the effectiveness of TL to the non-TL approach when a target dataset is relatively scarce. To test the hypothesis, small antiviral and antimicrobial datasets were artificially created by sampling the previously collected datasets, while the catalytic dataset was used in its entirety. TL was analyzed in the context of the improvement in model performance and compatibility among the selected three peptide functions. The relationship between the relative improvement and the size of the target dataset was also investigated to determine the maximum target dataset size up to which TL enhances performance. Statistical tests were used to confirm the difference in performance metrics between non-TL and TL model and to draw conclusions about TL effectiveness.

1.5. Thesis Structure Overview

Chapter 1. provides an introduction to the field of discovery of novel active peptides and defines the basic concepts. The problem of peptide function prediction is defined and ML approaches for peptide function prediction are briefly discussed.

In Chapter 2., the problem of predicting peptide function is put in the context of supervised ML task. Relevant theoretical ML and deep learning concepts, evaluation procedures, metrics, and statistical tests that will be employed in this thesis are introduced.

Chapter 4. introduces the problem of peptide sequence similarity. The datasets used in the thesis are introduced, and their statistical properties and similarities are discussed. A manually curated dataset of catalytic peptides is presented, and its properties are comprehensively analyzed.

Chapter 3. introduces three widely used peptide representation schemes and analyzes the information they capture. An information gap present in these representation schemes is identified and a novel representation scheme called *sequential properties* is introduced to bridge this gap.

In Chapter 5., the procedure used to evaluate representation schemes and employed models is detailed. An extensive analysis is conducted on the results of sequential properties, focusing on predictive performance and optimal hyperparameters.

The TL evaluation methodology and the corresponding results are presented in Chapter 6. Furthermore, the optimal TL strategy found by a grid search is analyzed and the dependency of TL effectiveness on the target dataset size is explored.

Finally, Chapter 7. provides a summary of the research conducted in this thesis and highlights the potential future directions for improving peptide function prediction.

2. Chapter

MACHINE LEARNING

Artificial intelligence is a broad field that focuses on the development of computer systems capable of mimicking human intelligence and performing tasks that would typically require human intelligence. For example, this includes tasks related to natural language processing, decision making, image and speech recognition, autonomous driving, game playing and medical diagnosis. Machine learning (ML) is a subset of artificial intelligence which encompasses algorithms that use statistical learning to learn from and model data. Unlike traditional problem-solving methods, these systems are not explicitly programmed on how to perform a task. Instead, they learn from a dataset, and once built, they are used to make predictions or decisions on unseen data.

A dataset is a collection of instances or data points representing some observation. An instance is characterized by a set of independent variables, also known as features or attributes. The selection of informative and discriminative features is crucial for a model to learn the mapping from these features to target values. Furthermore, it is important to use high-quality features as they directly condition model performance. The process of designing, selecting and transforming features with the aim of enhancing model performance is referred to as feature engineering and is often a part of the ML process.

Features can be broadly categorized into numerical and categorical types based on their nature. Numerical features are those that use numerical values, which can be either continuous or discrete. On the other hand, categorical features represent categories. These can be further subdivided into ordinal and nominal features. Ordinal features are those where an order exists between the categories, while nominal features do not imply any order.

In ML, especially when dealing with complex models such as DNNs that have a large capacity, there is a risk that the model might memorize instances seen during training. This means that the model fits the training data too closely, and this phenomenon is known as overfitting. It can lead to overly optimistic results when the model is evaluated on the same data on which it was trained, as it might make correct predictions for the training data, but fail on unseen data. To ensure a fair and unbiased evaluation of the model, it is necessary to split the dataset into at least two sets known as a training set and a test set.

The training set is used to train the model, while the test set is kept aside for the final evaluation of the model at the end of the experiment. This ensures that the model never comes into contact with the test set during training, simulating real-world data that the model may encounter once deployed. However, it is often necessary to estimate model performance during the experiment for various reasons, such as hyperparameter optimization. In such cases, an additional set of data is introduced, known as a validation set. For example, it can be used to evaluate multiple model configurations and to select the one that produces the highest score on the validation set. The test set cannot be used for this purpose because any choices made based on the test set results would be biased towards optimizing the score on the test set. In that case, the final evaluation on the test set would not reflect the performance that can be expected in production.

When splitting the data, a random split may result in training, validation, and test sets that do not have an equal ratio of classes. This can be especially problematic for datasets with high class imbalances as any of the sets may end up not being representative of the whole population. This can be especially important in terms of the minority class. To address this issue, a stratified split can be used, which guarantees that the ratio of classes in the dataset will be preserved across all splits. The split is usually 70:15:15 for the training, validation and test sets, respectively. The largest portion of the dataset is assigned to the training set because model training requires significant quantities of data. It is important that all sets are representative of the population to ensure that the model learns effectively and generalizes well to new data.

Various ML algorithms exist, including neural networks, decision trees, SVMs and many more. Deep neural networks (DNNs) are a specific type of neural network that consists of at least two layers. A subfield encompassing DNNs is known as deep learning. DNNs offer multiple advantages over other ML algorithms, usually at the cost of computational complexity and a higher demand for training data. Due to their deep and complex architecture, they are able to learn and better model non-linear relationships in the data. Furthermore, they can automatically learn complex features from the data in tasks where it is difficult to define features. Over the past decade, DNNs have achieved state-of-the-art performance on such tasks, which include image classification, natural language processing and speech recognition.

ML algorithms are usually divided into four categories:

1. Supervised learning - involves using labeled data, where both inputs and expected output values are provided. The task of the model is to learn to map the input to the output values.

2. Unsupervised learning - only input data is provided without the output. The goal of the model is to learn the structure of the data and is often used for clustering and dimensionality reduction.

3. Semi-supervised learning - a combination of supervised and unsupervised learning in which most data is not labeled. It is often used when it is costly to collect labeled data, but unlabeled data are available and can be used to improve the model.

4. Reinforcement learning - a model interacts with the environment by making decisions and receives rewards or penalties. The goal of the model is to learn a policy that maximizes rewards.

2.1. Supervised Machine Learning

Since the goal of this thesis is to develop a model that can distinguish between peptides that exhibit a certain activity and those that do not, the peptide activity prediction problem is treated as a supervised binary classification task. In the supervised learning setting, each instance in the dataset is assigned one or more dependent variables, also known as output or target variables. The objective of supervised learning is to learn to predict the target variables based on the input features. Depending on the nature of the problem and the dependent variables, supervised learning can be divided into classification and regression tasks.

In a classification task, the goal is to assign instances into specific categories. Binary classification is a subtype of classification in which an instance can belong to one of two classes. In contrast, multi-class classification allows an instance to belong to one of multiple classes. Furthermore, if an instance can belong to multiple categories simultaneously, the problem is known as multi-label classification as multiple labels are assigned to each instance. In peptide activity prediction, binary classification is performed more often, as most of the available datasets provide only a label indicating whether a peptide has a certain activity(Table 2.1). Such datasets are mostly used for binary, but they can also be compiled together for multi-label classification [27, 30, 31]. For example, a two-step method was developed for the identification of antimicrobial peptides and their functional subtypes [32]. In the first step, a binary classification model is used to distinguish antimicrobial from non-antimicrobial peptides. In the second step, 14 binary classification SVM and XGBoost models are used to perform multi-label classification based on a one-vs-rest approach. In such a setting, each classifier in the second step is trained to distinguish only one functional subtype from the other subtypes. Such implementation enables a single sequence to belong to multiple classes at the same time.

In contrast to classification tasks, models in regression tasks learn to map the input features to a continuous numerical value representing peptide activity, which introduces the requirement for the dataset to provide such values for training. For instance, a regression-based model for the prediction of IC_{50} values for antiviral peptides was developed [33]. The authors experimented with SVM, RF, k-Star, and k-NN. They evaluated various combinations of compositional and physico-chemical properties as well as one-hot encoding. The results showed that combining compositional, physico-chemical and binary features correlated with better prediction.

2.2. Artificial Neural Networks

Artificial neural networks (ANNs) were selected for this study due to their ability to model complex and non-linear relationships in data, automatically learn peptide rep-

Table 2.1: Overview of related research articles that employed ML for the peptide function prediction. **CO**=compositional properties, **OE**=one-hot encoding, **PC**=physico-chemical properties, **EV**=evolutionary properties, **EM**=embedding

Research article	**Task**	**Max. similarity**	**Model**	**Representation**	**Dataset size**
Wei et al. [27]	Anticancer peptides prediction	90%	SVM	CO OE	500 164 (test)
Boopathi et al. [30]	Anticancer peptides prediction	80% 90%	SVM LR RF k-NN	CO OE PC	532 314 (test)
Qureshi et al. [33]	IC50 prediction for antiviral peptides	Unknown	SVM RF k-Star k-NN	CO OE PC	683 76 (only testing)
Gull et al. [32]	Prediction of antimicrobial activity (14 subactivities)	40%	SVM XGBoost	CO	786
Hu et al. [34]	Peptide MHC class I binding affinity prediction	Not employed	CNN	EV	186 685
Yu et al. [31]	Anticancer peptides prediction	90%	BiLSTM CNN CNN+BiLSTM	EM	500 164 (only testing)
Beltran et al. [35]	Antimicrobial, anticancer and bacteriocin peptides prediction	Not employed	RF k-NN SVM MLP	PC	8 220 6 935 2 612 1 497 737 147

resentations, extract useful patterns and generalize them to the entire peptide, achieve state-of-the-art performance, and can be used to leverage the knowledge gained on one task to improve the performance on the other task. ANNs are inspired by the human brain, although they differ significantly in the way they learn. ANNs consist of artificial neurons that can be categorized into input neurons that receive the data, hidden neurons which transform the data, and output neurons which produce the final result. Each neuron consists of multiple input connections x, weights w assigned to the input connections, an activation function f, and a single output y. Input connections are connected to the outputs of other neurons, while one of the inputs may also be used as a bias term. Weights are usually initialized to random values to break the symmetry. The neuron first computes the weighted sum of the inputs z as shown in Equation 2.1, and then applies the activation sum to obtain the output value as shown in Equation 2.2.

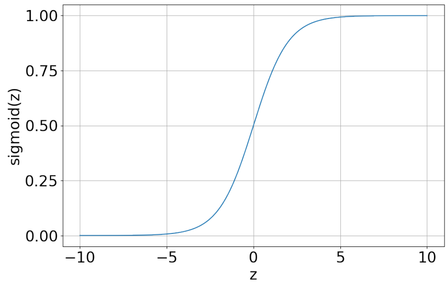
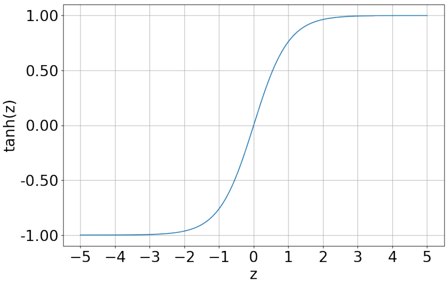
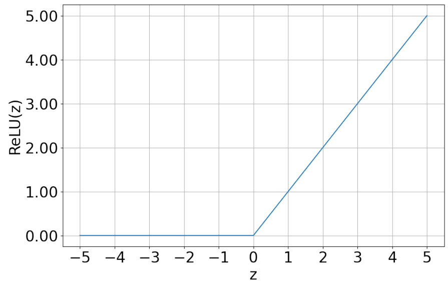
$$z = \sum_i x_i w_i \tag{2.1}$$

$$y = f(z) \tag{2.2}$$

The activation function is usually a non-linear function which transforms the computed linear combination of inputs z to obtain the final output y. Without using non-linear activation functions, a neuron would perform a linear transformation. It can be mathematically shown that a neural network which does not employ non-linear activation functions, regardless of the number of neurons, can be summed up into a single neuron exhibiting the same behavior. A non-linear behavior is introduced into the neural network by using non-linear activation functions which allows for modeling of non-linear functions. Generally speaking, neural networks with more neurons are capable of modeling more complex functions.

Various activation functions exist and the ones used in this thesis are shown in Table 2.2. Each one has its own characteristics and the choice of activation function depends on its purpose and mathematical properties. For example, the sigmoid function is often used for the output of binary classification neural network where the expected output value is between 0 and 1, representing the probability that an instance belongs to a positive class. The probability of instance belonging to the negative class can be simply

Table 2.2: Definition, domain, co-domain and visualization for sigmoid, tanh and ReLU activation functions.

Activation function	Domain	Co-domain	Visualization
$sigmoid(z) = \frac{1}{1+e^{-z}}$	$[-\infty, \infty]$	$[0, 1]$	
$tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$[-\infty, \infty]$	$[-1, 1]$	
$ReLU(z) = max(0, z)$	$[-\infty, \infty]$	$[0, \infty]$	

computed by subtracting the obtained probability from 1. If the predicted probability is above the customary classification threshold, the instance is classified in the positive class. The classification threshold is usually set to 0.5, but it can be optimized for better performance. Tanh is usually used when the expected output value should be centered around zero. ReLU is computationally more efficient, outputs only unbounded positive values and unlike tanh and sigmoid, does not suffer from the vanishing gradient problem. It is a common choice for the activation function of hidden neurons.

Even though the connections between neurons can be arbitrary and unstructured, they are usually arranged into layers and this type of neural network is known as multi-layer perception (MLP). Such a layered structure enables a simpler and computationally efficient learning process, while at the same time multiple layers facilitate the learning of more complex functions.

The learning process consists of the forward and the backward pass. In the forward pass, instances from the training set are propagated through the neural network to compute the outputs. After each instance x_i, a loss function L is computed to measure the prediction error as a discrepancy between the predicted and the ground truth values y_i. During the backward pass, the backpropagation algorithm is used to propagate the prediction error back through the network to adjust the weights and reduce the error. The chain rule (Equation 2.3) is used to automatically compute the gradient of the error with respect to each weight. The partial derivative of the loss function with respect to each of the weights is computed for the current set of weights W, input features x_i and expected output values y_i. The partial derivatives form a vector known as the gradient of the loss function as shown in Equation 2.4 and point in the direction of the steepest increase in the loss. Therefore, the loss can be minimized by updating the weights in the opposite direction of the gradient. The update rule for a single weight w_j is given in Equation 2.5. The magnitude of the update to the weights W is influenced by the magnitude of the gradient G and a customary hyperparameter α, as it can be seen from Equation 2.5. With an appropriately set α, the process eventually converges to a local optimum after multiple iterations of the algorithm. This algorithm is known as Stochastic Gradient Descent (SGD) and serves as a basis for more advanced training algorithms.

$$[f(g(x))]' = f'(g(x)) \cdot g'(x) \tag{2.3}$$

$$G = (\frac{\partial L(W; x_i, y_i)}{dw_0}, \frac{\partial L(W; x_i, y_i)}{\partial w_2}, ..., \frac{\partial L(W; x_i, y_i)}{\partial w_{|W|}}) \tag{2.4}$$

$$w_j \leftarrow w_j - \alpha \cdot \frac{\partial L(W; x_i, y_i)}{\partial w_j} \tag{2.5}$$

One such advanced algorithm is the widely used adaptive moment estimation (Adam) algorithm. Adam introduces momentum, allowing the direction of the gradient update from the previous iteration to influence the current update. This accelerates convergence in the relevant direction by dampening oscillations. Adam also uses a moving average of squared gradients to normalize the gradient and adjust the learning rate adaptively for each weight in the network. Instead of updating the weights after each instance, it employs mini-batches where a batch of instances is used to compute the gradients. These gradients are then averaged to obtain the gradient for the weight update. This approach leads to a better estimation of the direction of the gradient, less noisy updates, and consequently, faster convergence.

Categorical cross-entropy and binary cross-entropy loss functions are usually used in classification tasks. These loss functions are based on the concept of entropy, which is a measure of uncertainty in the context of information theory. A lower entropy signifies a lower degree of uncertainty. Therefore, the objective of the training process is to minimize this uncertainty by minimizing the loss function.

Categorical cross-entropy loss is given in Equation 2.6 and is commonly used for multi-class classification tasks. Each instance in the dataset is associated with a target y that contains C elements, each corresponding to a specific class. Only one position in y is assigned a value of one, given that classes in multi-class classification are mutually exclusive, while all other positions contain zero. The position with a value one assigned, signals the class to which this instance belongs. The activation function for the output layer of the model is typically a softmax function, which ensures that the sum of all predicted values equals one. Consequently, the predicted values in $\hat{y}$ can be interpreted as a probability distribution indicating the likelihood that a given instance belongs to each class.

$$L(y, \hat{y}) = -\sum i_C y_i log(\hat{y}) \tag{2.6}$$

Binary cross-entropy loss, on the other hand, is a special case of categorical cross-entropy loss and is utilized for binary classification tasks. There are only two classes in binary classification, and since the sum of their predicted probabilities equals one, knowing the probability of one class allows for the computation of the probability of the other class. Therefore, the variable y is a single number, which equals one if the instance belongs to the positive class and zero if it belongs to the negative class. The variable $\hat{y}$ is a single real-valued number denoting the predicted probability for the positive class. By considering these facts and applying Equation X, Equation 2.7 for binary cross-entropy loss is derived.

$$L(y, \hat{y}) = -(1 - y)log(1 - \hat{y}) - ylog(\hat{y}) \tag{2.7}$$

2.3. Overfitting Control

Overfitting is a common problem in ML where a model learns the training data and its noise so closely that it negatively impacts the model's performance on unseen data. It can be identified by evaluating the model on a validation set that was not used during training. If the model performs well on the training set but poorly on the validation set, it is likely overfitting.

Various strategies exist to combat overfitting. One approach is to reduce the complexity of the model by decreasing the number of parameters. This prevents the model from learning overly complex patterns and memorizing the training data. Regularization techniques, such as L1 and L2 penalties, can also be introduced into the loss function to discourage the model from becoming too complex. Another regularization technique is dropout, where randomly selected neurons are temporarily dropped out during backpropagation and their connections are not updated. This makes the neural network less sensitive to the activations of specific neurons and promotes better generalization. The dropout rate is a user-defined probability that controls the number of randomly selected neurons to be deactivated. Another strategy is to implement early stopping which monitors the model's performance on a validation set during training. Once the performance on the validation set did not improve for a defined number of epochs, the training is halted to prevent the model from overfitting. Additionally, the model can be saved during each epoch so that the one with the best performance on the validation set can be used. However, it is important to note that applying overfitting control methods too aggressively can lead to underfitting, where the model performs poorly on both the training and validation sets due to its inability to learn the underlying patterns in the data. Therefore, it is important to strike a balance between underfitting and overfitting and often requires experimentation.

2.4. Convolutional Neural Networks

Some types of data, such as images or audio recordings, have a spatial dependency where the context of each input value is influenced by its surrounding values. For example, the ordering of pixels in images or samples in audio recordings carries meaningful information on the surrounding values that influence how the current value is interpreted. Exploiting these dependencies is often necessary to achieve good performance in tasks involving such data.

Convolutional layers are usually used in neural networks to specifically exploit these dependencies. Each convolutional layer consists of a set of learnable filters that are randomly initialized at the beginning and then learned during the training process. Each filter slides over the input data and a dot product is computed at each step. This operation produces an activation map that gives the response of that filter at each spatial position. The activation maps of all filters are then stacked along the depth dimension to produce the output of the convolutional layer.

For example, suppose 1D data with $n_{features}$ features (i.e. channels) and a length of l_{input} and a convolutional layer with $n_{filters}$ filters, where each filter is of length l_{filter}. The output of a convolutional layer will contain $n_{filters}$ features and will have a length of l_{input}. The length of the output may vary because it depends on the step size chosen for the sliding operation, as well as the handling of edge cases. Output features represent sequences containing activations of the associated filter.

One of the benefits of convolutional layers is their ability to automatically learn and extract features from raw data. This is particularly useful in situations where it is challenging to manually engineer relevant features, such as in image processing. Furthermore, this increases the adaptability of the model as convolutional layers can automatically learn the patterns most relevant for the prediction, leading to higher prediction performance. The fact that each filter is used across the entire length of the input provides better generalizability since each filter is learned from the entire input data.

The receptive field of a convolutional layer is the part of the input data that is covered by the layer in a single step and it plays a crucial role in the ability of the model to detect certain patterns in the input data. If the receptive field is too small, it may not be enough to capture relevant patterns. On the other hand, if the receptive field is too large, it may miss important information in the input as it aggregates the information. The receptive field of a neural network can be increased by either increasing the size of the convolutional filters or by stacking multiple convolutional layers. It is usually preferred to stack multiple convolutional layers as this facilitates hierarchical feature learning where layers closer to the input data learn simple features, while deeper convolutional layers combine those simple features into more complex ones.

2.5. Word Embeddings

Word embeddings concept refers to the technique of representing categorical values as multi-dimensional real-value vectors. In such a method, each categorical value is mapped to a unique vector that is learned automatically during the training process. During prediction, word embeddings act as lookup tables that convert a categorical value into a real-value vector that represents the given categorical value. Although word embeddings are most commonly used in natural language processing, they can also be applied in many other fields where categorical values are used. For example, in cases where categorical features have a large number of possible values, embedding layers may be used to more efficiently represent categorical values in comparison to the one-hot vector encoding.

There are various implementations of word embeddings, such as Word2Vec and FastText, that differ in the way they are trained and in the type of context they capture. These approaches tend to group semantically similar words closer in the embedding space, allowing for subsequent analyses. However, these approaches require separate training before they can be used in a downstream task.

In many research papers in the field of peptide function prediction, an already-implemented embedding layer from a ML library is often used [31, 36, 37]. In this thesis, an implementation of word embedding from the Keras library is used. It represents each categorical value as an n-dimensional vector, which means that $n \cdot m$ weights must be learned to represent m different categorical values.

Unlike the aforementioned approaches that require separate training, a Keras embedding layer is trained simultaneously with the rest of the neural network in which it is used. This is achieved by optimizing the embedding layer weights during the training process towards the values that minimize the loss. Although this implementation

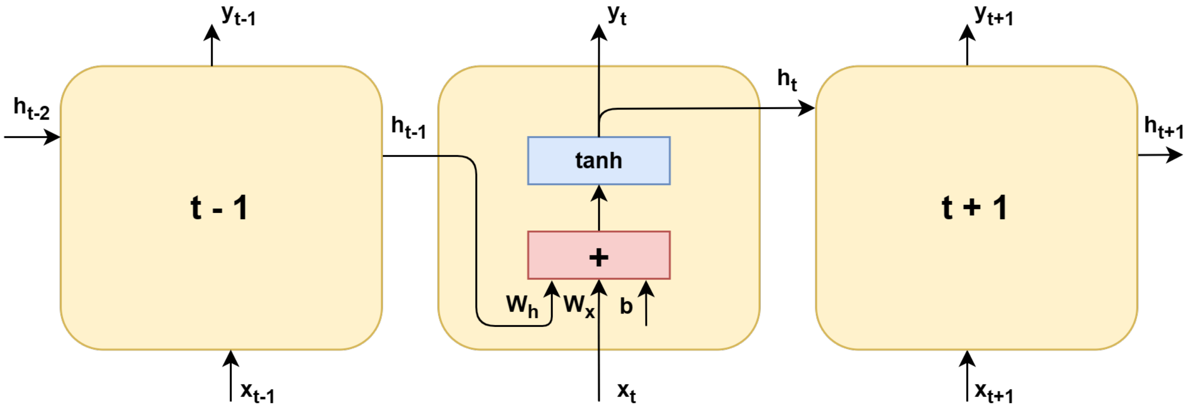


Figure 2.1: The schematic of standard RNN.

allows for the simultaneous training of the embedding layer and the neural network, it does not group semantically similar categorical values together. Therefore, a meaningful subsequent analysis of the embedding space is not possible with this approach.

2.6. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a specialized type of neural network designed for modeling single- or multi-channel sequential or temporal data. For example, they are usually used for tasks such as machine translation, natural language processing, and speech recognition, but they are also applicable to peptide function prediction [31, 36, 38, 39]. RNNs process sequences one step at a time, maintaining a memory in the form of a numerical vector that contains the results of computation up to the current step. The next state of the memory is computed from its current state and the current input. This sequential processing makes RNNs difficult to parallelize and potentially computationally intensive, especially when dealing with long sequences. RNNs use the same weights at each time step to update the memory and compute the output. This weight-sharing approach significantly reduces the complexity of the model and the number of parameters that need to be learned. Furthermore, it allows the network to make predictions on inputs of variable sizes.

The most basic form of an RNN is a *standard RNN*, which is depicted in Figure 2.1. In this model, memory is implemented as a hidden state. It is computed at each time step and then fed as input at the next step, along with the current input data. The size of the hidden state vector, denoted as $hidden\_size$, is a hyperparameter that needs to be configured and depends on the complexity of the task. The next hidden state is obtained by summing the previous hidden state, input for the current sequence step, and bias vector and applying tanh activation function element-wise. Matrix multiplication of the hidden state and the current input with their weight matrices W_h and W_x is performed before the bias vector is added. The matrix W_h has a dimension of $(hidden\_size, hidden\_size)$, while matrix W_x has dimensions of $(hidden\_size, input\_size)$ where $input\_size$ denotes the size of the input vector x. Therefore, the result of matrix multiplication with vectors h_{t-1} and x_t are the vectors of size $hidden\_size$. Now they can be summed together element-wise since they have the same size as the bias vector b. The memory implemented in the form of hidden state h_t is passed to the next sequence step along with the input data x_{t+1} to compute the next hidden state h_{t+1}. The hidden state in standard RNN also serves as the output from RNN. Therefore, after each sequence step is processed, a corresponding output y_t is also computed. The matrices W_h and W_x, and a bias vector b are randomly initialized and converge to optimal values during model training.

This type of RNN implementation suffers from vanishing and exploding gradient problems [40]. In the case of vanishing gradients, the gradient becomes smaller as the sequence becomes longer, leading to very small updates to the weights. This reduces the RNN's ability to learn long-term dependencies, resulting in poor performance. On the other hand, exploding gradients occur when the gradients become larger with the length of the input sequence, leading to divergence during training. These problems are influenced by the size of weight values, learning rate, sequence length and activation functions. The problem of exploding gradients can be mitigated by a technique known as gradient clipping.

A more advanced variation of RNN is the Long Short-Term Memory (LSTM), shown in Figure 2.2. Unlike a standard RNN, an LSTM additionally implements a cell state C_t, which enables long-term memory capabilities. It has three mechanisms that control its content: the input gate controls the amount of input data that should be stored in the cell state, the forget gate controls the amount of cell state that should be kept, and the output gate controls the amount of cell state that should be passed to the output at the current sequence step. These mechanisms allow the LSTM to mitigate the problem of vanishing gradients.

A limitation of traditional RNN and LSTM networks is that they process the input

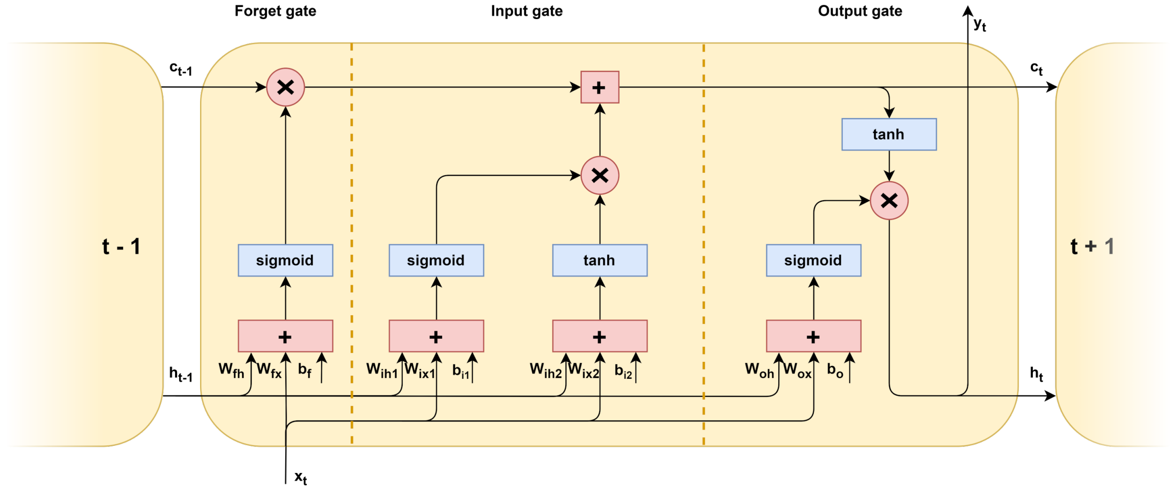


Figure 2.2: The schematic of LSTM.

sequence in one single direction. This can be restrictive for applications where understanding the data requires context from both the past and the future. This issue can be addressed by implementing a bidirectional RNN, which in the case of LSTM is known as Bidirectional LSTM (BiLSTM). BiLSTM mitigates this limitation by incorporating two LSTM layers: one processes the sequence from start to end (forward direction), and the other processes it from end to start (backward direction). The outputs of these two layers are then concatenated to form the final output, which effectively captures information from both past and future contexts.

When training neural networks, it is necessary for all sequences in a single batch to be of the same size. This requirement presents a challenge that can be addressed in several ways. One approach is to use SGD which updates the weights after each data instance, thereby avoiding the problem by eliminating the need for batches. However, this method usually has a slower convergence and is more sensitive to noise. Another approach is the bucketing method, where a batch is formed of sequences of the same length. This method reduces randomness as sequences of different lengths cannot appear together in a batch, potentially leading to batches that are not representative of the overall population. Furthermore, scarce sequence lengths will produce small batches. A third approach is to pad sequences with a special token to make them of the same length. While padding introduces additional computational complexities, it enables parallel processing by using batches and allows for greater flexibility as sequences of different lengths can be used together.

The output of an RNN at each step represents a rich, context-aware representation of the sequence data processed so far. Since the cell and hidden states do not depend on the length of the input sequence, the output from the RNN is also of a fixed size. This means that RNNs generate a new sequence at the output that is the same length as the input sequence, which can be used for sequence-to-sequence tasks. However, for applications relevant to this thesis, only the final output is used as it contains the accumulated information from the entire sequence. Therefore, standard RNNs or LSTMs are usually followed by fully connected layers. In this setting, the RNN part processes the sequential data and outputs a fixed-size vector containing information about the sequence. This vector can then be used by the fully connected layers to make a prediction. The purpose of RNN in this approach is to process the sequential data, while the fully connected part makes a prediction.

2.7. Evaluation Metrics

An evaluation metric is a quantitative measure used to assess the predictive performance of a model. Classes in binary classification are usually referred to as positive and negative classes, where the positive class is usually the class of interest. In the context of this thesis, active peptides are treated as a positive class because they exhibit the desired activity.

Binary classification tasks yield four possible outcomes for each prediction:

1. True Positive (TP) - the model correctly predicted the instance to be positive.
2. False Positive (FP) - the model incorrectly predicted a negative instance to be positive and it is also known as type 1 error.
3. True Negative (TN) - the model correctly predicted the instance to be negative.
4. False Negative (FN) - the model incorrectly predicted a positive instance to be negative and it is also known as type 2 error.

These four outcomes can be arranged into a 2D confusion matrix, as shown in Table 2.3, and serve as the basis for deriving more complex classification metrics.

Table 2.3: Confusion matrix for binary classification.

		Predicted	
		Positive	**Negative**
Ground truth	**Positive**	**TP**	**FN**
	Negative	**FP**	**TN**

For an ideal classifier, the confusion matrix would contain only true positives and true negatives, while the counts of false positives and false negatives would be zero, indicating that all instances have been correctly classified.

It is necessary to use multiple evaluation metrics to evaluate the model from various perspectives since each metric has its own advantages and disadvantages. This thesis will employ eight metrics, namely recall, precision, specificity, accuracy, F-1 score, Matthew's correlation coefficient, geometric mean and area under the receiver operating characteristic curve.

Accuracy (Acc), is a commonly used metric to assess the predictive performance of classification models. It is defined as shown in Equation 2.8 and represents the fraction of predictions that the model predicted correctly. However, it is usually used to compare the models on balanced or nearly-balanced datasets as it can produce over-optimistic or misleading results in the case of imbalanced datasets. For example, consider a model predicting only a negative class and an imbalanced dataset having 90% of instances in the negative class and 10% of instances in the positive class. Such a model would have an accuracy of 90% on a given dataset, even though it incorrectly classified all of the positive instances. It can be seen that the class imbalance influences the accuracy and makes it unsuitable for comparison of models across datasets with different distributions of classes.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.8}$$

In imbalanced datasets, the class of interest is usually greatly outnumbered by the negative class and assessing the predictive performance solely with accuracy can be misleading. This issue is tackled by defining three scores each being focused only on one aspect of prediction which are later used to define more complex scores for imbalanced datasets. Precision measures the proportion of correctly predicted positive instances out of all instances that the model predicted as positive. A high precision score indicates fewer false positives and is especially useful when the cost of a false positive is high. Recall reflects the proportion of correctly predicted positive instances out of all actual positive instances. This metric is crucial when the cost of false negatives is high as it measures the model's ability to capture positive instances in the data. Similarly to recall, but for the negative class, specificity assesses the model's ability to correctly identify the members of the negative class. It is crucial when the cost of false positives is high. All three metrics output the value in the range from 0 to 1 and they are defined in Equations 2.9, 2.10 and 2.11, respectively.

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.11}$$

F-1 score (F1) is a harmonic mean of precision and recall and is shown in Equation 2.12. It is more robust than accuracy for an imbalanced dataset because it considers how accurately the model predicts positive class (precision) and how good it is at identifying positive instances (recall). Therefore, predicting the positive class for all instances would result in lower precision and consequently in a lower F-1 score. F-1 score assumes that precision and recall are equally important which may not be true for all problems. Furthermore, it also assumes that it is more important to correctly identify the positive class over the negative class since it does not include true negatives in the calculation.

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{2.12}$$

Geometric mean (GM), shown in Equation 2.13 is another robust metric that computes the geometric mean between recall and specificity showing how good the model is at identifying the members of positive and negative classes. Unlike the F-1 score, it assumes that the ability to correctly identify positive and negative classes is equally important, as it includes both recall and specificity.

$$GM = \sqrt{recall \cdot specificity} \tag{2.13}$$

Matthews correlation coefficient (MCC) is a measure that takes into account all four fields of the confusion matrix and does not prefer one class or one error type over the other. Its value ranges from -1 to 1, where -1 indicates that a model misclassified all instances, while 1 indicates a perfect classification. However, a negative value in the case of binary classification denotes that all classes are opposites of the true values, which means that the score can be turned into a positive value by negating the output. Subsequently, the value of 0 in the case of binary classification suggests that the predictions are no better than a random choice.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.14}$$

The seven previously described metrics are dependent on the chosen classification threshold as they consider predicted classes that are obtained by applying the threshold to the predicted probability. On the other hand, area under the rectifier operating characteristic curve (ROC-AUC) is a measure that allows for the comparison of models without depending on any particular threshold. ROC curve shows the trade-off between $1 - specificity$ and $recall$ and is constructed by plotting those values on x and y axis for every possible classification threshold. ROC curve always starts at (0, 0) since the maximal threshold results in only the negative class being predicted which would result in $1 - specificity = 0$ and $recall = 0$. The curve always ends at (1, 1) because classifier with the threshold set to minimal value leads to only the positive class being predicted which results in $1 - specificity = 1$ and $recall = 1$. The ideal classifier would make correct predictions at any threshold in between which would produce $1 - specificity = 0$ and $recall = 1$ making the curve look like a square. ROC-AUC score is computed by computing the area under the constructed ROC curve, and higher values indicate better overall performance across all thresholds.

2.8. Cross-Validation

Cross-validation is a method used in ML to accurately assess the performance of a model by splitting the dataset into k partitions. In each of the k iterations of the algorithm, one partition is used for testing, while the remaining ones are used for training. This is done in a way that ensures that each partition is used exactly once for testing and $k-1$ times for training. This process yields k measurements, from which the average score and deviation are computed. Using a higher number of k iterations can help reduce variance of the score estimate, providing a more reliable measure of model performance. In the case of imbalanced datasets, where classes are not equally represented, a stratified split may be used. This type of split guarantees that the ratio of classes will be preserved in all folds, ensuring that each fold is a good representative of the whole dataset.

To further enhance the reliability of the performance estimate, a procedure known as repeated cross-validation can be employed. In this procedure, the cross-validation process is repeated n times, with each repetition involving a new instance of cross-validation. This results in $n \cdot k$ measurements, further reducing the variance of the estimate.

Cross-validation is often used to repeat the experiment multiple times as this allows for the conduction of statistical tests and the statistical comparison of multiple approaches. It also enables the estimation of the model's generalization ability to unseen data, the performance of the model in hyperparameter optimization and feature selection and helps to avoid overfitting.

However, the downside of cross-validation is that it can be computationally expensive, especially in the case of repeated cross-validation. This problem can be mitigated to some extent by parallelization. Since each iteration of cross-validation is independent of other iterations, multiple iterations can be run simultaneously. This requires hardware powerful enough to run multiple instances of the experiment, which may not always be feasible due to lack of access to such hardware or the costs associated with obtaining access to such hardware.

2.9. Leave-One-Cluster-Out

In peptide prediction, datasets often contain highly similar peptide sequences [41]. This similarity typically arises from manual design processes in which known active sequences are modified to create novel active sequences with potentially higher activity. For instance, a conservative substitution might replace an amino acid with another that has similar physico-chemical properties. Consequently, the original and modified sequences appear highly similar, both in sequence and in their physico-chemical properties.

This high similarity between training and test sets can affect the model's predictive capabilities and its generalization power. Clusters of similar sequences of the same class might lead the model to base predictions on patterns common to that cluster, while overlooking the differences between individual sequences within the cluster. Such a model would base its decisions entirely on the similarity of a given peptide to the peptide clusters seen during training. However, this approach has limited value in real-world applications, as there are infinitely many possible sequences that were not covered in the training set. Therefore, the main goal is to overcome the prediction based on similarity and develop a model that can learn how the interplay of amino acids influences peptide activity and generalize this knowledge to unseen data.

Simply discarding highly similar sequences from the dataset to control similarity can be problematic in the case of small datasets where a significant portion of data may be excluded further reducing already too small dataset. To avoid this issue, a method such as leave-one-cluster-out cross-validation (LOCOCV) is required for model evaluation [32]. This method groups peptides in the dataset by their similarity into clusters, which act as folds. Like regular cross-validation, each cluster is used exactly once for testing and forms a part of the training set in all other iterations.

LOCOCV employed in this thesis clusters peptides based on similarity that is later defined in Chapter 4. The algorithm starts with no clusters and iterates over the peptides in the dataset. Each peptide is checked against the representatives of existing clusters. If the similarity between a given peptide and a representative exceeds a user-defined threshold, the peptide is added to that cluster. If no suitable cluster is found, a new cluster is created, and the given peptide becomes the representative of this cluster. The number of clusters is variable and depends on the data, the user-defined similarity threshold, and the order in which peptides are processed during clustering.

While LOCOCV does not directly solve the problem of a model basing its decisions purely on similarity, it enables a realistic estimate of the model's performance on unseen data. It can be used in hyperparameter optimization to tune the hyperparameters in a way that maximizes the model's generalization power. In this way, it can indirectly mitigate the problem and encourage the model to base its decision on other factors as well. However, it is important to note that this approach can be computationally expensive for large and diverse datasets.

2.10. Feature Selection

Feature selection methods are a crucial aspect of ML as they can improve predictive performance and generalization while reducing model complexity and computational cost of training and testing. They can be categorized into filter and wrapper methods. Filter methods rank features based on statistical analysis and are not dependent on a ML model. They are computationally efficient, but they may require the data to comply with some requirements for them to be applicable. In contrast, wrapper methods use a ML model to rank features, resulting in a feature set that is optimized for that particular model. These methods are typically more flexible as they make fewer assumptions about the data, but they are more computationally expensive compared to filter methods.

In this study, a wrapper method known as sequential feature selection is employed. This method is chosen because it produces an optimal feature set for a given model and, unlike many filter methods, it is applicable to sequential data.

Sequential feature selection can operate in two directions, forward and backward. The forward search begins with an empty set of features and in each iteration, it identifies the unselected feature that contributes most to the model's performance. This process involves training and evaluating the model separately using a set of already selected features, complemented with each of the unselected features. The feature that leads to the highest increase in performance is then added to the set of selected features. Conversely, the backward search starts with a set of all features and, in each iteration, removes the feature that contributes the least to the performance. Both of these approaches are greedy and have a complexity of $\mathcal{O}(n*m)$, where n is the total number of available features and m is the number of desired features. The algorithm is based on a greedy approach and does not consider the correlation and dependencies among the features. Therefore, there is no guarantee that it will find a global optimum in the form of a feature subset that achieves the best predictive performance. Forward search is more efficient when a small number of features is desired as it has to do a small number of iterations to append the desired number of features to the set. On the other hand, backward search is more suitable when a smaller number of features has to be discarded from the feature set.

Its complexity makes sequential feature selection unfeasible when a large number of evaluations is required or when the time for a single evaluation is unacceptably long. Such problems arise when the pool of features is too large or when dealing with large models and datasets that take a long time to evaluate. Although the search can be accelerated by employing parallelization, it is usually insufficient for large problems.

The random initialization of model weights may cause oscillations in model performance, even when the model is retrained using the same data and features. To get a better estimate of model performance on each evaluated feature subset, a nested stratified cross-validation can be used. The downside of this procedure is that it increases the number of necessary evaluations by the number of folds and repetitions of cross-validation. Since each iteration of cross-validation is completely independent of the other iterations, it can be efficiently parallelized to run multiple iterations simultaneously. This approach helps to mitigate the computational cost while ensuring a robust estimate of model performance.

2.11. Transfer Learning

Transfer learning (TL) is a ML method in which a pre-trained model is used instead of a randomly initialized model as a starting point for a learning process. Therefore, TL is a process that involves two sequential stages: a model is pre-trained on a source task (S) in the first stage, and then it is fine tuned on a target task (T) in the second stage. Using a pre-trained model as a starting point usually leads to an increase in performance on the target task as well as faster convergence to the solution which in turn leads to a shorter training time. TL is often used to mitigate the challenges of small datasets as they do not contain enough data instances for a proper training of a randomly initialized neural network [42].

To devise an optimal TL strategy, one needs to address the following three questions: *When to transfer*, *What to transfer*, and *How to transfer* [43]. *When to transfer* focuses on identifying the domains and tasks that are mutually compatible for the transfer of knowledge. Tasks and domains that are subjectively deemed similar are usually seen as suitable for TL. However, not all domains and tasks are mutually compatible, or in the case of some domains, they may be compatible only in one way. Applying TL to such incompatible domains and tasks may not yield any improvement, or may even result in negative transfer which is the degradation of performance. Therefore, it is crucial to assess the compatibility of the domains and tasks before performing TL. However, most of the existing studies focus on the other two aforementioned questions, and there is no general method to assess the compatibility of the domains and tasks, except by conducting empirical experiments. TL between incompatible domains may result in a decline in target performance, a phenomenon known as negative transfer.

What to transfer question is concerned with identifying the parts of a neural network that can be reused on a target task. The knowledge in neural networks is encoded in the form of topology and weights, which define its function and behavior. In many application domains, commonly used model topologies are already established and fine-tuned for their specific needs. Therefore, these topologies are usually reused for different tasks within the same or similar domains, and only their weights must be optimized to adapt the neural network behavior. Consequently, TL is concerned with identifying parts of a neural network that can be reused from a source task to help solve the target task. A naive approach would be to search for the optimal subset of nodes and connections to transfer, but this would be computationally expensive and impractical. A more efficient approach is to leverage the insights from the topology and functioning of the network. It is known that neural networks learn hierarchical representations of the data, where shallow layers capture basic features and deeper layers capture more abstract and complex features. This means that shallow layers contain more general knowledge common to a wider range of tasks, while deeper layers contain task-specific knowledge. Hence, a commonly used strategy is to transfer the shallow layers and randomly initialize the deeper layers. This assumes that the source and target tasks share some commonality in the type of task that needs to be solved or the domain of the application.

In the context of peptide function prediction, TL has been applied to transfer knowledge across both closely related and more distinct peptide functions, as well as from proteins to peptides(Table 2.4). For example, knowledge from protein toxicity prediction has been transferred to peptide toxicity prediction, leveraging the similarity between proteins and peptides [39]. In another case, the knowledge gained on antifungal peptides datasets was leveraged to fine-tune the model for anticancer peptides prediction, exploiting common patterns within different subfunctions of antimicrobial peptides [38]. Additionally, TL has also been explored for more unrelated peptide activities, such as transferring knowledge from antimicrobial to ion channel-modulating peptides [36]. This thesis will examine two TL scenarios involving highly related tasks, specifically the transfer from AMP-ExAVP to AVPPred and vice versa, as well as the transfer of knowledge from AMP and AVPPred to unrelated CAT. Given that antiviral peptides are a subtype of antimicrobial peptides, transferring knowledge from AMP-ExAVP to AVPPred is expected to be beneficial. However, the transfer from a single-function-focused AVPPred dataset to a more diverse AMP-ExAVP dataset may not yield as good results due to the limited diversity of patterns learned on the AVPPred dataset. Although the catalytic function is unrelated to antimicrobial and antiviral functions, it is anticipated that the transfer from AMP to the CAT dataset will perform better due to the broader range of patterns learned during pretraining, compared to the transfer from AVP.

How to transfer refers to the way in which TL is performed once the relevant parts of a neural network have been identified. Various methodologies are available, depending on the level of adaptation and fine-tuning required. The simplest way is to transfer the identified layers from a source neural network and then train the whole network on a target task. However, this may cause the transferred layers to diverge from their near-optimal states and effectively lead to the loss of knowledge from the source task. This is a result of training transferred layers simultaneously with the rest of the network which is still not tuned to the target task. To minimize this, a common practice is to freeze the transferred layers so that they do not change during training, or to use a lower learning rate for them so that the changes in weights are smaller. This allows the randomly initialized part of the network to converge fast to the optimal solution, while the transferred layers preserve the learned knowledge and fine-tune gradually to the target task. The adjusted learning rate α_{TL} used in transferred layers is usually expressed as a learning rate α multiplied by

Table 2.4: Overview of related research articles that employed TL for the peptide function prediction. **CO**=compositional properties, **OE**=one-hot encoding, **PC**=physico-chemical properties, **GR**=graphical properties, **EV**=evolutionary properties, **EM**=embedding

Research article	Source task	Source dataset size	Target task	Target dataset size	TL remark	Max. similarity	Model	Representation
Lobo et al. [22]	Prediction of masked amino acids	Unknown	Antifungal peptides prediction	Unknown	Pretraining embeddings	Min. 3 amino acids difference	SVM RF LR k-NN MLP	EM
Salem et al. [23]	Embedding training	2 billion 28 357	Peptide hemolytic activity prediction	1 902	Pretraining embeddings Two stage fine-tuning	40%	Adapted BERT LLM	EM
Wei et al. [39]	Protein toxicity prediction	10 813	Peptide toxicity prediction	3864	Model pretraining Model fine-tuning	40% 90%	CNN+BiGRU	CO PC EV GR
Lee et al. [1][36]	/	/	IMPs [2] prediction	410 - 1 506	Multi-task learning (4 targets)	50% 70% - 95%	CNN BiLSTM CNN+BiLSTM	EM
	Antimicrobial peptides prediction	4 402	IMPs [2] prediction	72 - 506	Pretraining model Fine-tuning model			
Lane et al. [38]	Antifungal peptide prediction	2 880	Anticancer peptides prediction	240 331 586 740 844	Model pretraining Model fine-tuning	Not employed	CNN+LSTM	EM
Chen et al. [44]	/	/	Anticancer peptides prediction (6 targets)	158	Multi-task learning	90%	CNN	CO OE PC
Pang et al. [37]	Prediction of masked amino acids	31 million	Antimicrobial peptides prediction (7 subactivities)	22 381	BERT pretraining Model fine-tuning with frozen BERT backbone	40%	BERT LLM	EM

[1] Lee et al. investigated two settings for knowledge transfer.

[2] Ion channel-modulating peptides.

a learning rate multiplier ω, as shown in Equation 2.15. An established learning rate for this model and the task at hand can be used for alpha, while the optimal learning rate multiplier needs to be determined empirically and is typically within the range of [0.01, 1].

$$\alpha_{TL} = \alpha \cdot \omega \tag{2.15}$$

Alternatively, pretraining in a self-supervised learning fashion can be used to utilize large volumes of unlabeled proteins for model pretraining [23, 37]. It is a form of unsupervised learning in which randomly selected amino acids in protein sequences are masked and the model is trained to predict them. Such training provided the model with useful knowledge of common patterns in sequences which was later exploited for peptide classification. The advantage of this approach is that it requires only the sequences which enables it to utilize large databases of known protein and peptide sequences that have not been characterized yet. Most of the parameters of the employed Bidirectional Encoder Representations from Transformers (BERT) large language model (LLM) have been frozen during fine-tuning as the size of the target dataset was not enough for a whole-model fine-tuning. Furthermore, a two-layer neural network was appended at the end of the BERT backbone to perform classification based on representation outputted by BERT [37]. However, a downside of this approach is that LLM models require large amounts of data to train and are computationally expensive.

Multi-task learning has also been shown to be a viable option to facilitate knowledge-sharing across multiple targets for the same peptide function (Table 2.4) [36, 44]. Unlike TL which aims to leverage the knowledge from the source task to enhance the performance on a target task, multi-task learning exploits shared representation and common patterns by simultaneously training on multiple tasks allowing each task to benefit from the other tasks. It enables the data from different targets to be combined resulting in a larger and more diverse dataset. This approach allows the model to share knowledge across the tasks, leveraging the similarities between the targets. As all of the targets pertained to the same peptide activity, they are considered similar tasks with shared underlying patterns, enabling the model to effectively learn and generalize common features across tasks. However, class imbalances in the dataset may cause the multi-task learning model to be biased towards more numerous classes, affecting the performance on less frequent classes [45].

2.12. Statistical Tests

A statistical test is a formal procedure that compares observed data with a hypothesis about the data. These tests usually define two competing hypotheses: the null hypothesis and the alternative hypothesis. Depending on whether they assume the data follow a specific distribution, statistical tests can be categorized into parametric and non-parametric tests. Furthermore, these tests can operate on paired or unpaired samples. Paired samples are those where there is a one-to-one correspondence between sets of samples, and the measurements are made on the same subject with different treatments being applied.

In the context of ML, statistical tests are often used to check if there is a significant difference in performance between two or more models or the same model with different configurations. To conduct a statistical test, multiple measurements of model performance are required and these can be collected by repeating the experiment multiple times using cross-validation.

The result of a statistical test is a p-value that represents the probability of obtaining the observed data if the null hypothesis is true. If the p-value is less than a customary threshold known as the significance level α, the null hypothesis is rejected and the alternative hypothesis is accepted. Otherwise, the null hypothesis is accepted and the alternative hypothesis is rejected. Therefore, there is a probability of at most α that the null hypothesis will be rejected when it should have been accepted, which constitutes a type I error. In contrast, not rejecting a null hypothesis when it should have been rejected constitutes a type II error.

The choice of a statistical test depends on the research question, the type of data, and the assumptions that the data must adhere to. The Friedman test is a non-parametric test used to compare the means of three or more paired groups. The null hypothesis is that the means of all groups are equal, while the alternative hypothesis is that the mean of at least one group is different. If the Friedman test determines that one group differs, then a post-hoc analysis is needed to determine which groups are different from each other. One such test suitable for post-hoc analysis is the Wilcoxon signed-rank test. It is a non-parametric test used to compare two paired samples. The null hypothesis is that the median difference between pairs is zero, while the alternative hypothesis is that the difference is not zero.

When conducting a single test, the probability of falsely rejecting the null hypothesis (Type I error) is equal to the significance level α. However, the probability of falsely rejecting at least one null hypothesis, known as the Family-Wise Error Rate (FWER), increases. Consequently, this increases the likelihood of falsely identifying a significant difference, leading to a false discovery.

This can be mitigated by controlling the FWER. One of the simplest and most commonly used methods for this is the Bonferroni correction. This method adjusts the significance level α by the number of tests performed. The adjustment is shown in Equation 2.16, where α_{adj} denotes the adjusted significance level and n denotes the number of performed statistical tests.

$$\alpha_{adj} = \frac{\alpha}{n} \tag{2.16}$$

While the Bonferroni correction successfully reduces the FWER to the significance level, it is known to be quite conservative, especially when a large number of tests are performed. This leads to lower statistical power, increasing the probability of accepting the null hypothesis when it should have been rejected (Type II error).

The Holm-Bonferroni method is a modification of the Bonferroni correction that reduces the probability of Type II errors while maintaining the same FWER, making it generally more powerful than the Bonferroni correction.

The key difference between these two methods lies in how they adjust the significance level. In the Holm-Bonferroni method, a list of n p-values is first sorted from the lowest to the highest. The lowest p-value has an index of 0 and the highest p-value has an index of $n - 1$. The algorithm then iteratively checks if each p-value is lower than $\frac{1}{m-i}$, where i is the index of the p-value in the sorted list. If it is lower, the corresponding hypothesis is rejected and the algorithm proceeds. Otherwise, the corresponding and all remaining hypotheses are accepted and the algorithm finishes. In contrast to the Bonferroni correction which uses the same significance threshold for all tests, this one gradually adjusts its threshold reducing the number of Type II errors.

When multiple Friedman tests are performed, the correction is applied first to these tests. Post-hoc pairwise tests are then conducted in cases where the Friedman test found a difference, and the correction is applied to all these pairwise comparisons together. When multiple models are compared in terms of multiple metrics, this approach ensures that the probability of falsely rejecting the null hypothesis is α for the entire set of results.

3. Chapter

REPRESENTATION SCHEMES

In the context of ML, a peptide sequence can be perceived as a series of nominal values. However, these nominal values cannot be directly processed by ML models. Therefore, it is necessary to transform them into a numerical representation that can be utilized by ML models.

Peptide properties-based feature sets based on compositional, theoretical physico-chemical properties and categorical features have been widely used in a wide range of models, including support-vector machine (SVM), random forest (RF), k-nearest neighbors (k-NN), k-Star, XGBoost, linear regression (LR) and deep neural networks(Table 2.1) [27, 30, 32, 33, 35]. To date, various compositional properties have been employed, such as amino acid and dipeptide composition which reflect the frequency of each amino acid or dipeptide in the sequence, respectively. Furthermore, more sophisticated compositional features have been used as well, including g-gap dipeptide composition and composition-transition-distribution properties [27, 30]. The first considers dipeptides that are composed of amino acids separated by g residues allowing it to capture long-range dependencies. The latter approach uses physico-chemical properties of amino acids to group them and then uses this grouping to calculate the share and distribution of each group in the sequence. It also computes the percentage of transitions between groups when the sequential nature of the peptide is taken into account. Even though this encoding is still considered compositional, its heavy reliance on physico-chemical properties to compute the composition enables it to indirectly supply the model with information on the physico-chemical properties of the sequence. Furthermore, theoretical physico-chemical important for the prediction have also been used to complement compositional properties [30, 33]. One-hot encoding has been widely used to derive binary profile features indicating which amino acid occupies each position in a sequence [27, 30, 33]. While the number of compositional and physico-chemical features is not influenced by the length of the sequence, the number of such binary features is proportional to the number of residues. As ML models expect the same number of features, the commonly used approach is to pad all sequences to predefined maximum length resulting in a fixed number of binary features for each sequence. Furthermore, as the calculation of some compositional and physico-chemical properties is dependent on user-specified parameters, multiple feature sets with different parameters can be generated. For example, multiple groups of compositional properties with different user-supplied parameters have been combined and feature selection was employed to reduce the feature set and identify the most relevant features [27].

Sequence-based representations encode amino acids in the sequence to retain information about the peptide's primary structure. Although one-hot encoding has been used to derive a fixed set of binary features in peptide properties-based schemes for non-sequential models, one-hot encoding can be applied in models that are capable of directly handling sequence data. One-hot encoding and embeddings have been used to generate sequences of encoded amino acids to represent peptide sequences (Tables 2.1 and 2.4). Scoring matrices, like BLOSUM60, have been used as embeddings for amino acids. However, they can also be automatically learned by deep learning models in such a way that the embeddings are optimal for a given task. The sequential nature of these representations can be effectively exploited by deep learning models such as CNNs and LSTMs to extract patterns that can be generalized to different positions in the sequence as well as to sequences of different lengths [4]. Simpler models, such as SVM and RF, have also been used even though they do not have such generalization capabilities since they are not designed to model sequential data and do not account for the sequential nature of these representations [22].

Feature representation learning is an alternative to embedding approaches, aiming to automatically learn more informative features from a set of manually crafted features. This method has also been applied to peptide function prediction (Table 2.1) [27, 46, 47, 48]. For example, feature representation learning that combines physico-chemical, compositional properties and one-hot encoding was employed for the identification of anticancer peptides [27]. The method employed 40 SVM predictors trained on different feature subsets and their predictions formed a 40-dimensional vector, which served as a high-level representation of a given peptide. Feature selection was performed to discard uninformative and irrelevant features to make the representation more compact, which was then used for a final SVM model to make a final prediction.

This thesis explores the application of three prevalent representation schemes, which includes one peptide properties-based and two sequence-based representation schemes. Furthermore, a novel representation scheme is introduced, and compared to other three representation schemes in terms of model performance.

3.1. Peptide Properties

Peptide properties-based representation schemes represent a peptide through a set of its properties which typically encapsulate the physico-chemical and compositional characteristics of the peptide. The effectiveness of this approach depends on the inclusion of relevant and discriminative properties, which involves feature engineering and requires domain knowledge. Therefore, physico-chemical properties which significantly influence the structure, function and environmental interactions of the peptide should be included. In the context of this thesis, 10 physico-chemical and 18 compositional properties are used, resulting in a representation of each peptide by 28 numerical features.

The utilized physico-chemical properties include:

1. Three Cruciani properties - they are computed as the average value of corresponding properties of all the amino acids in the sequence. They reflect the polarity, hydrophobicity and hydrogen bonding capacity of the peptide.

2. Instability index - provides an estimate of the stability of a peptide in a test tube and is derived from the dipeptide composition.

3. Hydrophobicity - a property of molecules to repel water. In this case, it is computed on the Eisenberg scale.

4. Hydrophobic moment - a quantitative measure of the distribution of hydrophobic residues. A high hydrophobic moment indicates a peptide or protein with a high

hydrophobicity on one side and a high hydrophilicity on the other. It is required to specify a rotational angle and a window length depending on the secondary structure [49]. For alpha-helix structures, a typical rotational angle is 100°, while for beta-sheets it is expected to be between 160° and 180° [49, 50]. Short peptides can be analyzed at once, while longer peptides or proteins are analyzed using a sliding window. In this case, a default value of 100° is used for the rotational angle and 11 for the window size.

5. Aliphatic index - characterizes the relative volume occupied by aliphatic side chains. A higher aliphatic index has been found to correlate positively with the thermostability of proteins [51].

6. Net charge - the total charge of the peptide. In this thesis, it is computed for a neutral pH of 7.4 using the Henderson–Hasselbalch equation on Lehninger scale.

7. Isoelectric point - represents the pH level at which a peptide has a net charge of zero.

The classification of amino acids into 9 groups was taken from the Peptides package written in R, and is shown in Table 3.1 [52]. This is used to compute the absolute and relative occurrence of each group in the peptide, resulting in 18 compositional properties. The absolute value is computed by counting the number of amino acids present in the sequence that are from a specific group. The relative value ranges from 0 to 1 and is a ratio between the absolute count and the length of the peptides. The inclusion criteria for a group are based on the size and properties of the peptide side chains. Each amino acid is assigned to at least one group.

An example is given in Figure 3.1. The peptide ADDC is encoded into a real-value vector of a fixed size representing its physico-chemical and compositional properties.

All of the aforementioned theoretical physico-chemical and compositional properties, except the hydrophobic moment, are agnostic to the ordering of amino acids in the sequence. This is due to the facts that these properties are mostly computed as the average or the sum of individual properties, not taking the ordering into account. This means that two peptides composed of the same amino acids, but in different order, cannot be distinguished based on order-agnostic peptide properties. Such a representation makes all

Table 3.1: Classification of amino acids into groups.

Group	**Amino acids**
Tiny	A, C, G, S, T
Small	A, C, D, G, N, P, S, T, V
Aliphatic	A, I, L, V
Aromatic	F, H, W, Y
Non-polar	A, C, F, G, I, L, M, P, V, W, Y
Polar	D, E, H, K, N, Q, R, S, T
Charged	D, E, H, K, R
Basic	H, K, R
Acidic	D, E

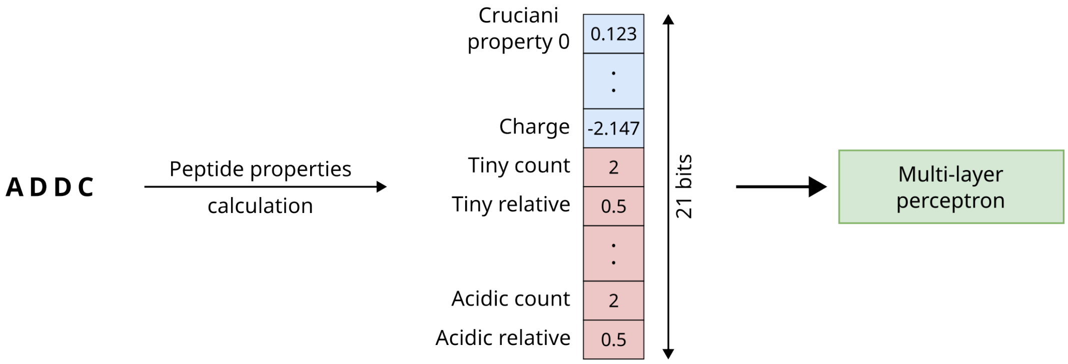


Figure 3.1: Peptide sequence ADDC encoded using its physico-chemical and compositional properties which are shown in blue and red, respectively.

Table 3.2: Physico-chemical properties of two peptides with the same amino acid composition, but with the different ordering of amino acids. All the examined physico-chemical properties have equal values except the hydrophobic moment marked in bold.

Property	**Peptide $ACCD$**	**Peptide $ACDC$**
Cruciani property 1	-0.2650	-0.2650
Cruciani property 2	-0.6475	-0.6475
Cruciani property 3	-0.0775	-0.0775
Instability index	165.5	165.5
Boman property	1.0875	1.0875
Hydrophobicity (Eisenberg scale)	0.075	0.075
Aliphatic index	25	25
Isoelectric point (Lehninger scale)	2.994992	2.994992
Charge (Lehninger; pH=7.4)	-1.289589	-1.289589
Hydrophobic moment	**0.2444636**	**0.3993655**

peptide permutations appear to be the same to the model. Such loss of information on the ordering of amino acids limits the model in its performance as it is known that the ordering influences the activity of the peptide as well. An example of two peptides with the same composition is given in Table 3.2.

3.2. One-Hot Encoding

One-hot vector encoding is a widely used technique to transform categorical data into a numerical form that ML algorithms can process. It involves creating a binary vector for each possible value that a categorical variable can take. Each vector consists of n binary variables, where n represents the number of possible categorical values. A specific categorical value is encoded by setting its corresponding binary variable to one, while all other variables are set to zero. Consequently, each categorical value is mapped to a unique binary vector.

Peptides can be viewed as sequences of nominal variables, which may be encoded into a sequence of binary vectors using one-hot vector encoding, with each vector representing a single amino acid. For instance, if we consider only 20 natural amino acids and a peptide sequence of length l, the peptide can be encoded into a matrix of dimensions $(20, l)$. An example of encoding for peptide ADDC padded to the length of 6 is shown in Figure 3.2. It is encoded as a sequence of 6 sequence steps that are processed sequentially by a sequence modeling neural network.

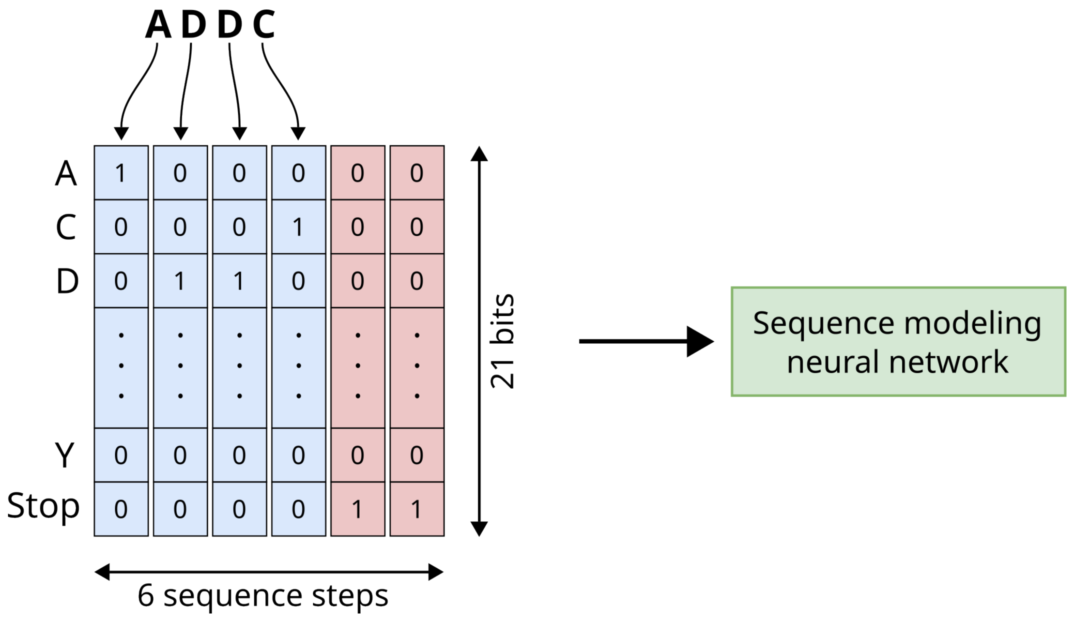


Figure 3.2: Peptide sequence ADDC encoded using one-hot vector encoding and padded to the length of 6. Vectors encoding amino acids are depicted in blue color, while vectors used for padding are depicted in red color. First 20 bits are used to encode the amino acids, while the 21^{st} bit encodes a stop signal.

One of the advantages of one-hot encoding is its simplicity and that it does not require any additional information about the categorical variable beyond the possible values it can take. Furthermore, this representation indirectly carries the compositional information. However, this encoding method also has some drawbacks. The size of the matrix used to represent a peptide is proportional to the number of amino acids used. While the number of rows is limited to 20 if only natural amino acids are used, this number can quickly grow to hundreds if non-natural amino acids are included. This can lead to high memory consumption, slower execution times, and may require a more complex prediction model. Furthermore, the resulting matrix is sparse, with most elements being zeros, which is an inefficient use of computational resources and memory. This type of representation preserves only the information on the ordering and composition of the amino acids in the sequence, while the information on physico-chemical properties is not available.

To handle sequences of varying lengths, an additional categorical value to represent a stop signal is added to the encoding and used for padding the sequence to the required length. Therefore, after the last amino acid is encoded, the sequence is appended with binary vectors where the stop signal is set to one. This enables to represent all sequences with the matrices of the same dimensionality.

3.3. Word Embedding

Even though feature representation learning enabled the automatic learning of high-level peptide representation to some degree, the application of deep learning and natural language processing methods enabled the learning of more complex representations. The word embedding approach preserves information on the ordering of amino acids and, indirectly, the composition of the peptide sequence. Since this approach adapts the representation scheme to the data and is influenced by the relationships between amino acids, it may be assumed that it is more informative than one-hot encoding. Nonetheless, this approach does not provide any information on any peptide property to the model. In contrast to one-hot vector encoding, where all amino acids are equidistant in the feature space and are treated equally, the word embedding approach exploits the relationships between amino acids and captures their similarities to position them in the feature space. However, the captured similarities may not necessarily correlate to the physico-chemical properties.

This approach requires additional processing steps, which are handled by a tokenizer and a word embedding layer. The role of the tokenizer is to convert one-letter amino acid codes into tokens that are then converted by the word embedding layer into continuous real-value vectors. As a result, a peptide is represented as a sequence of vectors, each representing a single amino acid. While predefined binary vectors are used in one-hot encoding, the word embedding approach automatically learns the representations. To handle peptide sequences of different lengths, a new token is introduced which is used for padding and also serves as a stop signal. Its representation is also automatically learned during training. This ensures that all sequences, regardless of their original length, can be represented with the same dimensionality. An example for peptide ADDC is given in Figure 3.3. The sequence is first padded to the length of 6 by appending it with the padding token. Embedding matrix is then used to encode each token in the sequence into a real-value vector.

One of the advantages of this approach is that the representations are learned automatically, removing the need for manual feature engineering. The only requirement for their application is knowledge about the possible categorical values in the input. The effectiveness of word embedding was investigated in conjunction with a BiLSTM model, a

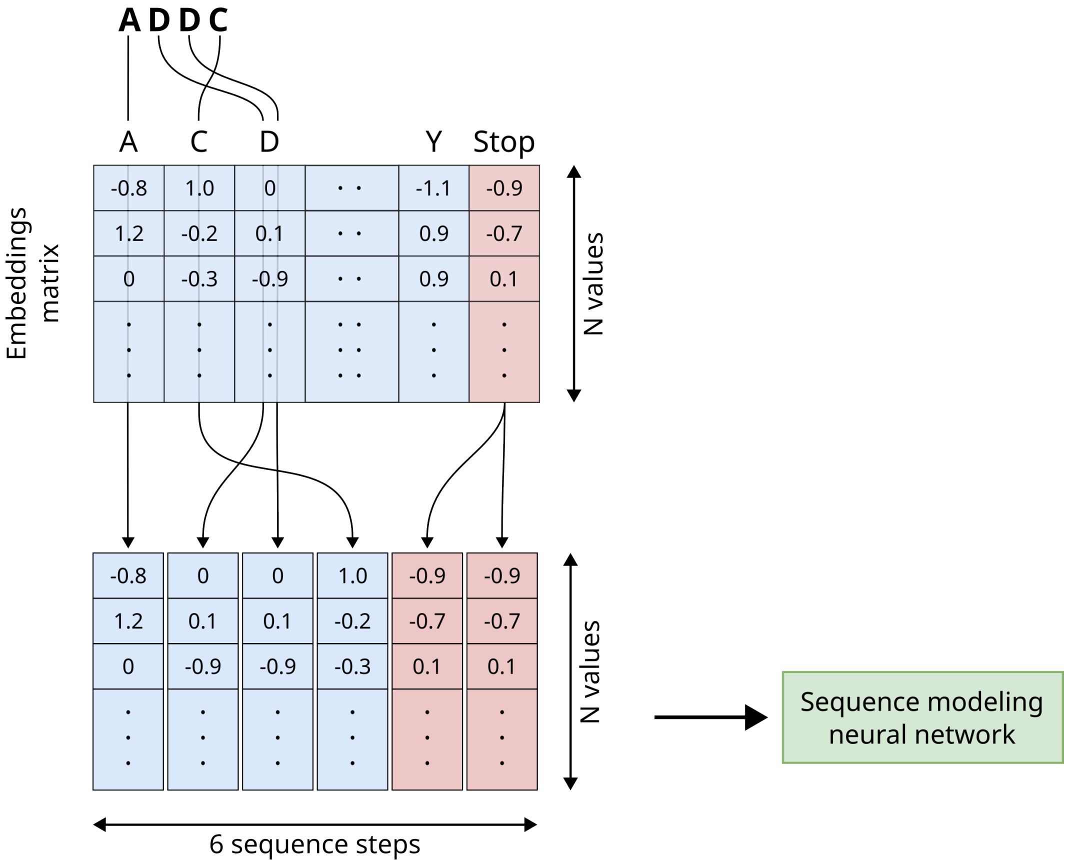


Figure 3.3: Peptide sequence ADDC encoded using word embeddings and padded to the length of 6. Amino acids and stop signals are encoded with values from the embeddings matrix. Vectors encoding amino acids are depicted in blue color, while vectors used for padding are depicted in red color. The size of the embeddings is controlled by the variable N.

CNN model and their combination which consisted of CNN followed by LSTM [31]. The results indicated that the combination of word embedding with the LSTM model yielded the best results. However, this approach can be biased by the distributions in the learning data, and training becomes more computationally expensive as the representations also have to be learned. Furthermore, as deep learning methods require large amounts of data in general, their performance can heavily be influenced by the quality and size of the training datasets.

Multiple algorithms are available for training the word embedding layer and this thesis will employ the one implemented in Keras ML library. It is trained at the same time as the rest of the neural network as a part of end-to-end training. The learned representations capture the semantic meaning of the amino acids that are optimized for the task. Furthermore, amino acids similar in the context of a given task are usually grouped closer together in the embedding space. Other unsupervised learning-based embedding algorithms, such as Word2Vec and FastText can be employed as well. However, unlike the embeddings employed in this thesis, they require separate training of the embeddings. Once trained, they can be applied across a wide range of prediction tasks [53, 54].

3.4. Sequential Properties

It has been shown that theoretical physico-chemical and compositional properties are in most cases agnostic to the ordering of amino acids [26]. This does not enable the model to distinguish between different peptides of the same composition. On the other hand, amino acid ordering-based representation schemes, while preserving the information on the ordering of amino acids in the sequence, they lack the physico-chemical descriptors. The information loss present in both methods might decrease the predictive performance of the model. To bridge this information gap, a hybrid *sequential properties* peptide representation scheme is developed and presented in this thesis. The name of the scheme stems from the fact that it encodes each amino acid in the sequence by its individual physico-chemical, topological, geometrical and constitutional properties. In this way, both the information on the theoretical properties and the order of amino acids in the sequence is preserved. In this thesis, 94 per-amino acid properties were extracted from Peptides package, including:

1. **Hydrophobicity** on 38 scales that capture different aspects on hydrophobicity of amino acid. The included scales are Aboderin, Abraham-Leo, Argos, Black-Mould, Bull-Breese, Casari, Chothia, Cid, Cowan at a pH of 3.4 and 7.5, Eisenberg, Engelman, Fasman, Fauchere, Goldsack, Guy, Hopp-Woods, Janin, Jones, Juretic, Kiddera, Kuhn, Kyte-Doolittle, Levitt, Manavalan, Miyazawa, Parker, Ponnuswamy, Prabhakaran, Rao, Rose, Roseman, Sweet, Tanford, Welling, Wilson, Wolfenden and Zimmerman [52].
2. 3 **Cruciani properties** that are scaled principal component scores reflecting polarity, hydrophobicity and H-bonding [55].
3. 5 **Z-scale** values that include lipophilicity, steric properties, electronic properties and two properties related to electronegativity, heat of formation, electrophilicity and hardness [56].
4. 6 **FASGAI** properties that describe hydrophobicity, alpha and turn propensities, bulky properties, compositional characteristics, local flexibility and electronic properties [57].
5. 8 **VHSE** values reflecting hydrophobicity, steric and electronic properties [58].
6. 8 **ProtFP** descriptors derived from amino acid properties in AAindex database [59].
7. 10 **BLOSUM62** indices derived from a VARIMAX analysis of physico-chemical properties and BLOSUM62 substitution matrix [60].
8. 3 **MS-WHIM** properties derived from 36 electrostatic potential properties of the 3D structure [61].
9. 5 **t-scale** values derived from 67 common topological descriptors [62].
10. 8 **st-scale** values derived from a set of 827 physico-chemical, topological, geometrical and constitutional properties [63].

An example of peptide ADDC being encoded with sequential properties using 94 amino acid physico-chemical features and padded to the length of 6 is shown in Figure 3.4. It is encoded as a sequence of 6 sequence steps that are processed sequentially by a sequence modeling neural network.

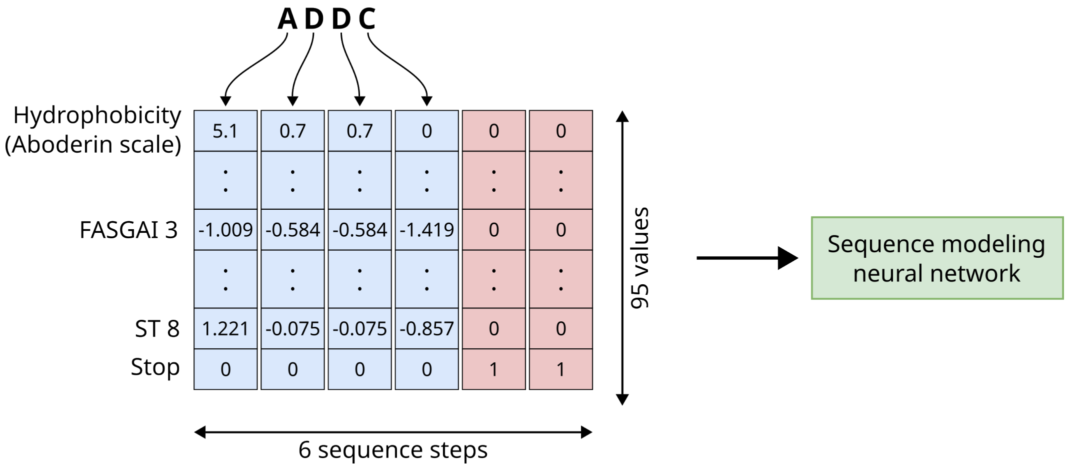


Figure 3.4: Peptide sequence ADDC encoded using sequential properties and padded to the length of 6. Each amino acid is encoded with 94 physico-chemical properties. The 95^{th} is used to signal the end of the sequence. Vectors encoding amino acids are depicted in blue color, while vectors used for padding are depicted in red color.

Unlike one-hot encoding and word embedding approaches, which limit pattern matching to subsequences of amino acids, this representation enables the model to exploit patterns within relevant theoretical properties that correlate to activity. This gives a model more detailed insight, as similarity between amino acids in the context of their properties can also be taken into account. Furthermore, the proposed scheme does not require learning during training as word embedding does, which can shorten execution time and be beneficial when there is insufficient data to train embeddings. However, this approach requires the properties to be collected for amino acids present in the input data, which can be challenging in the case of modified or non-natural amino acids. In comparison to the peptide properties, where a single value of a physico-chemical property was used to describe the entire peptide, this approach allows for a more detailed representation. It allows the model to focus on the subsequences and their properties, which are relevant for the activity that is being predicted. Although compositional properties are not directly present in this representation scheme, they can be indirectly deduced by the model from the input data.

Additionally, an extra *stop signal* feature is introduced to signal the end of the sequence. This feature is set to 0 until the end of the sequence is reached, at which point it becomes 1. The sequence is padded to the desired length by appending vectors with all features set to 0, except the stop signal which is set to 1.

4. Chapter

DATASETS AND PREPROCESSING

4.1. Feature Scaling

Features of different scales can lead to a slower convergence, an unstable learning process and features with larger scales dominating features with low scales. This can be mitigated by applying scaling techniques to bring all features into the same range, typically [0, 1] or [-1, 1]. In this thesis, feature scaling is used only in the case of peptide properties and sequential property representations because they contain features of different scales. The one-hot encoding representation by definition contains values that are either 0 or 1, while feature scaling is not applicable in the case of word embedding as the numerical representations are learned by the model itself.

In the case of peptide properties, feature values are computed based on the properties of individual amino acids and the length of the peptide, while some of them are also influenced by the ordering of amino acids in the peptide. An analysis of the underlying equations used to compute features would be needed to determine the theoretical range of values for each feature separately. To avoid this tedious and computationally expensive process, standardization is used instead because it centers each feature around 0 by removing the mean and scales it to the unit standard deviation. Feature i of data instance x is scaled as shown in Equation 4.1, where μ_i denotes the mean value of feature i and σ_i denotes the standard deviation of feature i.

$$x'_i = \frac{x_i - \mu_i}{\sigma_i} \tag{4.1}$$

In the case of sequential properties, the physico-chemical properties of each amino acid are known a priori and they are combined to form the input sequence to the model representing a given peptide. Furthermore, the length of the peptide only influences the length of the input sequence and does not influence the physico-chemical properties of individual amino acids. Based on this, it is possible to determine the minimal and maximal value of each feature, and Min-Max scaling can be applied to proportionally transform the features to the range [0, 1]. It scales feature i of data instance x according to the Equation 4.2, where m_i denotes the minimal and M_i denotes the maximal value of feature i.

$$x'_i = \frac{x_i - m_i}{M_i - m_i} \tag{4.2}$$

The parameters used for standardization (μ and σ) are usually inferred from a training set and then used to scale the entire dataset. In this way, the potential outliers or data instances with feature values outside ranges found in the training set will not affect the scaling and will end up outside the desired range. This preserves the integrity of the validation and test sets and allows for an unbiased estimate of the performance of the model on unseen data. As all amino acid properties that may appear in the input sequence are known in advance for sequential properties, the m and M values do not need to be estimated and can be easily determined for each property separately.

4.2. Similarity

Once a ML model has been trained, it is evaluated on the test set with the goal of estimating how well it would perform in a real-world scenario. Therefore, the test set should not contain data that was already fed to the model during training. The similarity between training and test sets may also influence the evaluation of the model as the model may be more prone to categorize similar peptides into the same class. The similarity between two peptide sequences can be easily exploited by the model, especially when sequence-based representations are used, as they transparently provide information on the ordering of amino acids in the sequence. Although classification based on sequence similarity is sometimes used, it is applicable only to sequences that are similar to those in the data set and fails for sequences with a low level of similarity. Therefore, it is necessary to take this into account during model evaluation, in order to create a model that fully exploits all the features of the input data and generalizes well even to the sequences not similar to the dataset.

The similarity of peptide sequences in the dataset is computed by performing pairwise sequence alignments, and then a percent identity is calculated to quantify the degree of similarity between the aligned sequences. Various definitions of percent identity have been suggested and adopted to various degrees in different software packages. For example, CD-HIT (Cluster Database at High Identity with Tolerance) and BLAST (Basic Local Alignment Search Tool) are two widely used software packages for sequence analysis that employ different definitions of percent identity. CD-HIT is a program designed to efficiently cluster sequences on the basis of their similarity [64]. It uses a global pairwise alignment algorithm, such as Needleman-Wunsch, to align two sequences end-to-end. The percentage of identity is then calculated as the number of matching residues $N_{matches}$ divided by the length of the shorter sequence (len_1 or len_2) as shown in Equation 4.3. On the other hand, BLAST is a program for finding regions of local similarity between sequences [65]. It employs its own heuristic-based local pairwise alignment algorithm optimized execution time-wise. The percentage of identity is computed as the number of matches $N_{matches}$ divided by the total length of the alignment $N_columns$ as shown in Equation 4.4 [66]. In comparison to other local alignment algorithms, such as Smith-Waterman, BLAST alignment algorithm produces less accurate results, but has a shorter execution time making it suitable for analyzing large databases of sequences [67, 68]. However, there is no guarantee that it will find the best possible alignment due to its heuristic nature. While there are multiple flavors of BLAST available for different types of sequence similarity searches, the rest of the comparison will focus on BLASTp (Protein BLAST) because the peptides are in the focus of this thesis.

$$PID = \frac{N_{matches}}{min(len_1, len_2)} \tag{4.3}$$

$$PID = \frac{N_{matches}}{N_{columns}} \tag{4.4}$$

The example shown in Figure 4.1 demonstrates the difference between global alignment

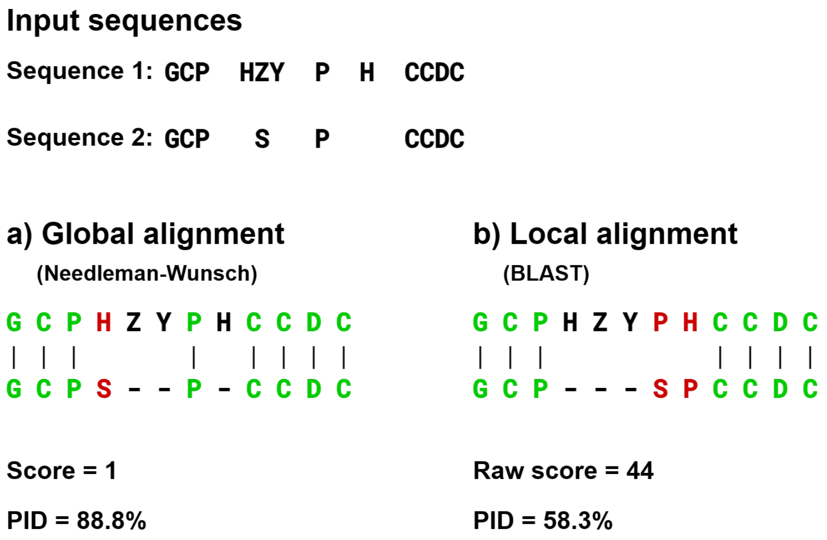


Figure 4.1: An example of a) a global alignment using Needleman-Wunsch algorithm, and b) a local alignment using BLASTp algorithm. For the sake of example, Needlamn-Wunsch algorithm uses a basic scoring scheme with a match score set to 1, a mismatch penalty set to -1 and a linear gap penalty set to -1. The resulting alignment score is 1 and percent identity of 88.8%. BLASTp algorithm used the default settings which include BLOSUM62 substitution matrix for scoring and affine gap penalty with opening penalty set to -11 and extension penalty set to -1. The resulting bit score for BLASTp is 21.8 and percent identity is 58.3%. Input sequences *GCPHZYPHCCDC* and *GCPSPCCDC* are intentionally segmented to contrast their identical and different parts. Matching residues are shown in green, mismatches are shown in red and gaps are shown with dashes (-).

using Needleman-Wunsch algorithm and local alignment using the BLASTp algorithm. While global alignment aligns the sequences globally from end-to-end, a local alignment identifies and aligns the local similarities between sequences. In the given example, a global alignment algorithm identified the matching parts shown in green (*GCP*, *P* and *CCDC*), while the rest of the aligned residues are either mismatching residues shown in red (*H-S* pair) or gaps indicated as black dashes which indicate residues in one sequence that do not have counterpart in the other sequence. BLASTp resulted in two local matching regions (*GCP* and *CCDC*), two mismatching residues (*P-S* and *H-P* pair) and three gaps that separate regions of local similarity. In the context of evolutionary relationship, gaps in aligned sequences can be interpreted as amino acids that were removed from one sequence or inserted into the other sequence. Even though both alignments resulted in only the second sequence having gaps, alignment of more divergent sequences can result in gaps being present in both sequences.

Internally, the alignment process in both algorithms is guided by a scoring scheme that has to be maximized in order to achieve the optimal alignment. The most fundamental scoring scheme for Needleman-Wunsch algorithm requires manually set scores that will be assigned to columns depending on whether there is a match, a mismatch, or a gap. However, more complicated scoring schemes based on substitution matrices are used to achieve more nuanced scoring by taking into account chemical, structural and evolutionary properties. The substitution matrix defines a score for each pair of amino acids separately allowing for each combination of amino acids to contribute differently to the final score thus taking into the account biological relationship between sequences. In both cases, the type and value of the gap penalty have to be manually configured. If a linear gap penalty is used, each gap in the alignment is penalized equally and their distribution in the alignment does not influence the final score. On the other hand, an affine gap penalty defines an opening penalty used to penalize the opening of each new gap and an extension penalty which penalizes the extension of an already opened gap. The opening penalty is usually higher than the extension penalty which directs the algorithm to prefer an alignment with fewer longer gaps over many shorter ones. The total score of an alignment is the sum of previously assigned scores to individual columns in the alignment. Similarly, the raw BLASTp alignment score is also defined as the sum of substitution scores and gap penalties. BLASTp also computes a normalized bit score from the raw score which allows for the consistent comparison of scores across different searches and databases.

Local sequence alignments are better suited to study homology and evolutionary relationship of sequences since local alignment successfully identifies local regions of similarity separated by unrelated or divergent subsequences. Insertions or removals of amino acids that cause divergence are usually caused by mutations or rearrangements that naturally occur in the evolutionary process. However, local alignment may not capture the overall similarity or diversity of the sequences, as it focuses only on the best-matching regions. In contrast, global alignment algorithms are better suited to assess the similarity of two sequences as they align them globally. For this reason, in this thesis the Needleman-Wunsch algorithm implemented in *scikit-bio* Python library will be used to compute the percentage of identity defined in Equation 4.3. The default settings will be used which use BLOSUM50 as a substitution matrix, a gap opening penalty of 11 and a gap extension penalty of 1. Additionally, terminal gaps at the beginning and ending of the alignment are not penalized to allow for the alignment of sequences of different lengths and to accommodate possible sequence extensions or truncations.

Peptide similarity is often controlled during data preprocessing to mitigate potential biases that could result in overfitting and an overestimation of model performance [22, 23, 27, 30, 31, 32, 36, 37, 39, 44]. It involves clustering peptides and eliminating sequences with high similarity to minimize homology bias. As there is no established standard similarity threshold in the field, each study selects its own threshold, which is sometimes influenced by the dataset size and the number of sequences that would be excluded. The used thresholds vary from a conservative 40% to a liberal 90% (Tables 2.1 and 2.4). In this thesis, a similarity threshold of 70% will be employed as it provides a balance between overly liberal and overly conservative thresholds.

4.3. Catalytic Dataset

Proteins with catalytic properties that speed up chemical reactions are known as enzymes. They achieve this by providing an alternative reaction pathway with a lower activation energy, thereby facilitating the reaction to occur more easily and quickly. They occur naturally in human bodies and are involved in a wide range of processes such as digestion. Beyond their biological roles, they are also widely utilized in industry and manufacturing. For instance, the enzyme rennin is used in the food industry for the coagulation of milk in cheese production, whereas cellulase is used in the textile industry to remove waxes, oils, and starch coatings from fabrics to enhance the final product's appearance. Despite their broad applications, enzymes have certain limitations. They often become unstable at varying pH levels and temperatures, and their production can be costly due to their complexity.

In contrast to proteins, peptides are much smaller chains of amino acids making them cheaper and easier to produce. They also exhibit better stability under varying pH levels and temperature conditions. These advantages of peptides over proteins suggest that the discovery of low-cost peptide catalysts could potentially replace enzymes in various industries and lead to new applications where enzymes were previously not applicable due to their limitations. One such application is ester hydrolysis, a chemical reaction in which an ester is broken down into a carboxylic acid and an alcohol component by reacting with water. This reaction has a wide range of applications in various industries. For example, it can be used to synthesize soaps, manufacture fire extinguishers, serve as softening agents in resins and plastics, or synthesize other chemical compounds.

A manually curated dataset is presented containing experimentally verified peptides that catalyze ester and phosphoester hydrolysis, two widely studied and important reactions in biological systems [1] [41]. The sequences were manually collected from published research papers up to the year 2023 found by searching the Google Scholar database with the keywords *catalytic peptides*, *p-NPA*, *p-NPP* and *self-assembly*. The search was specifically limited to peptides confirmed to have catalytic activity by standard colorimetric assays. These assays change color in the presence of esters and phosphoesters, thereby confirming that ester hydrolysis has occurred. They can also be used to quantify the catalytic activity of the peptides.

Since this dataset was manually collected through a literature search, it can be considered highly reliable and can serve as a benchmark for models that were trained on automatically collected data. Being the first dataset of its kind, it also has the potential to initiate the application of ML methods in the research and development of next-generation peptide-based catalysts. By focusing solely on ester hydrolysis, it provides an opportunity for analysis of the sequence-to-activity relationship for this particular cat-

---

[1] The dataset is available as CSV files at `https://data.mendeley.com/datasets/6s9kxj2ndr/2`

alytic activity. Furthermore, it also offers insight into existing design strategies important for the development of new catalytic sequences.

The catalytic dataset (CAT) contains purely peptidic sequences for the catalysis of ester and phosphoester hydrolysis. The dataset consists of 126 peptide sequences, of which 110 comprised of only proteinogenic amino acids and 16 contain non-proteinogenic amino acids. All the sequences follow Michaelis-Menten kinetics and 114 of them have their catalytic efficiency (k_{cat}/K_M) determined. For one sequence only an upper bound for catalytic efficiency is provided, and for 11 sequences only a label indicating that they exhibit catalytic function is provided as the exact efficiency is not known. Therefore, sequences showing any level of catalytic activity are treated as active for the purpose of this thesis, while the rest are treated as inactive. This criterion yields 97 active and 29 inactive sequences. Even though it is expected for the inactive peptides to be prevalent in nature, this skewness can be attributed to the bias in scientific literature towards publishing only positive results. For each peptide sequence, the following 9 are provided: a one-letter amino acid code, N- and C- termini modifications, tested substrate, catalytic parameter (k_{cat}/K_M), mechanism of catalysis, ability to form secondary structures and/or self-assemble and DOI of paper it was collected from. SMILES notation is also provided for the peptides containing only proteinogenic amino acids.

The mechanisms of catalytic reaction were analyzed and are presented in Table 4.1. The majority of the peptides in the dataset exhibit catalytic activity that is dependent on the presence of zinc ions (Zn^{2+}) which facilitates catalysis. The second most prevalent mechanism of reaction involves the catalytic triad residues, which involves three specifically arranged amino acids located in the active site. The least number of peptides in the dataset is dependent on the presence of histidine in their sequence. Given that the pH of histidine is close to 7, it can act as either a base or an acid in catalysis [69]. The mechanism of reaction is unknown for the remaining 30 peptides. The analysis of peptide lengths and amino acid composition are shown in Figures 4.2 and 4.3. It can be seen that most of the sequences have length from 7 to 9. The amino acid composition of sequences is diverse in the case of active sequences, while it is limited to 7 amino acids in the case of inactive sequences. Furthermore, the function of the peptide is also influenced by the substrate it is evaluated on, and C- and N-termini modifications. The dataset covers 13 substrates and their distribution is shown in Table 4.2, while the distribution of C- and

Table 4.1: Distribution of peptides by catalysis mechanism separately for the peptides containing only natural amino acids and those that contain non-natural amino acids.

Mechanism of catalysis	Peptides containing only natural amino acids	Peptides containing non-natural amino acids
Zn^{2+} dependent	47	0
Catalytic triad residues	23	8
Histidine	10	8
Not known	30	0
Total	110	16

Table 4.2: The number of peptides evaluated with a specific substrate.

Substrate	Sequences containing only natural amino acids	Sequences containing non-natural amino acids	Total
p-NPA	86	10	96
p-NPP	5	4	9
p-NPB	4	0	4
p-NPH	3	0	3
p-NPO	3	0	3
p-NPS	3	0	3
BNPP	1	0	1
HPNPP	1	0	1
Indoxyl acetate	1	0	1
p-NPF	1	0	1
p-NPMA	0	2	2
p-NPPalmitate	1	0	1
p-NPPropionate	1	0	1

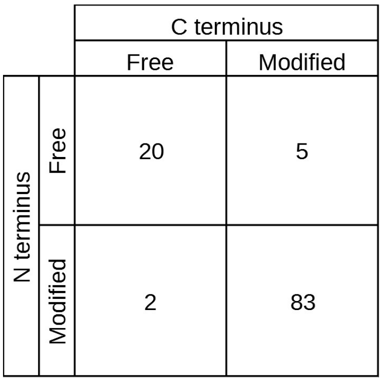
N-termini modifications is shown in Table 4.3.

The most numerous substrate is p-NPA, while all other substrates have 5 or fewer instances in total. Moreover, negative instances are only available for sequences composed of proteinogenic amino acids and they relate only to p-NPA and p-NPP substrates. Substrate is a nominal feature, and a low number of instances with the same category value can introduce bias and noise during training, as well as overfitting. This does not allow neural network to learn the decision boundary to distinguish positive from negative instances for substrates other than p-NPA. Furthermore, most of the available software libraries support only proteinogenic amino acids. Due to these reasons, only peptides tested on p-NPA containing only proteinogenic amino acids will be utilized in this thesis.

Theoretical physico-chemical properties that are important for peptide activity were computed by using peptide.py Python library and the distributions are shown in Fig-

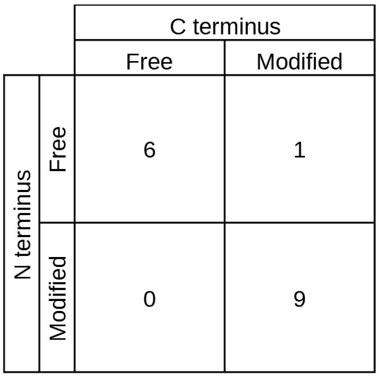
Table 4.3: Number of peptides with a specific combination of N- and C- termini.

(a) Sequences containing only proteinogenic amino acids.

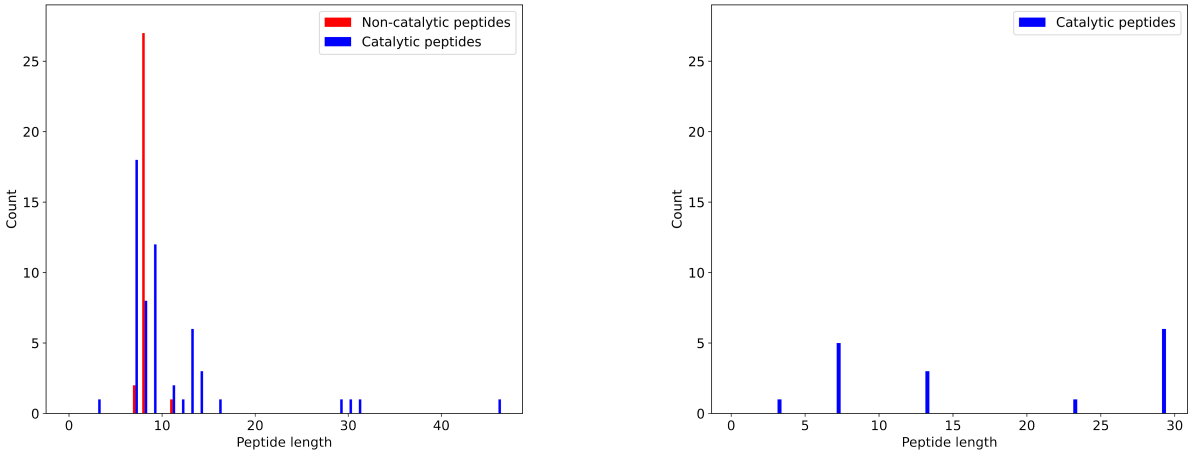


		C terminus	
		Free	Modified
N terminus	Free	20	5
	Modified	2	83

(b) Sequences containing at least one non-proteinogenic amino acid.



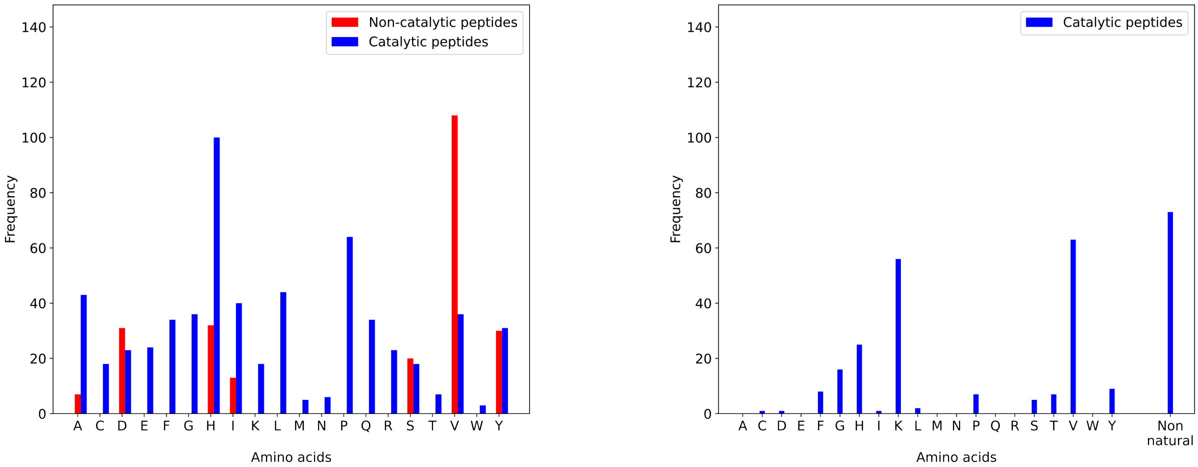
		C terminus	
		Free	Modified
N terminus	Free	6	1
	Modified	0	9



(a) Sequences containing only natural amino acids.

(b) Sequences that contain at least one non-natural amino acid.

Figure 4.2: Distribution of peptide lengths in the dataset.



(a) Sequences containing only natural amino acids.

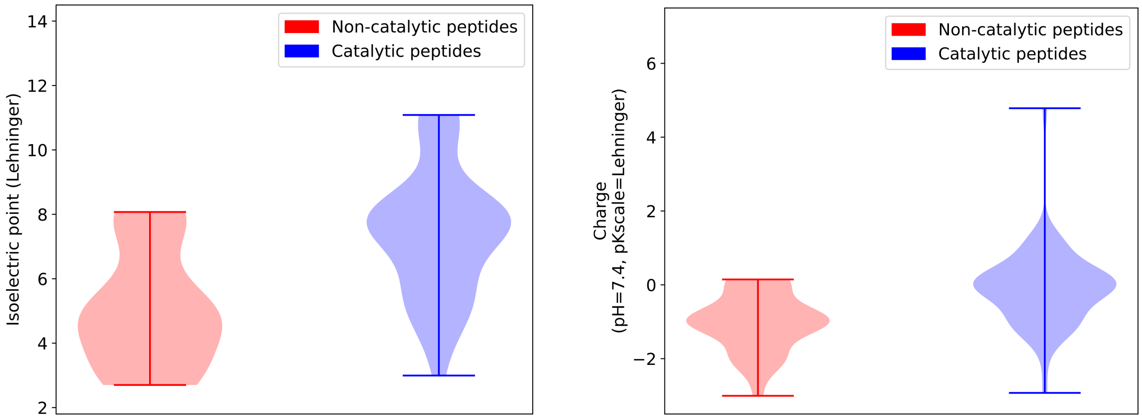
(b) Sequences that contain at least one non-natural amino acid.

Figure 4.3: The distribution of amino acids in the dataset.

ure 4.4. The charge of the peptide can influence the interaction between a peptide and a substrate or other components of the reaction and is dependent on the pH level of the surrounding environment. To capture this, isoelectric point which represents the pH at which the peptide carries no net charge, and the total net charge were computed. The calcualtion of net charge was performed for the neutral pH of 7.4 using the Henderson–Hasselbalch equation. Hydrophobicity is another important property for catalysis as it is known that higher hydrophobicity can improve catalysis [70]. It quantifies the tendency of a chemical compound to repel water molecules. The GRAVY hydrophobicity index reflecting the hydrophobicity of a peptide is computed by summing the hydrophobicity values of individual amino acids in the sequence and dividing the sum by the number of residues.

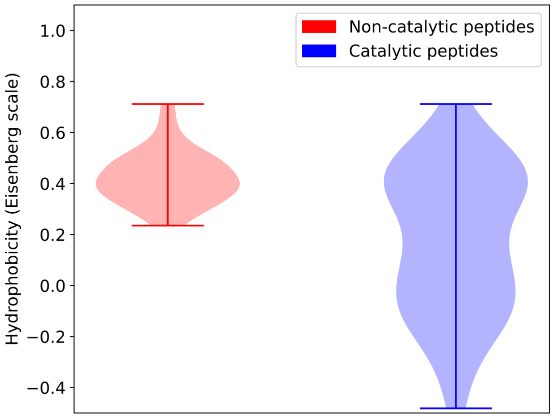
Figures 4.4a and 4.4b reveal that catalytic peptides tend to have a zero net charge in acidic conditions and a negative net charge at neutral pH. Furthermore, Figure 4.4c indicates that the average hydrophobicity of active peptides is higher compared to inactive peptides, and its range is also narrower. However, it can be seen that none of these properties is discriminative enough to be solely used for prediction.

The analysis of similarity for p-NPA sequences (Figure 4.5) containing only natural amino acids shows that the average inter-sequence similarity is 0.22 and 13.54% of the sequences have more than 70% similarity indicating the presence of highly similar sequences. Figure 4.6 demonstrates the number of clusters in CAT for various similarity thresholds used in the clustering algorithm employed in LOCOCV. To account for the fact that the result from the clustering algorithm depends on the ordering in which it processes sequences, 100 permutations of CAT sequences were randomly created and they were used to run the algorithm multiple times for various similarity thresholds. It can be seen that the number of clusters grows almost linearly up to the threshold of 0.7 and then it rapidly increases to a point where each peptide is its own cluster. In the case of a very high threshold, LOCOCV starts to lose its purpose as similar peptides are put into different clusters and therefore marginally helps to reduce the similarity between training and test sets. On the other hand, picking a threshold that is too low results in peptides with very low similarity being grouped together in the same cluster. Such a setting results in a small number of bigger clusters which means that leaving out one cluster as a test set will reduce more the size of the training set and may lead to poorer performance of the model. This further reinforces the choice to use the similarity threshold of 70% as it is a



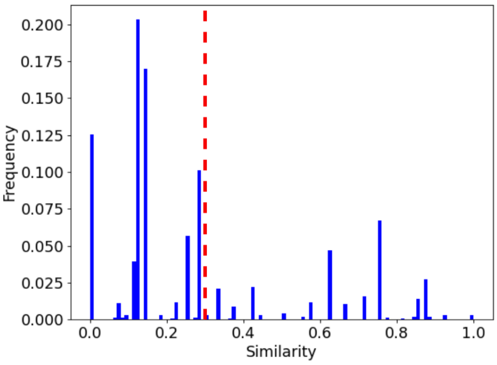
(a) Isoelectric point on Lehninger scale.

(b) Charge at pH=7.4 on Lehninger pKa scale.

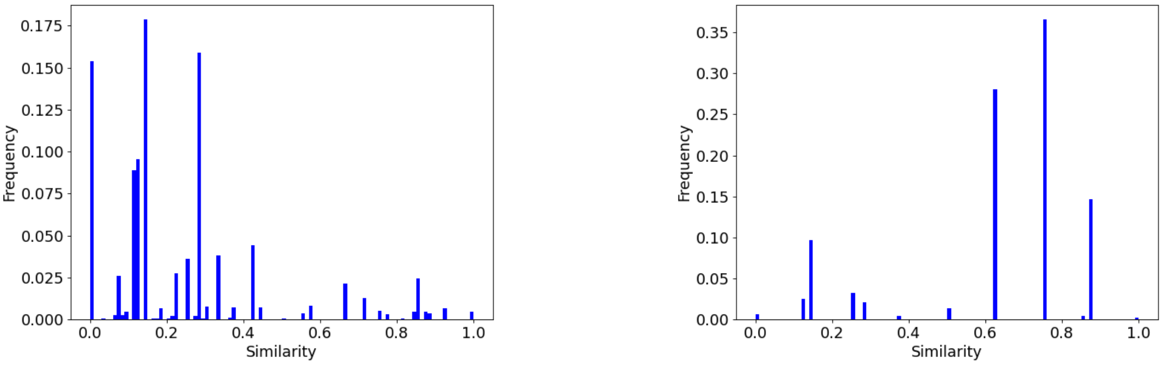


(c) Hydrophobicity on Eisenberg scale.

Figure 4.4: The distribution of theoretical physico-chemical properties relevant for the catalytic activity. Sequences containing only natural amino acids and tested with p-NPA substrate were included in this analysis.



(a) Similarity for all peptides. Peptides to the left of the red dashed line are less than 30% similar and represent 73% of the dataset.



(b) Similarity for only active peptides.

(c) Similarity for only inactive peptides.

Figure 4.5: The analysis of peptide similarities in CAT dataset. The analysis is limited to the sequences tested with p-NPA and containing only natural amino acids.

point of balance between the number of clusters and the number of peptides per cluster. The average number of clusters at this threshold is 21.5, while the average size of cluster is 3.81.

The final dataset after the aforementioned filtrations consists of 59 active and 27 inactive sequences.

4.4. Antiviral Dataset

The AVPpred database is used as the antiviral dataset in this thesis [71]. It is a comprehensive collection of experimentally verified antiviral peptides. These peptides

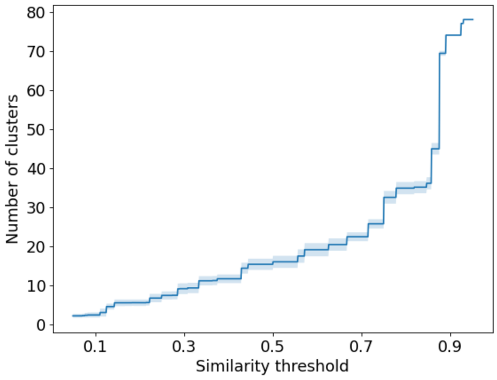


Figure 4.6: The dependency of the number of clusters for LOCOCV in CAT dataset on a chosen similarity threshold. The clustering algorithm was run 100 times each time with a random ordering of sequences. The blue line shows the average number of clusters, while the greyed area represents one standard deviation.

were collected from research articles on PubMed and patents in Patent Lens. About 91% of them are derived from natural sources, with the remaining originating from synthetic sources. After the removal of duplicates, the dataset comprises 604 active and 452 inactive peptides. Notably, the active peptides target human viruses such as HIV, HCV, and SARS-CoV. The dataset also provides a reference to a corresponding research article or patent for each peptide. The list of known targets is also given for antiviral peptides. While the dataset is pre-divided into training and test sets, it was necessary to concatenate these two sets for this study as cross-validation will be employed.

The diversity of the dataset in terms of peptide length can be seen in Figure 4.7. The similarity analysis (Figure 4.8) has shown that inter-sequence similarity is 0.08 and that only 0.58% of all peptides share more than 70% similarity indicating high diversity in the dataset. The similarity threshold of 70% was used to discard highly similar sequences and to limit inter-sequence similarity to 70%. The final dataset consists of 240 active and 358 inactive sequences.

4.5. Antimicrobial Dataset

Antimicrobial dataset (AMP) was manually created by combining active sequences from Data Repository of Antimicrobial Peptides (DRAMP) 3.0 with the inactive sequences collected from Uniprot repository.

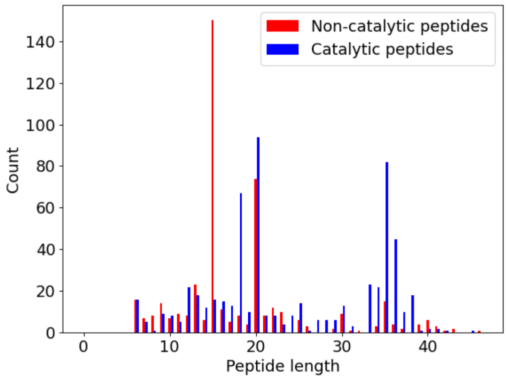
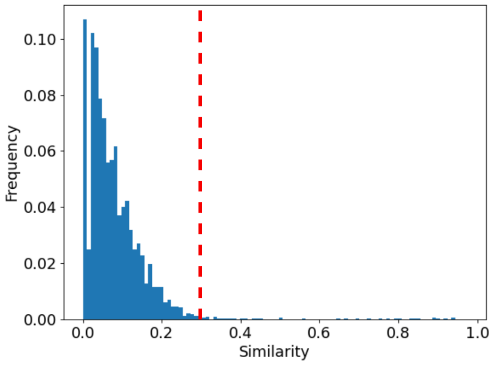
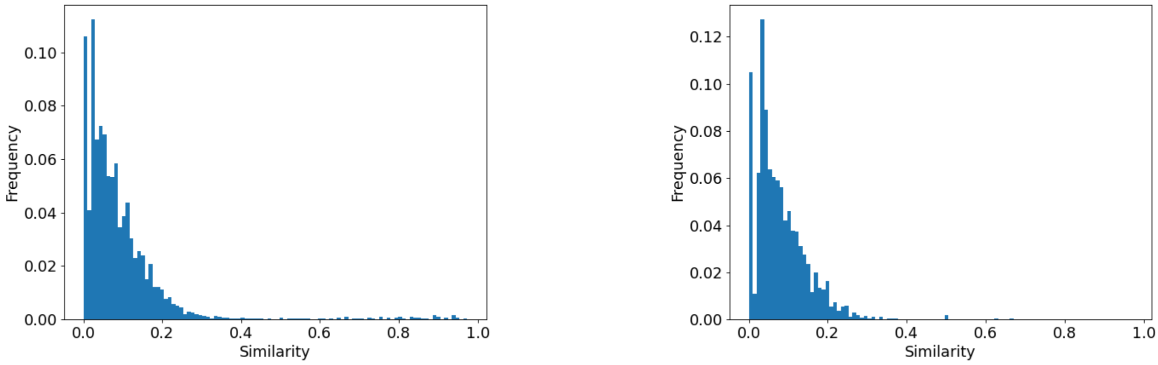


Figure 4.7: The distribution of peptide lengths in AVPPred dataset after similar peptides were removed.



(a) Similarity for all peptides. Peptides to the left of the red dashed line are less than 30% similar and represent 98% of the dataset.



(b) Similarity only active peptides.

(c) Similarity only inactive peptides.

Figure 4.8: The analysis of peptide similarities in AVPPred dataset.

DRAMP is a publicly accessible, manually curated database focused on antimicrobial peptides with a length of up to 100 residues [72]. Its latest version released on November 4th 2023 was utilized for this study. Peptides in the database come from publicly available dataset (Swiss-Prot, PDB, PubMed), clinical antimicrobial peptides in preclinical and clinical stages, and registered patents.

DRAMP database intends to provide the following information about each entry, but some entries may be missing some of the values including sequence:

1. **General information:** DRAMP ID, single-letter amino acid code, sequence length, peptide name, gene, source (whether synthetic or natural and its natural source), family, and Swiss-Prot ID.
2. **Activity information:** type of activity (e.g., antifungal), target organism, hemolytic activity, cytotoxicity, and binding targets.
3. **Structure information:** linear or cyclic peptide, modifications of C and N termini, non-terminal modifications, stereochemistry, structure information, and PDB ID.
4. **Physico-chemical information:** length, molecular weight, theoretical pI, amino acid composition, net charge, formula, extinction coefficient, estimated half-life, instability index, aliphatic index, and GRAVY value.
5. **Additional comments:** comments on the function, mode of action, etc.
6. **Reference and source information:** the reference and information on the source of the entry.

The entries in the repository are further categorized into subactivities, as shown in Table 4.4. These categories are not mutually exclusive, and a single peptide can be present in more than one category simultaneously. Peptides from all available categories were merged into a single set of antimicrobial peptides for the purpose of this thesis. Duplicate entries, peptides with non-natural or unknown amino acids, and those with more than 50 residues were removed. After this filtration process, a total of 4471 peptides remained in the set.

UniProt is a comprehensive, high-quality and publicly available resource of protein sequences, functional information and annotations [73]. It is a joint effort between the

Table 4.4: Number of instances in each of the subactivity categories in DRAMP 3.0 database.

Subactivity	Number of instances
Antimicrobial	5626
Antibacterial	4159
Anti-gram-positive	2733
Anti-gram-negative	2562
Antifungal	1836
Antiviral	312
Anticancer	163
Anti-SARS-CoV-2	90
Antiparasitic	52
Insecticidal	98

List of Code Listings 4.1: Query used to collect inactive sequences from UniProt repository.

```
NOT keyword:antimicrobial AND length:[0 TO 50] AND fragment:no
AND reviewed:yes
```

European Bioinformatics Institute (EMBL-EBI), the SIB Swiss Institute of Bioinformatics and the Protein Information Resource (PIR). For the purpose of this study, the inactive sequences were collected from Uniprot using the query shown in Code listing 4.1. *Antimicrobial* is a keyword in a Uniprot-controlled vocabulary encompassing sequences with deleterious effects on any type of microbe excluding viruses. *length:[0 TO 50]* limits the number of residues in the sequence, while *fragment:no* limits the result only to peptidic sequences by preventing the query from finding fragments of longer proteins. Database records missing *antimicrobial* keyword may either indicate sequences not having antimicrobial activity or it may indicate sequences that were not tested for antimicrobial activity. Therefore, “reviewed:yes” is used to limit the search results only to the sequences that were confirmed by UniProt to not have antimicrobial activity.

The query returned 9037 sequences that were then filtered to remove duplicates, sequences longer than 50 residues and sequences containing unknown or non-natural amino acids. This reduced the number of inactive sequences to 6036 and they in a combination with 4471 previously collected active sequences from DRAMP form AMP dataset with 10507 sequences in total.

The histograms of peptide lengths and a similarity analysis presented in Figures 4.9

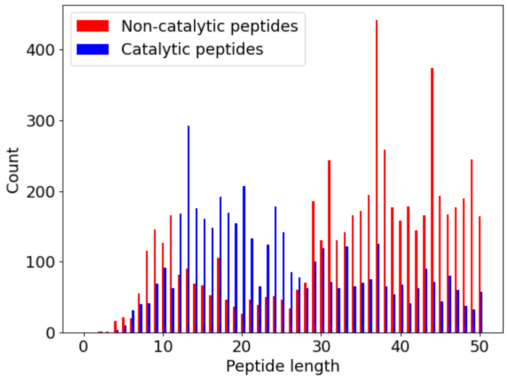


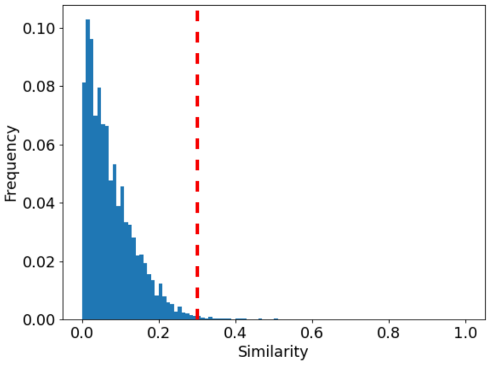
Figure 4.9: The distribution of peptide lengths in AMP dataset after similar peptides were removed.

and 4.10 indicate a high level of diversity in AMP dataset. Among all peptide pairs, 98% of them are less than 30% similar, while the average similarity is 0.08.

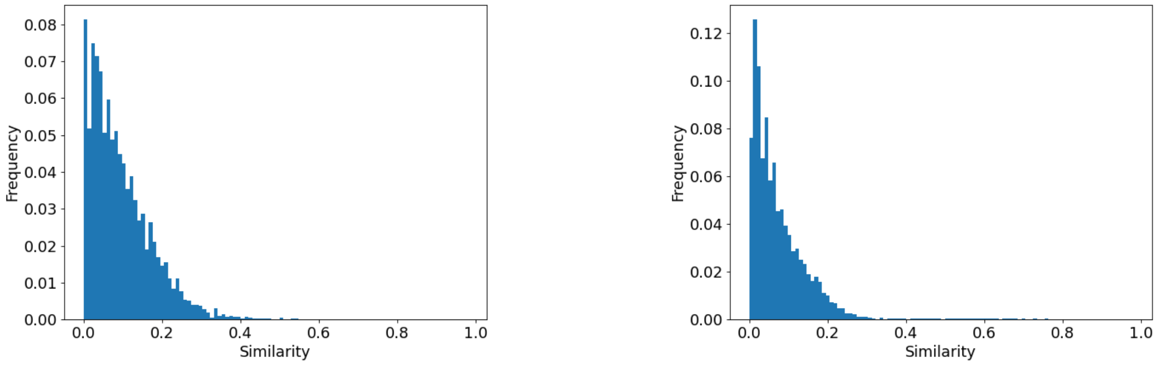
The histogram of peptide lengths shown in Figure 4.9 shows that active and inactive are diverse in terms of peptide lengths as they both cover all lengths. The similarity analysis (Figure 4.10) has shown that the average inter-sequence similarity is 0.08 and that only 0.13% of peptides share more than 70% similarity. The dataset was filtered to limit the inter-sequence similarity to 70%. The final dataset contains 2085 active and 2792 inactive sequences.

As AMP is going to be used in combination with AVPPred for TL in Chapter 6., it was necessary to create AMP-ExAVP dataset to ensure that there is no overlap between the source and target datasets, which could bias the model by allowing it to learn patterns specific to antiviral peptides during pretraining. Additionally, pretraining and fine-tuning on the same peptide subfunction would not reflect a real-world scenario, where TL is typically applied because there is not enough data for the target task. AMP-ExAVP excludes 312 antiviral peptides and any peptide overlapping with AVPPred. The same dataset filtering and processing steps that were used for other datasets were also used to process AMP-ExAVP.

The overview of the three used datasets after the filtration is given in Table 4.5.



(a) Similarity for all peptides. Peptides to the left of the red dashed line are less than 30% similar and represent 98% of the dataset.



(b) Similarity for only active peptides.

(c) Similarity for only inactive peptides.

Figure 4.10: The analysis of peptide similarities in AMP dataset.

Table 4.5: Overview of the used datasets, their source, and number of instances after data filtration.

Dataset	**Source**	**Positive instances**	**Negative instances**
AVPPred	AVPPred [71]	240	358
AMP	DRAMP 3.0 (positive instances) [72] Uniprot (negative instances) [73]	2085	2792
AMP-ExAVP	DRAMP 3.0 (positive instances excluding antiviral peptides) [72] Uniprot (negative instances) [73]	1991	2785
CAT	Manually curated dataset of catalytic peptides [41]	59	27

5. Chapter

REPRESENTATION SCHEMES EVALUATION

5.1. Models

The choice of model architecture is primarily influenced by the representation scheme employed. An MLP is used to learn and predict peptide activity from peptide properties representation scheme. When peptides are represented by their properties, the sequential nature of the data is lost and a peptide is represented through a set of numerical values in a tabular format. This makes MLPs to be used for activity prediction as they do not inhrently support sequential data. The architecture of the MLP model is shown in Figure 5.1a [35]. The original model consisted of three layers, each containing 100 neurons. The activation function for these layers was ReLU, while the output neuron employed a sigmoid activation function. As the information propagates through each layer of the MLP, the model learns to recognize more complex features and patterns. Hence, a different number of layers and neurons may be needed depending on the data and the task. A grid search process with the hyperparameters shown in Table 5.2a is employed to optimize the model. The number of layers in the model can vary between 1 and 3, while the number of neurons in each layer is optimized for each layer separately, resulting in 84 possible configurations. The rest of the model hyperparameters are preserved as they are in the original model. An additional 2-neurons binary input is employed in the case of CAT dataset to provide the information on C- and N- termini modifications.

Amidated C-terminus and acetylated N-terminus are represented by 1, while the free state is represented by 0.

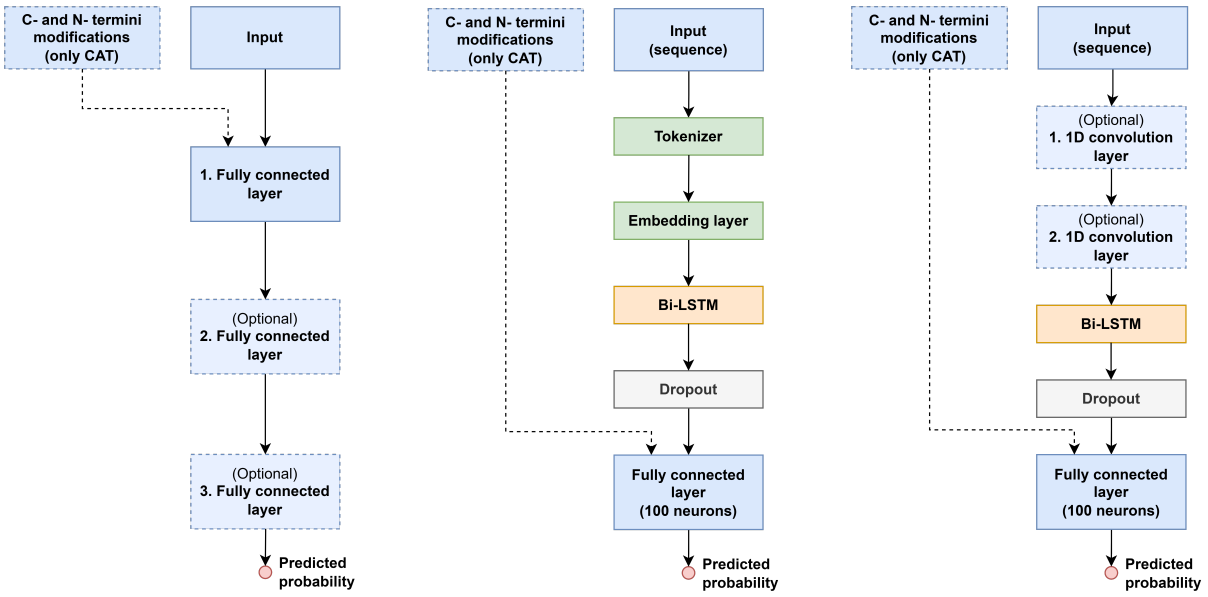
A neural network utilized for word embedding approach is shown in Figure 5.1b [31]. It starts with a tokenizer that converts amino acids into tokens. The tokens are then passed to an embedding layer, which automatically learns the real-value vector representation for each token. As a result, a peptide is encoded as a sequence of real-value vectors, each representing an amino acid. In this approach, the number of possible tokens is known in advance and corresponds to the number of amino acids with the addition of one more token for a padding sign. These vectors are then processed by a BiLSTM layer. This layer learns the sequential dependency between amino acids and summarizes the sequence into a single real-value vector of fixed size capturing the sequential characteristics of the input sequence. The final prediction is made by the output neuron, which employs a sigmoid activation function. A dropout of is applied to the outputs of the BiLSTM layer to combat overfitting. This approach can be considered a complete end-to-end solution as it eliminates the need for manual feature engineering. It handles the conversion from the amino acid sequence to a numerical representation internally by automatically learning the representation of each amino acid. The architecture of the model is fixed during the optimization process, while the size of the embedding layer, the number of units in the LSTM layer and the dropout factor are optimized using a grid search. The examined hyperparamter values shown in Table 5.2b allow for 54 different configurations. The additional input for termini modifications skips the sequence processing part of the model as it is not part of input sequential data and is connected directly to the fully connected layer at the end of the model. In this way, information on termini modifications is taken into consideration together with the sequence summary provided by the LSTM layer for the final prediction.

A neural network for one-hot encoding is shown in Figure 5.1c. This model is also suitable for sequential properties, as both one-hot encoding and sequential properties are in their essence multivariate sequential data. Its architecture comprises a variable number of convolutional layers, a BiLSTM layer and a single output neuron with a sigmoid activation function. The dropout is applied to the outputs of the BiLSTM layer to control overfitting. The model's architecture is optimized for the prediction tasks using a grid search, as depicted in Table 5.2c. It involves tuning the number of convolutional layers, the number of filters in each convolutional layer, the kernel size, the number of units in the LSTM layer and the dropout factor. The grid search is allowed to employ up to two convolutional layers which includes the possibility that no convolutional layer will be used. By adjusting the number of convolutional layers and the kernel size, the receptive field of the model is changed and tuned to enhance the detection of relevant patterns in the data. To reduce the computational complexity of this search, the kernel size is shared between the two convolutional layers and it is not considered in optimization if no convolutional layers are used. The total number of configurations that have to be evaluated is 333.

The role of the convolutional layers is to extract local patterns in the sequence. In the context of one-hot encoding, the convolutional layers are limited by the binary information provided from one-hot encoding. In that case, they learn to recognize patterns of consecutive amino acids, forming subsequences relevant to the prediction. In the context of sequential properties, more details about amino acids are available to the convolutional layers that enables them to learn patterns in physico-chemical properties relevant to the prediction. The output of the convolutional layers is a sequence representing the presence of patterns at each sequence step. This sequence is processed by a BiLSTM, which summarizes the sequence into a fixed-size real-value vector. Unlike the convolutional layer, the LSTM is not limited to only consecutive sequence steps as it operates at the level of the entire sequence. This enables it to capture long-term relationships between steps that are not necessarily consecutive. For the same reasons as in the embedding approach, the additional input for termini modifications is connected to the fully connected layer.

5.2. Evaluation Procedure

The evaluation procedure used in the case of AVPPred and AMP dataset to evaluate representation schemes is depicted in Figure 5.2a. A 10 times repeated 10-fold cross-validation is employed in the case of AVPPred and AMP datasets to split the data into training and test sets. In each iteration, the training set is used to perform feature selection and hyperparameter optimization, while the test set is used to assess the performance of the final model. Such a setting will produce 100 repeated measurements on various test sets that will be used for statistical analysis. Due to the small size of CAT dataset, a modified evaluation procedure employing LOCOCV is employed as shown in Figure 5.2b.


C- and N- termini modifications (only CAT)
Input
1. Fully connected layer
(Optional) 2. Fully connected layer
(Optional) 3. Fully connected layer
Predicted probability
C- and N- termini modifications (only CAT)
Input (sequence)
Tokenizer
Embedding layer
Bi-LSTM
Dropout
Fully connected layer (100 neurons)
Predicted probability
C- and N- termini modifications (only CAT)
Input (sequence)
(Optional) 1. 1D convolution layer
(Optional) 2. 1D convolution layer
Bi-LSTM
Dropout
Fully connected layer (100 neurons)
Predicted probability

(a) MLP for peptide properties.

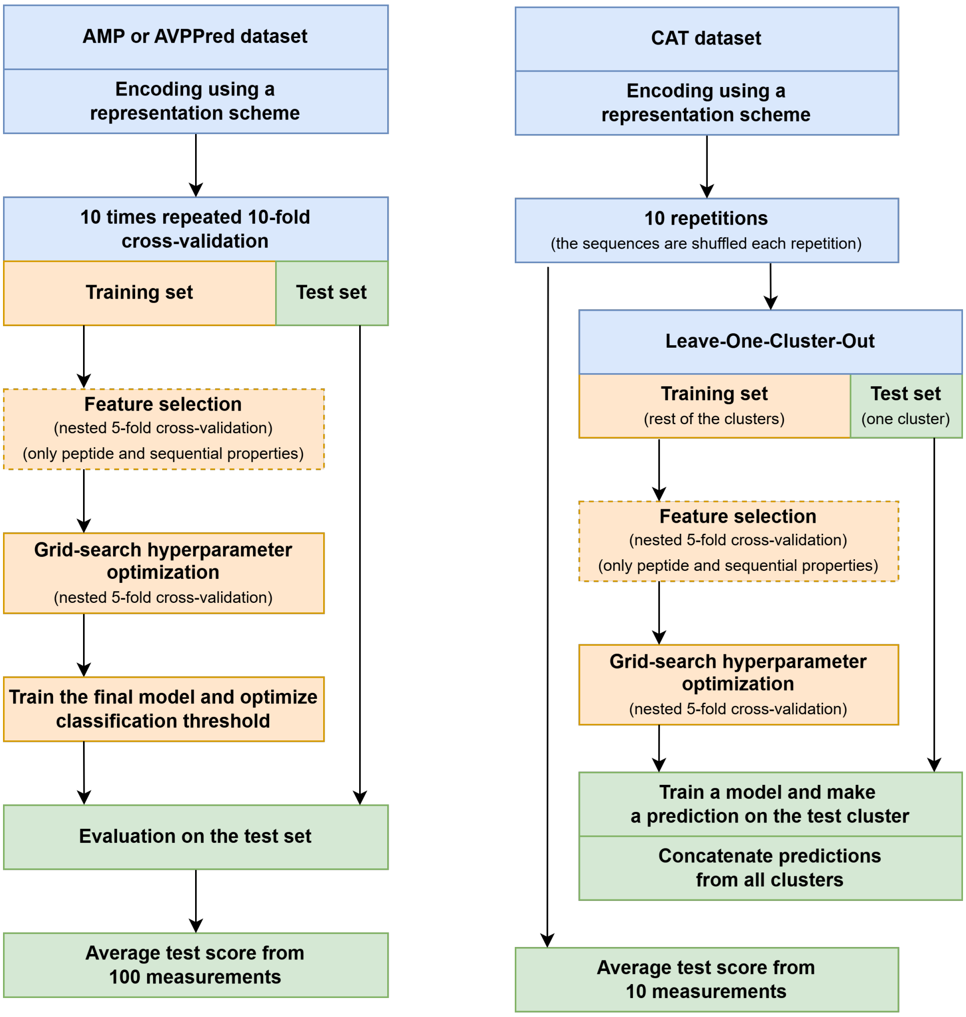
(b) Sequence modeling neural network with an embedding encoding.

(c) Sequence modeling neural network for one-hot encoding and sequential properties representation schemes.

Figure 5.1: Model architectures used for each of the representation schemes. The additional input for C- and N-termini modifications is used only in the case of CAT dataset.

It utilizes 10 times repeated LOCOCV with sequences being reshuffled at the start of each repetition to account for the fact that clusters produced by LOCOCV are influenced by the order in which sequences are processed. It is limited to 10 repetitions due to the longer execution time which is needed to iterate through all clusters. In each iteration of LOCOCV, one of the clusters is used as a test set, while the rest are used for feature selection and hyperparameter optimization. As clusters are expected to contain 3.81 peptides on average, their size is insufficient to assess the model performance individually on each of those clusters. For this reason, the predictions made on all test clusters during one repetition of the outer loop are concatenated and used to assess the model during that run of the outer loop. In the end, this produces 10 measurements that will be used for statistical analysis.

The complexity of the model is controlled through its hyperparameters and the complexity may be influenced by the number of features. A smaller number of features will generally require a less complex model. Conversely, the choice of hyperparameters configuration can influence the optimal feature set. As these two parameters are mutually


AMP or AVPPred dataset
Encoding using a representation scheme
10 times repeated 10-fold cross-validation
Training set
Test set
Feature selection
(nested 5-fold cross-validation)
(only peptide and sequential properties)
Grid-search hyperparameter optimization
(nested 5-fold cross-validation)
Train the final model and optimize classification threshold
Evaluation on the test set
Average test score from 100 measurements
CAT dataset
Encoding using a representation scheme
10 repetitions
(the sequences are shuffled each repetition)
Leave-One-Cluster-Out
Training set
(rest of the clusters)
Test set
(one cluster)
Feature selection
(nested 5-fold cross-validation)
(only peptide and sequential properties)
Grid-search hyperparameter optimization
(nested 5-fold cross-validation)
Train a model and make a prediction on the test cluster
Concatenate predictions from all clusters
Average test score from 10 measurements

(a) AVPPred and AMP datasets. **(b)** CAT dataset.

Figure 5.2: Procedure used to evaluate each of the representation schemes.

dependent, it would be optimal to perform feature selection and hyperparameter optimization simultaneously for the best results. However, due to the computational time required for this approach, it is not feasible and has to be implemented in two sequential steps. In the case of feature selection followed by a hyperparameter optimization, feature selection is performed using non-optimized hyperparameters to select features, and then they are optimized for the chosen feature set. In the case of a hyperparameter optimization followed by feature selection, a hyperparameter optimization would be performed with all features, and then feature selection would be performed based on the chosen hyperparameters. The total number of features in sequential properties is 94 and is expected to be reduced to a much smaller number as not all features are important for a given activity and there is overlap between the features. Therefore, a second approach would potentially first create an optimal model for a large feature set and then later perform feature selection to reduce the feature set. The resulting model could be more complex than necessary and more prone to overfitting. With that in mind, the first approach is implemented where the feature selection is followed by the hyperparameter optimization. In this way, the architecture of the model is adjusted to the size of the feature set.

In the case of peptide properties and sequential properties, a feature selection can be used to reduce the number of features. Therefore, a scenario including forward feature selection and a scenario without feature selection are used to assess the model and estimate the benefits of feature selection for the proposed representation scheme. In the first scenario when forward feature selection is employed, a nested stratified 5-fold cross-validation is used to estimate the performance of feature sets. The non-optimized model hyperparameters used for feature selection are shown in Table 5.1. In the latter scenario when feature selection is omitted, more emphasis is put on the model's ability to automatically extract and learn feature patterns needed for the prediction. A search grid for hyperparameter optimization is formed from the values shown in Table 5.2 and nested 5-fold cross-validation is used to evaluate each configuration. The configuration with the highest ROC-AUC score is chosen as the optimal, which makes the choice independent of the used classification thresholds. The optimal classification threshold is determined on the validation set once the final model is trained. However, a classification threshold of 0.5 was used for CAT dataset as the optimization cannot be performed reliably due to the small validation set.

Table 5.1: Hyperparameter values used in feature selection before hyperparameter optimization is performed.

(a) Peptide properties.

Hyperparameter	Value
Number of neurons (1st FC layer)	100
Number of neurons (2nd FC layer)	100
Number of neurons (3rd FC layer)	100

(b) Sequential properties.

Hyperparameter	Value
Number of filters (1st convolutional layer)	64
Number of filters (2nd convolutional layer)	64
Kernal size (both convolutional layers)	6
Number of units (the LSTM layer)	64
Dropout factor	0.2

Larger datasets in ML tend to provide more reliable results since they encompass a wider range of data points, thereby enabling the model to learn and generalize better. Furthermore, the AVPPred and AMP datasets were shown to be more diverse than the CAT dataset and are expected to be more generalizable. For these reasons, the statistical tests are performed only on AVPPred and AMP datasets to draw reliable conclusions.

10% of the training set is set aside as the validation set for early stopping. An early stopping mechanism is used to control overfitting during feature selection, hyperparameter optimization and final training of the model by stopping the training process when the validation loss does not decrease for 5 consecutive epochs. The model's weights are restored to the state when the validation loss achieved the lowest value. Moreover, a dropout is implemented to prevent the models from learning the noise in the training data. The model's tendency towards overfitting is also indirectly reduced through a grid search as it selects a model configuration with the highest cross-validation score.

The experiment was executed in parallel over 18 Bull DLC B720 servers that are part of the supercomputer Bura located at the University of Rijeka using the Dask Python library. Parallel computation was achieved by using Message Passing Interface (MPI).

Table 5.2: Values of hyperparameters used in grid search for hyperparameter optimization.

(a) MLP model used for peptide properties.

Hyperparameter	Values
Number of fully connected layers	1, 2, 3
Number of neurons (in each layer separately)	40, 80, 120, 160

(b) Sequential model used for word embedding scheme.

Hyperparameter	Values
Embedding size	30, 50, 70, 90, 110, 130
Number of units (the LSTM layer)	64, 128, 256
Dropout factor	0.1, 0.2, 0.3

(c) Sequential model used for one-hot vector encoding and sequential properties schemes.

Hyperparameter	Values
Number of convolutional layers	0, 1, 2
Number of filters (in each convolutional layer separately)	16, 32, 64
Kernal size (shared between convolutional layers)	4, 6, 8
Number of units (the LSTM layer)	64, 128, 256
Dropout factor	0.1, 0.2, 0.3

5.3. Analysis of Feature Selection

Peptide properties and sequential properties representation schemes have been evaluated with and without feature selection, and the results are compared in Tables 5.3a-f. The results show that for both representation schemes, a variant not using feature selection outperformed the one employing feature selection in terms of ROC-AUC, but also in the majority of the other metrics. Even though the variants without feature selection performed better on average, Wilcoxon signed-rank test with Holm-Bonferroni correction with an FWER of 0.01 did not find a statistically significant difference in any of the metrics for AVPPred and AMP datasets. Therefore, it can be concluded that feature selection has no notable impact in the case of peptide properties and sequential properties for the datasets used in this thesis.

Furthermore, the frequency at which each feature is selected was analyzed separately for peptide properties (Appendix A1a-c) and sequential properties (Appendix A2a-c). The frequency is computed by counting the number of cases in which a particular feature was selected and dividing it by the total number of repetitions, which is 100 in the case of AVPPred and AMP dataset, while it is a variable depending on the number of clusters in the case of CAT dataset.

When peptide properties encoding was used, the average number of selected features for AVPPred was 11.1±2.4. Cruciani properties, hydrophobic moment and hydrophobicity were the most often chosen physico-chemical properties, while the rest of them were never selected with the exception of Boman property which was selected only in 2% of the cases. 8 of 9 compositional properties on the relative scale were chosen in more than 50% of the cases, while only 1 of 9 was chosen from the absolute scale in more than 50% of the cases. These results may suggest that the relative scale is more suitable for the prediction when a dataset with peptides of various lengths is used as it directly provides the model with the information on the share of amino acid group in the peptide sequence. Even though the same information can be derived from the data provided on the absolute scale, it requires the model to learn the necessary transformations to do so, making the training process more complex.

An average of 23.4 ± 3.4 out of 28 features were selected in the case of AMP dataset, which is 2.1 times higher in comparison to AVPPred. Furthermore, the least used feature

Table 5.3: Average performance for peptide and sequential properties representations when feature selection is and is not employed. The classification threshold was optimized for AVPPred and AMP datasets as the validation sets were of sufficient size in their case. Bold values indicate the maximum value. No statistically significant difference was found by Wilcoxon signed-rank test between the two methods in any of the metrics for AVPPred and AMP datasets. Holm-Bonferroni correction is used to control for FWER with a significance level set to 0.01. Statistical tests were not performed for CAT dataset.

(a) Peptide properties on AVPPred.

	Recall	Prec.	Spec.	Acc.	F1	MCC	GM	ROCAUC
No feature selection	0.615	**0.693**	**0.797**	**0.725**	0.633	**0.433**	0.687	**0.782**
Feature selection	**0.637**	0.680	0.779	0.722	**0.640**	0.432	**0.691**	0.781

(b) Peptide properties on AMP.

	Recall	Prec.	Spec.	Acc.	F1	MCC	GM	ROCAUC
No feature selection	0.754	**0.766**	**0.822**	**0.793**	0.756	0.581	**0.785**	**0.883**
Feature selection	**0.759**	0.763	0.818	**0.793**	**0.757**	**0.582**	**0.785**	0.881

(c) Peptide properties on CAT.

	Recall	Prec.	Spec.	Acc.	F1	MCC	GM	ROCAUC
No feature selection	**0.938**	**0.845**	**0.663**	**0.842**	**0.887**	**0.641**	**0.771**	**0.885**
Feature selection	0.875	0.784	0.523	0.752	0.824	0.414	0.643	0.846

(d) Sequential properties on AVPPred.

	Recall	Prec.	Spec.	Acc.	F1	MCC	GM	ROCAUC
No feature selection	**0.662**	**0.697**	**0.792**	**0.740**	**0.665**	**0.468**	**0.714**	**0.801**
Feature selection	0.652	0.687	0.783	0.730	0.655	0.449	0.703	0.793

(e) Sequential properties on AMP.

	Recall	Prec.	Spec.	Acc.	F1	MCC	GM	ROCAUC
No feature selection	**0.845**	0.828	0.865	**0.857**	**0.834**	**0.711**	**0.854**	**0.936**
Feature selection	0.826	**0.836**	**0.875**	0.854	0.828	0.705	0.849	0.934

(f) Sequential properties on CAT.

	Recall	Prec.	Spec.	Acc.	F1	MCC	GM	ROCAUC
No feature selection	**0.927**	**0.912**	**0.830**	**0.893**	**0.919**	**0.764**	**0.876**	**0.933**
Feature selection	0.871	0.870	0.743	0.827	0.869	0.615	0.796	0.886

is instability index which was chosen 16% of the time, while there were 13 features that were selected less than 16% of the time in the case of AVPPred. All of this shows that the distribution of important features is spread more evenly for AMP in comparison to AVPPred which makes it harder to pinpoint important features.

On average, 8.7 ± 3 features were selected in the case of CAT dataset. Only 8 features were selected more than 50% of the time which includes hydrophobicity, all three cruciani properties and hydrophobic moment from physico-chemical properties and 3 compositional features on the relative scale. Due to the low diversity in peptide lengths in the CAT dataset, there is a high correlation between the absolute and relative scales for compositional features. As a result, the choice of scale may not be as critical here as it is for the AVPPred dataset.

Interestingly, aliphatic index was never selected into feature set for AVPPred and CAT, while it was selected in only 24% of evaluations for AMP dataset, making it the second least selected feature for AMP and the least selected feature for CAT. On the other hand, the number of aliphatic amino acids was selected 61% of the time for AVPPred on the relative scale, 100% for AMP on the absolute scale and 84.2% for CAT dataset on the relative scale. This may suggest that compositional aspects responsible for the aliphatic index may be more important for the prediction than exact aliphatic index value reflecting the relative volume occupied by aliphatic side chains.

The frequencies for features in sequential properties were computed in the same fashion as for peptide properties (Appendix A2a-c). The results were aggregated and summarized by the features groups (Appendix A3a-c) for easier analysis. On average, 10.2 ± 3.4 features were selected in the case of AVPPred, while the average for AMP was 16.9 ± 5.3 and 6.7 ± 1.3 for CAT. In all three cases, the average number of features was lower in comparison to the peptide properties model which suggests that sequential properties features are more informative and consequently less of them are necessary for classification.

The summarized table (Appendix A3a-c) shows that hydrophobicity was found to be the most significant feature in all three cases. On average, more than four hydrophobicity scales were found to be among the most selected features in each of the datasets. On the other hand, none of the other groups had two or more of its features selected multiple times on average. While this behavior can be explained by the fact that hydrophobicity group contains more features in comparison to the other groups and hence provides more possible choices that could be relevant for the classification. Physico-chemical descriptors from ProtFP and VHSE were found to be selected more than 50% of the time for AVPPred and AMP, while they ranked as the 2$^{\text{nd}}$ and 3$^{\text{rd}}$ in the case of CAT, respectively. Additionally, t-scale topological and Z-scale physico-chemical descriptors were found to be informative for AMP dataset. BLOSUM62 indices were found to be important for AVPPred and AMP with being included in the feature set 65% and 96% of the time, respectively.

Cruciani and MS-WHIM descriptors were found to be the least selected for all three datasets. Interestingly, Cruciani descriptors were selected at least 60% of the time in the case of peptide properties, with some descriptors being chosen 100% of the time for the AVPPred and AMP datasets. However, the selection frequency of Cruciani descriptors was lower for sequential properties, with rates of 40% and 41%, respectively. It can be speculated that the difference in the frequency between sequence-level and amino acid-level descriptors is due to the overlapping between multiple physico-chemical descriptors utilized in the latter case which were more informative than Cruciani properties.

5.4. Statistical Analysis

The evaluation results of four representation schemes are shown in Table 5.4 and the values written in bold indicate the best score in each of the metrics. Scores obtained without feature selection are reported for peptide properties and sequential properties as it was shown that feature selection does not improve the results and it was slightly

Table 5.4: Evaluation scores measured on all datasets with the best values for each metric marked in bold. The classification threshold was optimized for AVPPred and AMP datasets as the validation sets were of sufficient size in their case. Metrics that are significantly different from sequential properties, based on the p-values obtained from Friedman and Wilcoxon signed-ranks tests, are marked by an asterisk. Holm-Bonferroni correction is used for Friedman and Wilcoxon signed-rank tests to control FWER with a significance level of 0.01.

	Recall	**Precision**	**Specificity**	**Accuracy**	**F1**	**MCC**	**GM**	**ROC-AUC**	**Dataset**
Peptide properties	0.615*	0.693	**0.797**	0.725*	0.633*	0.433*	0.687*	0.782*	**AVPPred**
One-hot	**0.664**	0.640*	0.721*	0.698*	0.626*	0.401*	0.667*	0.783*	
Embedding	0.657	0.648*	0.732*	0.702*	0.630*	0.404*	0.672*	0.769*	
Sequential properties	0.662	**0.697**	0.792	**0.740**	**0.665**	**0.468**	**0.714**	**0.801**	
Friedman p-value	4.7e-2*	2.4e-4*	2.4e-3*	6.9e-7*	4.5e-3*	6.7e-5*	1.7e-4*	3.7e-9*	
Peptide properties	0.754*	0.766*	0.822*	0.793*	0.756*	0.581*	0.785*	0.883*	**AMP**
One-hot	**0.847**	0.815*	0.853*	0.850*	0.829*	0.700*	0.849*	0.930*	
Embedding	0.819*	0.808*	0.851*	0.837*	0.811*	0.671*	0.833*	0.917*	
Sequential properties	0.845	**0.828**	**0.865**	**0.857**	**0.834**	**0.711**	**0.854**	**0.936**	
Friedman p-value	9.4e-23*	2.1e-17*	1.5e-6*	3.9e-45*	6.4e-47*	1.5e-46*	2.0e-46*	9.7e-58*	
Peptide properties	0.938	0.845	0.663	0.842	0.887	0.641	0.771	0.885	**CAT**
One-hot	0.893	0.863	0.697	0.824	0.872	0.608	0.768	0.882	
Embedding	**0.952**	0.776	0.440	0.773	0.850	0.452	0.598	0.776	
Sequential properties	0.927	**0.912**	**0.830**	**0.893**	**0.919**	**0.764**	**0.876**	**0.933**	

Table 5.5: Ranking of evaluated representation schemes by their ROC-AUC score.

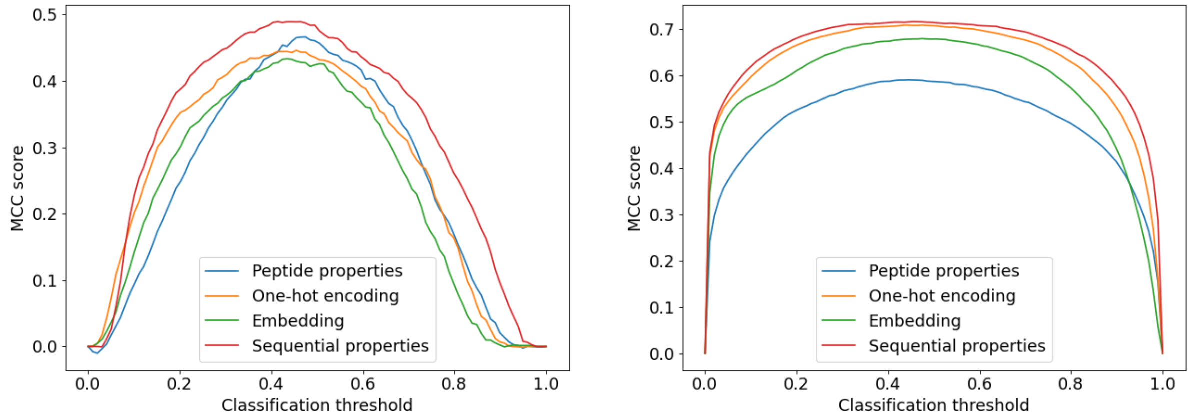
	Ranks			
	Sequential properties	**One-hot encoding**	**Peptide properties**	**Embedding**
AVPPred	1	2	3	4
AMP	1	2	4	3
CAT	1	3	2	4
Average rank	1	2.3	3	3.7

outperformed by the variants not employing feature selection. Friedman test was used to determine if there were any differences between the four examined representation schemes in any metric on the AMP and AVPPred dataset and Holm-Bonferroni correction was used to adjust the significance level for an FWER of 0.01. Friedman test found the significant difference in all of the metrics.

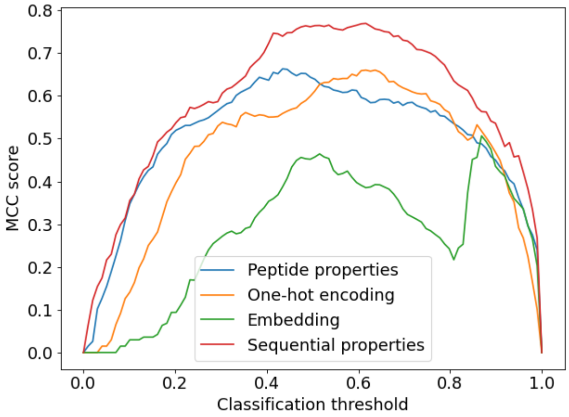
Hence, a post-hoc pair-wise Wilcoxon signed-rank test was performed to check for the differences between models based on peptide properties, one-hot encoding and word embedding against the sequential properties model individually. The Holm-Bonferroni method was used again to adjust the significance level to produce an FWER of 0.01. A statistically significant difference was found in 43 out of 48 pair-wise tests and sequential properties achieved higher scores in all those pair-wise comparisons. Notably, it achieved the highest ROC-AUC score for both datasets which was to be statistically significant by statistical tests. In the case of CAT dataset, sequential properties outperformed the other three models in all metrics except recall. The average improvement in ROC-AUC score was 3%, 2.9% and 10.5% for AVPPred, AMP and CAT datasets, respectively.

The ranking of the representation schemes by ROC-AUC score is shown in Table 5.5 and the average rank for each scheme is computed across the datasets. Sequential properties consistently achieved the highest ROC-AUC score, resulting in an average rank of 1. One-hot encoding ranked second with an average rank of 2.3, while peptide properties ranked third with an average rank of 3. Word embeddings ranked the lowest, with an average rank of 3.7. Interestingly, both one-hot encoding and word embeddings are sequence-based representations, yet peptide properties ranked better than word embedding and worse than one-hot encoding. Although word embedding was expected to be more informative than one-hot encoding due to its ability to learn the representation specialized for the task, it performed the worst on the smaller AVPPred and CAT datasets. However, on the largest dataset, AMP, word embedding ranked third. This suggests that word embedding might have better predictive peformance and outperform one-hot encoding if larger datasets were available for training.

The robustness of the developed representation scheme to a wider range of classification thresholds is not only evident from the ROC-AUC score, but also from metrics that are dependent on the classification threshold when it changes. A further visual analysis shown in Figures 5.3a-c provides an insight into the stability of the model at various classification thresholds. It can be seen that sequential properties scheme is less sensitive to the threshold because it consistently outperforms other schemes at various threshold values. Furthermore, this shows that the model is better at separating the classes, which can be beneficial in the case of noisy data to obtain more certain predictions in comparison to the other schemes.



(a) AVPPred dataset. **(b)** AMP dataset.



(c) CAT dataset.

Figure 5.3: Dependency of the average MCC score on the classification threshold for the test set.

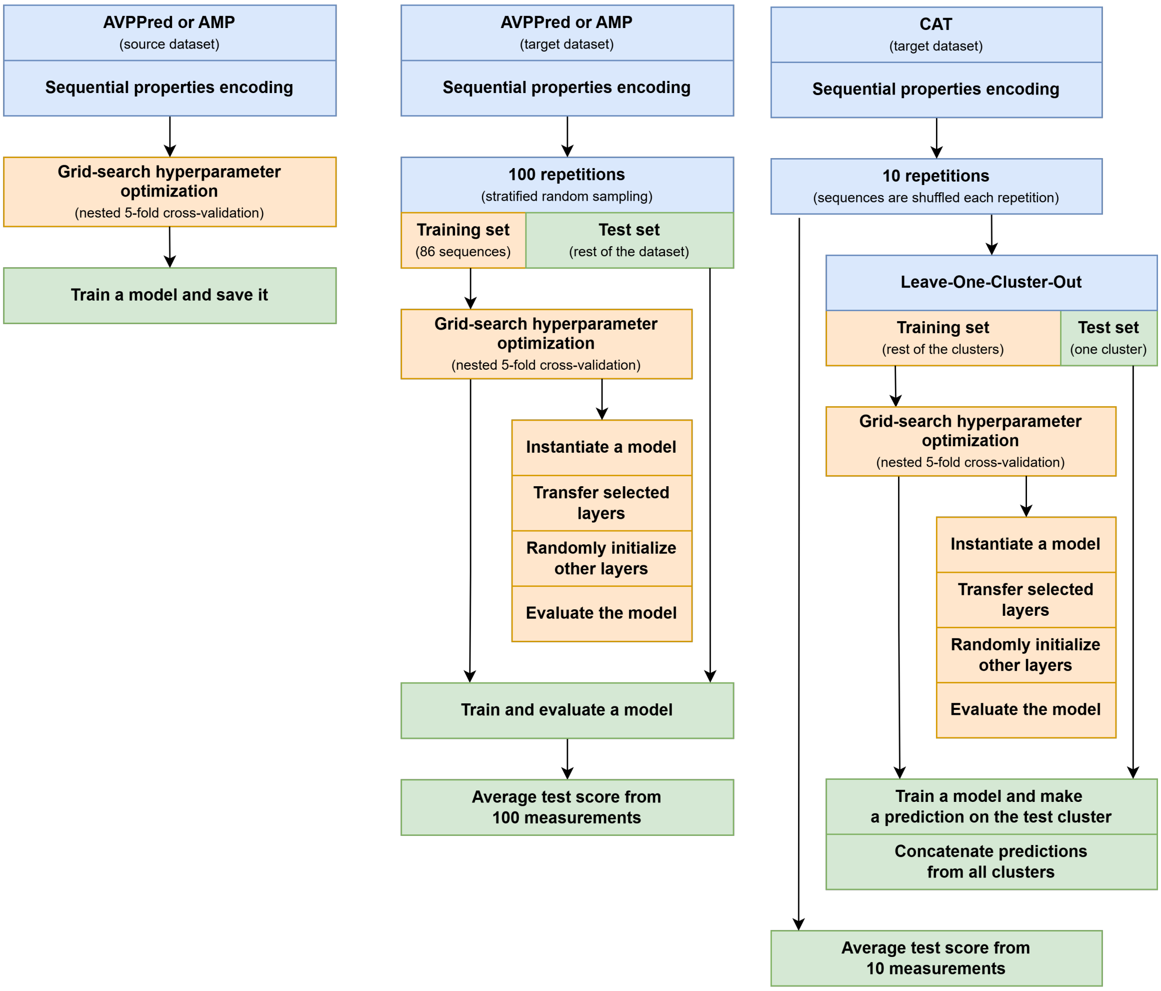
6. Chapter

TRANSFER LEARNING

6.1. Evaluation Procedure

The employed TL methodology involves two stages. In the first stage, a sequential properties model is pretrained on a source task, and then the pretrained model is adapted to the target task. In the pretraining phase, shown in Figure 6.1a, the entire dataset is used for model pretraining as the test set is not needed for the subsequent evaluation. A grid search is performed using nested 5-fold cross-validation to identify the optimal configuration. Once determined, a model is trained using found hyperparameters and its architecture and weights are saved.

In the second stage, the procedure outlined in Figure 6.1b is used for TL evaluation when the target dataset is either AVPPred or AMP-ExAVP. After loading the dataset and encoding sequences, a training set is created using stratified random sampling without replacement, while the remaining sequences that were not selected form the test set. The training set is fixed to contain 86 sequences, which corresponds to the size of the CAT dataset allowing for a simulation of a real-world scenario when a small quantity of data is available. Consequently, the test set is approximately 6 times larger than the training set for AVPPred and 56 times larger for AMP-ExAVP. These substantial test sets allow for a thorough evaluation of the TL models and reliable conclusions can be drawn from the results. The similarity between training and test sets is guaranteed to be below 70% as the sequences with a higher similarity were discarded during data preprocessing. Grid search is used to find the optimal TL configuration and then a final model is trained and


AVPPred or AMP
(source dataset)
Sequential properties encoding
Grid-search hyperparameter optimization
(nested 5-fold cross-validation)
Train a model and save it
AVPPred or AMP
(target dataset)
Sequential properties encoding
100 repetitions
(stratified random sampling)
Training set
(86 sequences)
Test set
(rest of the dataset)
Grid-search hyperparameter optimization
(nested 5-fold cross-validation)
Instantiate a model
Transfer selected layers
Randomly initialize other layers
Evaluate the model
Train and evaluate a model
Average test score from 100 measurements
CAT
(target dataset)
Sequential properties encoding
10 repetitions
(sequences are shuffled each repetition)
Leave-One-Cluster-Out
Training set
(rest of the clusters)
Test set
(one cluster)
Grid-search hyperparameter optimization
(nested 5-fold cross-validation)
Instantiate a model
Transfer selected layers
Randomly initialize other layers
Evaluate the model
Train a model and make a prediction on the test cluster
Concatenate predictions from all clusters
Average test score from 10 measurements

(a) Pretraining. **(b)** 10-fold cross-validation. **(c)** LOCOCV.

Figure 6.1: Procedures used in TL evaluation.

evaluated. This procedure is repeated 100 times to obtain multiple measurements that are used for statistical analysis.

The procedure shown in Figure 6.1c is used to evaluate TL when CAT is a target dataset. In comparison to the previously described procedure, it employs 10 times repeated LOCOCV with a similarity threshold of 70% to control for similarity between a training and test set. In each iteration, LOCOCV will use one cluster as a test set, while the rest of the clusters are put into a training set and it runs until each cluster has not been used for testing exactly once. Afterward, all test sets predictions from a single LOCOCV run are concatenated and used to evaluate the model. This procedure is repeated 10 times to obtain repeated measurements for statistical analysis. Since the training set and consequently a validation set are of small size, feature selection is not performed and the classfication threshold is not optimized in any of the two evaluation procedures.

In both cases, a grid search is performed to find the optimal TL strategy which involves identifying which layers should be transferred, the optimal learning rate multiplier and hyperparameters necessary to instantiate parts of the model that are not transferred. Each next layer in the model learns more complex features and it is assumed that the more complex feature representation becomes, the more it becomes task-specific. The exact layers that should be transferred ultimately depend on the compatibility between the source and target tasks as more compatible tasks can benefit from transfering more complex features. Therefore, it is necessary to transfer the consecutive layers as a whole because the correct functioning of each layer in a model is dependent on the outputs from the previous layers. The transfer of layers is limited to the convolutional and recurrent layers as these layers learn patterns from the sequence. The fully connected part, which classifies the instance based on the detected patterns, is always randomly initialized. Furthermore, it is allowed to transfer only a subset of transferable layers, while the rest are randomly initialized and their hyperparameters optimized (Table 5.2c), allowing for the adaptation of the model to the complexity of the target task. As the number of convolutional layers is allowed to vary in the pretrained model, this leads to the five possible cases for layer transfer shown in Figure 6.2.

In the first case (Figure 6.2a), both convolutional layers and a recurrent layer are transferred and only the dropout rate has to be optimized (3 configurations). In the second case (Figure 6.2b), the recurrent layer is randomly initialized and the number of

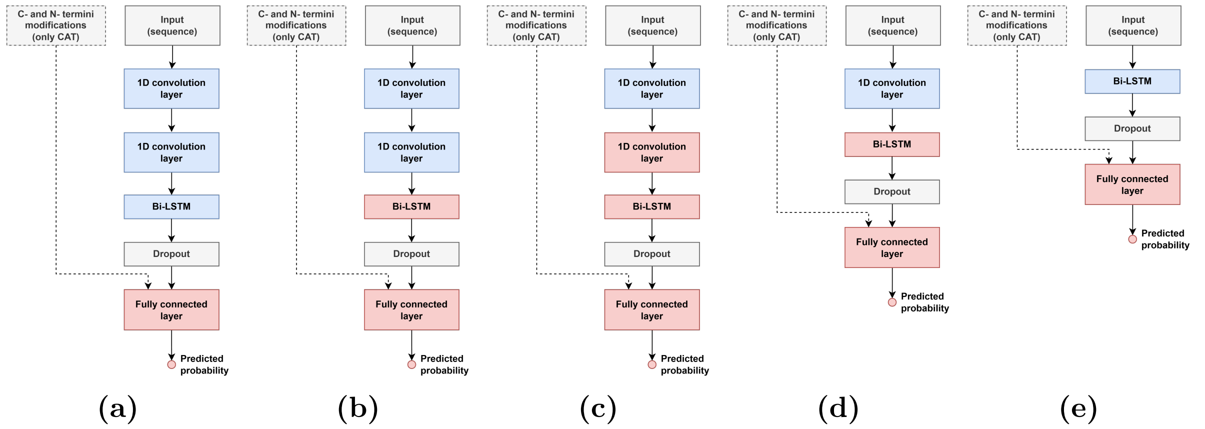


Figure 6.2: Options for selecting which layers to transfer and which layers to newly create. Blue color indicates layers that are transferred, red color indicates randomly initialized layers, and grey color depicts non-trainable parts of a model. The weights of connections going to the output neuron responsible for predicting probability are also reinitialized.

LSTM cells and dropout factor have to be optimized ($3 \cdot 3 = 9$ configurations). Both of these cases are possible only when the pretrained model contains two convolutional layers. In the third case (Figure 6.2c), the optimal number of filters and kernel size has to be determined for the second convolutional layer as well as the number of LSTM cells and the dropout factor ($3 \cdot 3 \cdot 3 \cdot 3 = 108$ configurations). The grid of hyperparameters in Table 5.2c allows for the case when the second convolutional layer is omitted and this case is separately considered in the fourth case (Figure 6.2d) where the number of LSTM cells and a dropout have to be optimized ($3 \cdot 3 = 9$ configurations). These two cases are possible if the pretrained model contains at least one convolutional layer. The fifth case (Figure 6.2e) is possible only when pretrained model does not contain a convolutional layer. In this case, a recurrent layer is transferred and only a dropout factor has to be optimized (3 configurations).

Six learning rate multipliers (0., 0.01, 0.05, 0.1, 0.5, 1.) are also included in the grid search which results in $6 \cdot (3+9+108+9) = 774$ possible configurations that cover different possibilities for transferring pretrained layers and optimizing the rest of the model.

6.2. Statistical Analysis

The distribution of optimal hyperparameters found during pretraining is available in Appendix B4 a-c. It shows that in pretraining on the AMP dataset, 1 convolutional layer

Table 6.1: Average performance for baseline and TL models. Metrics in which a statistically significant difference was found by Wilcoxon signed-rank test for cases with AVPPred and AMP as target datasets are marked by an asterisk. Holm-Bonferroni correction is used to control for FWER with a significance level set to 0.01. Statistical tests were not performed for cases with CAT as the target dataset. Values written in bold indicate the best performing of two models for a given metric.

(a) Transfer of knowledge from AMP to AVPPred.

	Recall*	**Prec.**	**Spec.**	**Acc.***	**F1***	**MCC***	**GM***	**ROCAUC***
Baseline	0.319	0.622	**0.875**	0.653	0.372	0.228	0.460	0.666
TL	**0.429**	**0.652**	0.837	**0.674**	**0.483**	**0.294**	**0.562**	**0.712**

(b) Transfer of knowledge from AVPPred to AMP.

	Recall	**Prec.***	**Spec.**	**Acc.***	**F1**	**MCC***	**GM**	**ROCAUC***
Baseline	**0.481**	**0.674**	**0.826**	**0.682**	**0.522**	**0.332**	**0.591**	**0.742**
TL	0.450	0.636	0.820	0.665	0.491	0.290	0.567	0.713

(c) Transfer of knowledge from AMP to CAT.

	Recall	**Prec.**	**Spec.**	**Acc.**	**F1**	**MCC**	**GM**	**ROCAUC**
Baseline	**0.927**	**0.912**	**0.830**	**0.893**	**0.919**	**0.764**	**0.876**	**0.933**
TL	0.927	0.891	0.773	0.873	0.907	0.715	0.838	0.928

(d) Transfer of knowledge from AVPPred to CAT.

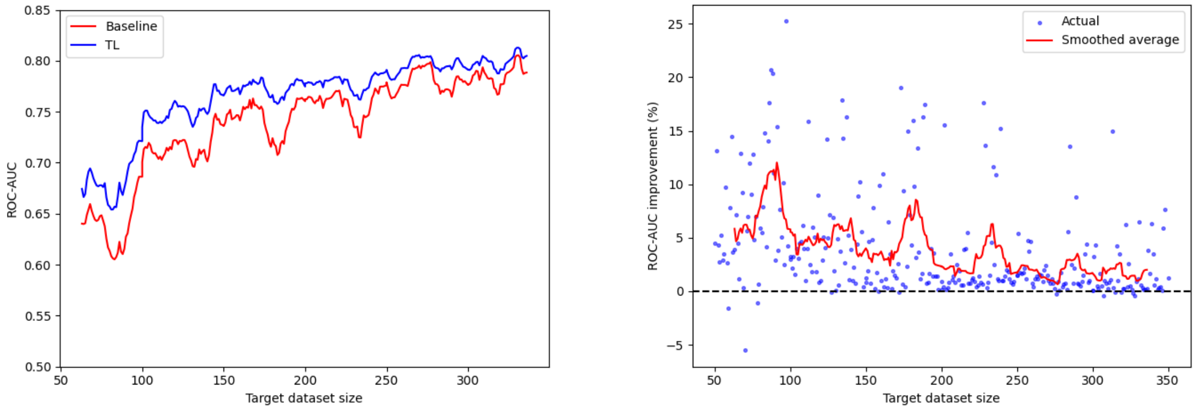
	Recall	**Prec.**	**Spec.**	**Acc.**	**F1**	**MCC**	**GM**	**ROCAUC**
Baseline	**0.927**	**0.912**	**0.830**	**0.893**	**0.919**	**0.764**	**0.876**	**0.933**
TL	0.920	0.881	0.747	0.859	0.897	0.688	0.814	0.908

was selected in 21% of cases, while two convolutional layers were used in 79% of cases. Similarly for the AMP-ExAVP dataset, 1 convolutional layer was chosen in 12% of cases, and two in 88% of cases. Notably, the option without convolutional layers was never selected for either AMP or AMP-ExAVP. In the case of AVPPred, no convolutional layers were selected in just 3% of cases, with 1 layer chosen in 35% and two layers in 62% of the cases. This indicates a strong preference for including convolutional layers. This means that in only these 3 cases, TL will be performed by transferring only the LSTM layer, as shown in Figure 6.2e, since it is the only option for models not containing convolutional layers. For the remaining models, the optimal set of layers to transfer will be determined individually, as previously explained.

TL results are presented in Tables 6.1a-d. Statistical tests were performed only in cases of AVPPred and AMP-ExAVP as those datasets had sufficiently large and diverse test sets to ensure the reliability of the conclusions. Holm-Bonferroni correction was used to adjust the FWER to 0.01 for each TL case separately, which implies that there is a 99% chance that none of the null hypotheses were falsely rejected. The statistical test found that the TL model on AVPPred target dataset significantly outperformed the baseline model in 6 of 9 metrics and ROC-AUC score increased by 6.9%. On the other hand, in the case of AMP-ExAVP target dataset, the baseline model significantly outperformed the TL model in 4 metrics whose ROC-AUC score was lower by 3.9%.

The model fine-tuning in the first case possibly benefited from the wider set of patterns learned during pretraining on AMP-ExAVP that has higher diversity as it covers 9 antimicrobial subfunctions. This resulted in a more general pretrained model whose more general set of patterns could then be leveraged during fine-tuning on the target AVPPred dataset to achieve better predictive performance. On the other hand, the negative transfer in the second case can be attributed to the narrowness of AVPPred dataset and a resulting pretrained model which learned a less diverse set of patterns causing it to struggle to adapt to the target dataset. Hence, training directly on the target dataset better captures the characteristics of the dataset and avoids the risk of transferring irrelevant patterns.

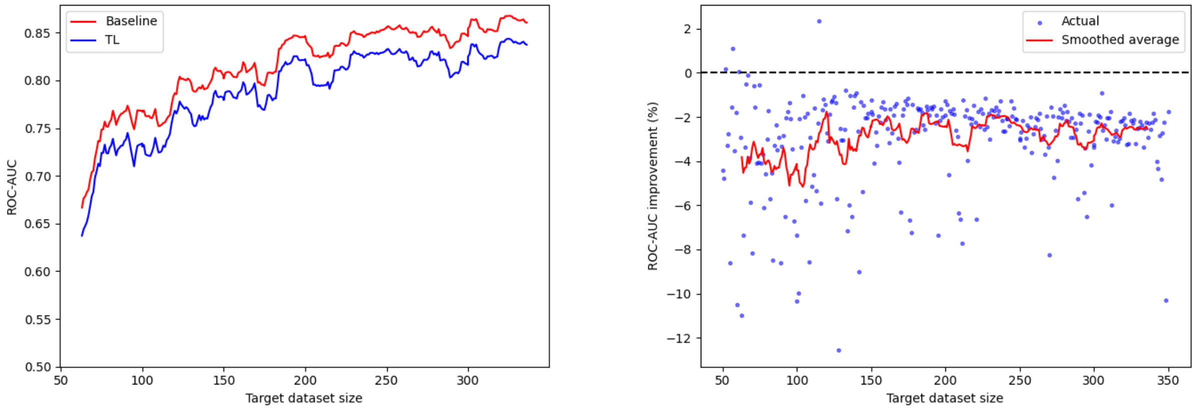
Baseline models outperformed TL models in both CAT cases for all metrics. The model pretrained on AMP achieved a 0.5% lower ROC-AUC score, while the one pretrained on AVPPred achieved a 2.7% lower ROC-AUC relative to the baseline model. Although TL did not improve performance in either case, the more substantial drop in ROC-AUC score is seen in the case of the model pretrained on AVPPred. This can be attributed to AVPPred being narrowly focused on antiviral function, resulting in more specialized pretrained model and making it less adaptable to the CAT. Conversely, the model pretrained on AMP was more generalizable as it had exposure to a broader set of patterns, leading to a smaller performance decline.



(a) Average ROC-AUC score. Lines have been smoothed to reduce noise and highlight underlying.

(b) Improvement in ROC-AUC score relative to the baseline model.

Figure 6.3: Dependency of ROC-AUC score for baseline and TL models on AVPPred target dataset at various sizes. For each dataset size, 100 runs have been performed and the average is reported in figures.



(a) Average ROC-AUC score. Lines have been smoothed to reduce noise and highlight underlying.

(b) Improvement in ROC-AUC score relative to the baseline model.

Figure 6.4: Dependency of ROC-AUC score for baseline and TL models on AMP-ExAVP target dataset at various sizes. For each dataset size, 100 runs have been performed and the average is reported in figures.

The effects of target dataset size were studied by varying the size of AMP-ExAVP and AVPPred datasets from 50 to 550. The experiment was repeated 100 times, each time evaluating the corresponding previously optimized TL and baseline models on the randomly sampled target dataset. Figure 6.3 shows that the highest improvement in performance due to TL is 6% when the target AVPPred dataset contains 50 peptides. As the target dataset increases in size, the difference becomes smaller and reaches 1% once the dataset contains 275 instances. From 275 to 350 instances, the average improvement oscillates between 1% and 3.4% with a downward trend towards 0%. Interestingly, Figure 6.3 shows that TL approach initially underperforms by -3.9% when dataset contains 50 instances, but this gap narrows to approximately -2.4% as the dataset size increases to 110 instances. Beyond this point, the performance of TL oscillates around -2.4% without converging towards the baseline model, regardless of further increases in dataset size.

6.3. Optimal Transfer Learning Strategy

The optimal configuration for the transfer of knowledge from AMP-ExAVP to AVPPred is further explored as this case significantly outperformed the baseline model. The distribution of optimal hyperparameter values found by grid search is shown in Tables 6.2a-e for each combination of transferred layers separately. The transfer of both convolutional layers and the LSTM layer was the most chosen one and was employed in 45% of the cases. The optimal learning rate multiplier was found to be either 0 or a value close to 1.0 in most cases. As pretrained layers are not changed during fine-tuning when learning rate multiplier is set to 0, it suggests that the pretrained model contained patterns that were already well-suited for the target task and do not require further adjustments. In the case where the learning rate multiplier is set to 0.5 or 1.0, the transferred patterns provided a good initalization and starting point for continued training on the target dataset. The mix of these two extreme cases is caused by the variability in the random data splits.

The combination with only the first layer being transferred was used in 44% of the cases, making it the second most used. In this setting, the second randomly initialized convolutional layer was introduced in 88.6% of cases and lower learning rate multipliers, such as 0 and 0.05, were used more often. This indicates that the low-level patterns captured by the first convolutional layer on AMP-ExAVP dataset can be reused and fine-tuned for use on AVPPred dataset. The transfer of both convolutional layers was found to be optimal by grid search in only 7% making it the least selected combination. This suggests that this combination usually scored lower on the validation set compared to the other two combinations. Even though the architecture of this combination employs two convolutional layers, which were shown to be beneficial in the other two combinations, the

Table 6.2: The distribution of optimal hyperparameters found by grid search for knowledge transfer from AMP-ExAVP to AVPPred. Each of the subtables corresponds to one combination of layers that were transferred. The second convolutional layer was randomly initialized and its number of filters and kernel size was optimized only when the layer was not transferred. The LSTM layer was reinstantiated and the number number of units was optimized only when the LSTM layer was not transferred.

(a) Frequency at which each combination of layers was transferred.

Transferred layers	**Frequency**
Both convolutional layers and the LSTM layer	45%
1st convolutional layer	44%
Both convolutional layers	7%
1st convolutional and the LSTM layer (only when pretrained model did not contain the 2nd convolutional layer)	4%

(b) Transfer of both convolutional layers and the LSTM layer.

Hyperparameter	**Values (Selection frequency)**					
Dropout factor	0.1 (33.3%)	0.2 (31.1%)	0.3 (35.6%)			
LR multiplier	0 (24.4%)	0.01 (6.7%)	0.05 (11.1%)	0.1 (15.6%)	0.5 (20%)	1 (22.2%)

(c) Transfer of the first convolutional layer.

Hyperparameter	**Values (Selection frequency)**					
Number of filters (2nd convolutional layer)	16 (45.5%)	32 (15.9%)	64 (27.3%)	Layer not employed (11.4%)		
Kernel size (2nd convolutional layer)	4 (38.5%)	6 (33.3%)	8 (28.2%)			
Number of units (LSTM layer)	64 (25%)	128 (20.5%)	256 (54.5%)			
Dropout factor	0.1 (36.4%)	0.2 (31.8%)	0.3 (31.8%)			
LR multiplier	0 (27.3%)	0.01 (15.9%)	0.05 (20.5%)	0.1 (18.2%)	0.5 (11.4%)	1 (6.8%)

(d) Transfer of both convolutional layers.

Hyperparameter	**Values** **(Selection frequency)**					
Number of units (LSTM layer)	64 (28.6%)	128 (28.6%)	256 (42.9%)			
Dropout factor	0.1 (42.9%)	0.2 (57.1%)	0.3 (0%)			
LR multiplier	0 (28.6%)	0.01 (14.3%)	0.05 (28.6%)	0.1 (14.3%)	0.5 (14.3%)	1 (0%)

(e) Transfer of the first convolutional layer and the LSTM layer when the pretrained model did not contain two convolutional layers.

Hyperparameter	**Values** **(Selection frequency)**					
Dropout factor	0.1 (50%)	0.2 (25%)	0.3 (25%)			
LR multiplier	0 (25%)	0.01 (25%)	0.05 (0%)	0.1 (25%)	0.5 (25%)	1 (0%)

model cannot achieve the same performance level. Possibly, the patterns learned in the second convolutional layer during pretraining may be too specialized for AMP-ExAVP making it hard for the convolutional layer to adapt and for the LSTM layer to learn the relation of these patterns to the new task. Furthermore, in the case of 12 pretrained models that contained only one convolutional layer, both the convolutional and the LSTM layer were transferred in 8 cases, while in the remaining 4 cases, only the LSTM layer was transferred. These results suggest that the optimal strategy for TL may be either transferring only layers with low-level patterns common to both tasks and training the rest of the model to learn task-specific patterns, or transferring convolutional and recurrent layers in their entirety as this preserves the entire feature extraction part of the model possibly making it easier for the model to adapt to the target task.

7. Chapter

CONCLUSION

In this thesis, three key contributions were made toward advancing peptide function prediction. A novel dataset of catalytic peptides for ester and phosphoester hydrolysis, the first of its kind, was introduced to support the study of catalytic activities. Moreover, three dominantly used peptide representation schemes for ML were investigated and a novel representation scheme that bridges gaps in the current schemes was proposed. The effectiveness of the proposed scheme was experimentally verified and confirmed by statistical tests. TL with the proposed scheme was implemented and four scenarios were explored, with one case demonstrating positive benefits of TL that were confirmed by statistical tests. To draw reliable conclusions during the experimental evaluation of the representation schemes and TL, large and diverse AVPPred and AMP datasets were employed for statistical testing, while the newly introduced CAT dataset served as an additional case study.

The introduced dataset contains 126 peptide sequences that were manually collected from 22 published research articles. Of these sequences, 110 are composed solely of natural amino acids. Each entry includes a single-letter amino acid code, N- and C-termini modifications, substrate, catalytic parameter (k_{cat}/K_M), mechanism of catalysis, ability to form secondary structures and/or self-assemble, and the reference to the corresponding research paper. Additionally, a SMILES notation of peptide is also provided for sequences containing only natural amino acids. Statistical, physico-chemical and compositional properties of the dataset were analyzed in the thesis. Notably, p-NPA is the most represented substrate and it comprises 96 entries, while reactions dependent on Zn^{2+} are the most common, with 47 entries. As peptides with enzymatic properties demonstrate better stability under varying pH levels and temperature conditions, as well as being easier and more cost-effective to produce compared to enzymes, they could become an attractive alternative in various industrial applications. Being the first manually curated dataset covering catalytic peptides, it has significant potential to help the application of ML in the development of next-generation peptide-based catalysts.

The analysis of dominantly used peptide properties, one-hot encoding and word embedding representation schemes revealed that none simultaneously captures the information on the order of amino acids and their physico-chemical properties, despite the known importance of both factors in peptide function prediction. In this thesis, a novel *sequential properties* representation scheme was proposed to address the identified limitations. The proposed scheme encodes sequences by representing amino acids by their physico-chemical properties, preserving information on both amino acid ordering and their properties in a complementary manner. Statistical tests demonstrated that the proposed scheme significantly outperformed other representations in 90% of cases on the AVPPred and AMP datasets, with average ROC-AUC score improvements of 3% and 2.9%, respectively. An improvement of 10.5% on average was achieved on the smaller CAT dataset, demonstrating the scheme's effectiveness even with limited amounts of data. Additionally, the scheme showed a higher level of insensitivity to classification thresholds, resulting in better predictive performance across a broader range of thresholds in comparison to the other three schemes. Furthermore, there was no significant difference in predictive performance when the feature selection was applied to the proposed scheme to reduce the number of features. This showcased the ability of DNNs to automatically identify input features from the scheme relevant for the task and to learn necessary internal representations.

The proposed representation scheme was also evaluated within a TL setting in which undersampled versions of the AMP and AVPPred datasets, matching the size of the smaller CAT dataset, were used as target datasets to simulate real-world data scarcity. Additionally, an AMP-ExAVP dataset was derived from AMP by removing antiviral peptides. Transfer of knowledge from AMP-ExAVP to AVPPred and vice versa was evaluated, as well as from AMP and AVPPred to CAT. The results indicated that only the transfer from AMP-ExAVP to AVPPred resulted in improved model performance, with a statistically significant 6.9% increase in ROC-AUC score. In contrast, negative transfer in the reverse scenario resulted in a 3.1% decrease in ROC-AUC. This was attributed to AMP-ExAVP diversity in terms of peptide functions, which enabled the model to learn wide diversity of patterns enabling it to leverage them during fine-tuning on single-function AVPPred dataset. A similar behaviour was observed when knowledge was transferred to the CAT. Transfers from AMP and AVPPred resulted in ROC-AUC reductions of 0.5% and 2.7%, respectively, likely due to the substantial difference between antimicrobial and antiviral functions and catalytic function, which made the learned patterns less usable. However, the smaller drop in performance with AMP, compared to AVPPred, was also attributed to its diversity. Interestingly, the optimal TL strategy determined by grid search involved one of two extremes, either transferring only the convolutional filters that capture low-level features or transferring all convolutional and recurrent layers, with intermediate approaches proving less effective.

Furthermore, the size of AMP-ExAVP and AVPPred target datasets was varied from 50 to 350 peptides to investigate the dependency of predictive performance improvement on the target dataset size. In the case of positive transfer, performance improvement was found to be inversely correlated with target dataset size. TL proved to be most effective when the dataset was the smallest, showing a 6 % improvement. It consistently achieved improvements greater than 1 % for datasets smaller than 275 instances, demonstrating its potential to enhance peptide function prediction in the early stages of research when only a limited amount of data is available. In the case of negative transfer, the TL model initially exhibited the highest underperformance, with a reduction in ROC-AUC score of -3.9% when the dataset was smallest. As the dataset size increased, this gap narrowed and stabilized around -2.4% at 110 instances. Despite further increases in dataset size, the TL model did not show additional improvement and fluctuated around the stabilization value. However, the future success of TL approaches depends on identifying source and target peptide functions compatible for knowledge transfer.

BIBLIOGRAPHY

[1] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick *et al.*, "Accurate structure prediction of biomolecular interactions with alphafold 3," *Nature*, pp. 1–3, 2024.

[2] M. Abolhasani and E. Kumacheva, "The rise of self-driving labs in chemical and materials sciences," *Nature Synthesis*, vol. 2, no. 6, pp. 483–492, 2023.

[3] E. Callaway, "Set it and forget it': automated lab uses ai and robotics to improve proteins," *Nature*, vol. 625, no. 7995, pp. 436–436, 2024.

[4] E. Otović, M. Njirjak, D. Kalafatovic, and G. Mauša, "Sequential properties representation scheme for recurrent neural network-based prediction of therapeutic peptides," *Journal of chemical information and modeling*, vol. 62, no. 12, pp. 2961–2972, 2022.

[5] M. d. C. Aguilera-Puga, N. L. Cancelarich, M. M. Marani, C. de la Fuente-Nunez, and F. Plisson, "Accelerating the discovery and design of antimicrobial peptides with artificial intelligence," in *Computational Drug Discovery and Design.* Springer, 2023, pp. 329–352.

[6] M. Ramakrishnan, A. van Teijlingen, T. Tuttle, and R. V. Ulijn, "Integrating computation, experiment, and machine learning in the design of peptide-based supramolecular materials and systems," *Angewandte Chemie*, vol. 135, no. 18, p. e202218067, 2023.

[7] A. Cesaro, M. Bagheri, M. Torres, F. Wan, and C. de la Fuente-Nunez, "Deep learning tools to accelerate antibiotic discovery," *Expert Opinion on Drug Discovery*, vol. 18, no. 11, pp. 1245–1257, 2023.

[8] V. Apostolopoulos, J. Bojarska, T.-T. Chai, S. Elnagdy, K. Kaczmarek, J. Matsoukas, R. New, K. Parang, O. P. Lopez, H. Parhiz, C. O. Perera, M. Pickholz, M. Remko, M. Saviano, M. Skwarczynski, Y. Tang, W. M. Wolf, T. Yoshiya, J. Zabrocki, P. Zielenkiewicz, M. AlKhazindar, V. Barriga, K. Kelaidonis, E. M. Sarasia, and I. Toth, "A global review on short peptides: Frontiers and perspectives," *Molecules*, vol. 26, no. 2, 2021. [Online]. Available: https://www.mdpi.com/1420-3049/26/2/430

[9] M. Mahlapuu, C. Björn, and J. Ekblom, "Antimicrobial peptides as therapeutic agents: Opportunities and challenges," *Critical reviews in biotechnology*, vol. 40, no. 7, pp. 978–992, 2020.

[10] L. Wang, N. Wang, W. Zhang, X. Cheng, Z. Yan, G. Shao, X. Wang, R. Wang, and C. Fu, "Therapeutic peptides: Current applications and future directions," *Signal Transduction and Targeted Therapy*, vol. 7, no. 1, p. 48, 2022.

[11] J. L. Lau and M. K. Dunn, "Therapeutic peptides: Historical perspectives, current development trends, and future directions," *Bioorganic & medicinal chemistry*, vol. 26, no. 10, pp. 2700–2707, 2018.

[12] G. Shi, X. Kang, F. Dong, Y. Liu, N. Zhu, Y. Hu, H. Xu, X. Lao, and H. Zheng, "Dramp 3.0: an enhanced comprehensive data repository of antimicrobial peptides," *Nucleic acids research*, vol. 50, no. D1, pp. D488–D496, 2022.

[13] M. M. He, S. X. Zhu, J. R. Cannon, J. K. Christensen, R. Duggal, M. Gunduz, C. Hilgendorf, A. Hughes, I. Kekessie, M. Kullmann *et al.*, "Metabolism and excretion of therapeutic peptides: current industry practices, perspectives, and recommendations," *Drug Metabolism and Disposition*, vol. 51, no. 11, pp. 1436–1450, 2023.

[14] W. C. Duckworth, R. G. Bennett, and F. G. Hamel, "Insulin degradation: progress and potential," *Endocrine reviews*, vol. 19, no. 5, pp. 608–624, 1998.

[15] H. Tan, W. Su, W. Zhang, P. Wang, M. Sattler, and P. Zou, "Recent advances in half-life extension strategies for therapeutic peptides and proteins," *Current Pharmaceutical Design*, vol. 24, no. 41, pp. 4932–4946, 2018.

[16] C. Brian Chia, “A review on the metabolism of 25 peptide drugs,” *International Journal of Peptide Research and Therapeutics*, vol. 27, no. 2, pp. 1397–1418, 2021.

[17] K. Frank and M. J. Sippl, “High-performance signal peptide prediction based on sequence alignment techniques,” *Bioinformatics*, vol. 24, no. 19, pp. 2172–2176, 2008.

[18] P. Wang, L. Hu, G. Liu, N. Jiang, X. Chen, J. Xu, W. Zheng, L. Li, M. Tan, Z. Chen *et al.*, “Prediction of antimicrobial peptides based on sequence alignment and feature selection methods,” *PloS one*, vol. 6, no. 4, p. e18476, 2011.

[19] S. S. Hannenhalli and R. B. Russell, “Analysis and prediction of functional sub-types from protein sequence alignments,” *Journal of molecular biology*, vol. 303, no. 1, pp. 61–76, 2000.

[20] N. Palmer, J. R. Maasch, M. D. Torres, and C. de la Fuente-Nunez, “Molecular dynamics for antimicrobial peptide discovery,” *Infection and Immunity*, vol. 89, no. 4, pp. 10–1128, 2021.

[21] S. Liu, J. Bao, X. Lao, and H. Zheng, “Novel 3d structure based model for activity prediction and design of antimicrobial peptides,” *Scientific reports*, vol. 8, no. 1, p. 11189, 2018.

[22] F. Lobo, M. S. González, A. Boto, and J. M. Pérez de la Lastra, “Prediction of antifungal activity of antimicrobial peptides by transfer learning from protein pretrained models,” *International Journal of Molecular Sciences*, vol. 24, no. 12, p. 10270, 2023.

[23] M. Salem, A. Keshavarzi Arshadi, and J. S. Yuan, “Ampdeep: hemolytic activity prediction of antimicrobial peptides using transfer learning,” *BMC bioinformatics*, vol. 23, no. 1, p. 389, 2022.

[24] M. Attique, M. S. Farooq, A. Khelifi, and A. Abid, “Prediction of therapeutic peptides using machine learning: computational models, datasets, and feature encodings,” *Ieee Access*, vol. 8, pp. 148 570–148 594, 2020.

[25] J. Yan, J. Cai, B. Zhang, Y. Wang, D. F. Wong, and S. W. Siu, “Recent progress in the discovery and design of antimicrobial peptides using traditional machine learning and deep learning,” *Antibiotics*, vol. 11, no. 10, p. 1451, 2022.

[26] I. Erjavac, D. Kalafatovic, and G. Mauša, “Coupled encoding methods for antimicrobial peptide prediction: how sensitive is a highly accurate model?” *Artificial intelligence in the life sciences*, vol. 2, p. 100034, 2022.

[27] L. Wei, C. Zhou, H. Chen, J. Song, and R. Su, “Acpred-fl: a sequence-based predictor using effective feature representation to improve the prediction of anti-cancer peptides,” *Bioinformatics*, vol. 34, no. 23, pp. 4007–4016, 2018.

[28] P. G. Aronica, L. M. Reid, N. Desai, J. Li, S. J. Fox, S. Yadahalli, J. W. Essex, and C. S. Verma, “Computational methods and tools in antimicrobial peptide research,” *Journal of Chemical Information and Modeling*, vol. 61, no. 7, pp. 3172–3196, 2021.

[29] G. Mauša, M. Njirjak, E. Otović, and D. Kalafatovic, “Configurable soft computing-based generative model: The search for catalytic peptides,” *MRS Advances*, vol. 8, no. 19, pp. 1068–1074, 2023.

[30] V. Boopathi, S. Subramaniyam, A. Malik, G. Lee, B. Manavalan, and D.-C. Yang, “macppred: a support vector machine-based meta-predictor for identification of anticancer peptides,” *International journal of molecular sciences*, vol. 20, no. 8, p. 1964, 2019.

[31] L. Yu, R. Jing, F. Liu, J. Luo, and Y. Li, “Deepacp: a novel computational approach for accurate identification of anticancer peptides by deep learning algorithm,” *Molecular Therapy-Nucleic Acids*, vol. 22, pp. 862–870, 2020.

[32] S. Gull, N. Shamim, and F. Minhas, “Amap: Hierarchical multi-label prediction of biologically active and antimicrobial peptides,” *Computers in biology and medicine*, vol. 107, pp. 172–181, 2019.

[33] A. Qureshi, H. Tandon, and M. Kumar, “Avp-ic50pred: Multiple machine learning techniques-based prediction of peptide antiviral activity in terms of half maximal inhibitory concentration (ic50),” *Peptide Science*, vol. 104, no. 6, pp. 753–763, 2015.

[34] Y. Hu, Z. Wang, H. Hu, F. Wan, L. Chen, Y. Xiong, X. Wang, D. Zhao, W. Huang, and J. Zeng, “Acme: pan-specific peptide–mhc class i binding prediction through attention-based deep neural networks,” *Bioinformatics*, vol. 35, no. 23, pp. 4946–4954, 2019.

[35] J. A. Beltran, L. Aguilera-Mendoza, and C. A. Brizuela, "Optimal selection of molecular descriptors for antimicrobial peptides classification: an evolutionary feature weighting approach," *BMC genomics*, vol. 19, pp. 79–92, 2018.

[36] B. Lee, M. K. Shin, T. Kim, Y. J. Shim, J. W. J. Joo, J.-S. Sung, and W. Jang, "Prediction models for identifying ion channel-modulating peptides via knowledge transfer approaches," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 12, pp. 6150–6160, 2022.

[37] Y. Pang, L. Yao, J. Xu, Z. Wang, and T.-Y. Lee, "Integrating transformer and imbalanced multi-label learning to identify antimicrobial peptides and their functional activities," *Bioinformatics*, vol. 38, no. 24, pp. 5368–5374, 2022.

[38] N. Lane and I. Kahanda, "Deepacppred: a novel hybrid cnn-rnn architecture for predicting anti-cancer peptides," in *Practical Applications of Computational Biology & Bioinformatics, 14th International Conference (PACBB 2020) 14*. Springer, 2021, pp. 60–69.

[39] L. Wei, X. Ye, T. Sakurai, Z. Mu, and L. Wei, "Toxibtl: prediction of peptide toxicity based on information bottleneck and transfer learning," *Bioinformatics*, vol. 38, no. 6, pp. 1514–1524, 2022.

[40] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. Pmlr, 2013, pp. 1310–1318.

[41] P. Janković, E. Otović, G. Mauša, and D. Kalafatovic, "Manually curated dataset of catalytic peptides for ester hydrolysis," *Data in brief*, vol. 48, p. 109290, 2023.

[42] E. Otović, M. Njirjak, D. Jozinović, G. Mauša, A. Michelini, and I. Štajduhar, "Intra-domain and cross-domain transfer learning for time series data—how transferable are the features?" *Knowledge-Based Systems*, vol. 239, p. 107976, 2022.

[43] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[44] J. Chen, H. H. Cheong, and S. W. Siu, “xdeep-acpep: deep learning method for anticancer peptide activity prediction based on convolutional neural network and multitask learning,” *Journal of chemical information and modeling*, vol. 61, no. 8, pp. 3789–3803, 2021.

[45] J.-C. Li, “Imbalanced toxicity prediction using multi-task learning and oversampling,” in *2020 International Conference on Machine Learning and Cybernetics (ICMLC)*. IEEE, 2020, pp. 1–7.

[46] Y. P. Zhang and Q. Zou, “Pptpp: a novel therapeutic peptide prediction method using physicochemical property encoding and adaptive feature representation learning,” *Bioinformatics*, vol. 36, no. 13, pp. 3982–3987, 2020.

[47] L. Wei, C. Zhou, R. Su, and Q. Zou, “Pepred-suite: improved and robust prediction of therapeutic peptides using adaptive feature representation learning,” *Bioinformatics*, vol. 35, no. 21, pp. 4272–4280, 2019.

[48] P. Charoenkwan, C. Nantasenamat, M. M. Hasan, M. A. Moni, B. Manavalan, and W. Shoombuatong, “Umpred-frl: A new approach for accurate prediction of umami peptides using feature representation learning,” *International journal of molecular sciences*, vol. 22, no. 23, p. 13124, 2021.

[49] D. Eisenberg, R. M. Weiss, T. C. Terwilliger, and W. Wilcox, “Hydrophobic moments and protein structure,” in *Faraday Symposia of the Chemical Society*, vol. 17. Royal Society of Chemistry, 1982, pp. 109–120.

[50] D. Eisenberg, R. M. Weiss, and T. C. Terwilliger, “The hydrophobic moment detects periodicity in protein hydrophobicity.” *Proceedings of the National Academy of Sciences*, vol. 81, no. 1, pp. 140–144, 1984.

[51] A. Ikai, “Thermostability and aliphatic index of globular proteins,” *The Journal of Biochemistry*, vol. 88, no. 6, pp. 1895–1898, 1980.

[52] D. Osorio, P. Rondón-Villarreal, and R. Torres, “Peptides: a package for data mining of antimicrobial peptides,” *Small*, vol. 12, pp. 44–444, 2015.

[53] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[54] T. Mikolov, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[55] G. Cruciani, M. Baroni, E. Carosati, M. Clementi, R. Valigi, and S. Clementi, "Peptide studies by means of principal properties of amino acids derived from mif descriptors," *Journal of Chemometrics*, vol. 18, no. 3-4, pp. 146–155, 2004.

[56] M. Sandberg, L. Eriksson, J. Jonsson, M. Sjöström, and S. Wold, "New chemical descriptors relevant for the design of biologically active peptides. a multivariate characterization of 87 amino acids," *Journal of medicinal chemistry*, vol. 41, no. 14, pp. 2481–2491, 1998.

[57] G. Liang and Z. Li, "Factor analysis scale of generalized amino acid information as the source of a new set of descriptors for elucidating the structure and activity relationships of cationic antimicrobial peptides," *QSAR & Combinatorial Science*, vol. 26, no. 6, pp. 754–763, 2007.

[58] H. Mei, Z. H. Liao, Y. Zhou, and S. Z. Li, "A new set of amino acid descriptors and its application in peptide qsars," *Peptide Science: Original Research on Biomolecules*, vol. 80, no. 6, pp. 775–786, 2005.

[59] G. J. van Westen, R. F. Swier, J. K. Wegner, A. P. IJzerman, H. W. van Vlijmen, and A. Bender, "Benchmarking of protein descriptor sets in proteochemometric modeling (part 1): comparative study of 13 amino acid descriptor sets," *Journal of cheminformatics*, vol. 5, pp. 1–11, 2013.

[60] A. G. Georgiev, "Interpretable numerical descriptors of amino acid space," *Journal of Computational Biology*, vol. 16, no. 5, pp. 703–723, 2009, pMID: 19432540. [Online]. Available: https://doi.org/10.1089/cmb.2008.0173

[61] A. Zaliani and E. Gancia, "Ms-whim scores for amino acids: a new 3d-description for peptide qsar and qspr studies," *Journal of chemical information and computer sciences*, vol. 39, no. 3, pp. 525–533, 1999.

[62] F. Tian, P. Zhou, and Z. Li, "T-scale as a novel vector of topological descriptors for amino acids and its application in qsars of peptides," *Journal of molecular structure*, vol. 830, no. 1-3, pp. 106–115, 2007.

[63] L. Yang, M. Shu, K. Ma, H. Mei, Y. Jiang, and Z. Li, "St-scale as a novel amino acid descriptor and its application in qsam of peptides and analogues," *Amino acids*, vol. 38, pp. 805–816, 2010.

[64] W. Li and A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences," *Bioinformatics*, vol. 22, no. 13, pp. 1658–1659, 2006.

[65] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[66] Y. Wang, H. Wu, and Y. Cai, "A benchmark study of sequence alignment methods for protein clustering," *BMC bioinformatics*, vol. 19, pp. 95–104, 2018.

[67] I. Lobo, "Basic local alignment search tool (blast)," *Nature Education*, vol. 1, no. 1, 2008.

[68] G. Wieds, "Bioinformatics explained: Blast versus smith-waterman," *CLCBio. http://www. clcbio. com/index. php*, 2007.

[69] C. Mideros-Mora, L. Miguel-Romero, A. Felipe-Ruiz, P. Casino, and A. Marina, "Revisiting the ph-gated conformational switch on the activities of hiska-family histidine kinases," *Nature communications*, vol. 11, no. 1, p. 769, 2020.

[70] L. Leonova, Z. Moravec, P. Sazama, J. Pastvova, L. Kobera, J. Brus, and A. Styskalik, "Hydrophobicity boosts catalytic activity: The tailoring of aluminosilicates with trimethylsilyl groups," *ChemCatChem*, vol. 15, no. 13, p. e202300449, 2023.

[71] N. Thakur, A. Qureshi, and M. Kumar, "Avppred: collection and prediction of highly effective antiviral peptides," *Nucleic acids research*, vol. 40, no. W1, pp. W199–W204, 2012.

[72] G. Shi, X. Kang, F. Dong, Y. Liu, N. Zhu, Y. Hu, H. Xu, X. Lao, and H. Zheng, "DRAMP 3.0: an enhanced comprehensive data repository of antimicrobial peptides," *Nucleic Acids Research*, vol. 50, no. D1, pp. D488–D496, 08 2021. [Online]. Available: https://doi.org/10.1093/nar/gkab651

[73] U. Consortium, "Uniprot: a hub for protein information," *Nucleic acids research*, vol. 43, no. D1, pp. D204–D212, 2015.

LIST OF FIGURES

1.1 Number of publications per year indexed in Web of Science that satisfy query shown in Code listing 1.1. The 2024 (gray) data is incomplete as it was retrieved on July 11th, 2024. 6

1.2 Broad categorization of peptide representation schemes according to how they encode data, illustrated using the peptide sequence ACDAAC. 7

2.1 The schematic of standard RNN. 25

2.2 The schematic of LSTM. 27

3.1 Peptide sequence ADDC encoded using its physico-chemical and compositional properties which are shown in blue and red, respectively. 47

3.2 Peptide sequence ADDC encoded using one-hot vector encoding and padded to the length of 6. Vectors encoding amino acids are depicted in blue color, while vectors used for padding are depicted in red color. First 20 bits are used to encode the amino acids, while the 21^{st} bit encodes a stop signal. . . 49

3.3 Peptide sequence ADDC encoded using word embeddings and padded to the length of 6. Amino acids and stop signals are encoded with values from the embeddings matrix. Vectors encoding amino acids are depicted in blue color, while vectors used for padding are depicted in red color. The size of the embeddings is controlled by the variable N. 51

3.4 Peptide sequence ADDC encoded using sequential properties and padded to the length of 6. Each amino acid is encoded with 94 physico-chemical properties. The 95^{th} is used to signal the end of the sequence. Vectors encoding amino acids are depicted in blue color, while vectors used for padding are depicted in red color. 54

4.1 An example of a) a global alignment using Needleman-Wunsch algorithm, and b) a local alignment using BLASTp algorithm. For the sake of example, Needlamn-Wunsch algorithm uses a basic scoring scheme with a match score set to 1, a mismatch penalty set to -1 and a linear gap penalty set to -1. The resulting alignment score is 1 and percent identity of 88.8%. BLASTp algorithm used the default settings which include BLOSUM62 substitution matrix for scoring and affine gap penalty with opening penalty set to -11 and extension penalty set to -1. The resulting bit score for BLASTp is 21.8 and percent identity is 58.3%. Input sequences *GCPHZYPHCCDC* and *GCPSPCCDC* are intentionally segmented to contrast their identical and different parts. Matching residues are shown in green, mismatches are shown in red and gaps are shown with dashes (-). 58

4.2 Distribution of peptide lengths in the dataset. 64

4.3 The distribution of amino acids in the dataset. 64

4.4 The distribution of theoretical physico-chemical properties relevant for the catalytic activity. Sequences containing only natural amino acids and tested with p-NPA substrate were included in this analysis. 66

4.5 The analysis of peptide similarities in CAT dataset. The analysis is limited to the sequences tested with p-NPA and containing only natural amino acids. 67

4.6 The dependency of the number of clusters for LOCOCV in CAT dataset on a chosen similarity threshold. The clustering algorithm was run 100 times each time with a random ordering of sequences. The blue line shows the average number of clusters, while the greyed area represents one standard deviation. 68

4.7 The distribution of peptide lengths in AVPPred dataset after similar peptides were removed. 69

4.8 The analysis of peptide similarities in AVPPred dataset. 69

4.9 The distribution of peptide lengths in AMP dataset after similar peptides were removed. 72

4.10 The analysis of peptide similarities in AMP dataset. 73

5.1 Model architectures used for each of the representation schemes. The additional input for C- and N-termini modifications is used only in the case of CAT dataset. 78
5.2 Procedure used to evaluate each of the representation schemes. 79
5.3 Dependency of the average MCC score on the classification threshold for the test set. 90

6.1 Procedures used in TL evaluation. 92
6.2 Options for selecting which layers to transfer and which layers to newly create. Blue color indicates layers that are transferred, red color indicates randomly initialized layers, and grey color depicts non-trainable parts of a model. The weights of connections going to the output neuron responsible for predicting probability are also reinitialized. 94
6.3 Dependency of ROC-AUC score for baseline and TL models on AVPPred target dataset at various sizes. For each dataset size, 100 runs have been performed and the average is reported in figures. 97
6.4 Dependency of ROC-AUC score for baseline and TL models on AMP-ExAVP target dataset at various sizes. For each dataset size, 100 runs have been performed and the average is reported in figures. 97

LIST OF TABLES

2.1 Overview of related research articles that employed ML for the peptide function prediction. **CO**=compositional properties, **OE**=one-hot encoding, **PC**=physico-chemical properties, **EV**=evolutionary properties, **EM**=embedding 17

2.2 Definition, domain, co-domain and visualization for sigmoid, tanh and ReLU activation functions. 19

2.3 Confusion matrix for binary classification. 29

2.4 Overview of related research articles that employed TL for the peptide function prediction. **CO**=compositional properties, **OE**=one-hot encoding, **PC**=physico-chemical properties, **GR**=graphical properties, **EV**=evolutionary properties, **EM**=embedding 38

3.1 Classification of amino acids into groups. 47

3.2 Physico-chemical properties of two peptides with the same amino acid composition, but with the different ordering of amino acids. All the examined physico-chemical properties have equal values except the hydrophobic moment marked in bold. 48

4.1 Distribution of peptides by catalysis mechanism separately for the peptides containing only natural amino acids and those that contain non-natural amino acids. 63

4.2 The number of peptides evaluated with a specific substrate. 63

4.3 Number of peptides with a specific combination of N- and C- termini. . . . 64

4.4 Number of instances in each of the subactivity categories in DRAMP 3.0 database. 71

4.5 Overview of the used datasets, their source, and number of instances after data filtration. 74

5.1 Hyperparameter values used in feature selection before hyperparameter optimization is performed. 81
5.2 Values of hyperparameters used in grid search for hyperparameter optimization. 82
5.3 Average performance for peptide and sequential properties representations when feature selection is and is not employed. The classification threshold was optimized for AVPPred and AMP datasets as the validation sets were of sufficient size in their case. Bold values indicate the maximum value. No statistically significant difference was found by Wilcoxon signed-rank test between the two methods in any of the metrics for AVPPred and AMP datasets. Holm-Bonferroni correction is used to control for FWER with a significance level set to 0.01. Statistical tests were not performed for CAT dataset. 84
5.4 Evaluation scores measured on all datasets with the best values for each metric marked in bold. The classification threshold was optimized for AVPPred and AMP datasets as the validation sets were of sufficient size in their case. Metrics that are significantly different from sequential properties, based on the p-values obtained from Friedman and Wilcoxon signed-ranks tests, are marked by an asterisk. Holm-Bonferroni correction is used for Friedman and Wilcoxon signed-rank tests to control FWER with a significance level of 0.01. 87
5.5 Ranking of evaluated representation schemes by their ROC-AUC score. . . 88

6.1 Average performance for baseline and TL models. Metrics in which a statistically significant difference was found by Wilcoxon signed-rank test for cases with AVPPred and AMP as target datasets are marked by an asterisk. Holm-Bonferroni correction is used to control for FWER with a significance level set to 0.01. Statistical tests were not performed for cases with CAT as the target dataset. Values written in bold indicate the best performing of two models for a given metric. 95

6.2 The distribution of optimal hyperparameters found by grid search for knowledge transfer from AMP-ExAVP to AVPPred. Each of the subtables corresponds to one combination of layers that were transferred. The second convolutional layer was randomly initialized and its number of filters and kernel size was optimized only when the layer was not transferred. The LSTM layer was reinstantiated and the number number of units was optimized only when the LSTM layer was not transferred. 99

A1 Frequencies of properties being selected by the feature selection in peptide properties representation scheme. The corresponding number of cases in which a feature was selected is provided in parentheses. 129

A2 Frequencies of individual properties being selected by the feature selection in sequential properties representation scheme. The corresponding number of cases in which a feature was selected is provided in parentheses. 131

A3 Frequencies of property categories being selected by the feature selection in sequential properties representation scheme. Each category is counted as selected if at least one if its features was selected and the corresponding number of such cases is provided in parentheses. The average number of selected features from each category is also provided. 139

B4 The distribution of optimal hyperparameters values found by grid search in a pretraining stage for transfer learning. 141

LIST OF CODE LISTINGS

1.1 Query used to find publications that simultaneously mention *machine learning*, *deep learning* or *neural network* and *peptide* in their abstracts or titles. 5
4.1 Query used to collect inactive sequences from UniProt repository. 71

LIST OF ABBREVIATIONS

Adam	Adaptive moment estimation
ANN	Artificial neural network
BiLSTM	Bidirectional LSTM
CNN	Convolutional neural network
DNN	Deep neural network
GM	Geometric mean
LLM	Large language model
LOCOCV	Leave-one-cluster-out cross-validation
LSTM	Long short-term memory
MCC	Matthews correlation coefficient
MD	Molecular dynamics
ML	Machine learning
MLP	Multi-layer perceptron
RF	Random forest
RNN	Recurrent neural network
ROC-AUC	Area under the rectifier operating characteristic curve
SGD	Stochastic gradient descent
SMILES	Simplified molecular input line entry system
SVM	Support vector machine
TL	Transfer learning

APPENDIX

A. Feature Selection Results

Table A1: Frequencies of properties being selected by the feature selection in peptide properties representation scheme. The corresponding number of cases in which a feature was selected is provided in parentheses.

(a) AVPPred dataset.

Property	Frequency
Cruciani 2	100.0% (100)
Hydrophobic moment	97.0% (97)
Aromatic rel.	89.0% (89)
Small rel.	87.0% (87)
Nonpolar abs.	83.0% (83)
Acidic rel.	79.0% (79)
Nonpolar rel.	72.0% (72)
Cruciani 3	72.0% (72)
Hydrophobicity	70.0% (70)
Basic rel.	70.0% (70)
Aliphatic rel.	61.0% (61)
Polar rel.	60.0% (60)
Cruciani 1	60.0% (60)
Charged rel.	53.0% (53)
Tiny rel.	32.0% (32)
Tiny abs.	10.0% (10)
Aliphatic abs.	6.0% (6)
Basic abs.	4.0% (4)
Aromatic abs.	3.0% (3)
Boman	2.0% (2)
Polar abs.	1.0% (1)
Charged abs.	1.0% (1)
Acidic abs.	1.0% (1)
Small abs.	0.0% (0)
Isoelectric point	0.0% (0)
Instability index	0.0% (0)
Charge	0.0% (0)
Aliphatic index	0.0% (0)

(b) AMP dataset.

Property	Frequency
Cruciani 3	100.0% (100)
Aliphatic abs.	100.0% (100)
Hydrophobic moment	99.0% (99)
Cruciani 2	98.0% (98)
Charge	98.0% (98)
Boman	98.0% (98)
Polar abs.	97.0% (97)
Nonpolar abs.	96.0% (96)
Small rel.	95.0% (95)
Small abs.	95.0% (95)
Basic abs.	94.0% (94)
Isoelectric point	92.0% (92)
Acidic abs.	90.0% (90)
Nonpolar rel.	88.0% (88)
Aromatic abs.	88.0% (88)
Tiny rel.	86.0% (86)
Aliphatic rel.	83.0% (83)
Cruciani 1	82.0% (82)
Charged rel.	82.0% (82)
Aromatic rel.	82.0% (82)
Tiny abs.	80.0% (80)
Basic rel.	79.0% (79)
Hydrophobicity	77.0% (77)
Polar rel.	76.0% (76)
Charged abs.	73.0% (73)
Acidic rel.	71.0% (71)
Aliphatic index	24.0% (24)
Instability index	16.0% (16)

(c) CAT dataset.

Property	Frequency
Hydrophobicity	77.7% (167)
Cruciani 2	77.2% (166)
Hydrophobic moment	66.0% (142)
Tiny rel.	60.9% (131)
Cruciani 1	55.8% (120)
Aliphatic rel.	55.3% (119)
Small rel.	54.9% (118)
Cruciani 3	54.4% (117)
Acidic rel.	48.4% (104)
Small abs.	46.5% (100)
Polar rel.	43.7% (94)
Basic rel.	34.9% (75)
Nonpolar rel.	29.8% (64)
Aromatic rel.	29.3% (63)
Charged rel.	27.4% (59)
Tiny abs.	21.9% (47)
Aliphatic abs.	16.3% (35)
Boman	15.3% (33)
Charge	14.0% (30)
Acidic abs.	9.8% (21)
Polar abs.	7.9% (17)
Isoelectric point	6.0% (13)
Nonpolar abs.	5.6% (12)
Basic abs.	5.1% (11)
Aromatic abs.	4.7% (10)
Charged abs.	4.2% (9)
Instability index	0.0% (0)
Aliphatic index	0.0% (0)

Table A2: Frequencies of individual properties being selected by the feature selection in sequential properties representation scheme. The corresponding number of cases in which a feature was selected is provided in parentheses.

(a) AVPPred dataset.

Property	Frequency
Hydrophobicity Cowan, pH=3.4	37.0% (37)
ProtFP 1	35.0% (35)
Hydrophobicity Fauchere	35.0% (35)
Z-scale 4	32.0% (32)
Cruciani 1	32.0% (32)
Hydrophobicity Cid	31.0% (31)
st-scale 1	27.0% (27)
Hydrophobicity Rose	26.0% (26)
t-scale 3	21.0% (21)
Hydrophobicity Janin	21.0% (21)
Hydrophobicity Casari	19.0% (19)
VHSE 5	17.0% (17)
VHSE 2	17.0% (17)
ProtFP 4	17.0% (17)
Hydrophobicity Welling	17.0% (17)
BLOSUM 5	17.0% (17)
Z-scale 3	16.0% (16)
ProtFP 7	16.0% (16)
Hydrophobicity Argos	16.0% (16)
Hydrophobicity Abraham-Leo	16.0% (16)
BLOSUM 2	16.0% (16)
t-scale 1	15.0% (15)
MSWHIM 3	15.0% (15)
Hydrophobicity Bull-Breese	15.0% (15)
FASGAI 5	15.0% (15)
FASGAI 3	15.0% (15)
Hydrophobicity Fasman	14.0% (14)
Cruciani 3	14.0% (14)
BLOSUM 9	14.0% (14)
VHSE 7	13.0% (13)
FASGAI 1	13.0% (13)
BLOSUM 6	13.0% (13)
st-scale 4	12.0% (12)
Hydrophobicity Wilson	12.0% (12)
Hydrophobicity Eisenberg	12.0% (12)
Hydrophobicity Miyazawa	11.0% (11)
Hydrophobicity Chothia	11.0% (11)

Property	Frequency
t-scale 2	10.0% (10)
ProtFP 3	10.0% (10)
BLOSUM 1	10.0% (10)
Hydrophobicity Tanford	9.0% (9)
Hydrophobicity Kidera	9.0% (9)
Hydrophobicity Engelman	9.0% (9)
BLOSUM 10	9.0% (9)
t-scale 4	8.0% (8)
st-scale 3	8.0% (8)
VHSE 8	8.0% (8)
VHSE 3	8.0% (8)
ProtFP 2	8.0% (8)
MSWHIM 1	8.0% (8)
Hydrophobicity Parker	8.0% (8)
FASGAI 4	8.0% (8)
Hydrophobicity Wolfenden	7.0% (7)
Hydrophobicity Roseman	7.0% (7)
Hydrophobicity Cowan, pH=7.5	7.0% (7)
FASGAI 6	7.0% (7)
BLOSUM 4	7.0% (7)
t-scale 5	6.0% (6)
st-scale 7	6.0% (6)
st-scale 8	6.0% (6)
Z-scale 2	6.0% (6)
MSWHIM 2	6.0% (6)
Hydrophobicity Sweet	6.0% (6)
Hydrophobicity Levitt	6.0% (6)
Hydrophobicity Juretic	6.0% (6)
Hydrophobicity Jones	6.0% (6)
BLOSUM 7	6.0% (6)
st-scale 5	5.0% (5)
Z-scale 1	5.0% (5)
ProtFP 6	5.0% (5)
ProtFP 5	5.0% (5)
Hydrophobicity Rao	5.0% (5)
Hydrophobicity Prabhakaran	5.0% (5)
Hydrophobicity Ponnuswamy	5.0% (5)
Hydrophobicity Kyte-Doolittle	5.0% (5)
Hydrophobicity Hopp-Woods	5.0% (5)
Hydrophobicity Aboderin	5.0% (5)
BLOSUM 3	5.0% (5)

Property	Frequency
VHSE 4	4.0% (4)
VHSE 1	4.0% (4)
ProtFP 8	4.0% (4)
Hydrophobicity Kuhn	4.0% (4)
Hydrophobicity Goldsack	4.0% (4)
FASGAI 2	4.0% (4)
Z-scale 5	3.0% (3)
Hydrophobicity Manavalan	3.0% (3)
st-scale 6	2.0% (2)
st-scale 2	2.0% (2)
VHSE 6	2.0% (2)
Hydrophobicity Guy	2.0% (2)
Hydrophobicity Black-Mould	2.0% (2)
Hydrophobicity Zimmerman	1.0% (1)
BLOSUM 8	1.0% (1)
Cruciani 1	0.0% (0)

(b) AMP dataset.

Property	Frequency
t-scale 3	91.0% (91)
Z-scale 5	77.0% (77)
BLOSUM 5	56.0% (56)
Hydrophobicity Aboderin	46.0% (46)
VHSE 5	42.0% (42)
t-scale 4	38.0% (38)
st-scale 4	33.0% (33)
Z-scale 3	33.0% (33)
Hydrophobicity Levitt	31.0% (31)
Cruciani 1	30.0% (30)
Z-scale 4	29.0% (29)
Hydrophobicity Hopp-Woods	29.0% (29)
FASGAI 4	29.0% (29)
MSWHIM 2	28.0% (28)
Hydrophobicity Miyazawa	28.0% (28)
BLOSUM 7	28.0% (28)
ProtFP 6	27.0% (27)
Z-scale 2	24.0% (24)
Hydrophobicity Casari	24.0% (24)
t-scale 2	23.0% (23)
t-scale 1	23.0% (23)
FASGAI 6	23.0% (23)
FASGAI 5	23.0% (23)
BLOSUM 2	23.0% (23)
st-scale 7	22.0% (22)
st-scale 2	22.0% (22)
VHSE 8	21.0% (21)
Hydrophobicity Fauchere	21.0% (21)
BLOSUM 6	21.0% (21)
VHSE 7	19.0% (19)
FASGAI 3	19.0% (19)
ProtFP 2	18.0% (18)
Hydrophobicity Wilson	18.0% (18)
ProtFP 7	17.0% (17)
Hydrophobicity Welling	17.0% (17)
Hydrophobicity Jones	17.0% (17)
st-scale 8	16.0% (16)
VHSE 3	16.0% (16)
Hydrophobicity Rose	16.0% (16)
st-scale 5	15.0% (15)
st-scale 1	15.0% (15)
VHSE 2	15.0% (15)

Property	Frequency
VHSE 1	15.0% (15)
Hydrophobicity Parker	15.0% (15)
Cruciani 3	15.0% (15)
BLOSUM 4	15.0% (15)
t-scale 5	14.0% (14)
Z-scale 1	14.0% (14)
Hydrophobicity Ponnuswamy	14.0% (14)
Hydrophobicity Kyte-Doolittle	14.0% (14)
Hydrophobicity Cowan, pH=3.4	14.0% (14)
Hydrophobicity Chothia	14.0% (14)
BLOSUM 3	14.0% (14)
st-scale 3	13.0% (13)
Hydrophobicity Tanford	13.0% (13)
Hydrophobicity Kidera	13.0% (13)
Hydrophobicity Janin	13.0% (13)
Hydrophobicity Fasman	13.0% (13)
Hydrophobicity Argos	13.0% (13)
Hydrophobicity Abraham-Leo	13.0% (13)
MSWHIM 3	12.0% (12)
Hydrophobicity Manavalan	12.0% (12)
Hydrophobicity Bull-Breese	12.0% (12)
MSWHIM 1	11.0% (11)
FASGAI 2	11.0% (11)
BLOSUM 1	11.0% (11)
ProtFP 4	10.0% (10)
Hydrophobicity Wolfenden	10.0% (10)
Hydrophobicity Rao	10.0% (10)
Hydrophobicity Prabhakaran	10.0% (10)
BLOSUM 8	10.0% (10)
VHSE 4	9.0% (9)
ProtFP 5	9.0% (9)
ProtFP 1	9.0% (9)
Hydrophobicity Zimmerman	9.0% (9)
Hydrophobicity Sweet	9.0% (9)
Hydrophobicity Guy	9.0% (9)
VHSE 6	8.0% (8)
Hydrophobicity Engelman	8.0% (8)
Hydrophobicity Cowan, pH=7.5	8.0% (8)
Hydrophobicity Kuhn	7.0% (7)
Hydrophobicity Juretic	7.0% (7)
Hydrophobicity Black-Mould	7.0% (7)
BLOSUM 10	7.0% (7)
st-scale 6	6.0% (6)

Property	Frequency
Hydrophobicity Kuhn	7.0% (7)
Hydrophobicity Juretic	7.0% (7)
Hydrophobicity Black-Mould	7.0% (7)
BLOSUM 10	7.0% (7)
st-scale 6	6.0% (6)
Hydrophobicity Goldsack	6.0% (6)
FASGAI 1	6.0% (6)
ProtFP 8	5.0% (5)
ProtFP 3	5.0% (5)
Hydrophobicity Roseman	5.0% (5)
BLOSUM 9	5.0% (5)
Hydrophobicity Eisenberg	4.0% (4)
Hydrophobicity Cid	4.0% (4)
Cruciani 1	0.0% (0)

(c) CAT dataset.

Property	Frequency
Hydrophobicity Bull-Breese	58.1% (125)
ProtFP 7	30.2% (65)
Hydrophobicity Aboderin	24.2% (52)
VHSE 4	20.5% (44)
Z-scale 3	20.0% (43)
Hydrophobicity Argos	19.5% (42)
Hydrophobicity Abraham-Leo	16.7% (36)
Hydrophobicity Tanford	16.3% (35)
Hydrophobicity Black-Mould	14.9% (32)
t-scale 3	14.0% (30)
st-scale 7	13.5% (29)
Hydrophobicity Eisenberg	13.5% (29)
Hydrophobicity Fauchere	12.6% (27)
Hydrophobicity Engelman	12.6% (27)
Hydrophobicity Sweet	12.1% (26)
FASGAI 5	12.1% (26)
Hydrophobicity Jones	11.2% (24)
Hydrophobicity Ponnuswamy	10.2% (22)
BLOSUM 5	10.2% (22)
Hydrophobicity Janin	9.8% (21)
Hydrophobicity Goldsack	9.8% (21)
Hydrophobicity Prabhakaran	9.3% (20)
Hydrophobicity Chothia	9.3% (20)
Hydrophobicity Manavalan	8.8% (19)
Hydrophobicity Casari	8.8% (19)
Cruciani 1	8.8% (19)
Hydrophobicity Wilson	8.4% (18)
Hydrophobicity Miyazawa	8.4% (18)
Hydrophobicity Hopp-Woods	8.4% (18)
Hydrophobicity Cid	8.4% (18)
Z-scale 5	7.4% (16)
Hydrophobicity Parker	7.4% (16)
Hydrophobicity Kidera	7.4% (16)
Hydrophobicity Guy	7.4% (16)
Hydrophobicity Juretic	7.0% (15)
VHSE 7	6.5% (14)
Hydrophobicity Cowan, pH=3.4	6.5% (14)
Hydrophobicity Wolfenden	6.0% (13)
Hydrophobicity Levitt	6.0% (13)
FASGAI 3	6.0% (13)
ProtFP 4	5.6% (12)
t-scale 1	5.1% (11)

Property	Frequency
VHSE 2	5.1% (11)
VHSE 1	5.1% (11)
MSWHIM 3	5.1% (11)
Hydrophobicity Rao	5.1% (11)
BLOSUM 9	5.1% (11)
BLOSUM 7	5.1% (11)
t-scale 2	4.7% (10)
Z-scale 1	4.7% (10)
Hydrophobicity Cowan, pH=7.5	4.7% (10)
FASGAI 6	4.7% (10)
st-scale 3	4.2% (9)
VHSE 8	4.2% (9)
Hydrophobicity Welling	4.2% (9)
Hydrophobicity Rose	4.2% (9)
BLOSUM 10	4.2% (9)
st-scale 2	3.7% (8)
Hydrophobicity Kyte-Doolittle	3.7% (8)
Hydrophobicity Kuhn	3.7% (8)
Hydrophobicity Fasman	3.7% (8)
FASGAI 2	3.7% (8)
BLOSUM 2	3.7% (8)
BLOSUM 1	3.7% (8)
Z-scale 2	3.3% (7)
VHSE 5	3.3% (7)
ProtFP 6	3.3% (7)
Hydrophobicity Roseman	3.3% (7)
Cruciani 3	3.3% (7)
ProtFP 3	2.8% (6)
ProtFP 1	2.8% (6)
BLOSUM 3	2.8% (6)
st-scale 5	2.3% (5)
ProtFP 2	2.3% (5)
FASGAI 4	2.3% (5)
BLOSUM 8	2.3% (5)
BLOSUM 6	2.3% (5)
st-scale 8	1.9% (4)
VHSE 6	1.9% (4)
ProtFP 5	1.9% (4)
Hydrophobicity Zimmerman	1.9% (4)
BLOSUM 4	1.9% (4)
st-scale 6	1.4% (3)
VHSE 3	1.4% (3)
t-scale 4	0.9% (2)

Property	Frequency
st-scale 1	0.9% (2)
Z-scale 4	0.9% (2)
MSWHIM 2	0.9% (2)
t-scale 5	0.5% (1)
ProtFP 8	0.5% (1)
FASGAI 1	0.5% (1)
st-scale 4	0.0% (0)
MSWHIM 1	0.0% (0)
Cruciani 1	0.0% (0)

Table A3: Frequencies of property categories being selected by the feature selection in sequential properties representation scheme. Each category is counted as selected if at least one if its features was selected and the corresponding number of such cases is provided in parentheses. The average number of selected features from each category is also provided.

(a) AVPPred dataset.

Property	Frequency (count)	Average number of selected features
Hydrophobicity	99.0% (99)	4.23
ProtFP	67.0% (67)	1.49
BLOSUM62	65.0% (65)	1.51
VHSE	56.0% (56)	1.30
st-scale	54.0% (54)	1.26
Z-scale	50.0% (50)	1.24
FASGAI	50.0% (50)	1.24
t-scale	49.0% (49)	1.22
Cruciani	40.0% (40)	1.15
MS-WHIM	26.0% (26)	1.12

(b) AMP dataset.

Property	Frequency (count)	Average number of selected features
Hydrophobicity	100.0% (100)	5.33
t-scale	100.0% (100)	1.89
BLOSUM62	96.0% (96)	1.98
Z-scale	94.0% (94)	1.88
VHSE	80.0% (80)	1.81
st-scale	75.0% (75)	1.89
FASGAI	71.0% (71)	1.56
ProtFP	63.0% (63)	1.59
MS-WHIM	41.0% (41)	1.24
Cruciani	41.0% (41)	1.17

(c) CAT dataset.

Property	Frequency (count)	Average number of selected features
Hydrophobicity	100.0% (215)	3.93
ProtFP	42.8% (92)	1.15
VHSE	36.7% (79)	1.30
BLOSUM62	34.4% (74)	1.20
Z-scale	33.0% (71)	1.10
FASGAI	26.0% (56)	1.12
st-scale	25.1% (54)	1.11
t-scale	22.8% (49)	1.10
Cruciani	12.1% (26)	1.00
MS-WHIM	6.0% (13)	1.00

B. Optimal Hyperparameters Found During Pretraining

Table B4: The distribution of optimal hyperparameters values found by grid search in a pretraining stage for transfer learning.

(a) Pretraining on AVPPred.

Hyperparameter	**Values (Selection frequency)**			
Number of convolutional layers	0 (3%)	1 (35%)	2 (62%)	
Number of filters (1st convolutional layer)	16 (45%)	32 (28%)	64 (24%)	Layer not employed (3%)
Number of filters (2nd convolutional layer)	16 (18%)	32 (26%)	64 (18%)	Layer not employed (38%)
Kernel size (shared between convolutional layers)	4 (29%)	6 (30%)	8 (38%)	
Number of units (the LSTM layer)	64 (33%)	128 (26%)	256 (41%)	
Dropout factor	0.1 (40%)	0.2 (40%)	0.3 (20%)	

(b) Pretraining on AMP.

Hyperparameter	**Values (Selection frequency)**			
Number of convolutional layers	0 (0%)	1 (21%)	2 (79%)	
Number of filters (1st convolutional layer)	16 (0%)	32 (20%)	64 (80%)	Layer not employed (0%)
Number of filters (2nd convolutional layer)	16 (0%)	32 (11%)	64 (68%)	Layer not employed (21%)
Kernel size (shared between convolutional layers)	4 (44%)	6 (38%)	8 (18%)	
Number of units (the LSTM layer)	64 (1%)	128 (62%)	256 (37%)	
Dropout factor	0.1 (28%)	0.2 (38%)	0.3 (34%)	

(c) Pretraining on AMP-ExAVP.

Hyperparameter	**Values (Selection frequency)**			
Number of convolutional layers	0 (0%)	1 (12%)	2 (88%)	
Number of filters (1st convolutional layer)	16 (3%)	32 (17%)	64 (80%)	Layer not employed (0%)
Number of filters (2nd convolutional layer)	16 (0%)	32 (11%)	64 (77%)	Layer not employed (12%)
Kernel size (shared between convolutional layers)	4 (50%)	6 (29%)	8 (21%)	
Number of units (the LSTM layer)	64 (3%)	128 (63%)	256 (34%)	
Dropout factor	0.1 (31%)	0.2 (42%)	0.3 (27%)	

CURRICULUM VITAE

Erik Otović was born on June 19th, 1996, in Pula, Croatia. He enrolled in undergraduate studies at the University of Rijeka, Faculty of Engineering, in 2015. Erik earned his bachelor's degree in 2018 and his master's degree in 2020. That same year, he began his postgraduate doctoral studies in engineering sciences, specializing in computer science, at the Faculty of Engineering.

During his postgraduate studies, Erik was employed as an assistant and a researcher at the University of Rijeka, Faculty of Engineering, Department of Computer Engineering. He actively participated in the project "Design of short catalytic peptides and peptide assemblies (DeShPet)" under the leadership of Assoc. Prof. Goran Mauša, PhD and Assist. Prof. Daniela Kalafatović, PhD. His doctoral research focused on machine learning for peptide discovery, but he also conducted research in fields such as neuroevolution and energy-efficient computing.

LIST OF PUBLICATIONS

[1] E. Otović, M. Njirjak, D. Jozinović, G. Mauša, A. Michelini, and I. Štajduhar, "Intra-domain and cross-domain transfer learning for time series data—How transferable are the features?" *Knowledge-Based Systems*, vol. 239, p. 107976, 2022.

[2] E. Otović, M. Njirjak, D. Kalafatovic, and G. Mauša, "Sequential properties representation scheme for recurrent neural network-based prediction of therapeutic peptides," *Journal of chemical information and modeling*, vol. 62, no. 12, pp. 2961-2972, 2022.

[3] M. Njirjak, E. Otović, D. Jozinović, J. Lerga, G. Mauša, A. Michelini, and I. Štajduhar, "The choice of time—frequency representations of non-stationary signals affects machine learning model accuracy: A case study on earthquake detection from LEN-DB data" *Mathematics*, vol. 10, no. 6, p. 965, 2022.

[4] E. Otović, J. Lerga, D. Kalafatovic, and G. Mauša, "Neuroevolution for the Sustainable Evolution of Neural Networks," *Proceedings of 46th MIPRO ICT and Electronics Convention (MIPRO)*, pp. 1045-1051, 2023.

[5] P. Janković, E. Otović, G. Mauša, and D. Kalafatovic "Manually curated dataset of catalytic peptides for ester hydrolysis," *Data in brief*, vol. 48, p. 109290, 2023.

[6] G. Mauša, M. Njirjak, E. Otović, and D. Kalafatovic "Configurable soft computing-based generative model: The search for catalytic peptides," *MRS Advances*, vol. 8, no. 19, pp. 1068-1074, 2023.

[7] M. Negovetić, E. Otović, D. Kalafatovic, and G. Mauša "Efficiently solving the curse of feature-space dimensionality for improved peptide classification," *Digital Discovery*, vol. 3, pp. 1182-1193, 2024.