MACHINE-LEARNING-ASSISTED DE NOVO DESIGN OF MOLYBDENUM DISULFIDE BINDING PEPTIDES

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Biotechnology

By
Alp Deniz ÖĞÜT

July 2024
İZMİR

We approve the thesis of **Alp Deniz ÖĞÜT**

Examining Committee Members:

Asst. Prof. Deniz Tanıl YÜCESOY
Department of Bioengineering, İzmir Institute of Technology

Asst. Prof. Mehmet Serkan APAYDIN
Department of Computer Science, Acıbadem University

Asst. Prof. Emrah İNAN
Department of Computer Science, İzmir Institute of Technology

Prof. Dr. Uğur SEZERMAN
Department of Biostatistics and Medical Informatics, Acıbadem University

Prof. Dr. Engin ÖZÇİVİCİ
Department of Bioengineering, İzmir Institute of Technology

12 July 2024

Asst. Prof. Deniz Tanıl YÜCESOY
Supervisor, Department of Bioengineering
İzmir Institute of Technology

Asst. Prof. Mehmet Serkan APAYDIN
Co-Supervisor, Department of Computer Science
Acıbadem University

Assoc. Prof. Ceyda Öksel KARAKUŞ
Head of the Department of Bioengineering

Prof. Dr. Mehtap EANES
Dean of the Graduate School of Engineering and Sciences

ACKNOWLEDGEMENTS

I am grateful to my advisors Dr. Deniz Tanıl YÜCESOY and Dr. Mehmet Serkan APAYDIN for their constant assistance, guidance, and sincere efforts throughout this study, as they helped me gain wonderful insights into the complicated matters at hand.

I would also like to thank my dear colleagues Nursevim ÇELİK, Gizem ÇULHA, and İlker KAN along with valuable members of Yucesoy Laboratory, where I had the opportunity to share, discuss, and improve ideas and interpretations from different perspectives. Special appreciation is reserved for İlayda TÜLÜ, Kerem HAZNEDAR, and Eda YURDAMİL for their genuine interest, effort, and contributions in this work.

Finally, I'm deeply grateful to my family for being such loving, understanding, and encouraging parents. They have been the light for me with their empathy, critical thinking, sense of justice, and hard work. I am also eternally grateful to my partner and our son for filling my life with meaning, joy, and love.

ABSTRACT

MACHINE-LEARNING-ASSISTED DE NOVO DESIGN OF MOLYBDENUM DISULFIDE BINDING PEPTIDES

Peptides are molecular entities with a diverse set of functionalities vital for biological processes and biotechnological applications. Among their roles, the ability of peptides to bind to solid materials has gathered attention, particularly as building blocks in constructing bio-nano interfaces and molecular linkers. Directed evolution techniques such as iterative phage display, have emerged as capable tools for identifying peptides and proteins with specific affinities for various targets despite its constraints, particularly its low-throughput nature. Those limits have motivated the work on more advanced methodologies such as deep-directed evolution, which integrates high-throughput sequencing. By collecting massive amounts of data, deep-directed evolution provides a broad landscape of sequence information, thus enabling computational modeling and optimization of peptide sequences. This thesis aims to develop machine learning workflows that capture the sequence-function relationship from the data, allowing the design of peptides with desired functionalities. Two machine learning approaches were employed: the Random Forest algorithm (RF) and deep neural networks (DNN). By aggregating binding score predictions from the two models, the predictor achieved a Pearson correlation coefficient of 0.904 and a mean absolute error of 0.0304 on the high-confidence test set and was employed to design a candidate peptide as a proof of principle. Our findings emphasize the importance of including domain knowledge via peptide abundance weighting and amino acid encoding types while designing training strategies. The procedures outlined in this work demonstrate key steps towards designing a peptide sequence-function prediction platform with broad implications for bio-nanotechnology and engineering.

ÖZET

MOLİBDEN DİSÜLFİD BAĞLAYICI PEPTİTLERİN MAKİNE ÖĞRENİMİ DESTEKLİ DE NOVO TASARIMI

Kısa amino asit zincirleri, peptitler, biyolojik süreçler ve yüksek teknoloji uygulamaları için vazgeçilmez moleküllerdir. Geniş kullanım alanları arasında, moleküler tanıma özelliği ile bio-nano arayüzler oluşturmak ilgi toplayan bir araştırma konusu olmuştur. Yapılan çalışmalar sonucunda yönlendirilmiş evrim metodolojileri oluşturulmuş ve çeşitli hedeflere -enzim, antijen veya inorganik yapılar- bağlanan fonksiyonel peptit tanısı mümkün hale gelmiştir fakat bu geleneksel yaklaşım ölçeklenebilirlik ve sekans uzayındaki ilişkilerin anlaşılması konusunda zayıflıklar taşımaktadır. Bu zafiyetler, yüksek çıktılı sekanslama ve hesaplama verimlerinin artması ile beraber derin yönlendirilmiş evrim gibi daha güçlü teknolojilerinin geliştirilmesini motive etmiştir. Bu yöntemle üretilen büyük veri setleri, sekans-fonksiyon ilişkilerinin makine öğrenmesi ile modellenebilmesinin önünü açmıştır. Bu tezin amacı bu veri setlerine uygun bir makine öğrenmesi akışı oluşturmaktır. Bu düzlemde Random Forest algoritması ve derin nöral ağlar kullanılmış, eğitilen modellerin bağlanma puanı öngörüleri beraber kullanıldığında mutlak hata sırasıyla, 0.0304, Pearson korelasyon ölçütü 0.904 olarak elde edilmiştir. Bu modelleri kullanarak rastgele arama ve tekrarlayan optimizasyonlar ile güçlü bağlanan örnek bir peptit tasarlanmıştır. Bulgular alan bilgisinin makine öğrenme modeli eğitimdeki yerini vurgulamış, kullanılan örnek ağırlıklarının ve semantik amino asit vektörlerinin başarıya önemli katkıları gözlemlenmiştir. Bu çalışma çeşitli fonksiyonlara sahip peptit tasarlayabilen bir platform oluşturabilmek için temel noktaları göz önüne serer.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

Figure **Page**

LIST OF EQUATIONS

Equation **Page**

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Organisms have been evolving to adapt to their environment, which involves interacting with inorganic molecules in proximity and capability of weaving complex structures. As there are rare examples, such as magnetotactic bacteria producing structures that enable them to navigate and organize using magnetic forces, other examples are very much in our daily lives: the formation of shells among marine species, and of bones, nails, and teeth.[1,2,3]

This phenomenon can be exploited to craft specialized peptides that can interact with a vast variety of inorganic materials that can be useful for a wide spectrum of applications, such as in biosensing, bioremediation, and medical innovations.[4,5] This requires the ability to design peptides that can do well in terms of a particular function. For this purpose, various peptide design methodologies have been developed. Rational design is one of the approaches where molecular features of amino acids and their collective properties are considered. This approach enables researchers to adjust hydrophobicity, electric charge, and other properties to conform the designed peptide to the target in question.[6,7] Another widely used method to discover functional peptides is directed evolution. Directed evolution is a method that mimics the natural evolution process of biological molecules in the laboratory by iteratively diversifying genes and selecting improved variants. This method enables the development of genes, proteins, and peptides with desired functions by mimicking natural selection principles.[8] Upon advancements in computational tools, such as GPUs, and artificial intelligence, a significant amount of research has been shifted to in-silico methods. Along with traditional machine learning algorithms, deep neural networks by using various approaches such as direct classification and regression, generative adversarial networks (GANs), and protein language models have been proposed for amino acid sequence design by using peptide or protein databases for training.[9,10,11,12,13]

Deep-directed evolution proposed by Yucesoy et al.[14] describes an approach with only one iteration, where fallen phages are sequenced in every panning step to produce sequence data that can be utilized for training machine-learning models that are expected to predict function-based fitness scores for previously unseen functional amino acid sequences. This helps in avoiding multiple costly iterations of the experiment, decreases room for error that may be introduced in each of those iterations, expands the accessed sequence space, and enables extraction of key amino acid patterns by providing a massive amount of sequence information. This enables the application of innovative machine learning models and creates an immense potential for de novo peptide discovery.

In this study, we aim to take a deep look at the high-throughput phage-display experiment data, define a pre-processing methodology, and build and compare different predictive models that can reliably predict MoS_2 affinity of dodecapeptides, given the sequence in the context of deep-directed evolution.

1.1 Molybdenum Disulfide

There is a growing interest over the years in molybdenum disulfide in several areas of science, from materials science to nanotechnology. The layered structure of transition metal dichalcogenide is inherent and unique in providing good mechanical robustness and chemical stability, coupled with extraordinary electronic properties.[15,16] Such outstanding mechanical and electronic properties, together with the possibility of exfoliating MoS_2 into atomically thin layers, have opened up new ways for research and various technological applications.

In bulk form, MoS_2 is an indirect semiconductor, while in monolayer form, it is a direct bandgap semiconductor.[17] That makes monolayer MoS_2 very suitable for optoelectronic applications owing to its potential to enhance the efficiency of the light absorption and emission processes. Coupling the features of a direct bandgap together with high carrier mobility makes monolayer MoS_2 a very good material for fabricating high-performance field-effect transistors, photodetectors, and other electronic devices.[18,19]

The two-dimensional nature of MoS_2 monolayers retains several advantages over traditional bulk semiconductors with the improvement of electrostatic control, reduced

short-channel effects, and better scaling potential.[20] Some of these attributes seem to prefer MoS_2 for developing next-generation flexible electronics, wearable devices, and ultrasensitive sensors.[21] For example, on account of the atomic thickness of the MoS_2, it can integrate with a variety of heterostructures and composite materials, hence opening further possible applications.[22]

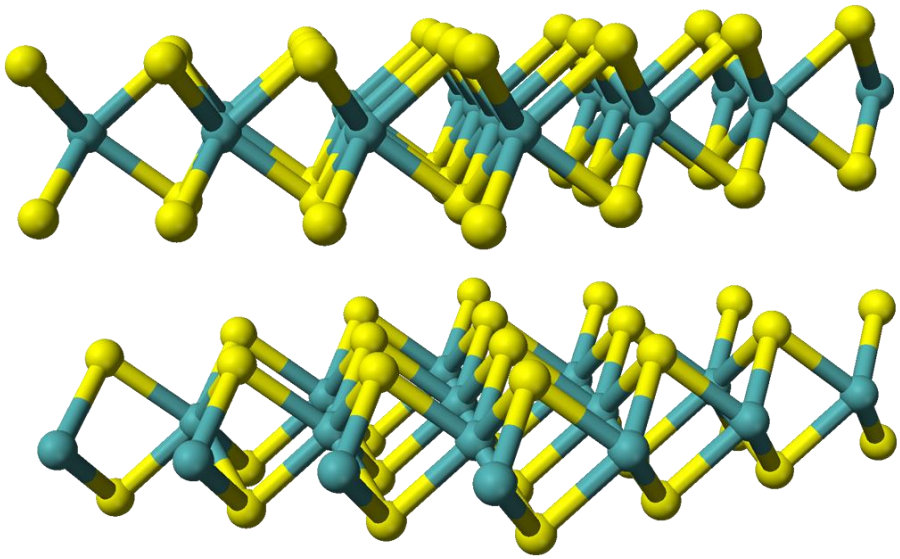

Figure 1.1. Molybdenum Disulfide layers with Sulphur atoms on the surfaces and Molybdenum atoms in between. By Ben Mills.
(Source: https://commons.wikimedia.org/w/index.php?curid=2976497**).**

A large amount of interest has gone into the interaction of MoS_2 with biological molecules, more so with peptides, in recent times. It vests from the rapid developments in the newly developing field of organic-inorganic interfaces. Various peptides have been designed, synthesized, and shown to bind specifically to MoS_2 surfaces, enabling a vast potential in terms of functionalization and bio-interfacing strategies of that material.[23] Such peptide-MoS_2 interactions open a way for the development of new bioelectronic interfaces, increased biocompatibility of MoS_2-based devices, and novel sensing platforms.[24]

Precise control over the surface properties of MoS_2 can be achieved through peptide binding and can lead to tuning of its electronic and optical properties for applications. Apart from this, functionalization with peptides helps in the integration of MoS_2 into biological systems and may enable a variety of new techniques that can be integrated into biosensing, drug delivery, and tissue engineering.[25]

Driven by this research, the importance of developing computational models to predict peptide-MoS_2 interactions and designing de novo peptides for achieving specific binding affinities grows. Such models would help accelerate the discovery of peptide sequences with optimal binding properties; therefore, one can design MoS_2-based devices and systems in a rational way to enhance their functionality and performance by employing them as molecular linkers.[26]

Coupled with the versatility of peptide-based functionalization, unique properties provided by monolayer MoS_2 create diverse possibilities for scientific explorations and technological innovations. Progressive research in peptide-MoS_2 interactions could have enormous potential for practical applications in nanoelectronics and biomedical engineering.

1.2 Directed Evolution

Directed evolution is a powerful method for protein engineering that exploits the process of natural selection to evolve proteins or nucleic acids for an artificial goal. The technique has become a valuable tool in biotechnology, drug design, and synthetic biology after its first emergence in the 1990s.[27] It was the result of the work of several researchers, including Willem P.C. Stemmer and Frances Arnold. While Stemmer[28] introduced DNA shuffling, a method for in vitro recombination of homologous genes, Arnold[29] took the lead in the use of iterative rounds of mutation and selection to engineer enzymes with enhanced properties.

Essentially, directed evolution follows the next three steps: (1) the generation of a variant library with high diversity; usually by random mutagenesis or recombination; (2) the selection or screening of variants showing the targeted properties; and (3) the diversification and amplification of successful variants with preference in preparation for sequential rounds of evolution.[30]

The self-directed cycle will permit the investigation of huge sequence spaces and probably finally attain solutions that rational design alone may not forecast. One of the primary advantages of directed evolution lies in how it engineers proteins without the requirement for detailed knowledge regarding protein structure-function relationships. This makes the technique particularly helpful for complex properties for which rational prediction or design might be very hard to do. Directed evolution can uncover novel mechanisms and strategies of protein function that enable an understanding of the fundamental principles in protein structure and evolution.[30,31]

However, the approach of directed evolution also has its limitations. Extensive labor is required for several rounds of selection and diversification. Selection or screening methods appropriate for targeting the relevant property are hard to design for many protein properties. In addition to the experimental difficulties, the number of sequences obtained is very low to be able to convey any information on functional patterns, and sequence space exploration is limited to induced mutations, hence not guaranteeing an optimal solution.[14,33]

In the last couple of years, further developments in this field have taken place, such as a variety of high-throughput screening techniques, increasing use of computation, and advancements in artificial intelligence that push guiding evolution procedures more efficiently.[34] These developments have greatly enhanced the scope and power of the method of directed evolution in the engineering of proteins with increasingly complex and diverse functions.

1.2 Deep-directed Evolution

Deep-directed evolution is an emerging approach that combines next-generation sequencing with machine learning to change the way we explore and optimize biological molecules, particularly peptides and proteins. It extends traditional methods of directed evolution, in terms of both scope and efficiency, as it also aims to map out the sequence-function relationship.

Similar to directed evolution, this technique is founded on the phage display method which depends on expressing peptides or proteins on the surface of viruses, and applying a selection towards a specific function, conventionally binding affinity.[34,35]

Deep-directed evolution differentiates itself by the integration of next-generation sequencing, and thus enables sophisticated data analysis. As described by Yucesoy et al., this approach allows the collection of massive amounts of data on sequence-function relationship.[14] By sequencing the phage DNA that is washed through successive rounds with increasing detergent concentrations, scientists gain access to a wealth of sequence information. The true potential of this technique arises when this vast data is utilized by advanced machine learning algorithms. Those algorithms can learn profound patterns within those sequences, build models, and map out the entire sequence-function landscape.[33]

Compared with most traditional directed evolution methods, the computational approach has the advantage of exploring much more extensive sequence space, reducing time-consuming and expensive experimental iterations, and providing optimum sequences that would otherwise have been missed by a purely experimental approach.[33,36]

The role of machine learning in deep directed evolution cannot be overemphasized. Such methods not only allow for the analysis of complex, high-dimensional data, but also afford a predictive power that might inform the design of future experiments. Deep-directed evolution combines experimental and computational strategies to form a very potent new paradigm in protein engineering and drug discovery, with immense potential for gross acceleration in the creation of novel therapeutics, enzymes, and other biologically active molecules.

1.3 Machine Learning Approaches

Machine learning is the part of the field of artificial intelligence that deals with the development of algorithms and statistical models for performing a certain job by learning from data. It is the process of training computer systems to recognize patterns and make predictions from data without explicit programming. Broadly, machine learning algorithms can be categorized under supervised, unsupervised, and reinforcement learning, each of which works in a specific manner to process and learn from data. This technology has become quite important in areas such as image and speech recognition; it has also found its way into recommendation systems, autonomous vehicles and natural language processing.

In this work, we have selected Random Forest algorithm and neural networks to model the result of the phage-display experiments due to their battle-tested robustness and performance.

1.3.1 Random Forests

Having been introduced by Leo Breiman[37] in 2001, Random Forest is an ensemble learning algorithm that combines multiple decision trees to make predictions, either a regression or a classification. The algorithm operates by constructing a multitude of decision trees during training and outputting the class that is the mode of the classes (classification) or average prediction (regression) of the individual trees.[38]

The key principles of Random Forest include bootstrap aggregating (bagging), where random subsets of the training data are selected with replacement to train each decision tree, reducing variance and overfitting.[37] Additionally, feature randomness is employed, meaning that at each node split, a random subset of features is considered, decorrelating the trees and improving generalization.[38] Finally, ensemble voting is used, where the predictions from all trees are aggregated through majority voting (classification) or averaging (regression) to make the final prediction.[39]

Random Forest offers several advantages. It often achieves high accuracy, outperforming individual decision trees and other algorithms.[40] The algorithm is robust, being less prone to overfitting and capable of handling noisy data and outliers well.[37] It can also measure the importance of each feature in the prediction process.[41] Moreover, Random Forest is versatile, as it can handle both classification and regression tasks with minimal hyperparameter tuning.[38]

However, Random Forest also has some disadvantages. The algorithm can be computationally expensive and memory-intensive, especially for large datasets.[42] The model is less interpretable than a single decision tree due to its ensemble nature.[43] In datasets with minority groups, Random Forest can produce results biased towards the majority groups due to suboptimal sampling.[44]

Despite these limitations, Random Forest remains a popular and effective algorithm in machine learning, with applications in various domains such as bioinformatics, remote sensing, and finance.[45]

1.3.2 Deep Neural Networks

Deep neural networks are an important part of machine learning models which are inspired by biological neural architectures. They consist of multilayer, interconnected nodes -neurons- with every succeeding layer transforming the input, based on a weighted connection with activation functions, progressively extracting features at each level.

Recent developments in DNNs have dramatically improved their capabilities. For example, attention mechanisms and transformer models have revolutionized natural language processing by allowing models to focus on the most relevant parts of an input sequence by using an attention mechanism.[46] GANs have enabled image generation and other creative capabilities by training two networks in tandem to produce realistic data.[47] Training schemes such as BERT and next-token prediction have set new benchmarks in understanding and generating human language, leading to advancements in chatbots, automated translation, and advanced text analysis.[48,49] These innovations in natural language processing are generating fundamental knowledge that is also applicable for decoding genes and proteins due to their shared sequence-based nature. DNNs can analyze vast amounts of biological data to predict the 3D structures of proteins from their amino acid sequences, a task that is computationally complex and critical for understanding protein function. Works such as Alphafold, OmegaFold, and ESM have ignited widespread enthusiasm in this regard.[12,50,51]

1.3.3 Language Models

Language models are a category of machine learning models that learn to predict the probability distribution of tokens in a sequence, given some context. The popular architectures used in language modeling consist of recurrent neural networks (RNNs) and long short-term memory networks (LSTMs), which have lately been overtaken by transformer models like BERT or GPT.[48,49,54,55,56] These models have produced state-of-the-art results in a very wide array of language tasks and revolutionized the field of natural language processing and artificial intelligence.

The task of predicting the next token in a sequence, known as next token prediction or autoregressive language modeling, is a fundamental approach to training these models. By learning to predict the next token based on the preceding context, language models can capture the statistical patterns, semantics, and long-range dependencies present in the training data.[52] This allows them to generate coherent and contextually relevant text and generate representative vectors of given text to perform various downstream natural language processing tasks such as text classification, named entity recognition, and machine translation.[49,56]

In the context of peptide function prediction and design, language models can be employed to learn the patterns and dependencies in peptide sequences. Through training with large datasets of peptide sequences, these models can capture the underlying statistics governing the relationship between sequence elements within the provided sequence library along with the unique description of the 12mer peptide as a whole. This knowledge can then be leveraged to calculate complex vector representations of given amino acid sequences for downstream objectives, or to generate new sequences on demand.[55]

1.4. Hypothesis & Aim

In this study, we hypothesize that by applying systematic pre-processing steps onto high-throughput phage-display experiment data and setting up various machine-learning models, we can accurately predict MoS_2 affinity of dodecapeptides based on their amino acid sequences, thereby proposing an efficient de novo peptide design methodology in the context of deep-directed evolution. The goal is to build a framework that achieves the following specific aims:

1. Process and prepare high-throughput phage display sequencing data
2. Train machine-learning models that capture the sequence-function landscape
3. Explore the models to design de novo peptides, optionally limited to a desired pattern.

CHAPTER 2

MATERIALS AND METHODS

Deep-directed evolution starts with a phage-display experiment with multiple panning rounds and high-throughput sequencing. The sequencing data is preprocessed into a filtered dataset of amino acid sequences to minimize noise while keeping the information as high as possible. The data is then used for training machine-learning models that can grasp the patterns within the sequences. The models are refined and confirmed with test sets.

2.1 Acquisition of Data

In this study, we aim to predict the binding scores of peptides given their amino acid sequence. To train and test our predictive models, we obtained high-throughput phage-display datasets produced by a previous study, through NCBI Sequence Reading Archive (SRA).[14,58] The run files are downloaded by using the SRA Toolkit software provided by NCBI.

The complete set consists of 24 FASTQ files, representing 3 biological and 2 technical replicate runs across panning steps (wash 1, wash 2, wash 3, eluate), totaling around 32GB. The experiment setup that produced the sequence data is described in Figure 2.1. Using such a setup, three experiments were performed with two technical replicates for each panning round as shown.

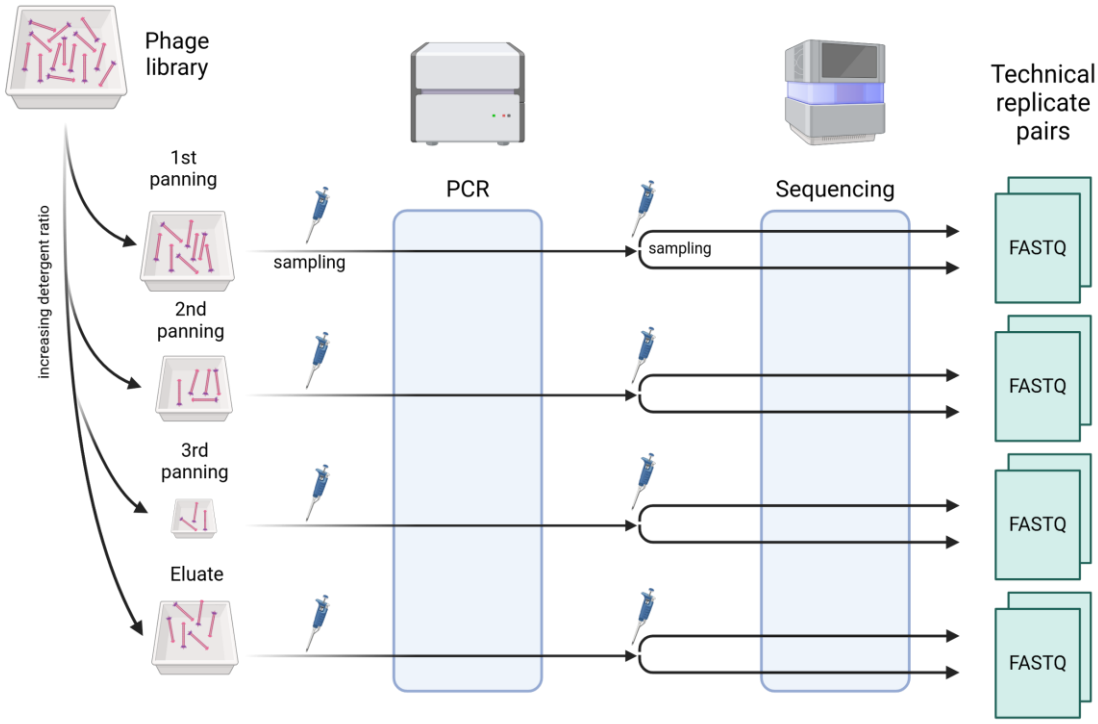

Figure 2.1. Source experiment data acquisition flow for a single biological experiment yielding 8 FASTQ files. The files contain sequence reads of a particular NGS run related to a set (biological experiment), wash (panning round), and technical replicate run. Created with biorender.com.

2.2 Pre-processing

The files are processed to count the number of observations of each DNA sequence for each NGS run using Python software, NumPy, and Pandas libraries. Sequence counts of technical replicate pairs are compared to check consistency and merged afterward, resulting in 12 files containing DNA sequence counts for each biological experiment (sets) and their respective panning steps. Additionally, sequence counts of sets are compared on each panning round to validate consistency across biological replicates. Next, these 12 files are further organized into 3 tables, representing 3 biological experiment datasets, each row containing reading counts of the panning rounds and eluate in that set.

Translation: DNA sequences are translated into amino acid sequences by mapping the codon table and counts of the ones that correspond to the same amino acid sequence are summed, while the ones that contain unidentified nucleotide readings or the ones that contain codons that do not correspond to valid amino acids are removed.

Concatenation of Datasets: Sequence readings across replicates, both biological and technical, have slightly different outcomes for the same sets of peptides as expected, especially for the peptides that are observed less, which correspond to noise in a machine learning perspective. To average out and reduce noise between biological experiment sets, the data is merged into one, resulting in a single data set with ~24 million unique peptides.

Cleaning: The dominant fraction of peptides has only 1 observation across all readings encompassing 24 files, and they have been filtered out from the dataset, yielding the root data set that contains ~8.7 million unique peptides. Removing singular peptides is also assumed to remove sequencing errors.

Binding Score Calculation: To establish a single value regression problem, a binding score definition is required to display phage-display data as a single numerical value to represent peptide fitness, in terms of a particular function, in this case, MoS_2 binding affinity, by utilizing the observation counts in each of the panning steps. This is achieved by using the center of abundance-mass (CoAM) metric as proposed by the study that produced the phage-display data.[14] The normalized CoAM equation is shown below:

$$binding\ score = \frac{1*wash2 + 2*wash3 + 3*eluate}{3*total}$$

Equation 2.1. Calculation of binding score per peptide, based on the number of observations per panning step.

Equation 2.1 assigns 0 score to peptides only observed in the first panning step (wash 1), and 1 to peptides observed only in the eluate, yielding a normalized binding score range ideal for machine learning applications.

Encoding: Encoding transforms peptide strings into vector forms that are suitable for machine learning algorithms. One popular method is one-hot encoding, where each

amino acid in a peptide is represented as a binary vector of size 20, with a single high (1) value corresponding to that acid, and all others are zero. In addition to one-hot encoding, amino acid sequences can also be encoded by other schemes such as VHSE8 (Vectors of Hydrophobic, Steric, and Electronic properties[59]), which projects measured molecular properties of amino acids onto the resulting 8-dimensional vector. Another candidate for encoding peptides is using protein language model embeddings produced by training such models with extensive protein sequence data and exporting its embedding layer. For such use, we have chosen Facebook Research's ESMv2 model embeddings.[12]

In this study, we employ multiple encoding schemes to represent peptide sequences for our predictive models. For the Random Forest algorithm, we utilize two primary encoding methods: one-hot encoding and VHSE8 (Vector of Hydrophobic, Steric, and Electronic properties). For peptides of 12 amino acids in length, the one-hot encoding produces a 240-dimensional vector (12 x 20, where 20 represents the number of standard amino acids), while VHSE8 results in a more compact 96-dimensional vector (12 x 8).

For our deep neural network models, we expand our encoding repertoire to include one-hot encoding, VHSE8, and embeddings from the ESM (Evolutionary Scale Modeling) framework. These encodings are utilized as 1D vectors for standard feed-forward neural networks and as 2D vectors for transformer-based language models to conform to Keras's natural language processing API, KerasNLP.[75] The ESM framework offers a variety of pre-trained models with different architectures and parameter counts. For this study, we selected an ESM model which comprises 12 layers and approximately 35 million parameters. This model accepts inputs with embedding vectors of size 480 for a rich representation of the peptide sequences. The choice of these diverse encoding schemes allows us to compare their effectiveness in capturing the relevant features of peptide sequences for functional score prediction. The one-hot encoding serves as a baseline, providing a simple and direct representation of the amino acid sequence. VHSE8 offers a more compact encoding that incorporates physicochemical properties, potentially capturing important functional characteristics of the peptides. The ESM embeddings, derived from large-scale unsupervised learning on evolutionary sequence data, provide a sophisticated representation that may capture complex patterns and relationships within protein sequences. By employing these varied encoding methods, we aim to identify the most effective approach for our peptide function prediction task and gain insights into the relative importance of different sequence features.

VHSE8 and ESM encodings are transformed using PaCMAP[60] (Pairwise Controlled Manifold Approximation Projection) to reduce their high-dimensional representations to 2D space, enabling visualization of how these encoding vectors represent amino acid residues. This dimensionality reduction allows for the observation of patterns, clusters, and relationships among residues based on their encoded properties. Projecting high-dimensional encoding vectors to 2D space offers several advantages, including improved interpretability, easier pattern recognition, facilitation of comparisons between encoding methods, feature analysis, quality assessment of encoding effectiveness, hypothesis generation, and enhanced communication of findings. The resulting 2D projections are plotted and analyzed, providing an intuitive way to visualize complex relationships and identify trends that might not be apparent in higher-dimensional spaces.

Sample Weights: In machine learning, sample weights are employed when fitting a model to ensure that certain observations are considered more "important" or "reliable" than others, or when the dataset is unbalanced, directly influencing the learning process. When a dataset includes numerical target values and confidence indicators, sample weights can be used.

$$w_i = C_i^{\varphi}$$

Equation 2.2. Sample weight calculation, where C_i is the total number of observations of *i*th peptide, φ is the suppression factor for the number of observations, and w_i is the resulting weight for *i*th peptide.

The total observation count is the number of encounters/observations of a particular peptide in all sequencing runs. Considering the entire process described in Figure 2.1., the number is assumed to be affected by the randomness introduced with fluid sampling, the following polymerase chain reaction (PCR), and variations in sequencing coverage. Since the binding score calculated from the small total number of observations can easily be altered by tiny amounts of additional readings, the phages -or the peptides- that are not abundant in the sequencing data are assumed to have less reliable binding

scores with higher variance. Therefore, we have proposed using total observation counts as sample weights after a simple transformation that intends to suppress the dramatic differences in populations:

Suppression factor φ helps to adjust the relation of the total number of observations of the peptides on sample weights. When φ equals 1, it applies the total number of observations, a value lower than one starts to suppress the weights and closes the gap between the effects of high and low-count peptides on the model. In this thesis work, φ is set to 0.7.

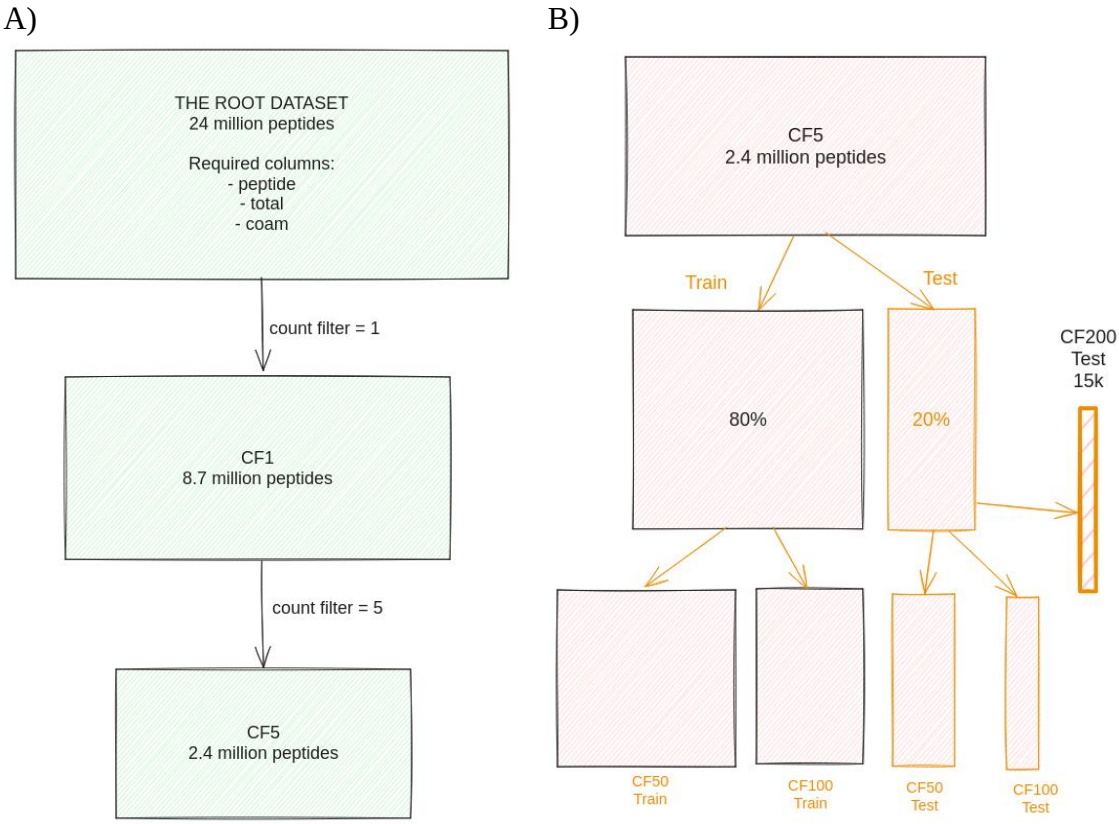

Figure 2.2. A) The root dataset and its sub-datasets. B) Train/Test split and its sub-datasets filtered with different count filter values.

Filtering: Additional filtering, also called *count filter, is* applied to reduce noise, and observe its effect on prediction performance, where peptides with less or equal to the filter value are removed from the dataset, trading off between the provided amount of

information and noise input to the machine learning models. The datasets filtered by the total observation count feature are tagged with their corresponding count filter values, abbreviated as *"cf"*. For example, a dataset obtained by using count filter 5 is referred to as *"cf5 dataset"*.

Validation: To validate the models, a 20% test set is randomly selected from the starting dataset. To be able to prevent data leaks between training and test sets, the main train/test split files are further filtered separately to build datasets with higher count filter values as shown in Figure 2.2.

2.3 Preliminary Analysis

The dataset is analyzed in terms of the number of readings, distribution of peptides and sample weights, and amino acid frequencies with varying count filter values. PacMap[60], a topology preserving alternative to PCA and t-SNE, is used for reducing the dimensionality of the peptide representations and visualizing them graphically. The same method is utilized for visualizing the phage-library coverage of the sequence space by superposing a subset of the experimental data along with randomly generated peptides.

Further analysis is performed to observe how amino acid frequencies differ between strong and weak binder groups. The frequencies for low (CoAM < 0.3) and high (CoAM > 0.7) scoring peptides are calculated by counting amino acids by the following formula:

$$AAF_i = \frac{N_i}{N_t}$$

Equation 2.3. Calculation of amino acid frequencies in a dataset.

where N_i is the total number of *i*th amino acid type, N_t is the total number of all amino acids, and AAF_i is the calculated frequency of the *i*th amino acid type in a complete list of amino acid sequences.

2.4 Model Training

To fit or train, validate, and test Random Forest and neural network models, NumPy, Scikit-learn[61], and Keras[62] Python libraries are used. Previously prepared datasets are fed into selected models after proper conversion of string inputs (peptides) to vectorized forms (encodings) along with sample weights -if applicable- and various metrics are logged, namely *mean square error*, *mean absolute error*, and *Pearson correlation coefficient*, for validation, test, and high-confidence test datasets.

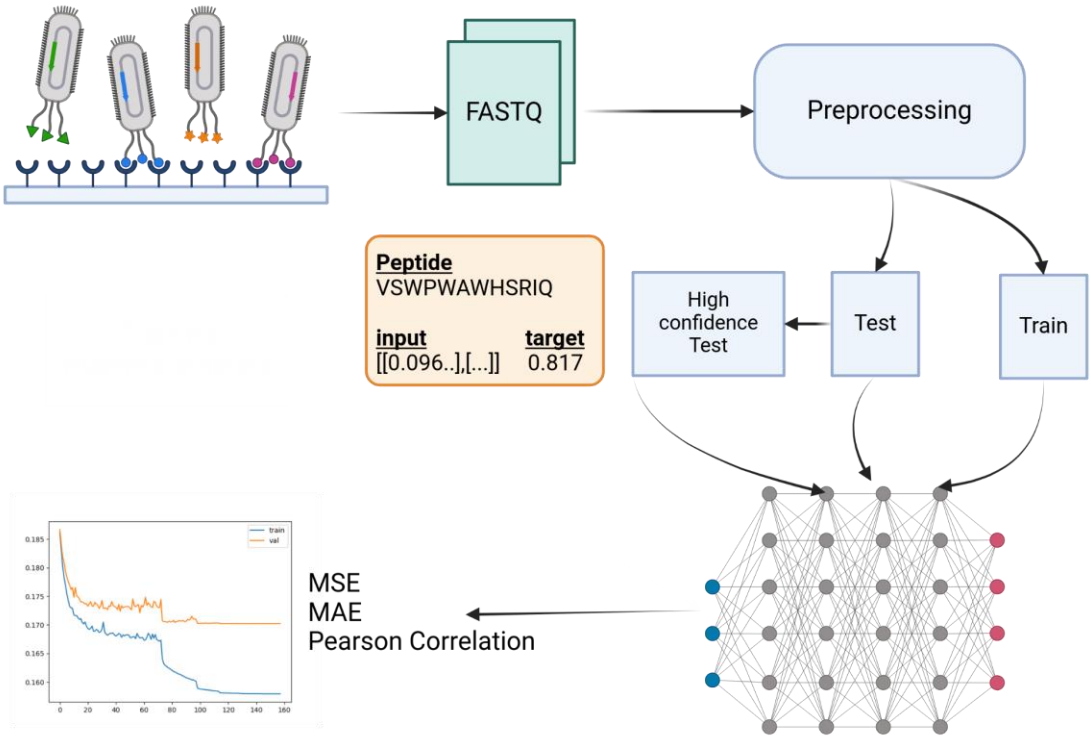

Figure 2.3. Regression model training/testing overview, displaying the data flow, and the training scheme applied to Random Forest and simple FFNN models. Created with BioRender.com.

Random Forests: Fitting Random Forest models is quite straightforward thanks to Scikit-learn module by utilizing its Random Forest regression class as described by the online documentation. Models are fitted to preprocessed datasets directly after application of the encodings, initially with default parameters. Hyperparameters are provided as arguments whenever required, during the hyperparameter optimization process. Feature importances which measure their significance by calculating the decrease in impurity - in decision trees, are visualized.

Baseline Regression: Random Forest models are built to observe the effects of parameters of the preprocessing, namely count filter, encoding scheme, and sample weighting to demonstrate the approximate effects of each parameter and draw a baseline for future experiments. Those trained models are evaluated principally with a common high-confidence test set to demonstrate their abilities to predict a peptide's affinity since validation sets differ and it becomes impossible to compare model performances when different count filters are provided.

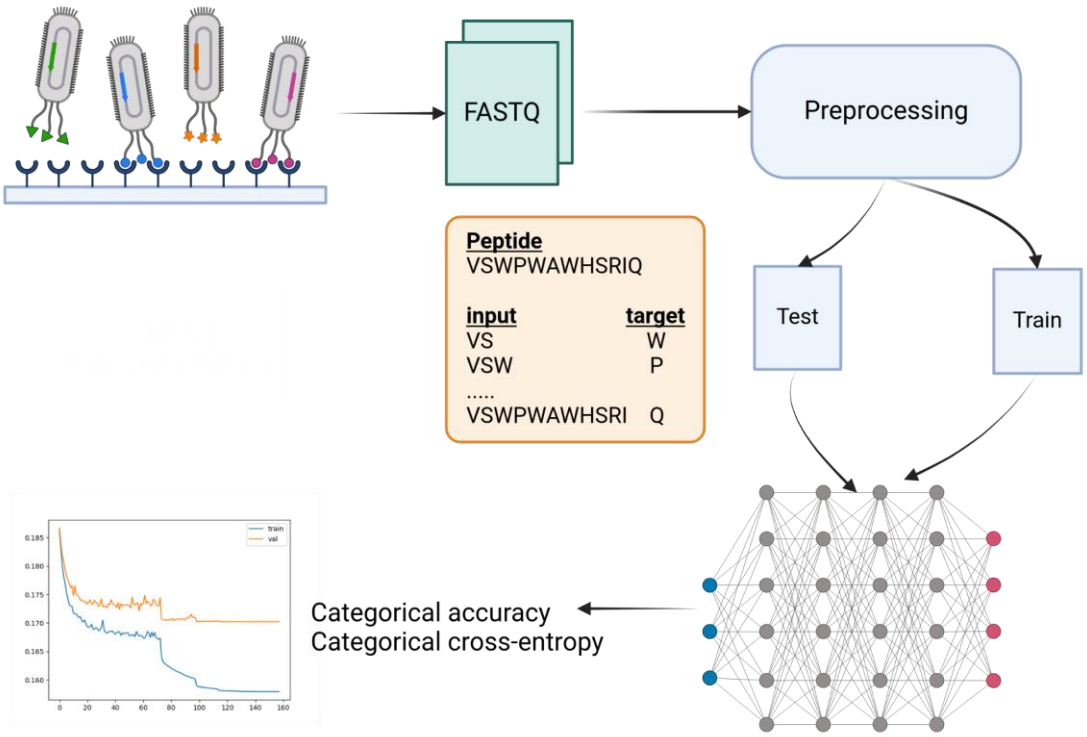

Figure 2.4. Peptide Language Model training overview, displaying the data flow, and autoregressive training scheme. Created with BioRender.com.

Neural Networks: Two types of neural network models are implemented in Python with Keras library: Simple feed-forward regression networks and feed-forward regression networks based on peptide language models (PepLM). The former, once built, trained with 1D input of size *sequence-length*encoding-size* and the binding score (CoAM) as the target output. Transformer-based PepLMs, on the other hand, are trained with an autoregressive strategy on 2D input, which consists of guessing the next amino acid, given the previous amino acids of the peptides in the dataset. The first *N* amino acids are provided to the model along with padding zeros, and the output expected is the *N+1*th amino acid. The two dimensions of the input matrix are sequence length and embedding vector size.

FFNN regression head is attached after converting the last hidden state of the resulting base model to a 1D vector as the input to FFNN regression head. For all models, early stopping is enabled, and training stops if validation loss does not decrease in the last 40 epochs. The learning rate is adjusted by a 0.8-fold decrease if validation loss does not decrease in the last 15 epochs. Dropout[63], weight regularization, and weight decay are applied in case of overfitting, during hyperparameter optimization.

Hyperparameter Optimization: Once the main dataset is decided for fitting or training the models, 10% of the data is split from the training dataset and used for evaluating the model during hyperparameter optimization. Hyperparameter optimization is restricted to *number of estimators* (trees) and *minimum samples per leaf* on Random Forest. Neural networks require much more intricate configuration parameters such as the *number of layers, number of hidden neurons, regularization coefficients, dropout rates*, application of *batch normalization*, and optimizer parameters such as *learning rate*, and *weight decay.*

Ensemble Averaging Model: To achieve the maximum prediction performance, the two trained models, the Random Forest and the combined PepLM+FFNN, are run in parallel and their results are averaged to produce the final prediction in the form of a simple ensemble setup, combining the powers of the different models.

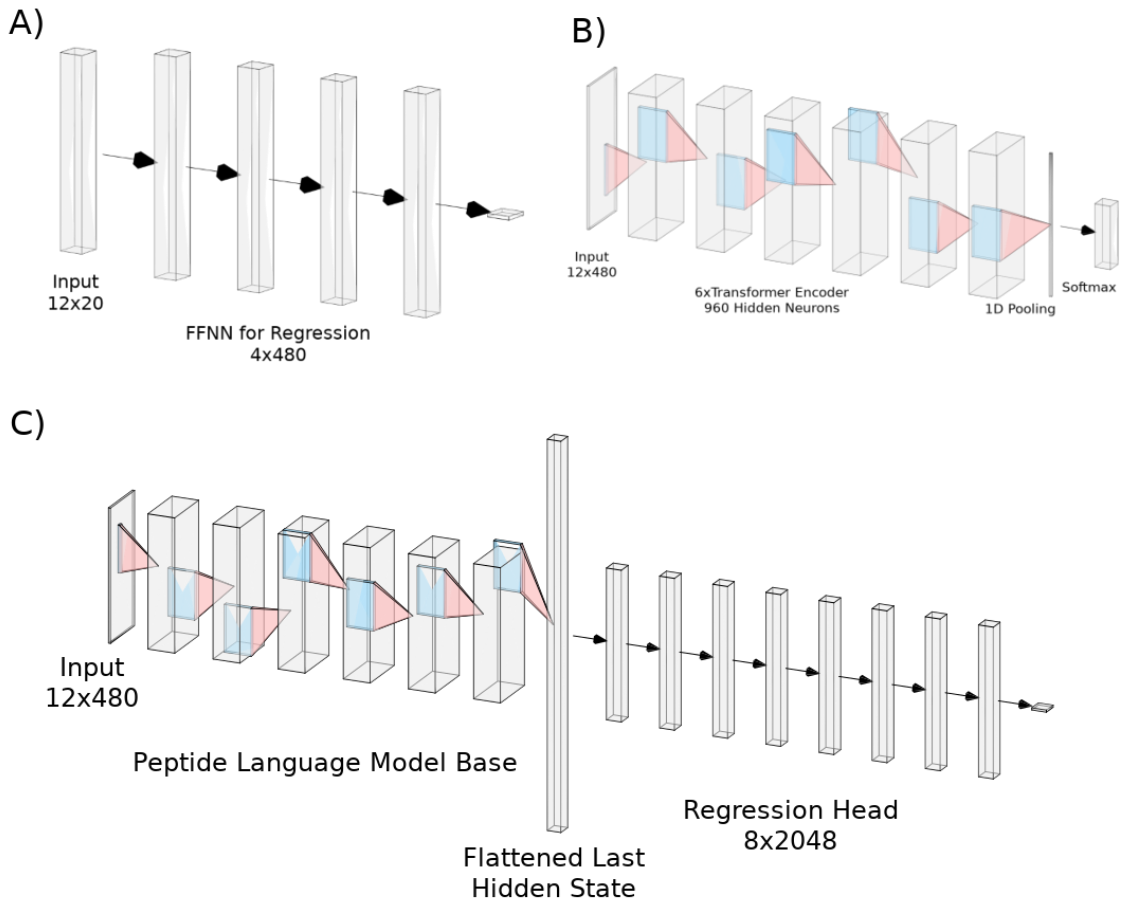

Figure 2.5. Examples of experimented models, demonstrating the neural network architectures used in this work. A) A simple feed-forward neural network. B) Peptide language model (PepLM) that is trained on next amino acid prediction. C) Combined PepLM and feed-forward network that yields the best prediction performance among neural networks.

2.5 Validation

In this study, the prediction performances of models that are trained with multiple different count filters, and ensemble model of Random Forest and neural network are compared. For this purpose, along with the test set, a high-confidence test set, *cf200 test set* with count filter 200 value is built from *cf5 test dataset* (Figure 2.2.). Both cf5 and cf200 test sets are used to measure the optimized models' performance, and we also report the training set performance to track overfitting. The metrics *mean square error*, *mean*

absolute error and *Pearson correlation coefficient* are calculated as in hyperparameter optimization. The test metrics in question are obtained without applying the sample weighting scheme to ensure comparability with models trained with different sample weight suppression coefficients, φ.

In decision-tree based models, a common method to observe the significance of each feature is to measure the *mean decrease in impurity* (MDI) of that feature. MDI is calculated as the average reduction in impurity each feature contributes across all trees in a forest, where impurity reduction is assessed using metrics like Gini impurity or entropy for classification and variance for regression. Features with higher MDI values are considered more important, as they significantly enhance the model's predictive accuracy.

Four peptides are collected from the literature and their binding affinities are predicted by selected models.[64,65,66] The predictions are compared with experimental observations. The peptides are provided in Table 2.1.

Table 2.1. Peptides that are collected from the literature with their assumed labels.[61,62,63]

Name	Sequence	Label
GRBP5-M6	IMVTASSAYDDY	Reference
MoS2-P15	GVIHRNDQWTAP	Strong
MoS2-P28	DRWVARDPASIF	Strong
MoS2-P3	SVMNTSTKDAIE	Weak

2.6 De Novo Design of Functional Peptides

Once the models are built, they can be used to search the sequence space for peptides with desired binding affinities. The sequence space can be explored with a stochastic search technique, such as Monte Carlo or the genetic algorithm, and can be accompanied by a post-exploration step with an evolutionary approach, that is mutating a candidate strong binder selected after the first stage. The justification for proposing stochastic/random search is that deterministic algorithms are NP-hard, whereas a non-deterministic run can be executed on average in polynomial time although without any

guarantee of global optima.[67] In this work, we have used a stochastic search approach that lists randomly generated peptides with highest predicted scores and further checked all possible mutations on the candidates iteratively. We have used the two trained machine-learning models to generate candidate peptides that are expected to bind strongest in two classes, random peptides, and random peptides excluding cysteine and aromatic amino acids, the residues which are observed to have dramatic effects on measurable binding score but are assumed to be non-specific. We later compared peptides found both by the Random Forest and the neural network model against each other's predicted scores, to lay out the differences. Finally, we have randomly explored the sequence space on the averaging ensemble model to produce two lists, consisting of random residues, and random residues excluding aromatic amino acids and sulfur-containing cysteine. The highest-scoring peptide is then further optimized by evaluating point mutations to increase the expected binding affinity to the maximum.

CHAPTER 3

RESULTS AND DISCUSSION

3.1 Datasets

Initial analysis displays the basic characteristics of the data generated by the experiment in the scope of deep-directed evolution. Twenty-four FASTQ files were downloaded from Sequence Read Archive (SRA)[58], totaling 32GB in size, and contain around 222 million DNA sequences, of which around 44 million are unique. Table 3.2 demonstrates the original distribution of sequences across sets and washes.

Tables 3.1. The number of total and unique DNA and amino acid sequences.

Number of total DNA sequences	~222M
Number of unique DNA sequences	~44M
Number of unique amino acid sequences	~24M

Table 3.2. Distribution of total number of readings across panning steps and sets.

	Number of Total DNA readings			
Set	Set 1	Set 2	Set 3	Total
Wash 1	~31.6M	~47.6M	~42.5M	~121.7M
Wash 2	~17.2M	~7.8M	~5M	~30.1M
Wash 3	~0.2M	~0.1M	~0.1M	~0.5M
Eluate	~20.8M	~23.1M	~25.7M	~69.7M

3.2 Pre-processing

Raw FASTQ files were processed to enumerate sequences for each panning step. Technical replicates were summed, and the resulting data were organized into tabular format with columns representing distinct panning steps. Subsequently, nucleotide sequences were translated into amino acid sequences. This process generated one CSV file per biological experiment.

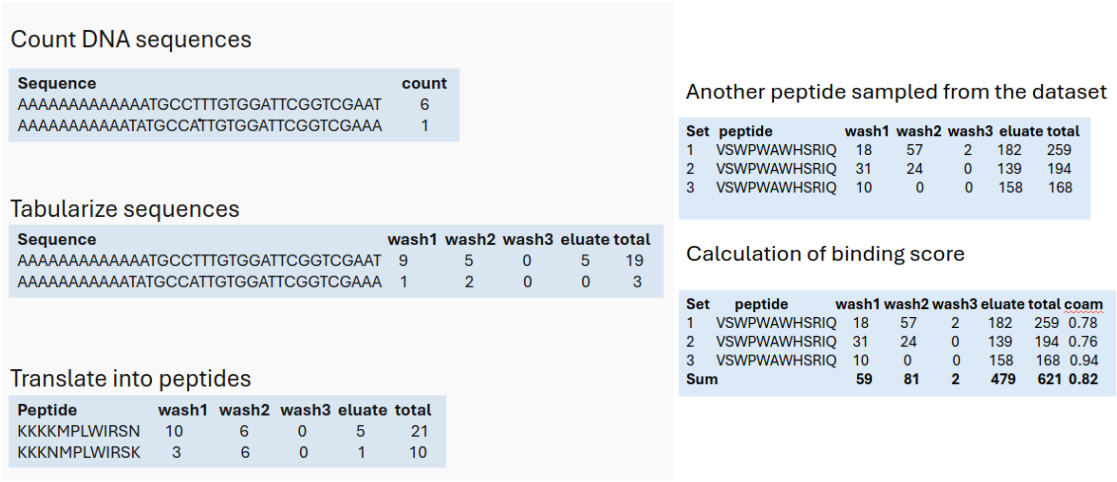

Figure 3.1. Left: Initial steps of preprocessing: Counting sequences of a sequencing run, tabularization, and translation into amino acid sequence. Right: sample peptide counts across sets and washes, along with candidate target values for training.

All biological sets are later summed to produce the aggregated dataset. Target scores were calculated using Equation 2.1. Representative data samples are illustrated in Figures 3.1.

Most sequences were detected in the first panning round, and the final eluate as shown in Table 3.2 and Figure 3.2. The abundance of sequences in the final round points out that a substantial portion of the peptides can stay on the MoS_2 surface. This can be attributed to the dynamics of binding of certain single amino acids to the material.

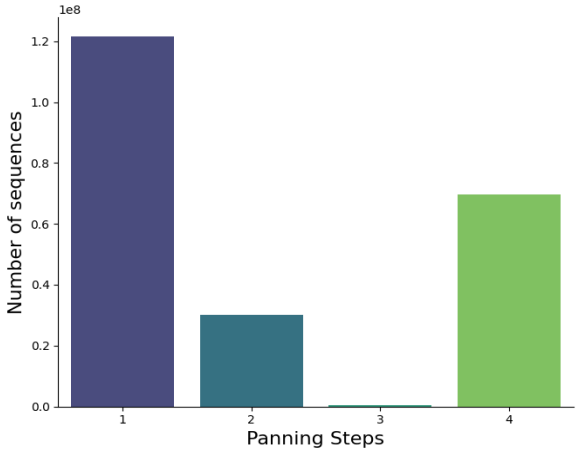

Figure 3.2. Distribution of reads across panning rounds.

To validate sequencing and to get an insight into the experiment data, the number of observations of distinct peptides across technical replicates (sequencing replicates within each set) and biological sets are compared. Figure 3.3 and 3.4. display scatter plots of the number of observations per peptide, and a corresponding slope calculated over the best fitting line.

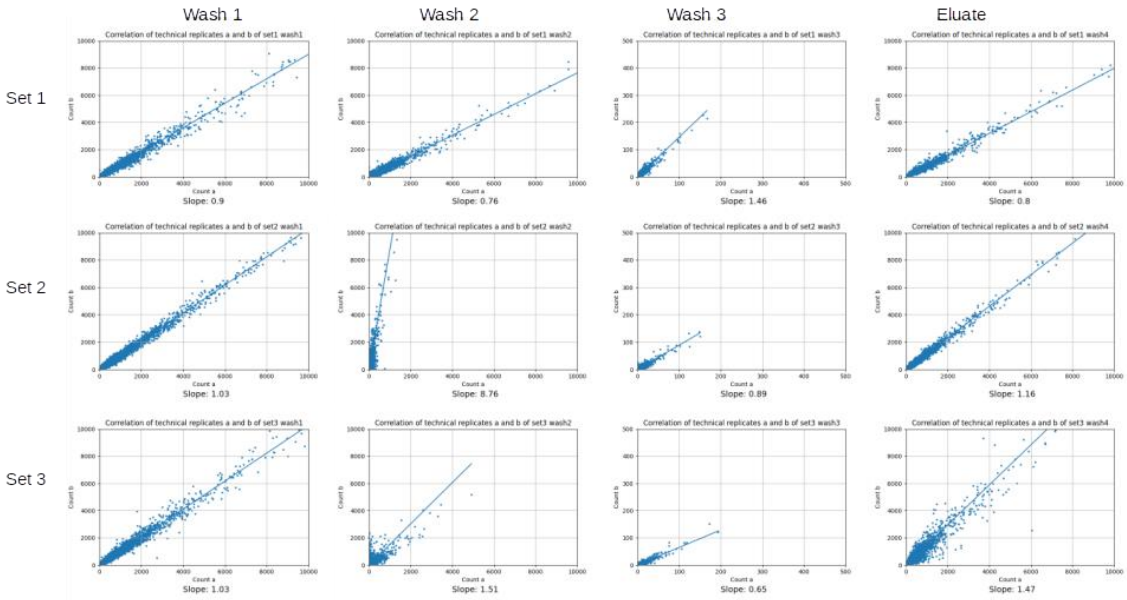

Figure 3.3. Correlation plots of technical replicate pairs.

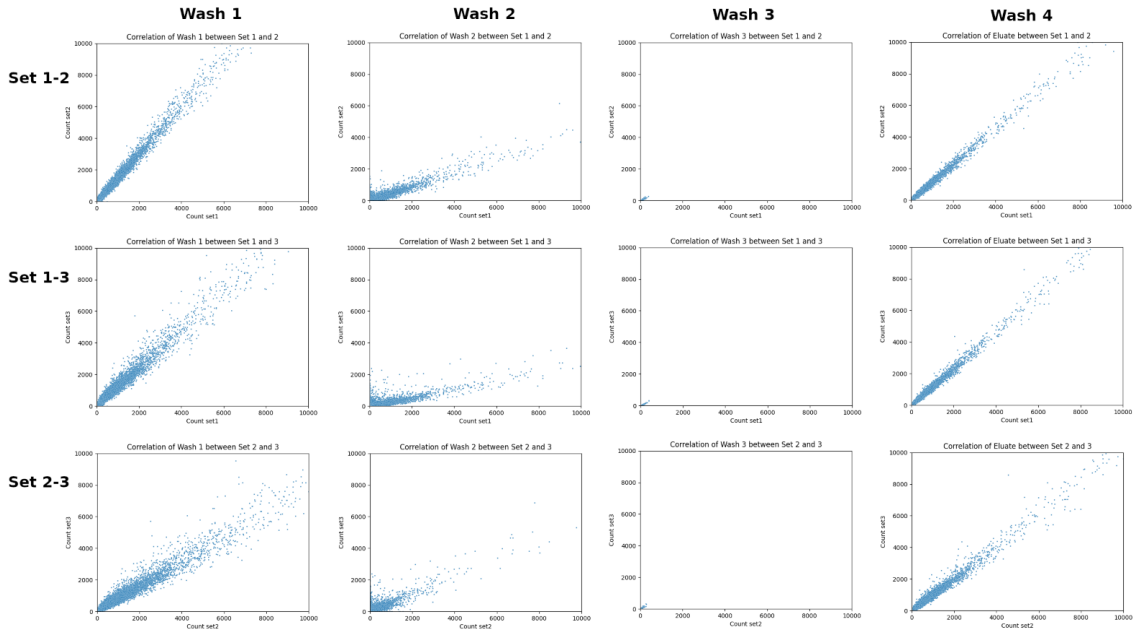

Figure 3.4. Correlation plots of combinations of biological sets on each panning round.

One-hot, VHSE8, and ESM encodings are applied to the data depending on the experiment. It is noteworthy that VHSE8 achieves significant dimensionality reduction when compared to one-hot representation, while incorporating physicochemical information in the input vectors. This compression is particularly crucial for the Random Forest algorithm, which is known for its memory-intensive nature. As the size of the embedding vectors increases, the number of features grows correspondingly, potentially limiting the algorithm's deployment due to practical hardware constraints.

Table 3.3. Input/output pair of an example peptide on different encoding schemes.

	Input		**Target**
	12mer peptide	Encoded peptide	Binding score
One-hot	VSWPWAWHSRIQ	[0, 0, 0, 0, ….] (240 elements)	0.817
VHSE8	VSWPWAWHSRIQ	[0.76, -0.92, 0.1…] (96 elements)	0.817
One-hot 2D	VSWPWAWHSRIQ	[[0, 0, 0, 0,…] (12x20 elements)	0.817
ESM480 2D	VSWPWAWHSRIQ	[[0.096..],[...]] (12x480 elements)	0.817

VHSE8 and ESM (vector size=480) encodings are transformed with PaCMAP and plotted in 2D in Figure 3.5 to inspect the representation of amino acids. 2D projections generated by PaCMAP align with known amino acid classes, conforming to expectations regarding distances between amino acids. Note the clusters of aromatic amino acids (W, Y, F), the relative proximity of negatively charged amino acids (D, E), the consistent distance of leucine and isoleucine (L and I, respectively).

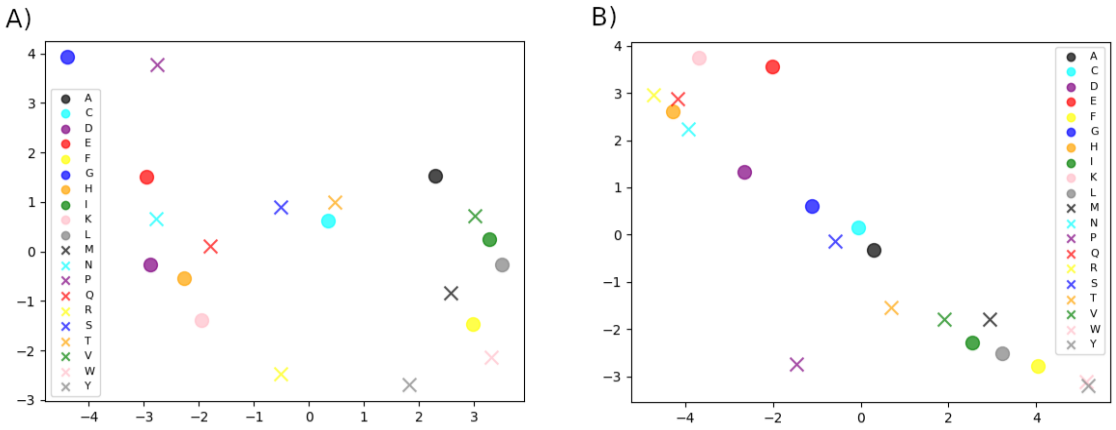

Figure 3.5. 2D projections of VHSE8 (A) and ESM480 (B) amino acid encodings, as produced by PaCMAP.

After pre-processing, we obtained a root dataset that was filtered to peptides with counts more than 5 (*count filter=5*), referred to as *cf5 dataset*. It is later split into training and test datasets. We have further filtered the resulting training and test sets separately whenever required.

3.3 Preliminary Analysis

Figure 3.6. clearly shows that the number of data rows dramatically decreases with increasing total observation count filter values, remarking the uneven distribution of sequences. The normalized distributions of binding scores for four datasets with different

count filters are shown in Figures 3.7, where the data reduces in size and accumulates towards the mean, while filter value increases. Additionally, Figure 3.8. displays the normalized distributions of the total number of observations per peptide on the binding score range.

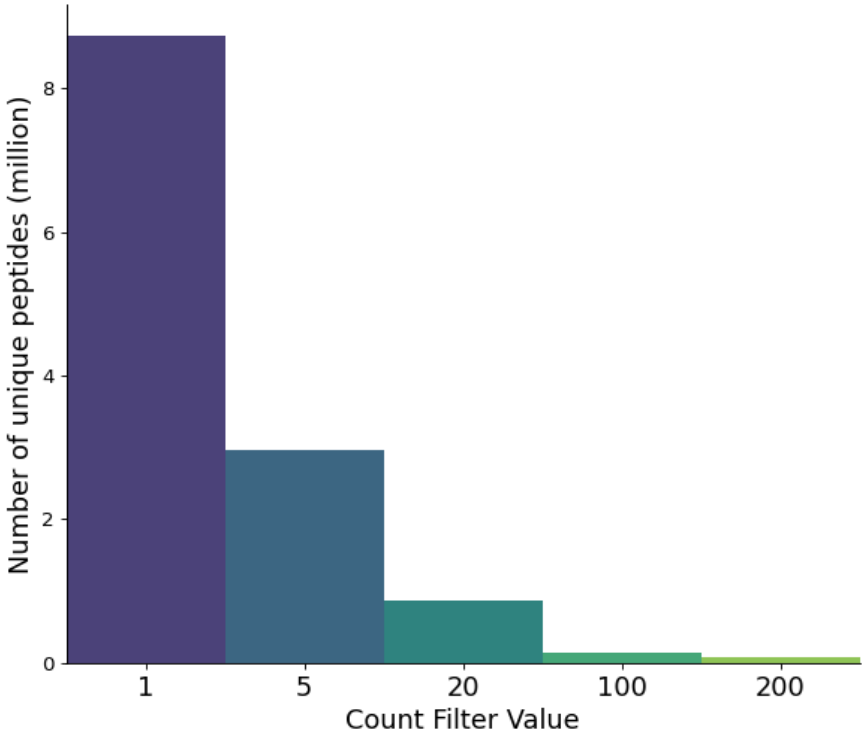

Figure 3.6. Number of distinct peptides vs applied count filter value.

Filtering results in a considerable decrease in the number of data points as the count filter value increases. This means that to get more confident data points, a huge portion of data that potentially bears significant signal, along with significant noise, is sacrificed. Filtering operations yield datasets that display the normal distribution characteristics which accumulate on the mean with decreasing standard deviation.

Amino acid frequencies show consistent differences between weak and strong binders. Among the strong binder subset, Sulphur-containing Cysteine (C) and aromatics Phenylalanine (F), Tryptophan (W) and Tyrosine (Y) are more frequent, whereas Aspartic Acid (D) and Glutamic Acid (E) are more frequent within the weak binder subset.

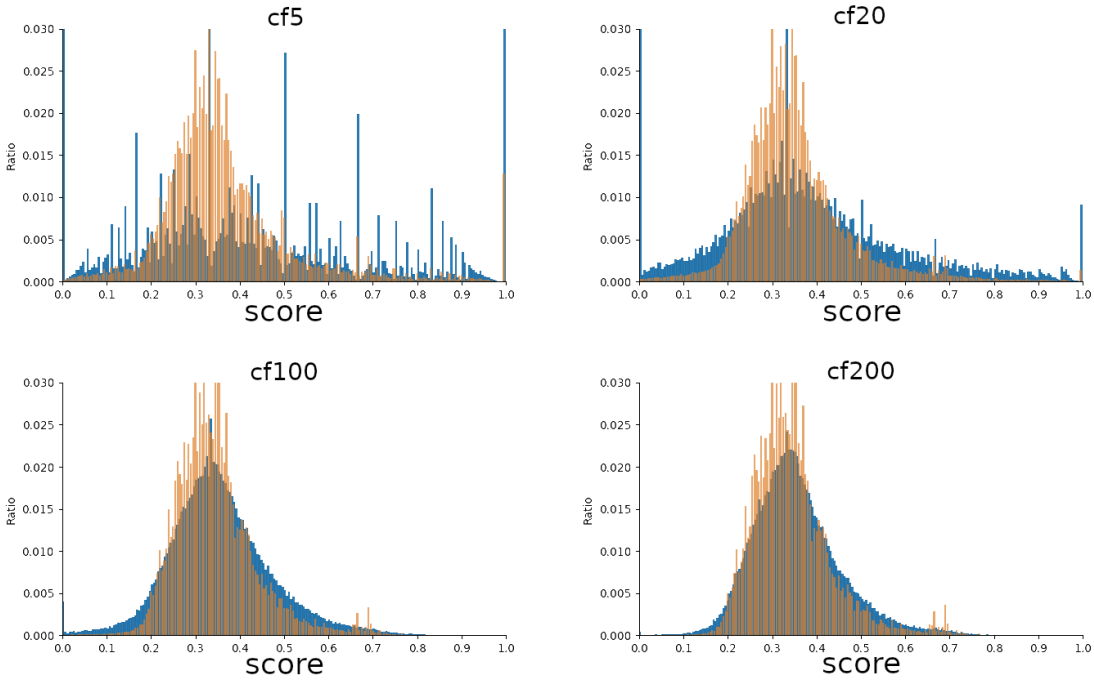

Figure 3.7. Normalized distribution of unique peptides (blue) and corresponding normalized total observation count distribution across CoAM (binding score) values (orange). Note that both unique peptides and observation counts are centered around the mean.

Amino acid frequencies on weak and strong binder peptides show small but consideration-worthy patterns on some amino acid types. All aromatic amino acid (W, Y, F) frequencies increase in the strong binder set, whereas all negatively charged amino acid (D, E) frequencies decrease. This may explain the abundance of peptides in the last panning round, pointing to strong binding characteristics of aromatic amino acids to MoS_2 surface. On the other hand, as illustrated in Figure 1.1., the 2D surface is made up of Sulphur atoms which tend to be negatively charged due to its higher electronegativity compared to molybdenum, explaining the negative effect of negatively charged amino acids on binding affinity.[68,69]

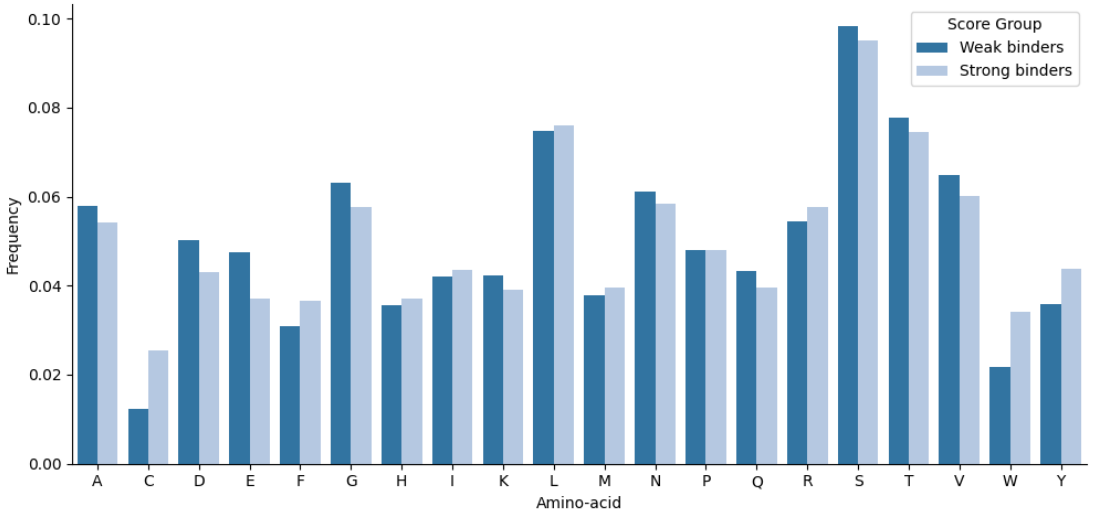

Figure 3.8. Amino acid frequencies among low affinity (dark blue, affinity < 0.3) and high affinity (light blue, affinity > 0.7) peptides.

Positively charged amino acids arginine and lysine do not show consistent differences in frequencies, where arginine (R) slightly increases, and lysine (K) slightly decreases in the strong binder group. Overall, the frequencies of amino acids align with a previous study on peptide-MoS_2 interactions where Juan Liu et al explain that amino acid binding to sulfur atoms on surfaces is influenced by aromatic residues like tryptophan and phenylalanine through favorable coordination, positively charged arginine via complementary charge and geometry, flexible weakly polar residues through hydrogen bonding with water, while negatively charged residues and bulky hydrophobic residues like proline, isoleucine, and leucine diminish binding due to electrostatic repulsion and reduced surface contact, with binding selectivity arising from the unique properties of both peptides and surfaces, beyond traditional hydrophobicity criteria.[70]

Dimensionality reduction of peptides in sequence space aids in observing if the sequences cluster in correlation to the function score. Figure 3.9 demonstrates the effect of increasing count filter values on emergence of strong and weak binder peptide clusters. As the count filter value rises, there is a gradual improvement in cluster border definitions. The dataset with the lowest count filter value fails to generate distinct clusters of strong binders. This observation aligns with the assumption that a higher observation count correlates with increased data point confidence.

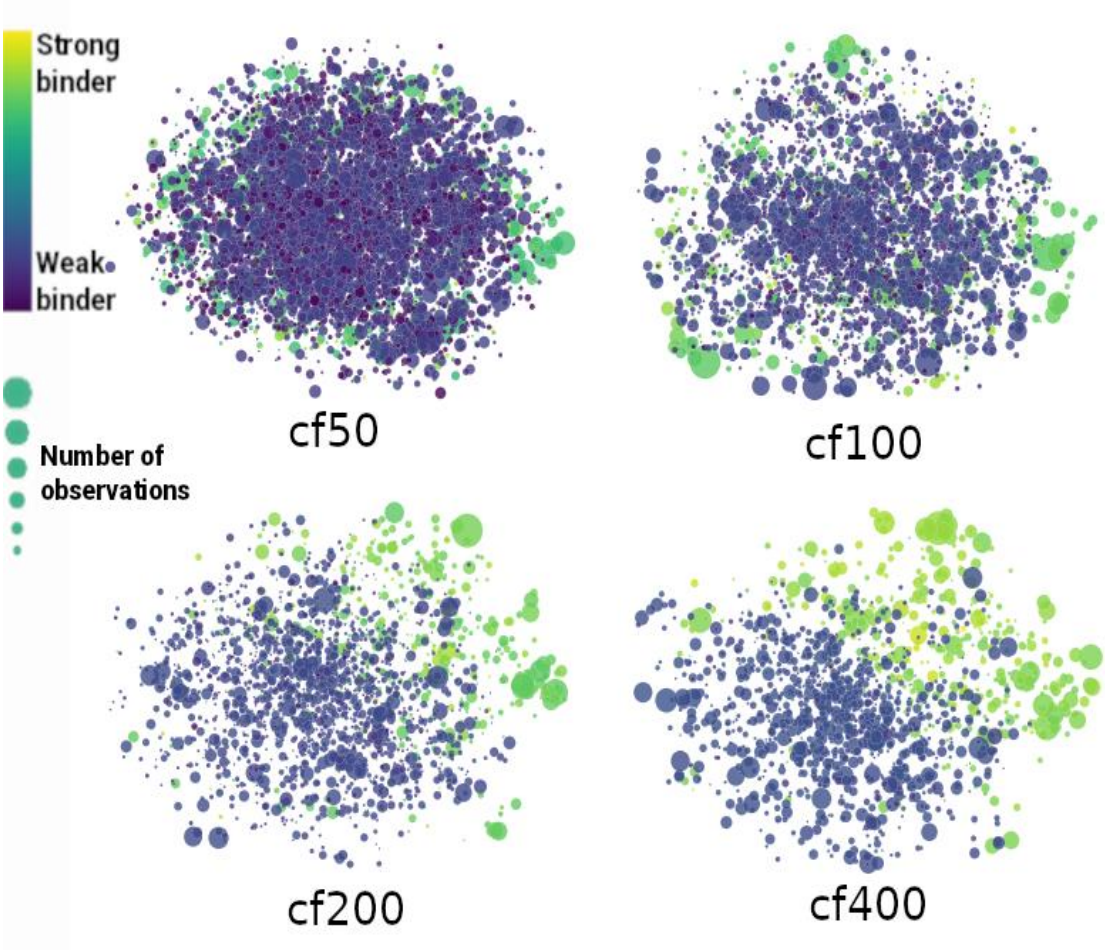

Figure 3.9. PacMap visualization of the dataset, using count filters 100, 150, 200, and 400. Note the green and blue clusters emerging through increasing count filter values.

Sequence space coverage of a subset of the library is visualized against random peptides in Figure 3.10. The figure does not display any obvious missing regions that are not covered by the diluted subset, nor the sequences from the actual library is a subset of a larger random set. This is favorable since any uncovered areas would implicate potentially larger prediction errors by the trained models over such regions.

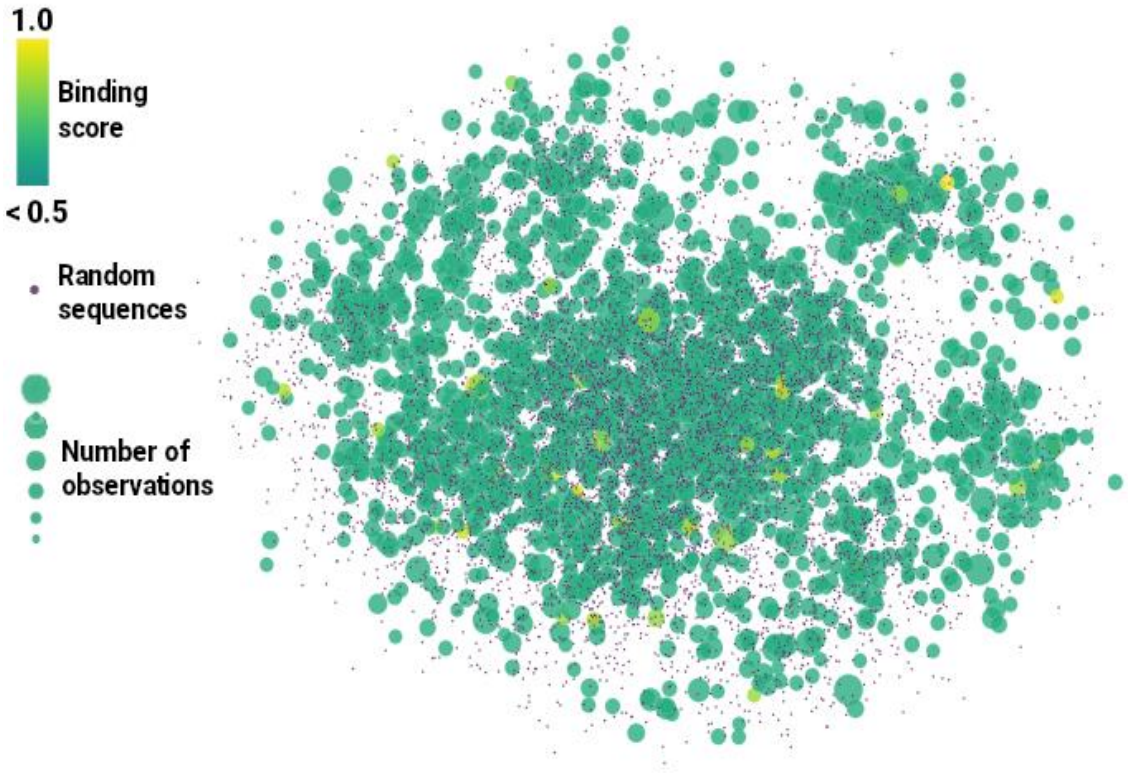

Figure 3.10. PacMap visualization of sequence space coverage of a sampled subset of cf200 set (green hues) against randomly generated peptides (tiny, dark blue dots) where lighter green spots indicate strong binders (score > 0.65).

3.4 Model Training

Baseline Regression with Random Forest: Prediction performance yields of various count filter values are computed by fitting the unweighted training sets with the Random Forest algorithm, initially with default parameters and one-hot encoding. This initial set of experiments yields a baseline that allows comparison with different solutions.

Table 3.4 demonstrates the negative correlation between the count filter parameter and the prediction performance on *cf200 test dataset*. It also communicates that validation loss is almost an order of magnitude larger than the training loss. Hence, in order to prevent overfitting, we have moved forward with optimizing the hyperparameters of the

RF algorithm by choosing the *minimum samples per leaf* parameter to reduce the gap between training & validation losses. Although *cf5 dataset* offers the minimum *cf200 test* loss, used datasets were filtered with higher count filter values during the optimization process due to limits on computational resources. Table 3.4 also emphasizes the Random Forest algorithm's robustness to noise, since the prediction performance improves with decreasing count filter values, although signal-to-noise ratio is expected to decrease with inclusion of data points with lower confidence. The *cf10* dataset is selected for hyperparameter exploration to reduce the required memory and experiment time.

Table 3.4. Performance and storage size of Random Forest models trained on datasets with different count filter values and tested on cf200 test set.

Training Set	cf5	cf10	cf20	cf50	cf100	cf200
Training Data Size (rows)	2.1M	1.3M	696K	261K	120K	73K
Training Loss	0.010	0.0084	0.0054	0.0025	0.0012	0.0007
Validation Loss	0.072	0.0600	0.0390	0.0170	0.0090	0.0048
Test cf200 Loss	0.0028	0.0029	0.0029	0.0037	0.0043	0.0049
Test cf200 Pearson C. C.	0.86	0.85	0.84	0.79	0.75	0.71
Model Storage Size (GB)	16.3	9.2	4.9	1.9	0.9	0.4

Hyperparameter optimization of Random Forest: Effects of *minimum samples per leaf* parameter are presented in Table 3.5. Training loss and validation loss values are minimal in the default configuration, since there is no leaf/depth limit, while the order of magnitudes of the training/test loss values are different, pointing to an overfit model. Then, stepping up the parameter value produces a slight increase in validation loss along with a dramatic increase in training loss that later slowly plateaus (Figure 3.11). Notably, the cf200 test set exhibits a more pronounced sensitivity to the *minimum samples per leaf* parameter. Considering the results, the optimum *minimum samples per leaf* is selected as 5, which is the smallest increment that provides the highest marginal utility in closing the gap between training and validation losses as well as significantly reducing the memory

and storage requirements. Increasing the *number of estimators* is effective when the value is small, it plateaus afterward (see Table 3.6.).

Table 3.5. Exploration of minimum sample per leaf parameter and its effects on prediction performance of models trained on cf10 dataset with number of estimators is set to 100.

Min sample per leaf	Default=1	**5**	10	20
Training Loss	0.0084	**0.0360**	0.0480	0.0560
Validation Loss	0.0600	**0.0620**	0.0630	0.0640
Test 200 loss	0.0029	**0.0034**	0.0041	0.0049
Test 200 Pearson	0.85	**0.82**	0.79	0.74

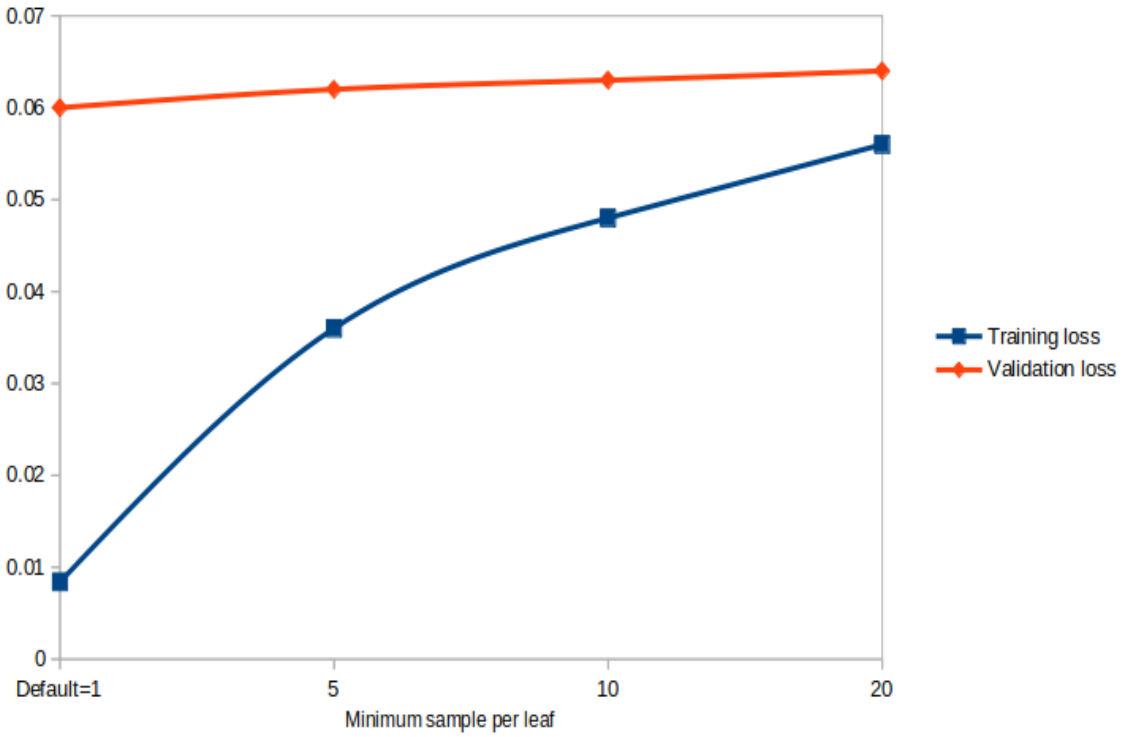

Figure 3.11. Trajectories of training and validation loss on increasing *minimum sample per leaf* value.

Table 3.6. Exploration of the number of estimators parameter and its effects on prediction performance of models trained on cf10 dataset where minimum sample per leaf is set to 10.

Number of estimators	20	50	100	200	400
Training Loss	0.0490	0.0480	0.0480	0.0480	0.0480
Validation Loss	0.0640	0.0630	0.0630	0.0630	0.0630
Test 200 loss	0.0045	0.0042	0.0041	0.0040	0.0040
Test 200 Pearson	0.762	0.783	0.789	0.794	0.795

As can be seen in Table 3.7, no significant reduction in validation loss is observed when using sample weights, however, high-confidence test set performance increased significantly when datasets with lower count filter values were used in combination with sample weighting. Positive effects of sample weighting seem to diminish with increasing dataset confidence (by adjusting the count filter value). It is then possible to speculate that sample weighting helps fit the model better to high-confidence data while keeping the low-confidence data in the training set. The performance impact on high-confidence test set MSE loss is around ~21% and ~17% respectively for cf5 and cf10 datasets (see Table 3.7).

Table 3.7. Comparison of models trained with and without sample weights, on cf5 and cf10 datasets where minimum sample per leaf is set to 10 and number of estimators is 100.

	cf5 training		cf10 training	
Sample Weights	No	Yes	No	Yes
Training Loss	0.041	0.047	0.048	0.052
Validation Loss	0.076	0.076	0.063	0.063
Test cf200 loss	0.0029	**0.0023**	0.0041	**0.0034**
Test cf200 Pearson	0.86	**0.88**	0.79	**0.82**

We have also compared one-hot and VHSE8 encoding schemes, omitting ESM encoding due to its infeasibility in training on standard hardware since it requires around 23 times more memory and storage than the one-hot scheme. Among the available encodings, VHSE8 performed significantly better, while decreasing model fitting time dramatically along with memory and storage demand, due to the smaller size of the input vector. Performance results of the models fitted with two different encodings are shown in Table 3.8, with VHSE8 offering around 3.5% and 20+% decrease in validation and high-confidence (*cf200*) test MSE losses, respectively, compared to one-hot encoding.

Table 3.8. Comparison of one-hot and VHSE8 encoding schemes and training with and without sample weights, on cf10 dataset.

	One-hot		VHSE8	
	Unweighted	Weighted	Unweighted	Weighted
Training Loss	0.048	0.052	0.032	0.037
Validation Loss	0.063	0.063	0.061	0.061
Test cf200 loss	0.0041	0.0034	0.0032	**0.0027**
Test cf200 Pearson	0.79	0.82	0.842	**0.86**

After exploration of the performance landscape, a more thorough hyperparameter search for models trained with *cf5 dataset* is conducted using 5-fold cross-validation. It displays how the *number of estimators* and *minimum samples per leaf* hyperparameters affect the validation MSE loss. While the error minimizes with an increasing number of estimators, the improvement of prediction performance also decreases. Considering the *minimum samples per leaf* value, the positive effects

Consequently, *minimum samples per leaf* parameter is selected as 5 and *number of estimators* parameter is selected as 200 by inspecting the results of the cross-validation runs (see Table 3.9.).

Hyperparameter optimization of Neural Networks: Initial experiments laid out the importance of application of the Dropout technique. Experiments with low or no-dropout models performed worse than the ones with high dropout. This view aligns with

the suggestion that dropout produces remarkable performance improvement when training on noisy targets.[71] Following the experiments, the dropout value was selected as high as 0.5 for the best demonstrated consistent validation performance (see Table 3.10).

Table 3.9. Hyperparameter search displaying mean-square-error (MSE) loss results of 5-fold cross-validation with weighted cf5 training dataset and VHSE8 encodings.

	Number Of Estimators				
Min sample per leaf	20	50	100	**200**	300
5	0.0812	0.0795	0.0789	0.0786	0.0785
10	0.0816	0.0805	0.0801	0.0798	0.0798
20	0.0822	0.0815	0.0813	0.0811	0.0811
50	0.0829	0.0826	0.0825	0.0824	0.0824
100	0.0834	0.0832	0.0831	0.0831	0.0831

Table 3.10. Effect of dropout value on model prediction performance.

Encoding	**Model**	**Dropout**	**Val MSE**	**Val MAE**
ESM480	FFNN 1x480	0.5	0.0824	0.227
ESM480	FFNN 1x480	0.0	0.0830	0.228

The number of layers, thus the number of parameters are of importance for obtaining better models. As it grows, a considerable amount of improvement is seen in prediction performance, which is demonstrated in Table 3.11. It is important to emphasize that minor differences in unweighted prediction performance, namely on metrics *MSE* and *MAE*, correspond to dramatic effects on tests with high-confidence data, as also demonstrated in Table 3.8.

Table 3.11. Comparison of training results with various encoding schemes and network configurations.

Encoding	Model	#Regression Params	Val MSE	Val MAE
One-hot	FFNN 1x480	116k	0.0828	0.229
One-hot	FFNN 8x480	1.73M	0.0827	0.228
VHSE8	FFNN 1x480	47k	0.0838	0.229
VHSE8	FFNN 8x480	1.66M	0.0828	0.227
ESM480	FFNN 1x480	2.77M	0.0824	0.227
ESM480	FFNN 8x480	22.1M	0.082	0.226
ESM480	PepLM+FFNN 1x480	2.77M	0.081	0.224
ESM480	PepLM+FFNN 8x480	22.1M	0.0802	0.221
ESM480	PepLM+FFNN 8x2048	41.2M	0.0757	0.21

3.5. Validation

Subsequently, the final models are trained with all the data, using the selected parameter values. It is then tested against the *cf5 test set* and the *cf200 test set.* The best Random Forest model achieved an MSE of 0.00205, and an MAE of 0.0325 on *cf200* test set, the minimum values observed throughout this work, while the Pearson correlation coefficient is as high as 0.895. The best neural network model performed slightly worse than the best Random Forest model (see Table 3.12). The averaging ensemble model performed slightly better than the two, with a Pearson correlation score of 0.904, an MSE of 0.00184, and an MAE of 0.0304.

The scatter plots of experimental vs predicted binding score (CoAM) values for the *cf200 test sets* are shown in Figure 3.12 where the diagonal is the perfect prediction line. It is clear from the figures that the models successfully predict the binding scores of the peptides with a higher total number of observations, however, it may result in larger errors for peptides with a lower number of total observations. On the other hand, Figure B.4 shows the experimental vs predicted binding score scatter plot for cf5 test set on the best Random Forest model along with cf200 test set, which demonstrates a big portion of

the low-confidence scores, that are coming from peptides with lower observation count, are not predicted correctly, while higher-confidence scores are predicted accurately.

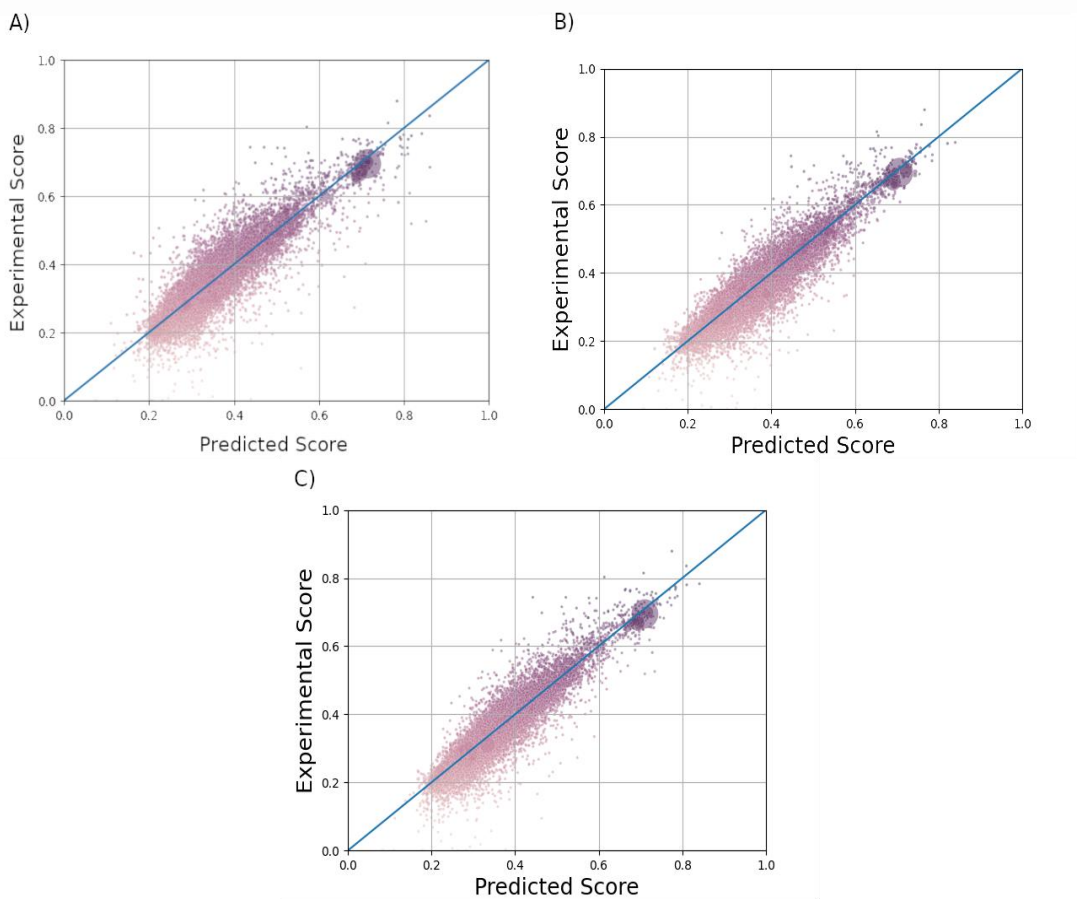

Figure 3.12. A, B, and C are experimental vs predictive binding score scatter plots of the best PepLM+FFNN, Random Forest and ensemble models, respectively, tested on the high-confidence (cf200) test set. The diameters of the points indicate the total count of the peptide, the hue projects the binding score, and the diagonal line marks the perfect prediction line.

An example feature importance set of a Random Forest model is visualized in Figure 3.13. It highlights the significance of tryptophan (W) on positions 1, 3, 5, 7, 9, serine (S) particularly on 1 and 2, positively charged arginine (R) and Sulphur containing cysteine (C) on 11 and 12, along with leucine, histidine, threonine, and tyrosine. The

results are not contradictory with a recent study on peptide-MoS_2 molecular dynamics simulations.[70]

Table 3.12. Prediction performance of the final Random Forest and PepLM-based neural network model.

	RF	PepLM+FFNN	RF + NN Ensemble
Training Loss	0.0295	0.0413	**0.0338**
Test cf5 MSE Loss	0.0729	0.0746	**0.0723**
Test cf5 MAE	0.210	0.210	**0.208**
Test cf200 MSE Loss	0. 00205	0.00242	**0.00184**
Test cf200 MAE	0.0325	0.0346	**0.0304**
Test cf200 Pearson	0.895	0.870	**0.904**

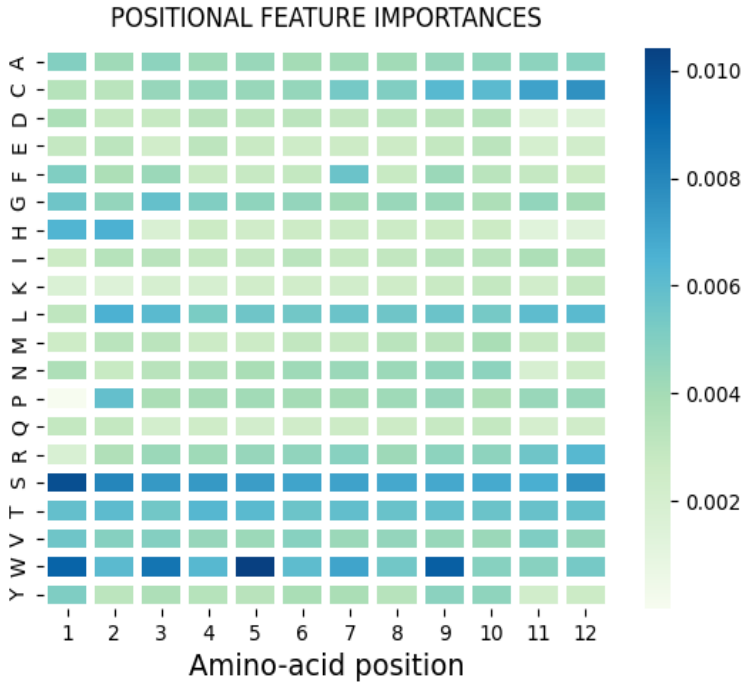

Figure 3.13. Feature importance matrix derived from mean decrease in impurity calculated over Random Forest trees. The model is trained on the cf20 dataset.

To further evaluate the models, binding scores of four experimentally evaluated peptides collected from the literature are predicted by the best random forest and neural network models as demonstrated in Table 3.13. The table demonstrates the differences between the two models' predictions, and consistency of expectations regarding the experimental analyses and predictions.

Table 3.13. Four peptides collected from the literature, experimental observation along with predictions by Random Forest and neural network models.[64,65,66]

Peptide	Sequence	Experiment Result	RF Prediction	NN Prediction	Ensemble Prediction
GrBP5-M6	IMVTASSAYDDY	Reference	0.41	0.32	0.36
MOS2-P15	GVIHRNDQWTAP	Strong	0.43	0.41	0.42
MOS2-P28	DRWVARDPASIF	Strong	0.44	0.39	0.41
MOS2-P3	SVMNTSTKDAIE	Weak	0.36	0.35	0.36

3.6. De Novo Design of Functional Peptides

The strongest predicted binders from randomly generated peptides are dominated by aromatic amino acids tryptophan (W) and phenylalanine (F) (see Figure 3.14). A random search for peptides without cysteine and aromatic residues produced different frequency magnitudes as expected, however, they still displayed similar patterns that emphasize arginine (R), methionine (M), leucine (L), isoleucine (I) and valine (V) as in Figure 3.15.

Building on the hypothesis that the binding scores of abundant peptides are closer to the hypothetically true binding scores (mean absolute error less than 4% for cf200 test set), peptides predicted to have similar binding scores over the two different models are selected as the first four candidate peptides and listed in Table 3.14. The data highlights that the scan on Random Forest model produces peptides that are evaluated similarly by the neural network model. Surprisingly, a scan on the NN model produces peptides that are estimated to have higher scores from the RF predictions.

Table 3.14. Peptides with the most consistent scores as predicted by the Random Forest and neural network models.

Model	Generator	Peptide	RF Score	NN Score	Difference
RF	Random	WNCWWYWFFYFD	0.729	0.729	0.000
RF	Random non-aromatic	MHILRTVASLAI	0.550	0.549	0.001
NN	Random	FWLWKCFIYFPD	0.653	0.894	0.241
NN	Random non-aromatic	MMLLLHMTTIDA	0.533	0.826	0.293

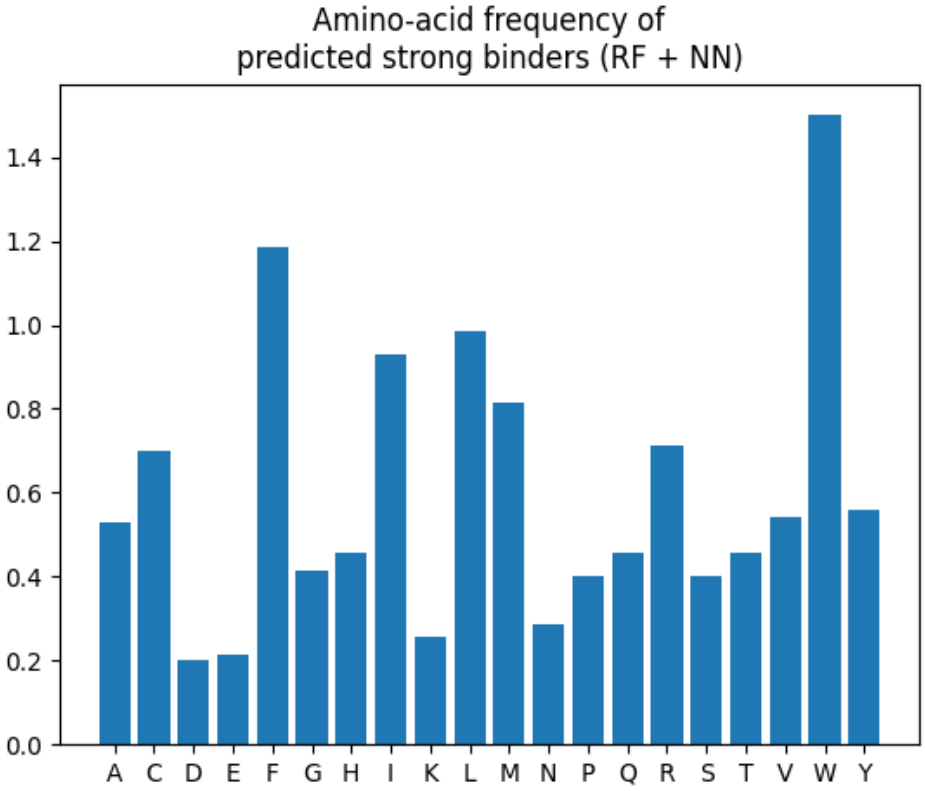

Figure 3.14. Amino acid frequencies of random peptides that are predicted to have strong binding characteristics by the ensemble model.

Scores of peptides obtained by scanning the Random Forest model are consistent with the scores predicted by the neural network model with a mean difference of 0.09 and 0.11, for completely random peptides and random peptides without “WYFC” residues.

However, this was not the case when the random search is applied to the neural network model, where the mean difference between predictors increases to 0.34 and 0.41 as the NN model's predictions tend to be further from the mean for a portion of sequences and predicted strongest binders of the RF model is a subset of that of the NN model (see full tables in Appendix C). Yet, strong binders predicted by the neural network are aligned with the Random Forest predictor, in terms of scores, whether a peptide is below or above the average binding score, 0.35.

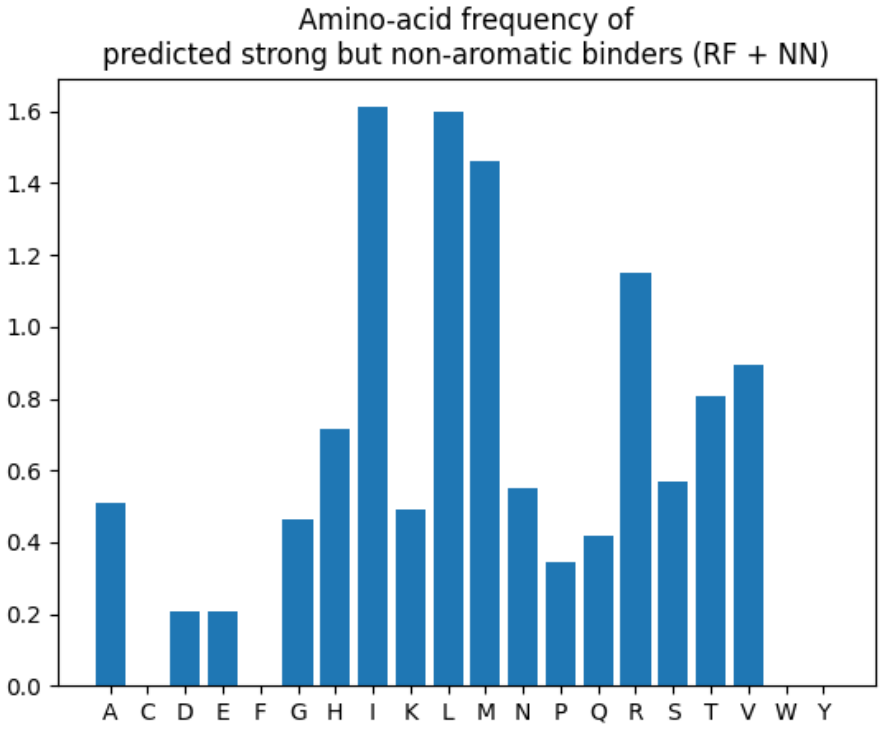

Figure 3.15. Amino acid frequencies of peptides, without cysteine and aromatic amino acids, are predicted to have strong binding characteristics by the ensemble model.

Along with consistently high scoring peptides, the highest scoring peptides for four categories (RF, NN, random, random without "WFYC") are selected from a random exploration run. These peptides, and their respective scores are displayed in Table 3.15.

Table 3.15. Highest scoring peptides as predicted by the Random Forest and neural network models, both with and without aromatic amino acids.

Model	Generator	Peptide	RF Score	NN Score	Difference
RF	Random	HAWEWLKMHHIL	0.742	0.353	0.389
RF	Random non-aromatic	LLLIEDTNPNLE	0.583	0.776	0.193
NN	Random	RNHYAYIHFCWL	0.547	0.925	0.378
NN	Random non-aromatic	VIMVMMNVKQMS	0.422	0.901	0.479

The highest scoring peptide of a random search run with around 5 million random peptides on the ensemble model, LRMLTRHLNVNN, without aromatic and sulphur-containing residues is selected for final optimization (see Appendix C, Tables C.5). A substitution matrix is built by evaluating all possible point mutations (Appendix C, Figure C.1). The matrix suggests mutating the 10^{th} residue valine (V) to leucine (L). Note that for this sequence, aromatic acids do not increase the predicted binding score, pointing out to the complex nature of amino acid sequences. After making the recommended substitution, a second matrix is built which proposes asparagine (N) instead of 8^{th} residue leucine (L).

The final matrix displays that the optimization is complete, and the binding score of the peptide is not expected to increase further, yielding the non-aromatic strong-binder candidate, **LRMLTRHNNLNN** with scores 0.55, 0.93, and 0.74 as predicted by the Random Forest, the deep neural network and the averaging ensemble model, respectively (see Appendix C, Figures C.1-3 for the substitution-prediction matrices). Figures 3.17 and 3.18 show the visualization of the candidate peptide's 3D structure as predicted respectively by Alphafold[50] and OmegaFold[51], rendered by UCSF Chimera.[72]

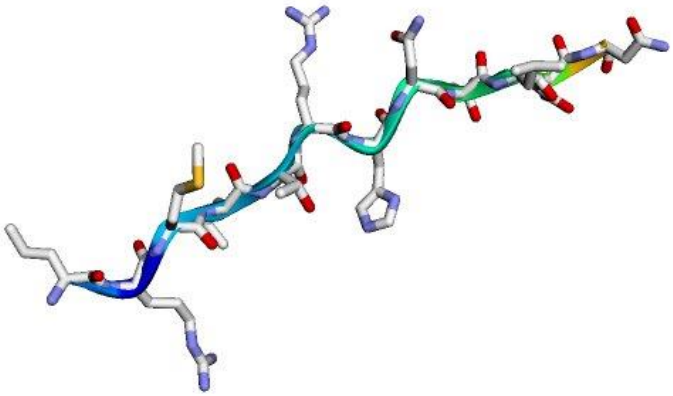

Figure 3.16. The optimized peptide, LRMLTRHNNLNN**,** after two substitutions to the initial peptide. The 3D structure is predicted by Alphafold[50], rendered by UCSF Chimera.[72]

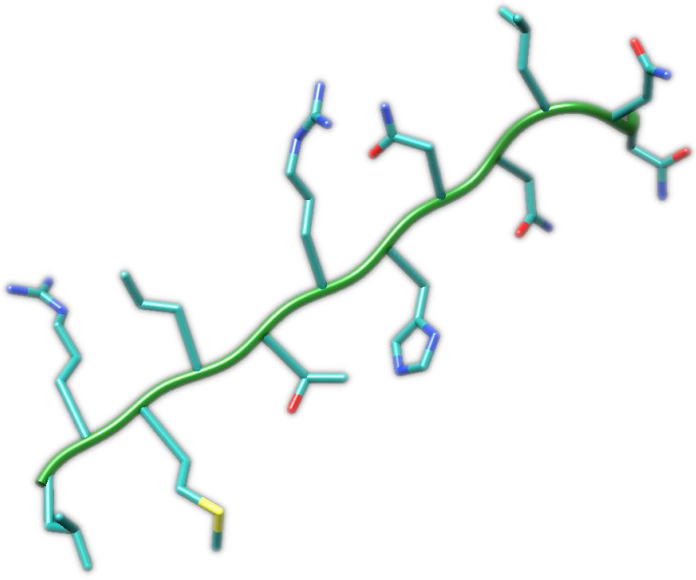

Figure 3.17. The optimized peptide, LRMLTRHNNLNN. The 3D structure is predicted by Omegafold[51], rendered by UCSF Chimera.[72]

CHAPTER 4

CONCLUSIONS

In this thesis, a preprocessing, fitting, and testing methodology was developed in the context of deep-directed evolution, to be able to predict binding scores with a deep learning model that employs a dodecapeptide language model within, a Random Forest model and an averaging ensemble of the two. The experiments emphasized the importance of amino acid embeddings and sample weighting, and thus, the significance of the inclusion of domain knowledge into the process.

A Pearson correlation score of 0.895 with a mean absolute error of 0.0325 is achieved on high-confidence test dataset with the Random Forest algorithm, successfully predicting the outcomes of frequently observed peptides of the phage-display experiment data with trees of about 6GB size in storage. On the other hand, the best deep learning network in this work is a two-stage neural network with an autoregressive peptide language model (PepLM) and an 8-layer 2048 hidden units regression head. It yielded 0.870 as the Pearson correlation score and 0.0346 as mean absolute error on the high-confidence test set. The information is compressed into 52M parameters (occupying around 200MB of memory including both PepLM and the regression head), which is significantly smaller when compared to the Random Forest models. When the two models are utilized in an ensemble setup, such that their predictions are averaged, the performance further increases, as the Pearson correlation score becomes 0.904, while the mean absolute error improves to 0.0304.

The prediction performances, overall, are satisfactory, considering the mean absolute error is smaller than 5% of the CoAM range [0, 1] on *cf200 test set*, and thus, the work demonstrates feasible machine learning approaches for predicting function scores through deep-directed evolution. To increase the performance even more, alternative sample weighting and encoding schemes can be utilized. It is also possible to improve existing models by further hyperparameter optimization, and with more

sophisticated ensemble approaches. As such, this work provides a proof of principle that can be optimized in future work.

The analysis emphasized that aromatic amino acids bind tightly to MoS_2, and especially tryptophan (W) dominates the higher binding score region. This is observed both in the dataset and the models' predictions. It is observed that the inclusion of tryptophan, or other aromatic amino-acids, along with sulfur-containing cysteine in a sequence increases the probability of a peptide remaining in the eluate in the performed phage-display experiment. This might be pointing out their particular binding mode and/or aromatic/hydrophobic mode of binding that is resistant to detergents.

Sequence space is explored with random peptides using the two trained models, and their averaging ensemble. To further refine the search, random peptides with non-aromatic and non-sulphur-mediated binding modes -peptides without cysteine, tryptophan, tyrosine, and phenylalanine (W, F, Y, C)- were targeted, as this type of selection of binding modes may be strategically significant while looking for optimal candidates. The exploration yielded thousands of candidate peptides that are suggested to perform well in terms of MoS_2 binding affinity, both with aromatic and non-aromatic binding modes. The highest-scoring peptide yielded by the exploration was optimized by evaluating iterative single amino acid mutations and resulted in a candidate peptide, LRMLTRHNNLNN. Additional experimental work is required to confirm the behavior of high-scoring peptides with different residue patterns, and question where and how the models' prediction differences arise.

Overall, deep-directed evolution has proved to be revolutionary as a peptide, and, potentially, protein design methodology, where millions of amino acid sequences are produced, enabling advanced machine learning techniques for crafting exceptional peptides in terms of a specific function. This work demonstrates a feasible and applicable set of procedures for designing de novo peptides with desired function scores, amino acid compositions and other properties by employing deep directed evolution.

REFERENCES

1. Scheffel, A.; Gruska, M.; Faivre, D.; Linaroudis, A.; Plitzko, J. M.; Schüler, D. An Acidic Protein Aligns Magnetosomes along a Filamentous Structure in Magnetotactic Bacteria. *Nature* **2005**, *440* (7080), 110–114. https://doi.org/10.1038/nature04382

2. Arias, J. L.; Fernández, M. S. Biomimetic Processes through the Study of Mineralized Shells. *Materials Characterization* **2003**, *50* (2-3), 189–195. https://doi.org/10.1016/s1044-5803(03)00088-3

3. Sharma, V.; Srinivasan, A.; Roychoudhury, A.; Rani, K.; Tyagi, M.; Dev, K.; Nikolajeff, F.; Kumar, S. Characterization of Protein Extracts from Different Types of Human Teeth and Insight in Biomineralization. *Scientific Reports* **2019**, *9* (1), 9314. https://doi.org/10.1038/s41598-019-44268-2

4. Yucesoy, D. T.; Karaca, B. T.; Cetinel, S.; Caliskan, H. B.; Adali, E.; Gul-Karaguler, N.; Tamerler, C. Direct Bioelectrocatalysis at the Interfaces by Genetically Engineered Dehydrogenase. *Bioinspired Biomim. Nanobiomaterials* **2015**, *4* (1), 79–89. https://doi.org/10.1680/bbn.14.00022

5. Yucesoy, D. T.; Akkineni, S.; Tamerler, C.; Hinds, B. J.; Sarikaya, M. Solid-Binding Peptide-Guided Spatially Directed Immobilization of Kinetically Matched Enzyme Cascades in Membrane Nanoreactors. *ACS Omega* **2021**, *6* (41), 27129–27139. https://doi.org/10.1021/acsomega.1c03774

6. Yucesoy, D. T.; Khatayevich, D.; Tamerler, C.; Sarikaya, M. Rationally Designed Chimeric Solid-binding Peptides for Tailoring Solid Interfaces. *Med. Devices Sens.* **2020**, *3* (3), e10065. https://doi.org/10.1002/mds3.10065

7. Hu, X.; Liao, M.; Gong, H.; Zhang, L.; Cox, H.; Waigh, T. A.; Lu, J. R. Recent Advances in Short Peptide Self-Assembly: From Rational Design to Novel Applications. *Current Opinion in Colloid & Interface Science* **2020**, 45, 1–13. https://doi.org/10.1016/j.cocis.2019.08.003

8. Wang, Y.; Xue, P.; Cao, M.; Yu, T.; Lane, S. T.; Zhao, H. Directed Evolution: Methodologies and Applications. *Chemical Reviews* **2021**, *121* (20), 12384-12444. https://doi.org/10.1021/acs.chemrev.1c00260

9. Witten, J.; Witten, Z. Deep Learning Regression Model for Antimicrobial Peptide Design. *bioRxiv* **2019**, 1-18. https://doi.org/10.1101/692681

10. Lin, E.; Lin, C.-H.; Lane, H.-Y. De Novo Peptide and Protein Design Using Generative Adversarial Networks: An Update. *J. Chem. Inf. Model.* **2022**, *62* (4), 761–774. https://doi.org/10.1021/acs.jcim.1c01361

11. Lin, Z.; Akin, H.; Rao, R.; Hie, B.; Zhu, Z.; Lu, W.; Smetanin, N.; Verkuil, R.; Kabeli, O.; Shmueli, Y.; dos Santos Costa, A.; Fazel-Zarandi, M.; Sercu, T.; Candido, S.; Rives, A. Evolutionary-Scale Prediction of Atomic-Level Protein Structure with a Language Model. *Science* **2023**, *379* (6637), 1123–1130. https://doi.org/10.1126/science.ade2574

12. Rives, A.; Meier, J.; Sercu, T.; Goyal, S.; Lin, Z.; Liu, J.; Guo, D.; Ott, M.; Zitnick, C. L.; Ma, J.; Fergus, R. Biological Structure and Function Emerge from Scaling Unsupervised Learning to 250 Million Protein Sequences. *Proceedings of the National Academy of Sciences* **2021**, *118* (15), e2016239118, 1-12. https://doi.org/10.1073/pnas.2016239118

13. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. *Generative Adversarial Networks*. *Advances in neural information processing systems*, **2014**, 27. https://doi.org/10.48550/arXiv.1406.2661

14. Yucesoy, D. T.; Rath, S. S.; Rodriguez, J. L.; Francis-Landau, J.; Nakano-Baker, O.; Sarikaya, M. Deep Directed Evolution of Solid Binding Peptides for Quantitative Big-Data Generation. *BioRxiv* **2021**, 1-26. https://doi.org/10.1101/2021.01.26.428348

15. Wang, Q. H.; Kalantar-Zadeh, K.; Kis, A.; Coleman, J. N.; Strano, M. S. Electronics and Optoelectronics of Two-Dimensional Transition Metal Dichalcogenides. *Nature Nanotechnology* **2012**, *7* (11), 699–712. https://doi.org/10.1038/nnano.2012.193

16. Chhowalla, M.; Shin, H. S.; Eda, G.; Li, L.-J.; Loh, K. P.; Zhang, H. The Chemistry of Two-Dimensional Layered Transition Metal Dichalcogenide Nanosheets. *Nature Chemistry* **2013**, *5* (4), 263–275. https://doi.org/10.1038/nchem.1589

17. Mak, K. F.; Lee, C.; Hone, J.; Shan, J.; Heinz, T. F. Atomically Thin MoS_2: A New Direct-Gap Semiconductor. *Physical Review Letters* **2010**, *105* (13), 136805. https://doi.org/10.1103/physrevlett.105.136805

18. Radisavljevic, B.; Radenovic, A.; Brivio, J.; Giacometti, V.; Kis, A. Single-Layer MoS2 Transistors. *Nature Nanotechnology* **2011**, *6* (3), 147–150. https://doi.org/10.1038/nnano.2010.279

19. Lopez-Sanchez, O.; Lembke, D.; Kayci, M.; Radenovic, A.; Kis, A. Ultrasensitive Photodetectors Based on Monolayer MoS2. *Nature Nanotechnology* **2013**, *8* (7), 497–501. https://doi.org/10.1038/nnano.2013.100

20. Akinwande, D.; Petrone, N.; Hone, J. Two-Dimensional Flexible Nanoelectronics. *Nature Communications* **2014**, *5* (1), 5678. https://doi.org/10.1038/ncomms6678

21. Sarkar, D.; Liu, W.; Xie, X.; Anselmo, A. C.; Mitragotri, S.; Banerjee, K. MoS2 Field-Effect Transistor for Next-Generation Label-Free Biosensors. *ACS Nano* **2014**, *8* (4), 3992–4003. https://doi.org/10.1021/nn5009148

22. Geim, A. K.; Grigorieva, I. V. Van Der Waals Heterostructures. *Nature* **2013**, *499* (7459), 419–425. https://doi.org/10.1038/nature12385

23. Hardy, J. G.; Amend, M. N.; Geissler, S.; Lynch, V. M.; Schmidt, C. E. Peptide-Directed Assembly of Functional Supramolecular Polymers for Biomedical Applications: Electroactive Molecular Tongue-Twisters (Oligoalanine–Oligoaniline–Oligoalanine) for Electrochemically Enhanced Drug Delivery. *Journal of materials chemistry. B* **2015**, *3* (25), 5005–5009. https://doi.org/10.1039/c5tb00106d

24. Ding, S.; Cargill, A. A.; Medintz, I. L.; Claussen, J. C. Increasing the Activity of Immobilized Enzymes with Nanoparticle Conjugation. *Current Opinion in Biotechnology* **2015**, *34*, 242–250. https://doi.org/10.1016/j.copbio.2015.04.005

25. Yin, W.; Yu, J.; Lv, F.; Yan, L.; Zheng, L. R.; Gu, Z.; Zhao, Y. Functionalized Nano-MoS2 with Peroxidase Catalytic and Near-Infrared Photothermal Activities for Safe and Synergetic Wound Antibacterial Applications. *ACS Nano* **2016**, *10* (12), 11000–11011. https://doi.org/10.1021/acsnano.6b05810

26. Marketa Hnilova; Deniz Tanil Yucesoy; Mehmet Sarikaya; Candan Tamerler. Controlling Biological Functionalization of Surfaces by Engineered Peptides. *Ceramic transactions /Ceramic transactions* **2013**, 137–150. https://doi.org/10.1002/9781118751015.ch15

27. Arnold, F. H. Design by Directed Evolution. *Accounts of Chemical Research* **1998**, *31* (3), 125–131. https://doi.org/10.1021/ar960017f

28. Stemmer, W. P. C. Rapid Evolution of a Protein in Vitro by DNA Shuffling. *Nature* **1994**, *370* (6488), 389–391. https://doi.org/10.1038/370389a0

29. Arnold, F. H.; Volkov, A. A. Directed Evolution of Biocatalysts. *Current Opinion in Chemical Biology* **1999**, *3* (1), 54–59. https://doi.org/10.1016/S1367-5931(99)80010-6

30. Romero, P. A.; Arnold, F. H. Exploring Protein Fitness Landscapes by Directed Evolution. *Nature Reviews Molecular Cell Biology* **2009**, *10* (12), 866–876. https://doi.org/10.1038/nrm2805

31. Bloom, J. D.; Arnold, F. H. In the Light of Directed Evolution: Pathways of Adaptive Protein Evolution. *Proceedings of the National Academy of Sciences* **2009**, *106* (Supplement_1), 9995–10000. https://doi.org/10.1073/pnas.0901522106

32. Avoigt, C.; Kauffman, S.; Wang, Z.-G. Rational Evolutionary Design: The Theory of in Vitro Protein Evolution. *Advances in protein chemistry* **2001**, 55, 79-160. https://doi.org/10.1016/S0065-3233(01)55003-2

33. Yang, K. K.; Wu, Z.; Arnold, F. H. Machine-Learning-Guided Directed Evolution for Protein Engineering. *Nature Methods* **2019**, *16* (8), 687–694. https://doi.org/10.1038/s41592-019-0496-6

34. Smith, G. Filamentous Fusion Phage: Novel Expression Vectors That Display Cloned Antigens on the Virion Surface. *Science* **1985**, *228* (4705), 1315–1317. https://doi.org/10.1126/science.4001944

35. Sidhu, S. S.; Lowman, H. B.; Cunningham, B. C.; Wells, J. A. [21] Phage Display for Selection of Novel Binding Peptides. *Methods in enzymology* **2000**, *328*, 333-IN5. https://doi.org/10.1016/S0076-6879(00)28406-1

36. Biswas, S.; Khimulya, G.; Alley, E. C.; Esvelt, K. M.; Church, G. M. Low-N Protein Engineering with Data-Efficient Deep Learning. *Nature Methods* **2021**, *18* (4), 389–396. https://doi.org/10.1038/s41592-021-01100-y

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

38. Liaw, A.; Wiener, M. Classification and Regression by RandomForest. R news, **2002**, *2 (3), 18-22.* https://journal.r-project.org/articles/RN-2002-022/RN-2002-022.pdf (accessed Apr 28, 2024).

39. Biau, G.; Scornet, E. A Random Forest Guided Tour. *TEST* **2016**, *25* (2), 197–227. https://doi.org/10.1007/s11749-016-0481-7

40. Fernández-Delgado, M.; Cernadas, E.; Barro, S.; Amorim, D.; Fernández-Delgado, A. Do We Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research* **2014**, *15*, 3133–3181. http://hdl.handle.net/10347/17792 (accessed on Jun 12, 2024)

41. Louppe, G.; Wehenkel, L.; Sutera, A.; Geurts, P. Understanding Variable Importances in Forests of Randomized Trees. *Advances in Neural Information Processing Systems* **2024**, *26*. https://hdl.handle.net/2268/155642 (accessed on Jun 25, 2024).

42. Segal, M. R. *Machine Learning Benchmarks and Random Forest Regression*. *Escholarship.org* **2004**, 1-14. https://escholarship.org/uc/item/35x3v9t4 (accessed on May 18, 2024).

43. Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; Pedreschi, D. A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys* **2018**, *51* (5), 1–42. https://doi.org/10.1145/3236009

44. Strobl, C.; Boulesteix, A.-L.; Zeileis, A.; Hothorn, T. Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution. *BMC Bioinformatics* **2007**, *8, 1-21*. https://doi.org/10.1186/1471-2105-8-25

45. Belgiu, M.; Drăguţ, L. Random Forest in Remote Sensing: A Review of Applications and Future Directions. *ISPRS Journal of Photogrammetry and Remote Sensing* **2016**, *114*, 24-31. https://doi.org/10.1016/j.isprsjprs.2016.01.011

46. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *Advances in Neural Information Processing Systems* **2017**, *30*, 5998–6008. https://doi.org/10.48550/arXiv.1706.03762

47. Aggarwal, A.; Mittal, M.; Battineni, G. Generative Adversarial Network: An Overview of Theory and Applications. *International Journal of Information Management Data Insights* **2021**, *1* (1), 100004. https://doi.org/10.1016/j.jjimei.2020.100004

48. Malach, E. *Auto-Regressive Next-Token Predictors are Universal Learners*. *arXiv.org* **2023**, 1-15. https://doi.org/10.48550/arXiv.2309.06979

49. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv.org* **2018**, 1-16. https://doi.org/10.48550/arXiv.1810.04805

50. Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; Bridgland, A.; Meyer, C.; Kohl, S. A. A.; Ballard, A. J.; Cowie, A.; Romera-Paredes, B.; Nikolov, S.; Jain, R.; Adler, J.; Back, T. Highly Accurate Protein Structure Prediction with Alphafold. *Nature* **2021**, *596* (7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2

51. Wu, R.; Ding, F.; Wang, R.; Shen, R.; Zhang, X.; Luo, S.; Su, C.; Wu, Z.; Xie, Q.; Berger, B.; Ma, J.; Peng, J. High-Resolution de Novo Structure Prediction from Primary Sequence. *BioRxiv* **2022**, 1-37. https://doi.org/10.1101/2022.07.21.500999

52. Bengio Y.; Ducharme R.; Vincent P.; Janvin C. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* **2003**, 3, 1137-1155. https://doi.org/10.5555/944919.944966

53. Mikolov, T.; Ilya Sutskever; Chen, K.; Corrado, G. S.; Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. *arXiv (Cornell University)* **2013**. https://doi.org/10.48550/arxiv.1310.4546

54. Mikolov, T.; Karafiát, M.; Burget, L.; Černocký, J.; Khudanpur, S. Recurrent Neural Network Based Language Model. *Interspeech* **2010**, 2 (3), 1045-1048. https://doi.org/10.21437/interspeech.2010-343

55. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Computation* **1997**, *9* (8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

56. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. *Preprint* **2018**, 1-12. https://static.aminer.cn/upload/pdf/1319/1601/76/5f8eab579e795e9e76f6f6a0_0.pdf (accessed on May 29, 2024).

57. Müller, A. T.; Hiss, J. A.; Schneider, G. Recurrent Neural Network Model for Constructive Peptide Design. *Journal of Chemical Information and Modeling* **2018**, *58* (2), 472–479. https://doi.org/10.1021/acs.jcim.7b00414

58. Leinonen, R.; Sugawara, H.; Shumway, M. The Sequence Read Archive. *Nucleic Acids Research* **2010**, *39* (Database), D19–D21. https://doi.org/10.1093/nar/gkq1019

59. Mei, H.; Liao, Z. H.; Zhou, Y.; Li, S. Z. A New Set of Amino Acid Descriptors and Its Application in Peptide QSARs. *Biopolymers* **2005**, *80* (6), 775–786. https://doi.org/10.1002/bip.20296

60. Wang, Y.; Huang, H.; Rudin, C.; Shaposhnik, Y. Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering T-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *Journal of Machine Learning Research* **2021**, *22* (201), 1–73. https://doi.org/10.48550/arXiv.2012.04456

61. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Duchesnay, É. Scikit-learn: Machine learning in Python. *Journal of machine Learning research* ***2011***, *12*, 2825-2830. https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf (accessed on Apr 5, 2024)

62. Chollet, F. Keras. **2015**. https://keras.io (accessed on Apr 5, 2024).

63. Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. R. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. *ArXiv.org* **2012**. https://doi.org/10.48550/arXiv.1207.0580

64. Hayamizu, Y.; So, C. R.; Dag, S.; Page, T. S.; Starkebaum, D.; Sarikaya, M. Bioelectronic Interfaces by Spontaneously Organized Peptides on 2D Atomic Single Layer Materials. *Scientific Reports* **2016**, *6* (1), 33778. https://doi.org/10.1038/srep33778

65. Sun, L.; Li, P.; Seki, T.; Tsuchiya, S.; Kazuki Yatsu; Narimatsu, T.; Mehmet Sarikaya; Yuhei Hayamizu. Chiral Recognition of Self-Assembled Peptides on MoS_2 via Lattice Matching. *Langmuir* **2021**, *37* (29), 8696–8704. https://doi.org/10.1021/acs.langmuir.1c00792

66. Cetinel, S.; Shen, W.-Z.; Aminpour, M.; Bhomkar, P.; Wang, F.; Borujeny, E. R.; Sharma, K.; Nayebi, N.; Montemagno, C. Biomining of MoS2 with Peptide-Based Smart Biomaterials. *Scientific Reports* **2018**, *8* (1), 3374. https://doi.org/10.1038/s41598-018-21692-4

67. Zabinsky, Z. B. Random search algorithms. *Department of Industrial and Systems Engineering, University of Washington, USA* **2009**, 1-16. https://people.bordeaux.inria.fr/pierre.delmoral/random-search-SO.pdf (accessed on Jun 11, 2024).

68. Mann, J. B.; Meek, T. L.; Knight, E. T.; Capitani, J. F.; Allen, L. C. Configuration Energies of the D-Block Elements. *J. Am. Chem. Soc.* **2000**, *122* (21), 5132–5137. https://doi.org/10.1021/ja9928677

69. Allen, L. C. Electronegativity Is the Average One-Electron Energy of the Valence-Shell Electrons in Ground-State Free Atoms. *J. Am. Chem. Soc.* **1989**, *111* (25), 9003–9014. https://doi.org/10.1021/ja00207a003

70. Liu, J.; Zeng, J.; Zhu, C.; Miao, J.; Huang, Y.; Heinz, H. Interpretable Molecular Models for Molybdenum Disulfide and Insight into Selective Peptide Recognition. *Chem. Sci.* **2020**, *11* (33), 8708–8722. https://doi.org/10.1039/d0sc01443e

71. Jindal, I.; Nokleby, M.; Chen, X. Learning Deep Networks from Noisy Labels with Dropout Regularization. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*; IEEE, **2016**; pp 967–972. https://doi.org/10.1109/ICDM.2016.0121

72. Pettersen, E. F.; Goddard, T. D.; Huang, C. C.; Couch, G. S.; Greenblatt, D. M.; Meng, E. C.; Ferrin, T. E. UCSF Chimera—A Visualization System for Exploratory Research and Analysis. *J. Comput. Chem.* **2004**, *25* (13), 1605–1612. https://doi.org/10.1002/jcc.20084

73. Basith, S.; Manavalan, B.; Hwan Shin, T.; Lee, G. Machine Intelligence in Peptide Therapeutics: A Next-generation Tool for Rapid Disease Screening. *Med. Res. Rev.* **2020**, *40* (4), 1276–1314. https://doi.org/10.1002/med.21658

74. Wei, L.; Zhou, C.; Su, R.; Zou, Q. PEPred-Suite: Improved and Robust Prediction of Therapeutic Peptides Using Adaptive Feature Representation Learning. *Bioinformatics* **2019**, *35* (21), 4272–4280. https://doi.org/10.1093/bioinformatics/btz246

75. Watson, M., Qian, C., Bischof, J., & Chollet, F. KerasNLP. **2022**. https://github.com/keras-team/keras-nlp (accessed on Jul 8, 2024).

APPENDIX A

In Figure A.1, feature importance maps of models fit to various datasets, with or without sample weights are shown in the form of heatmap. In comparison to the top two rows (models are fit by cf100 and cf50 respectively), in the bottom row (cf20), the importance of more amino acids emerges. In the right column, where outputs of models with sample weights are displayed, sample weighting output seems to balance out the dominance of amino acids other than tryptophan, cysteine, and histidine.

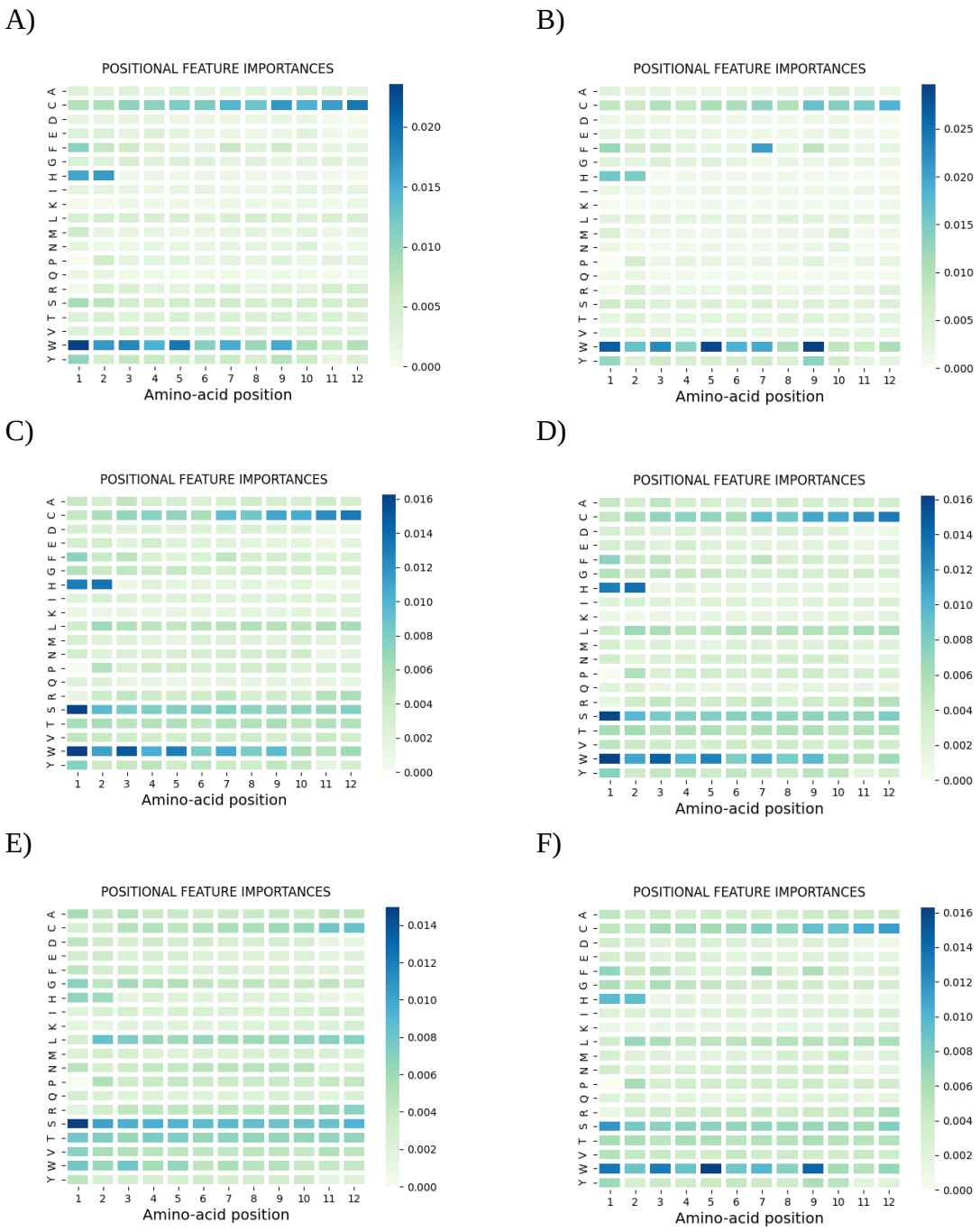

Figure A.1. Feature importance matrices derived from various Random Forest models fit to datasets with different count filters, with and without applying sampling weights. Applied datasets are cf100, cf50, and cf20 for the first row, second row and third row, respectively. While the heatmaps in the first column (A, C, E) are the result of fitting without sample weights, the heatmaps in the second column (B, D, F) are generated from models fit with sample weights.

APPENDIX B

Training and test data of simple feed-forward neural network models that are trained on *cf5* training set, and their respective experimental-vs-prediction scatter plots are shown in Figures B.1 and B.2. One-layer networks, as in the first columns of the two figures, seem to fit the training data, however, fail to generalize the validation data. The networks may not have enough capacity to capture the complex patterns and may require more neurons.

Adding more layers with dropout seems to prevent overfitting but still falls short of significantly improving the outcome.

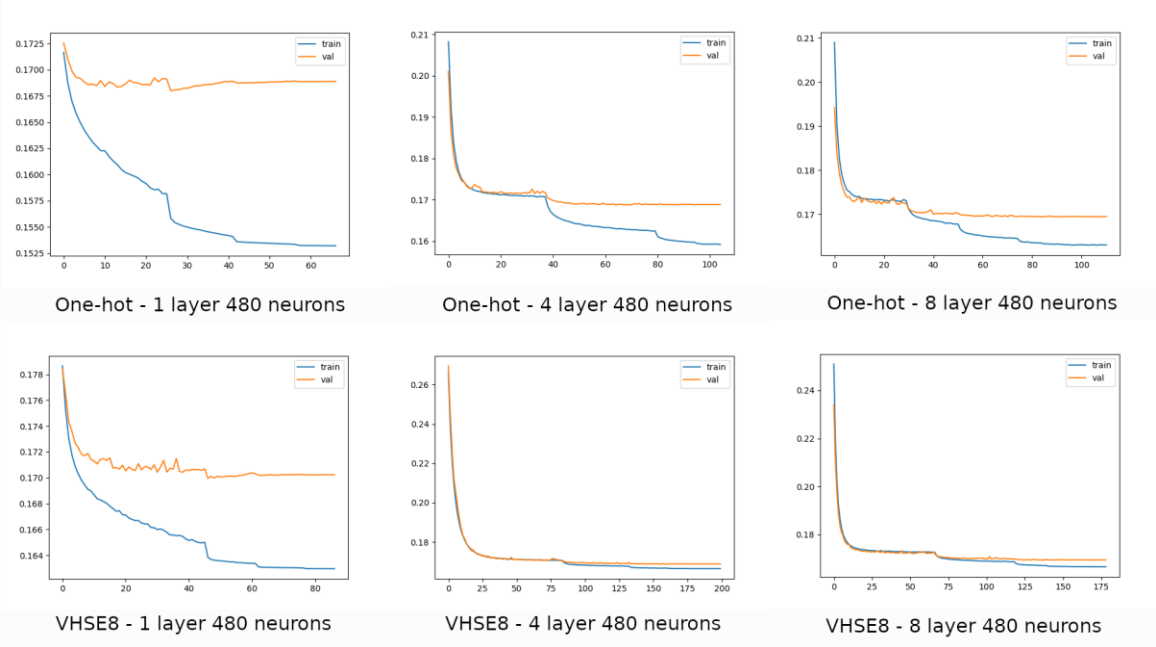

Figure B.1. Training and validation loss plots of one-hot and VHSE8 encoding schemes, and various network sizes. One-layer networks seem to overfit despite their small size. This is thought to stem from the fact that network capacity is not enough to capture the actual patterns but fits the training set.

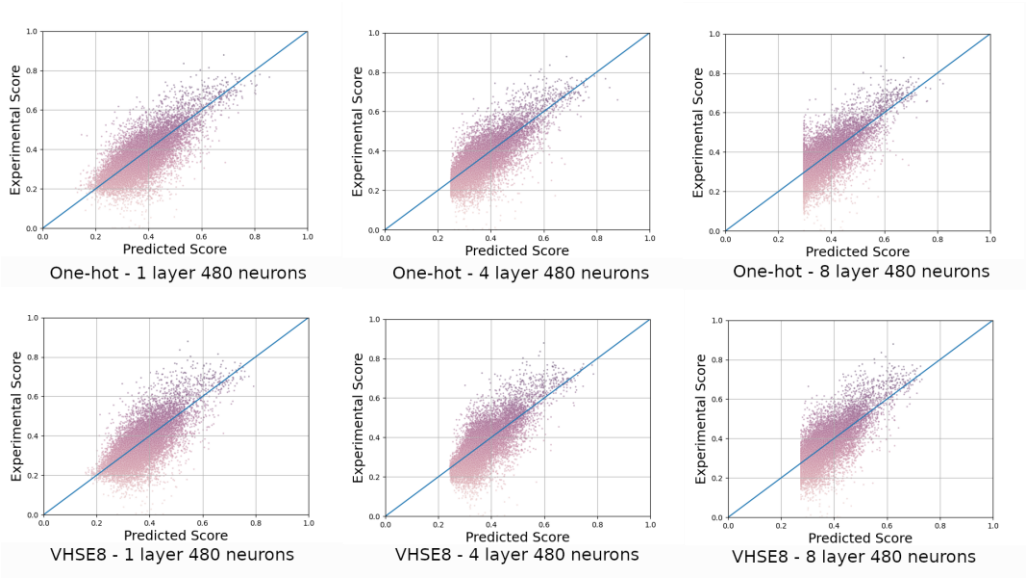

Figure B.2. Experimental vs prediction scatter plots of high-confidence test set evaluated on simple feed-forward neural networks with one-hot and VHSE8 encoding schemes, and various network sizes.

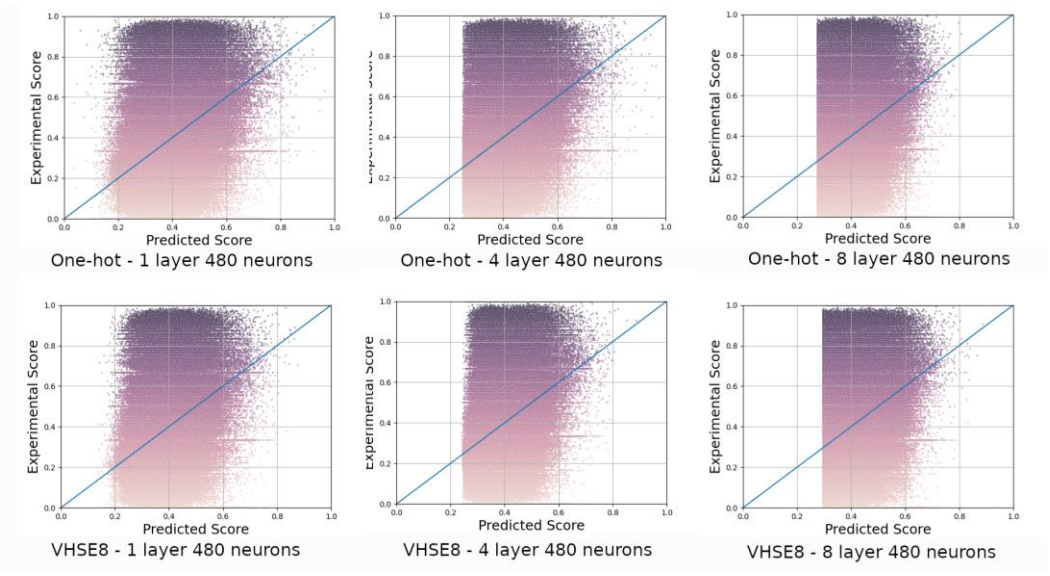

Figure B.3. *cf5* test scatter plots of experimental vs predicted values by simple feed-forward neural networks with one-hot and VHSE8 encoding schemes, and various network sizes.

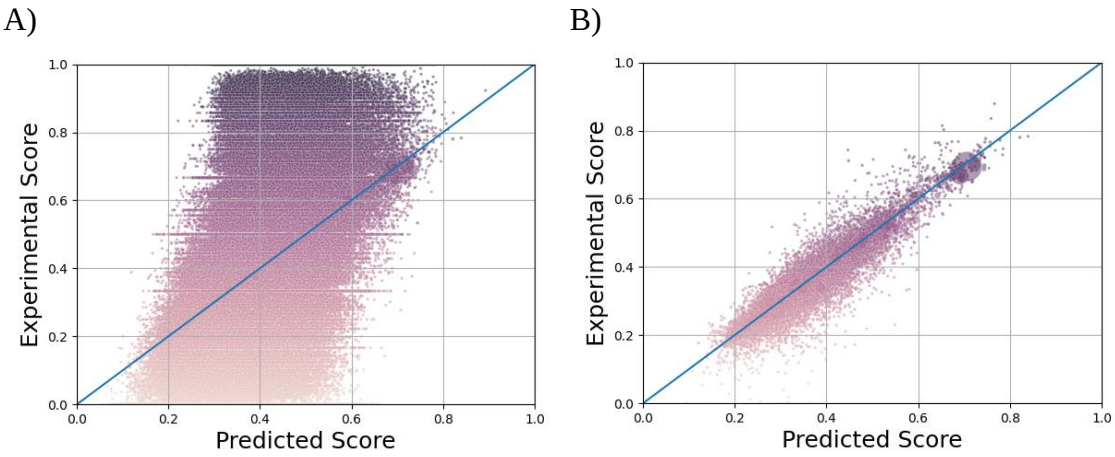

Figure B.4. Experimental vs prediction scatter plots of the optimized Random Forest model trained on *cf5* training set, tested on *cf5* test set (left) and *cf200* test set (right).

APPENDIX C

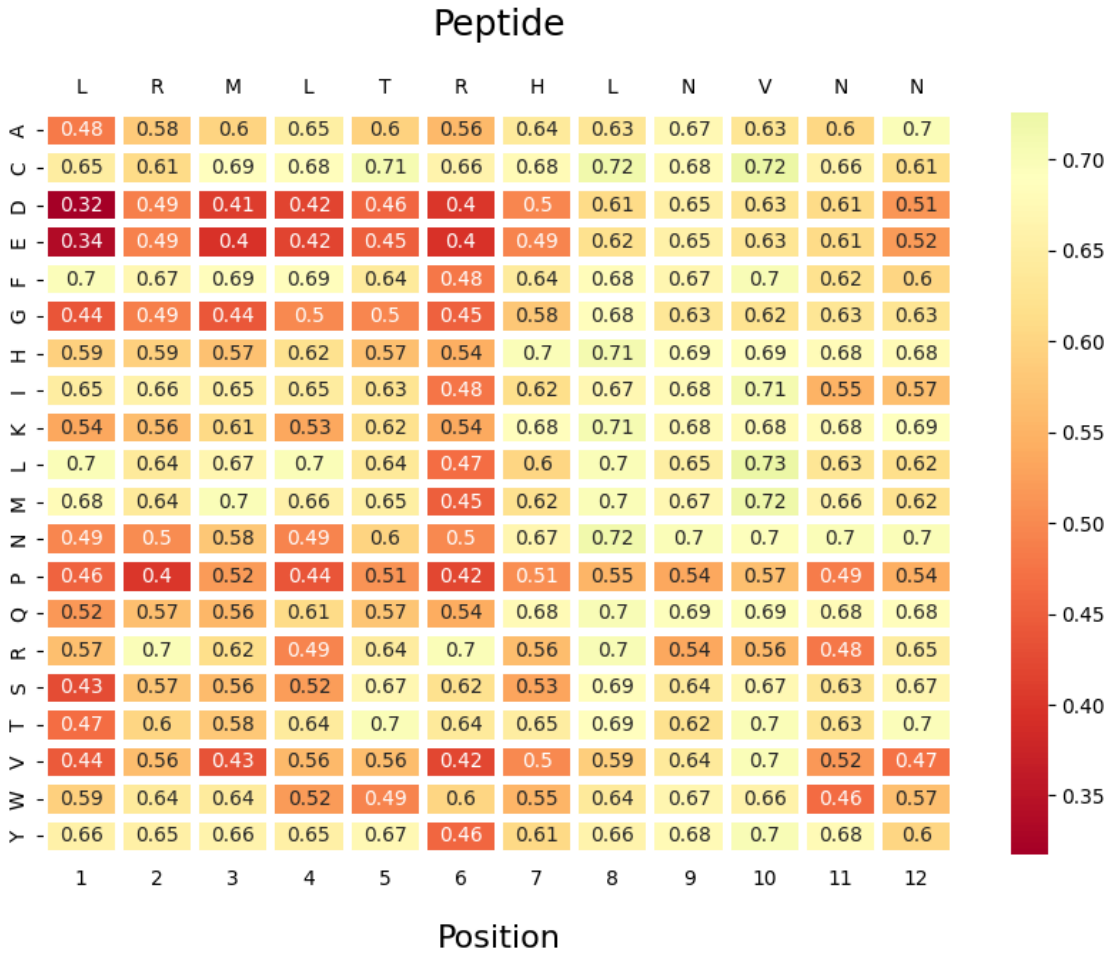

Figure C.1. Substitution matrix of LRMLTRHLNVNN, showing predicted scores by the ensemble model regarding each mutation in scope of the first optimization step after the random search. Note that aromatic residues are not expected to contribute to the peptides' binding affinity to MoS_2 for this particular sequence.

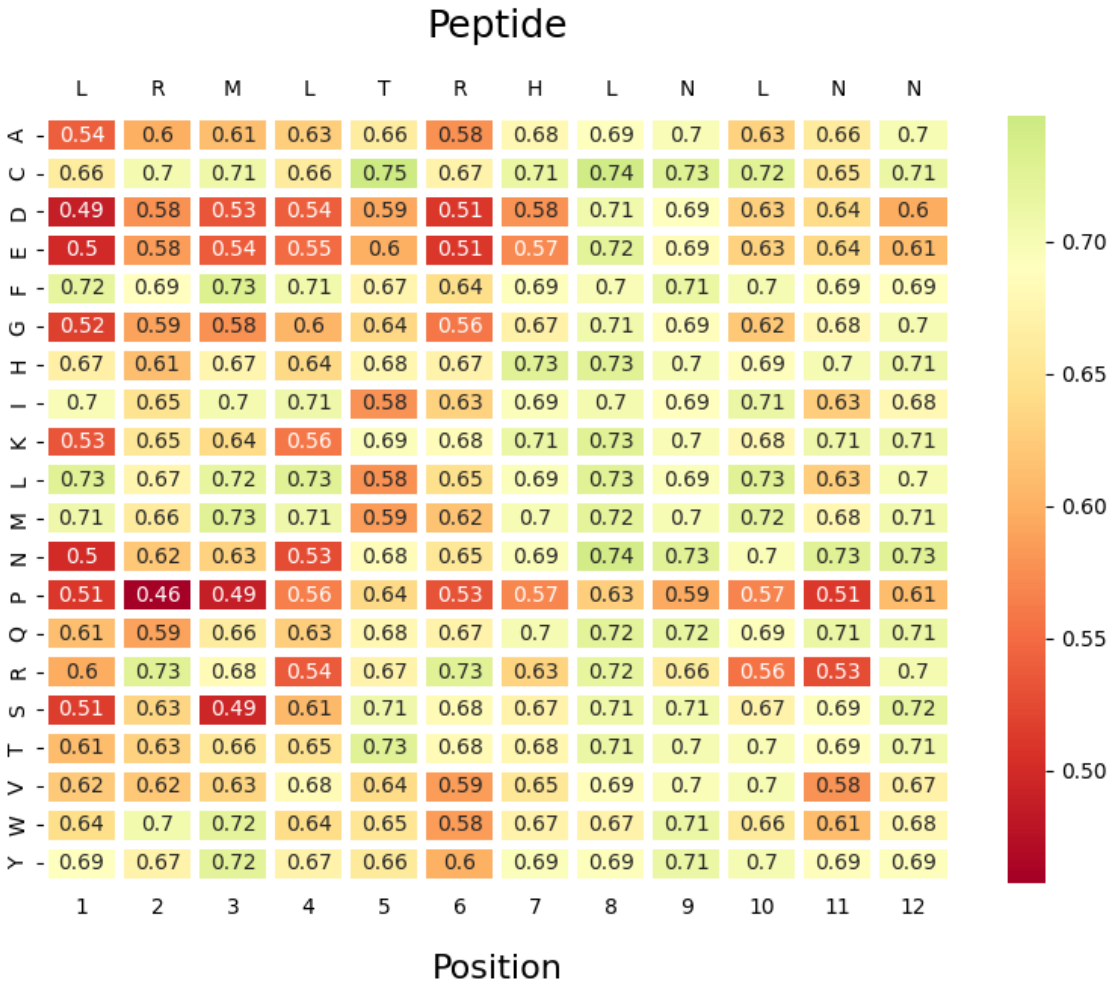

Figure C.2. Substitution matrix of LRMLTRHLNLNN, showing predicted scores by the ensemble model regarding each mutation.

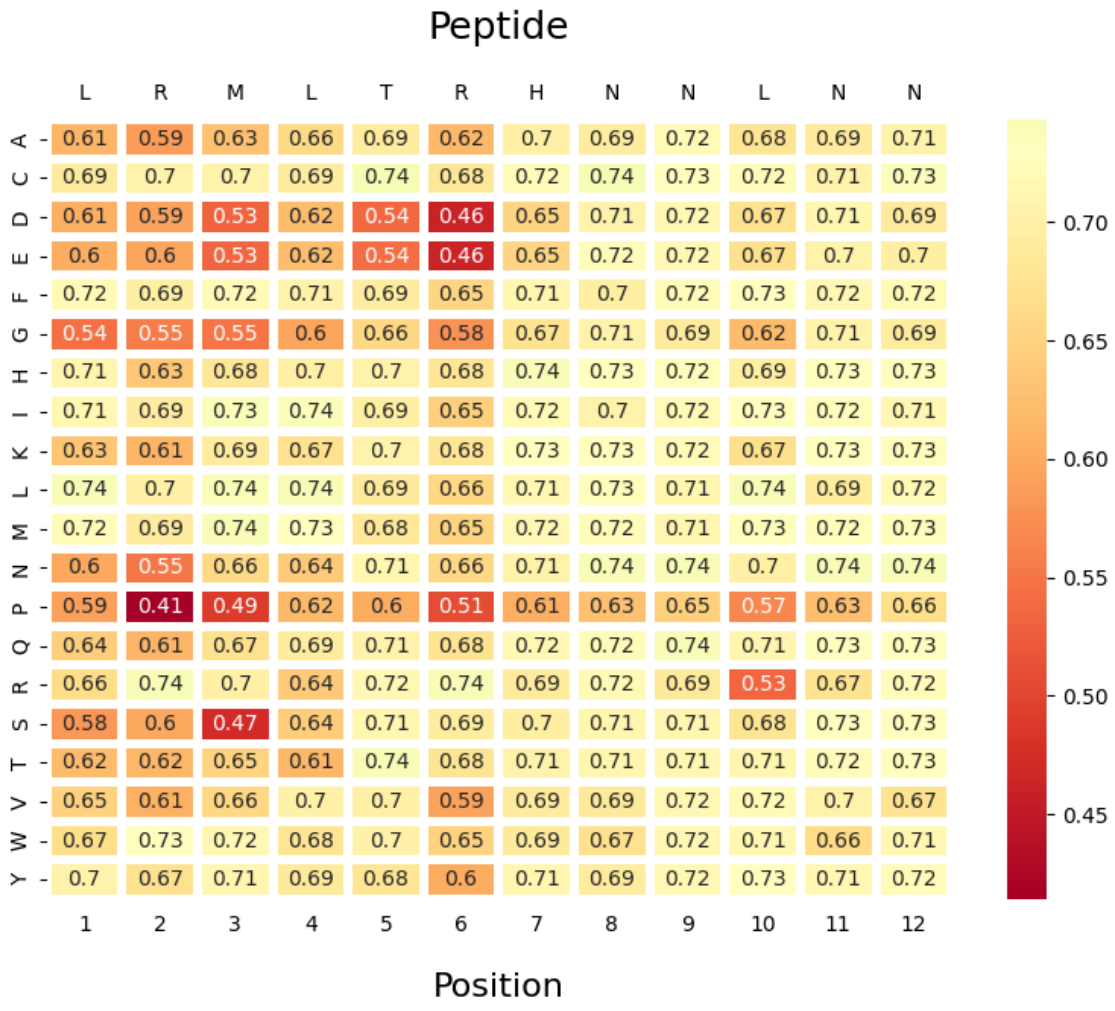

Figure C.3. Substitution matrix of LRMLTRHNNLNN, showing predicted scores by the ensemble model regarding each mutation after the second optimization step, projecting that the peptide is in the optimal form.