B-Cell Epitope Prediction for Improved Antibody Docking

by

Aristofanis Rontogiannis

BS, Computer Science and Engineering,
Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2023

© Massachusetts Institute of Technology 2023. All rights reserved.

Author . . . .
Department of Electrical Engineering and Computer Science
January 30, 2023

Certified by . . . .
Regina Barzilay
Professor, Department of Electrical Engineering and Computer Science;
Faculty Co-Lead, MIT Abdul Latif Jameel Clinic for Machine Learning in Health (J-Clinic)
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

B-Cell Epitope Prediction for Improved Antibody Docking

by

Aristofanis Rontogiannis

Submitted to the Department of Electrical Engineering and Computer Science
on January 30, 2023, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Predicting how antibodies bind to their targets is a fundamental problem of immunology, and a critical step in accelerating the development of vaccines and therapeutics against foreign pathogens. In particular, the task of predicting the 3D structure of an antibody-target complex, otherwise known as *docking*, is an important tool in drug design, providing valuable insights such as ways to increase antibody potency or methods to limit the likelihood of a mutational escape. State of the art models of antibody docking treat the task as a regression problem, outputting a single prediction. We hypothesized that while the performance after a single try might be poor, the likelihood of producing a good docking pose in K tries could be significantly higher. To achieve this without having to alter existing docking models, we propose to first train a B-Cell epitope predictor and to subsequently use it to produce a diverse set of candidate binding sites. Our epitope predictor achieves state of the art performance, with an ROC-AUC score of 76. We then show that, by properly post-processing the epitope model's predictions to select K promising candidate docking sites, the success rate of a docking model on an independent test set can be increased by a factor of almost 10, with as little as 10 tries. Our approach is compatible with any docking model and offers an alternative to pure generative modeling, while being able to guarantee a diverse set of solutions, without the need to leverage complex sampling strategies.

Thesis Supervisor: Regina Barzilay
Title: Professor, Department of Electrical Engineering and Computer Science; Faculty Co-Lead, MIT Abdul Latif Jameel Clinic for Machine Learning in Health (J-Clinic)

Acknowledgments

I would like to express my gratitude to my thesis supervisor, Regina Barzilay, for her invaluable support and advice throughout my studies. I would also like to thank PhD student Jeremy Wohlwend for his insight and mentorship during my research; without his help and guidance this thesis would have been exponentially more difficult to complete. I thank Wengong Jin for allowing me to use his model to test my hypothesis about improving antibody docking. Moreover, I am grateful to my labmates for the thought-provoking conversations we had.

On a more personal level, I would like to thank my parents, Lia and Panos, as well as my brother, Dimitris, for their continuous emotional support and unconditional love. I would also like to thank my partner, Aliai, for being by my side.

Finally, I would like to thank the Onassis Foundation for offering me a Graduate Scholarship that really helped me during my Masters' degree at MIT.

Contents

1 Introduction **9**

1.1 B-Cell Epitopes and Epitope Prediction 10

1.2 Antibody Docking 11

1.3 The Structure of the Thesis 12

2 Related Work **13**

2.1 Prior Work in Epitope Prediction 13

2.2 Prior Work in Antibody Docking 14

3 The B-Cell Epitope Model **17**

3.1 Problem Formulation 17

3.2 Model 18

3.2.1 Architecture 18

3.2.2 Surface Features 22

3.2.3 Putting it All Together 25

3.3 Data 26

3.3.1 Protein Data Bases 27

3.3.2 Data Processing 27

3.4 Training and Evaluation 31

3.5 Results 33

4 Antibody Docking **35**

4.1 The Problem of Docking 36

4.2 Model 36
4.2.1 The Docking Model 36
4.2.2 Post-Processing 38
4.3 Results 40
4.3.1 Improved Structure Predictions 47

5 Conclusions **49**
5.1 Summary of Results 49
5.2 Future Work 50
5.2.1 Incorporating Linear Epitope Data 50
5.2.2 Using Our Model to Improve AlphaFold Predictions 51

Bibliography **52**

A Implementation Details **57**
A.1 IEDB Experiments 57
A.2 Our Best Epitope Model 58
A.3 Our Best Seeding Model 59

Chapter 1

Introduction

B-cells, and the antibodies they produce, are the body's first line of defense against foreign pathogens. They are the main targets in the design of vaccines, and the modality of choice for many antivirals on the market. One of the key steps in the development of a new antibody, is to experimentally resolve its 3D pose when bound to its target antigen. This step, however, is expensive. Therefore, computational approaches to antibody-antigen structure prediction have emerged as a powerful alternative. One of the limitations of current methods, however, is that they only provide a single output. Instead, we hypothesize that producing multiple possible docking poses could allow for a more comprehensive evaluation of performance and open the door to refinement-based strategies.

One possible approach would be to formulate the task as a generative model. The issue with this strategy is that the model may sample many solutions at the same candidate binding site, which might be completely different than the true binding site, therefore requiring complex sampling strategies to produce a diverse set of solutions. It also requires training completely new models, which isn't always practical. Instead we propose to use a binding site (i.e epitope) predictor to seed a docking model. This has two advantages. First, it is compatible with any existing docking model, and second, we can guarantee diversity by simply picking high-ranking residues that are distant from one another, therefore covering multiple candidate binding sites.

In this first chapter we provide background information regarding antibodies, B-cells, and antibody docking. We also briefly describe how this thesis is structured.

1.1 B-Cell Epitopes and Epitope Prediction

B-cells, also known as B-lymphocytes, are a type of white blood cells produced in the bone marrow. They are part of the adaptive immune system, providing long-term immunity to various diseases we are exposed to over the course of our lives. To better understand how B-cells contribute to our protection, we will briefly go over their structure and function.

Embedded in the surface of B-cells, we find transmembrane proteins called B-Cell Receptors (BCRs). BCRs are Y-shaped proteins that bind to foreign substances, called *antigens*, and subsequently induce an immune response, activating the B-cell. Activated B-cells turn into plasma cells, which produce specialized proteins, called *antibodies*, that can recognize the antigen which triggered the immune response.

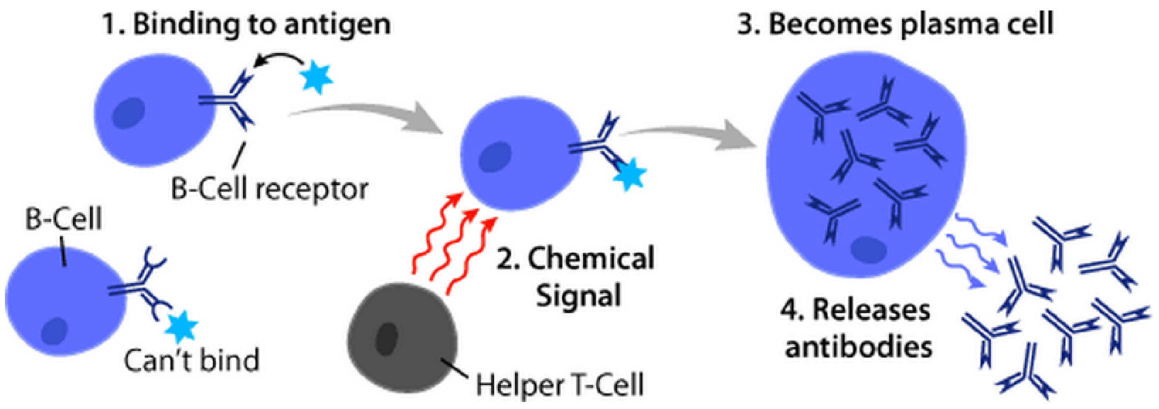


Figure 1-1: The function of a B-cell, illustrated [34]

A *B-cell epitope* (or simply *epitope*) can be defined as the part of the antigen that an antibody will bind to. In the following section, we will provide a more robust definition that will help us formulate our problem clearly.

The process of discovering the epitopes of an antigen is called *epitope mapping*. Mapping epitopes is crucial in medicine. Having knowledge on the epitopes of a virus vastly helps the vaccine developing process, and contributes to making better diagnoses. In fact, epitope-based vaccine development dates back to 1985 [22]. To locate the epitopes, in-silico mapping is commonly used to obtain a set of candidate epitopes, which are then engineered into vaccines and tested before entering production. This approach, however, is expensive and time-consuming, and does not scale well to large datasets.

A good B-cell epitope prediction model would therefore have numerous applications in therapeutics and vaccine design. In section 3, we will develop such a model from the ground up.

1.2 Antibody Docking

The problem of Antibody Docking belongs in the more broad category of *Macromolecular Docking* [8] tasks. Macromolecular Docking can be defined as the computational calculation of the three-dimensional structure that two macromolecules, such as proteins or nucleic acids, will form by interacting together. Antibody docking is a special case of the macromolecular docking problem, where the two interacting molecules are both proteins, one being an antigenic protein and the other an antibody.

There are multiple questions of interest in the problem of antibody docking. First of all, we would like to know, in a certain biological context, whether there will be binding between the antigen and antibody; it is important to answer this question before we start examining the structure of the resulting complex, as a false-positive answer to the question of binding could result in misleading results to the following questions we will be asking. Secondly, if they do indeed bind, can we accurately predict the three-dimensional structure of the crystal they form? This particular question is the one we will be addressing in this thesis. And finally, if they do not bind, is it possible to mutate or otherwise alter the antibody to induce binding? The last question is crucial in drug and vaccine design, and in the broader field of protein and antibody manufacturing.

1.3 The Structure of the Thesis

In chapter 2 we discuss related work, both in the field of epitope predicting as well as antibody docking. In chapters 3 and 4, we go over how we approached the respective tasks, as well as the experiments we ran and our thought process behind certain decisions we made. In chapter 5 we provide a summary of the results presented in this thesis, as well as plans for future work. Finally, in the Appendix, we provide some implementation specifics, as well as a more detailed look into various experiments we ran.

Chapter 2

Related Work

Machine Learning approaches are being widely used in many aspects of biology and medicine tangential to our area of focus. For instance, consider the problem of small molecule binding, a problem critical to the task of rational drug design, where one attempts to predict the binding structure of a small molecule ligand to a protein. DiffDock [6], by Corso et al, is a cutting-edge machine learning model, that uses diffusion, a technique typically used in image processing, to provide a state-of-the-art (SOTA) solution to the molecular docking problem.

Machine learning methods also have proven to be really effective in the task of protein design. H.C Hunt et al [12], in their recent publication, go over how they "designed, developed, and characterized potent, trivalent miniprotein binders that provide prophylactic and therapeutic protection against emerging SARS-CoV-2 variants of concern".

2.1 Prior Work in Epitope Prediction

The task of predicting B-cell epitopes is also a crucial one in the field of medicine and drug design. As we mentioned earlier, being able to accurately predict epitope residues has a multitude of applications, included but not limited to vaccine design, as well as the design of monoclonal antibodies, that is, engineered proteins that when injected into the organism induce immune response.

There have been numerous B-cell epitope prediction models in the current bibliography. Some of the most prominent ones are described in [4], [13], [23], [18], [30], however none of them generalize well, due to the innate noisiness of the task, as well as the lack of large, publicly available datasets that can be used for training. These models typically use Convolutional Neural Networks to encode their features, and Recurrent Neural Networks to decode them and produce output logits; however, this approach, especially when combined with poor featurization, has had little success (see section 3.5).

The current SOTA in the field of epitope prediction is a recently published deep neural network model, GraphBepi [36]. It achieves a fairly decent performance and its architecture is in many ways similar to the one we propose in this thesis. Since, however, GraphBepi was published less than a month ago, we can confidently say that the B-cell epitope model proposed in this thesis was discovered independently and in parallel. The proximity of our results and those of [36] suggests that the underlying techniques are a solid basis for approaching the B-cell epitope prediction problem.

2.2 Prior Work in Antibody Docking

Machine learning methods also have a wide array of applications in structure-predicting tasks like antibody docking, a problem crucial to generative protein/antibody design. In their recent work, Jin et al [14] propose a novel equivariant docking model that, given the structure of the antigen and the structure of the paratope, the part of the antibody which will come in contact with the antigen, can accurately predict the structure of the crystal complex the antigen and antibody form.

However, in many, if not most, practical applications, the paratope will not be known a-priori (*blind docking*). In fact, one usually only knows the sequence of the antibody, or maybe some common folding patterns it can take. In chapter 4 we go over how we use our epitope prediction model in tandem with a docking model to improve the task of blind docking.

When it comes to blind docking, the current SOTA is AlphaFold Multimer [15], by DeepMind. AlphaFold Multimer is an extension of the popular folding model AlphaFold2, which can predict the crystal structure of protein-protein complexes using only their amino-acid sequences.

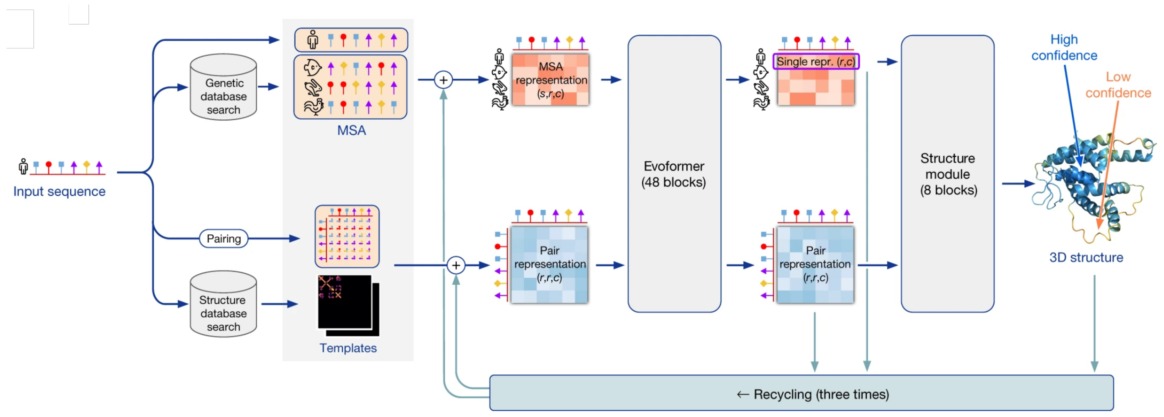

Input sequence
Genetic database search
MSA
Pairing
Structure database search
Templates
MSA representation (s,r,c)
Pair representation (r,r,c)
Evoformer (48 blocks)
Single repr. (r,c)
Pair representation (r,r,c)
Structure module (8 blocks)
High confidence
Low confidence
3D structure
← Recycling (three times)

Figure 2-1: The architecture of AlphaFold2, as visualized in DeepMind's original publication[15].

AlphaFold uses Multiple Sequence Alignment (MSA) against a large library of template sequences to generate embeddings for its input sequence(s). It additionally obtains some preliminary structural information for the protein(s) by searching into a vast library of structural templates. These embeddings, both the structural and the sequential, are then fed into a novel neural-network architecture containing attention-based [32] as well as non-attention-based components, called Evoformer, and in turn fed into a transformer-like module to produce an initial structural prediction. This prediction is then refined three times by being recycled back into the network, to eventually produce the final predicted structure.

The problem of antibody docking is however far from solved; in the AlphaFold paper it is explicitly mentioned that AlphaFold Multimer does not perform well when it comes to antigen-antibody binding. There is therefore a lot of room for improvement; in this thesis we provide a robust method for augmenting a docking model, that can be extended to apply to any docking model, such as AlphaFold Multimer.

Chapter 3

The B-Cell Epitope Model

In this section we study the problem of B-Cell epitope prediction and propose a model that outperforms the vast majority of other state-of-the-art models for the same task.

3.1 Problem Formulation

Before we can start building our model, we first need to concretely define the problem we are solving. Given an antigenic protein A, we say that an amino-acid residue r in A is part of a B-cell epitope, or equivalently, is an epitope residue, if there exists an antibody B such that:

1. A and B have been experimentally observed to form a crystal structure, that is, B can dock onto A, and,

2. the heavy carbon C_α of r is within x units of distance from any C_α atom in B.

Therefore, our goal is, given an antigenic protein, with its amino-acid sequence and three-dimensional structure, to predict which residues in the antigen are epitope residues.

3.2 Model

We will now discuss the architecture of our proposed epitope model, breaking down its components and elaborating on how each contributes to a robust representation of a protein. We will also discuss the various surface features we used, other than the primary and tertiary structure of the protein (i.e. sequence of amino-acids as well as the three dimensional structure of the protein).

3.2.1 Architecture

The epitope model consists of 4 components; ESM2 [21], a 150 million parameter language model by Facebook, pretrained on a large amount of protein sequences; a GNN (Graph Neural Network), modified to achieve $E(n)$-Equivariance; a bidirectional RNN (Recurrent Neural Network); and a fully connected multi-layer perceptron, which receives the resulting embeddings in order to produce the output logits. We will now discuss each of the components in greater detail.

Pretrained Language Model (ESM2)

ESM2 is a transformer-based language model [32] trained on more than 4 million protein sequences. For each protein sequence, approximately 15% of the amino-acids are masked and the model is tasked with recovering those hidden residues. This is a commonly used approach in Natural Language Processing, called Masked Language Modeling. In the context of proteins, this particular task allows the model to infer complex internal representations of the input sequence. Specifically, ESM2 performs really well in secondary structure prediction, that is, the identification of common arrangements of adjacent amino-acids in a polypeptide chain, as well as in binding site and contact prediction (i.e. the identification of residues between two interacting polypeptide chains that enable this interaction by being close together in the three-dimensional space).

The task we are tackling is very closely related to the tasks ESM2 excels at; we are interested in predicting residues on an antigenic protein that will interact with some antibody and our hypothesis is that ESM2 embeddings will, to some extent, have that information embedded in them.

$E(n)$-Equivariant Graph Neural Networks

Equivariant Graph Neural Networks (EGNNs for short) [24] is a novel architecture used to learn graph neural networks that are $E(n)$-equivariant. We say that a function $\phi : X \to Y$ is equivariant to a transformation T iff

$$\phi(T(x)) = T'(\phi(x)) \tag{3.1}$$

where T' is the equivalent transformation in the output space. In other words, a function is equivariant to a transformation if the order in which the function and the transformation are applied does not matter.

An $E(n)$-equivariant function is equivariant to translation, rotation, reflection, and permutation. $E(n)$-equivariance is crucial in our particular use-case, since it allows us to leverage the inherent symmetry of our task; in our structural dataset, proteins might appear translated, rotated, or otherwise symmetrically transformed, and we would like our model to recognize that a certain set of proteins corresponds to different transformations of the same original protein. EGNNs help us achieve that in a very elegant way.

Let us briefly go over the un-augmented architecture of Graph Neural Networks (which are by default permutation equivariant), as described in [10]. Consider a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ is the set of nodes and E=$\{(u_1, v_1), \ldots, (u_M, v_M)\}$, $(u_i, v_i) \in V^2$ the set of edges. A graph convolutional layer is defined as follows:

$$\begin{aligned} m_{i,j} &= \phi_e(h_i^l, h_j^l, a_{i,j}) \\ m_i &= \sum_{j \in N(i)} m_{i,j} \\ h_i^{l+1} &= \phi_v(h_i^l, m_i) \end{aligned} \tag{3.2}$$

where ϕ_e, ϕ_v are the edge and node functions respectively, commonly approximated by Multi-Layer Perceptrons, $N(i)$ the set of neighbors of node i, and h_i^l the embedding of node i at layer l; $a_{i,j}$ are edge features (e.g. distance between nodes, but also in the case of representing amino-acids it could also involve the types of amino-acids sharing an edge, that is, interacting); $m_{i,j}$ enables message passing between adjacent nodes, and m_i accumulates all the information passed to node i at a given layer.

Now let us examine the modified Equivariant Graph Convolutional Layer (EGCL), as proposed in [24]. Note that we have slightly tweaked the definition to account for the fact that our network is sparse and the adjacency matrix known, that is, we do not infer edges between residues. More details on how we build the adjacency matrix can be found in section 3.2.3

$$\begin{aligned} m_{i,j} &= \phi_e(h_i^l, h_j^l, \|x_i^l - x_j^l\|^2, a_{i,j}) \\ x_i^{l+1} &= x_i^l + \frac{1}{|V|}\sum_{j\neq i}(x_i^l - x_j^l)\phi_x(m_{i,j}) \\ m_i &= \sum_{j\in N(i)} m_{i,j} \\ h_i^{l+1} &= \phi_v(h_i^l, m_i) \end{aligned} \tag{3.3}$$

where ϕ_x is yet another Multi-layered Perceptron, and x_i^l are the coordinate embeddings at layer l (so x_i^0 are the initial coordinates of residue i). Notice that steps 1, 3, and 4 of equation 3.3 are $E(n)$-invariant; assuming that h^0 is invariant, we see that $m_{i,j}$ is also going to be invariant (since the distance between x_i^l and x_j^l is invariant to equivalent rotation and translation), and m_i as well as h_i^{l+1} are also going to be invariant, since they only depend on $m_{i,j}$ and h^l (the latter being invariant by induction). Finally, it is not difficult to see that the second step of 3.3 is $E(n)$-equivariant, that is, for some rotation matrix Q and a translation g we have that

$$Qx_i^{l+1} + g = Qx_i^l + g + \frac{1}{|V|}\sum_{j\neq i}(Qx_i^l + g - [Qx_j^l + g])\phi_x(m_{i,j}) \tag{3.4}$$

To see that this holds, we can simplify the right-hand side to get:

$$\begin{aligned}
&Qx_i^l + g + \frac{1}{|V|}\sum_{j\neq i}(Qx_i^l + g - [Qx_j^l + g])\phi_x(m_{i,j}) = \\
&Q\Big(x_i^l + \frac{1}{|V|}\sum_{j\neq i}(x_i^l - x_j^l)\phi_x(m_{i,j})\Big) + g = \\
&Qx_i^{l+1} + g
\end{aligned} \tag{3.5}$$

as desired. A more detailed proof of the $E(n)$-equivariance of this graph model can be found in appendix A of [24].

EGNNs naturally excel at structural protein modeling, since they allow us to produce embeddings that capture nuanced interactions between proximate residues in the three dimensional space.

Recurrent Neural Networks (RNN)

Since we are working with sequences, it is only natural to utilize a sequence model to augment our epitope model. For that purpose, we will use the Gated Recurrent Unit architecture, as described in [2].

GRUs solve the short-term memory problem that plain RNNs suffer from by introducing two types of gates: the update gate, responsible for determining how much past information should be passed along to future time steps, and the reset gate, responsible for determining how much of the past information should be forgotten. GRUs function very similarly to Long Short-Term Memory networks [11], however they are significantly more space-efficient compared to LSTMs, since they lack an output gate.

For completeness sake, we include the definition of the Fully-Gated Recurrent Unit, as described in [2].

$$
\begin{aligned}
z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\
\hat{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \hat{h}_t
\end{aligned}
\tag{3.6}
$$

Here $\odot$ denotes element-wise multiplication between vectors (also known as Hadamard Product).

Multi-Layer Perceptron

Finally, we would like to convert the embeddings produced by our model into output logits that, after feeding into a sigmoid function, we can interpret as the probability that a given residue is part of some epitope.

3.2.2 Surface Features

We experimented with various surface-level features for encoding residues. We will now go over the ones that we ended up using, as well as some honorable mentions that did not make into the final model.

Relative Accessible Surface Area

Relative Accessible Surface (or Solvent) Area (RASA or relative ASA) of an amino-acid residue within a polypeptide chain is the measure of the residue's solvent exposure. In simpler terms, it is a measure of how accessible this residue is to the solvent (most commonly water) that surrounds the protein. This is clearly a helpful feature to include in our model, since residues that are not very accessible to the solvent that surrounds them are less likely to come in contact with antibody residues in an antibody-antigen complex, and therefore less likely to be epitopes.

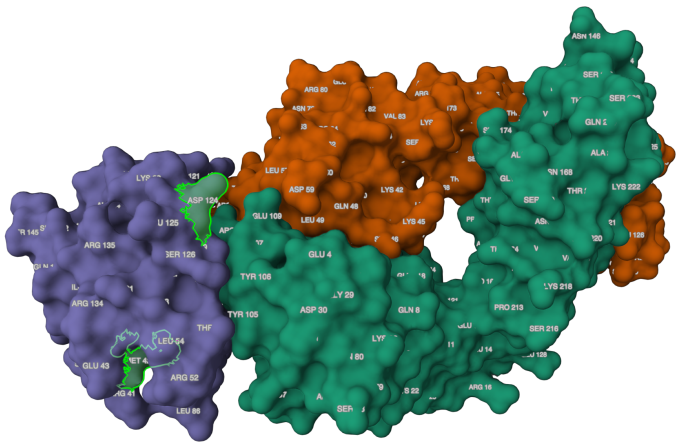


Figure 3-1: In many instances, the less accessible a residue is, the less likely it is to be part of an interaction site. Here, we have a structure taken from [27]. Notice that the less accessible MET 42 is not part of the interaction site between the two proteins, whereas the more accessible ASP 124 is. Of course this is not a general rule, however it demonstrates the usefulness of including RASA as a feature of our model.

Secondary Structure

Based on their amino-acid sequence, proteins fold into three-dimensional conformations. There are four different levels one can use to describe the structure of a protein, in order of increasing complexity, primary, secondary, tertiary, and quaternary. We have already mentioned the primary and tertiary structures of a protein (the protein sequence and the three-dimensional arrangement of residues in space, respectively). The secondary structure of a protein lies somewhere between the primary and tertiary structure, in regards to complexity; it is defined as the local spatial conformation of the polypeptide backbone excluding the side chains [29]. More intuitively, the secondary structure of a protein describes common formations that contiguous subsequences of the protein sequence take in space. The two most common such formations are α-*helixes* and β-*sheets*.

We utilized the Dictionary of Protein Secondary Structure (DSSP) classification [16] to classify conformations into 8 different types; 3_{10} helix, α helix, π helix, hydrogen bonded turn, extended strand in parallel and/or anti-parallel β-sheet conformation, residue in isolated β bridge, bend, and coil. To include this information in our model, we used 1-hot encoding to create 8-dimensional secondary structure class

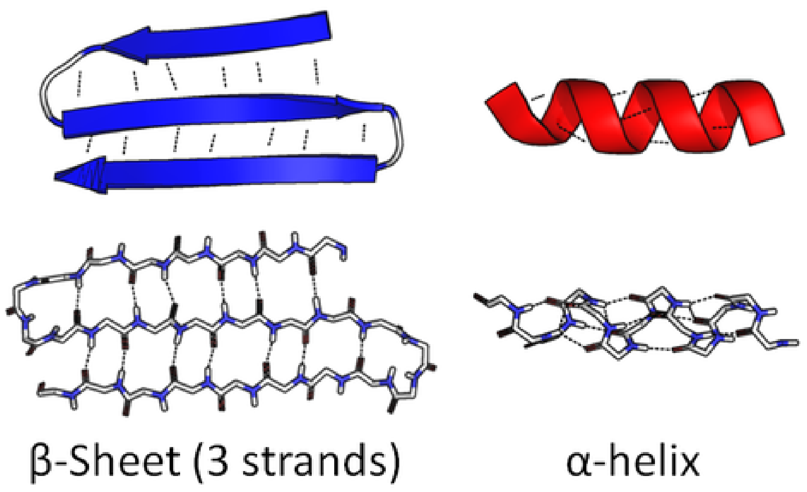


Figure 3-2: Two most common secondary structure conformations, the β-sheet and the α-helix [26].

embeddings for each residue.

Torsion Angles

We also included in our model the torsion angles of the N-C$_\alpha$(Φ) and C$_\alpha$-C(Ψ) bonds. Polypeptide chains can freely rotate around these bonds, therefore affecting the structure of the entire protein, so it is important that we include them in our residue representation.

Approximated Residue Depth

Residue Depth is a measure that describes how "burried" a residue is within the protein. Residue depth is complementary to relative ASA, a solvent accessibility measure we discuss above. In our experiments, we used an approximation of the actual residue depth (ρ), calculated as described in [9]. In the final model, we used relative ASA instead of ρ, as the latter ended up introducing noise which actually decreased model performance. It is worth noting, however, that, when not including the language model embeddings in our architecture, incorporating ρ as a residue feature actually did slightly improve model performance.

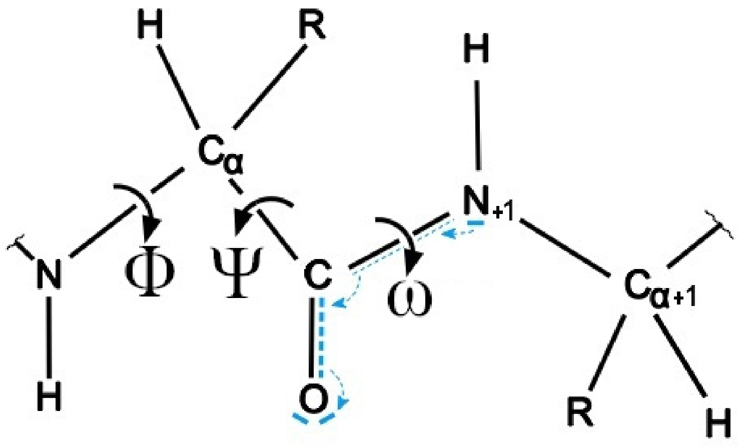


Figure 3-3: Torsion Angles; ω angles are generally fairly restricted, so we do not include them as features in our model [31].

3.2.3 Putting it All Together

Starting with the protein sequence, we feed it into the pretrained ESM2 language model (3.2.1), to produce 640-dimensional residue embeddings. The ESM2 embeddings are then fed into a simple fully-connected MLP, with no activation function or dropout, to reduce their dimension to D_{ESM2}. The reduced embeddings are then concatenated with the surface features (3.2.2) described above (relative ASA, 1-hot encoded secondary structure, Ψ, Φ) to obtain ($D_{\text{ESM2}} + 11$)-dimensional residue embeddings.

These embeddings are used as node features in our EGNN (3.2.1), so for each residue i, we initialize h_i^0 accordingly. The multilayer EGNN, with internal dimension D_{EGNN}, dropout p_{EGNN}, and n_{EGNN} layers, will update these embeddings, while keeping their dimension as is. For the edge features, as well as the overall graph construction, we took inspiration from [36]; for each residue, represented in the 3D space as a point with coordinates equal to the experimentally observed coordinates of its C_α atom, we add edges to its 10 nearest neighbors. We also add edges to all residues in a 10Å radius. Finally, we add a total of 4 sequential edges, 2 to the pair of residues preceding our residue in the amino-acid sequence and 2 to the pair of residues following it. We featurize each edge between residues i and j with a 45-dimensional 1-hot encoding (21 positions to represent the type of amino-acid of residue i, 21 more for residue j, as well as 3 positions to represent the type of edge connecting i and j, that is, sequential, nearest-neighbor, or within 10Å of each other).

The output of the EGNN is then be fed into a multi-layered bidirectional GRU (3.2.1) with hidden dimension D_{GRU}, dropout p_{GRU}, and n_{GRU} layers, to produce $2D_{\text{GRU}}$-dimensional embeddings.

Finally the $2D_{\text{GRU}}$-dimensional residue representations are fed into a fully-connected MLP (3.2.1) which reduces the dimension of the embeddings to D_{MLP}, applying the ReLU activation function and a dropout of p_{MLP}. Then, the dimension is further reduced with a second fully-connected convolution layer to produce the output logits.

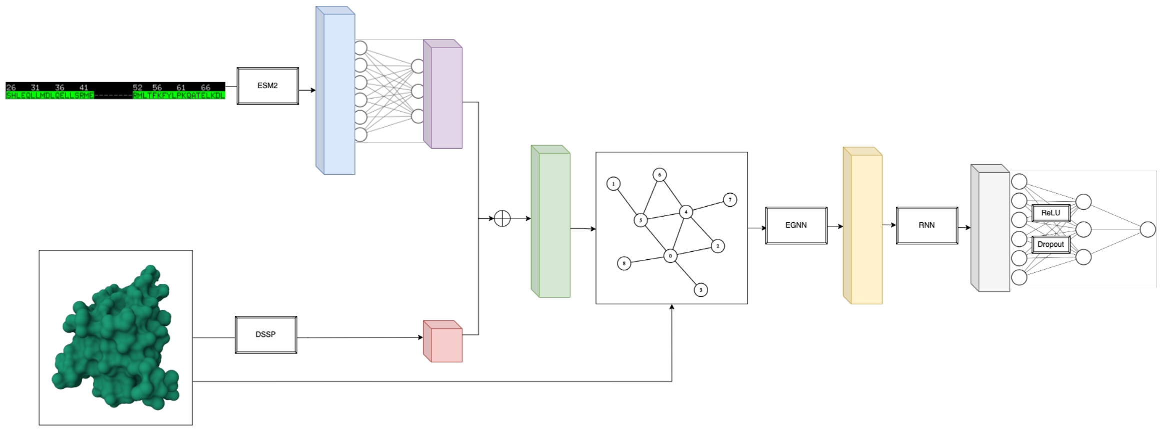


Figure 3-4: A visual representation of the model architecture demonstrating how the per-residue logits are calculated.

3.3 Data

For training, evaluating, and testing our model, we used a multitude of publicly available datasets. In this section, we will go over them, and explain how the data is split in a fair way to avoid data leaks.

3.3.1 Protein Data Bases

Our main source of protein structures was the Protein Data Bank (PDB) [1]. The PDB archive, currently the largest open access digital data resource in all of Biology, contains at least a terabyte of protein structural data, DNA, and RNA.

Due to the large scale of PDB, it can get tricky to accumulate a consistent set of protein-antibody complexes; for that purpose, we utilised The Structural Antibody Database (SAbDab) [25]. SAbDab contains all the antigen-antibody complexes found in PDB, consistently presented and annotated.

Finally, we explored using labeled data from the Immune Epitope Database (IEDB) [33]. IEDB is a carefully curated database of experimentally validated epitopes, containing a vast array of more than 200,000 linear epitopes (i.e. epitopes where all the residues are a contiguous subsequence of the protein sequence). We attempted using linear epitopes from IEDB to formulate a pretraining routine, with little to no success (see Appendix section A.1). Since the three-dimensional structures of the IEDB proteins were hard to obtain, we used ESMFold [21] to obtain approximate 3D structures for the proteins used.

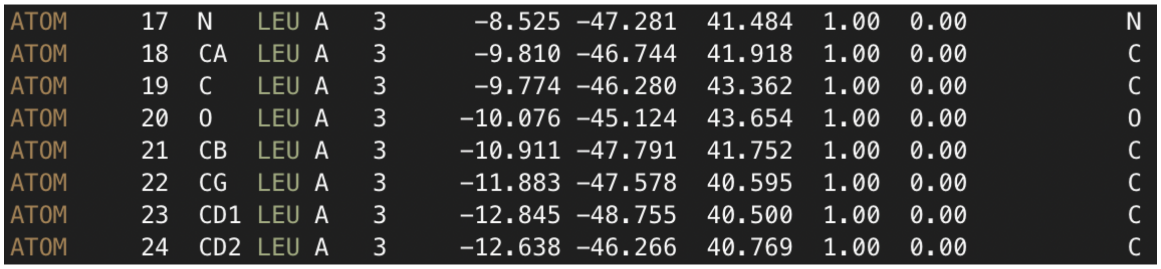
3.3.2 Data Processing

For our experiments, we used the train-test splits from [36] and [3]. We will however discuss here how we re-created these splits, as it will be relevant when we later talk about antibody docking.

Labeling the Data

First off, we start with the raw PDB files from SAbDab. Program DataBase (PDB) is a file format commonly used to store the three dimensional structure of proteins. Each PDB file contains two unique entities; the antigen, which commonly consists of a single amino-acid chain, and the antibody, which consists of two chains, the Heavy and the Light chain.

As we mentioned above, for simplicity, we represent residues in the 3D space by



```
ATOM     17  N   LEU A   3      -8.525 -47.281  41.484  1.00  0.00           N
ATOM     18  CA  LEU A   3      -9.810 -46.744  41.918  1.00  0.00           C
ATOM     19  C   LEU A   3      -9.774 -46.280  43.362  1.00  0.00           C
ATOM     20  O   LEU A   3     -10.076 -45.124  43.654  1.00  0.00           O
ATOM     21  CB  LEU A   3     -10.911 -47.791  41.752  1.00  0.00           C
ATOM     22  CG  LEU A   3     -11.883 -47.578  40.595  1.00  0.00           C
ATOM     23  CD1 LEU A   3     -12.845 -48.755  40.500  1.00  0.00           C
ATOM     24  CD2 LEU A   3     -12.638 -46.266  40.769  1.00  0.00           C
```

Figure 3-5: Consecutive entries in a PDB file representing the third residue in an amino-acid chain. Columns 7-10 contain the positions of the atoms comprising the residue in the three-dimensional space.

the location of their heavy carbon atom (C_α). With that in mind, we define an epitope residue on an antigenic protein Ag to be any residue in Ag within x units of distance from any other residue belonging to any antibody that forms a crystal structure with Ag. The distance x commonly ranges from 3 to 8 Å. Observe that to completely define all the epitope residues of an antigen, one would need structural information about all the antigen-antibody complexes this particular antigen can be a part of.

Sequence Identity

Now that we have established a labeling scheme for our residues, we need to make sure that the proteins in our training set are *non-redundant*. To define the notion of redundancy, we first need to talk about the concept of *sequence identity*. For two amino-acid (or DNA, or RNA) sequences, sequence identity is a measure of similarity between them. More specifically, after aligning the sequences, their sequence identity is equal to the ratio of the number of aligned residues that match and the total number of aligned residues. Here is a toy example to better illustrate the concept of sequence identity:

```
A: QVAAGQLGGTPPVKGQQLNASIIAQTR
B: QVAGQLGGTTPVKGQLNASIIIAQTR
```

First, we align the two sequences; to optimally do so, we can use a dynamic programming algorithm to minimize an alignment metric, such as the Damerau–Levenshtein distance [7]. This, however, is not in the scope of this thesis.

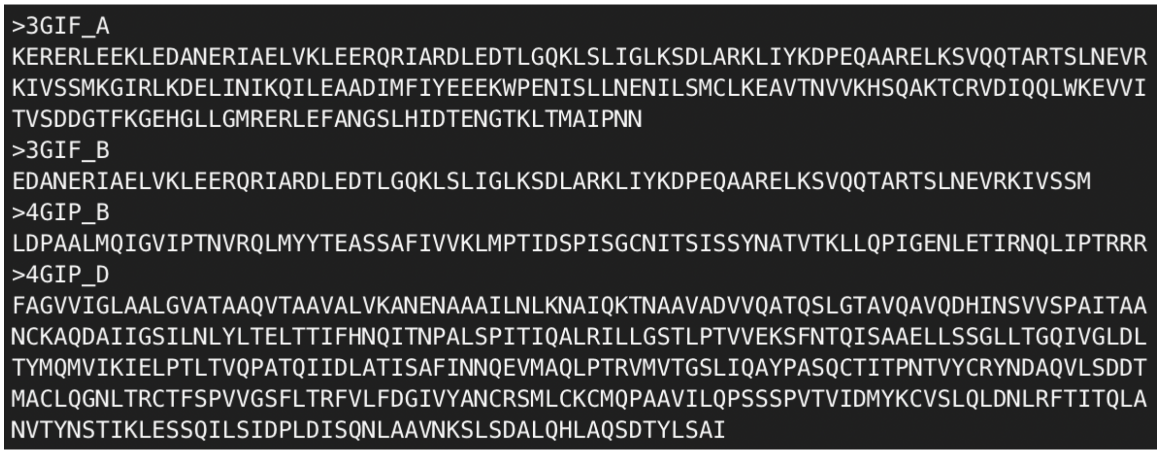
```
A: QVAAGQLGGTpPVKGQQLNASII-AQTR

B: QVA-GQLGGTtPVKGQ-LNASIIIAQTR
```

With the sequences now aligned, we can calculate their sequence identity as

$$\frac{\text{no. of matching residues}}{\text{no. of total aligned residues}} \times 100\% = \frac{24}{28} \times 100\% \approx 85.7\%.$$

MMSEQS2

For efficiently and accurately aligning and clustering our sequences based on sequence identity, we used the tool MMSEQS2 (Many-against-Many SEQuence Searching) [28]. MMSEQS2 takes a set of protein sequences in FASTA format and a minimum sequence identity cuttoff c, and clusters the sequences such that, within any cluster, all pairs of sequences in the cluster have sequence identity of at least c.



```
>3GIF_A
KERERLEEKLEDANERIAELVKLEERQRIARDLEDTLGQKLSLIGLKSDLARKLIYKDPEQAARELKSVQQTARTSLNEVR
KIVSSMKGIRLKDELINIKQILEAADIMFIYEEEKWPENISLLNENILSMCLKEAVTNVVKHSQAKTCRVDIQQLWKEVVI
TVSDDGTFKGEHGLLGMRERLEFANGSLHIDTENGTKLTMAIPNN
>3GIF_B
EDANERIAELVKLEERQRIARDLEDTLGQKLSLIGLKSDLARKLIYKDPEQAARELKSVQQTARTSLNEVRKIVSSM
>4GIP_B
LDPAALMQIGVIPTNVRQLMYYTEASSAFIVVKLMPTIDSPISGCNITSISSYNATVTKLLQPIGENLETIRNQLIPTRRR
>4GIP_D
FAGVVIGLAALGVATAAQVTAAVALVKANENAAAILNLKNAIQKTNAAVADVVQATQSLGTAVQAVQDHINSVVSPAITAA
NCKAQDAIIGSILNLYLTELTTIFHNQITNPALSPITIQALRILLGSTLPTVVEKSFNTQISAAELLSSGLLTGQIVGLDL
TYMQMVIKIELPTLTVQPATQIIDLATISAFINNQEVMAQLPTRVMVTGSLIQAYPASQCTITPNTVYCRYNDAQVLSDDT
MACLQGNLTRCTFSPVVGSFLTRFVLFDGIVYANCRSMLCKCMQPAAVILQPSSSPVTVIDMYKCVSLQLDNLRFTITQLA
NVTYNSTIKLESSQILSIDPLDISQNLAAVNKSLSDALQHLAQSDTYLSAI
```

Figure 3-6: The FASTA format.

Epitope Grafting and Redundancy Reduction

Now that we have talked about sequence identity, it is time to define the notion of *redundancy*. Generally speaking, we would like the sequences within our protein dataset to have a relatively small maximum pairwise sequence identity. We therefore define a set of protein sequences to be *redundant* if it contains a pair of sequences with identity more than $c_{\text{redundant}}$. To achieve non-redundancy, we cluster the protein sequences using MMSEQS2 at minimum sequence identity of $c_{\text{redundant}}$. Then, withing each cluster, we pick a single sequence as the representative (typically MMSEQS2 will pick it for us). For each sequence s in the remaining sequences within the cluster, we align it with the representative. For every matching residue in the alignment, if the residue is marked as an epitope in s, we mark it as an epitope in the representative. This step is referred to as "Epitope Grafting". Then, we keep only the representative sequences and discard the rest.

We perform this redundancy reduction procedure twice, the first time with 95% minimum sequence identity, and the second time at 70%.

Fair Data Splitting

Now that we have a non-redundant set of proteins, it is crucial that we split it into training/validation/test fairly; specifically, we would like to not have too similar sequences in our training and test sets, as this could be a cause of a data leak; to be able to deduce whether our model generalizes well, it is really important that the sequences in our test set are adequately different than the sequences in our training set. To achieve that, we cluster the sequences with minimum sequence identity of 30-50%, and split the data such that sequences belonging in the same cluster always end up in the same split.

After this procedure, we are left with 577 proteins for training and evaluation, and 66 proteins for testing. These numbers provide some much needed insight on why the problem of B-cell epitope prediction, and in extension the problem of antibody docking, is so challenging. Antibodies are proteins with very high specificity that can bind to only specific antigens. Without an adequate amount of data, it is very difficult to create a B-cell epitope model that will generalize well to unknown antigens, and 577 proteins are generally not enough data to easily achieve such a feat. However, as we will show in the following sections, we did the best we could with the data that was available to us and managed to create a competent model.

3.4 Training and Evaluation

Now that we have our model architecture set and our data cleaned up and properly split, it is time to decide the training regime, i.e. the loss function we will be minimizing. We ended up going with Binary Cross Entropy (BCE) loss, the most sensible candidate since our task is a binary classification one. Given two discrete probability distributions P (representing the data/observations) and Q (representing the estimations) in the same sample space $\mathcal{X}$, the cross-entropy between P and Q is defined as

$$H(P,Q) = H(P) + D_{\mathrm{KL}}(P\|Q) \tag{3.7}$$

where $H(P)$ is the entropy of distribution P (measuring the degree of randomness) and $D_{\mathrm{KL}}(P\|Q)$ the Kullback-Leibler divergence [19] between the two distributions (measuring the degree of surprise if one were to use Q instead of the ground truth P), respectively defined as

$$\begin{aligned} H(P) &= -\sum_{x\in\mathcal{X}} P(x)\log P(x) \\ D_{\mathrm{KL}}(P\|Q) &= \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)} \end{aligned} \tag{3.8}$$

Simplifying, we get:

$$\begin{aligned} H(P,Q) &= -\sum_{x\in\mathcal{X}} P(x)\log P(x) + \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)} \\ &= -\sum_{x\in\mathcal{X}} P(x)(\log P(x) - \log P(x) + \log Q(x)) \\ &= -\sum_{x\in\mathcal{X}} P(x)\log Q(X) \end{aligned} \tag{3.9}$$

We can use this definition of cross entropy to define a loss function over all the observations. Namely, for our ground truth distribution p_n and prediction distribution q_n, for observations indexed $n = 1 \dots N$, where N is the total number of observations, we get:

$$\begin{aligned} J(\mathbf{g}) &= \frac{1}{N} \sum_{n=1}^{N} H(p_n, q_n) \\ &= -\frac{1}{N} \sum_{n=1}^{N} \sum_{x \in \mathcal{X}} p_n(x) \log q_n(x) \\ &= -\frac{1}{N} \sum_{n=1}^{N} \Big[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \Big] \end{aligned} \tag{3.10}$$

where $\mathbf{g}$ is our differentiable model, $y_n \in \{0, 1\}$ (0: non-epitope, 1: epitope) the true label of the n^{th} observation, and $\hat{y}_n = \mathbf{g}(x_n)$ the predicted value from the model, given x_n, the features of the n^{th} observation.

We trained our model using the Adam variant of stochastic gradient descent [17] for 30 epochs with learning rate of 10^{-3}. We used the following set of hyperparameters (defined in section 3.2.3) for the different components of our model:

$$\begin{array}{lllllllll} D_{\text{ESM2}} & = & 512 & & & & & & \\ D_{\text{EGNN}} & = & 512 & p_{\text{EGNN}} & = & 0 & n_{\text{EGNN}} & = & 2 \\ D_{\text{GRU}} & = & 128 & p_{\text{EGNN}} & = & 0 & n_{\text{EGNN}} & = & 2 \\ D_{\text{MLP}} & = & 128 & p_{\text{MLP}} & = & 0 & & & \end{array}$$

As our evaluation metrics, we used the Area Under the Receiver Operating Characteristic Curve (AUROC) curve, as well as the Area Under the Precision Recall Curve (AUPRC).

3.5 Results

We compared our model with various state-of-the-art B-cell epitope prediction models ([4], [13], [23], [18], [30], [36]). For fairness, we used the training and test set from [36]. Bellow follows a table summarising the performance comparison results:

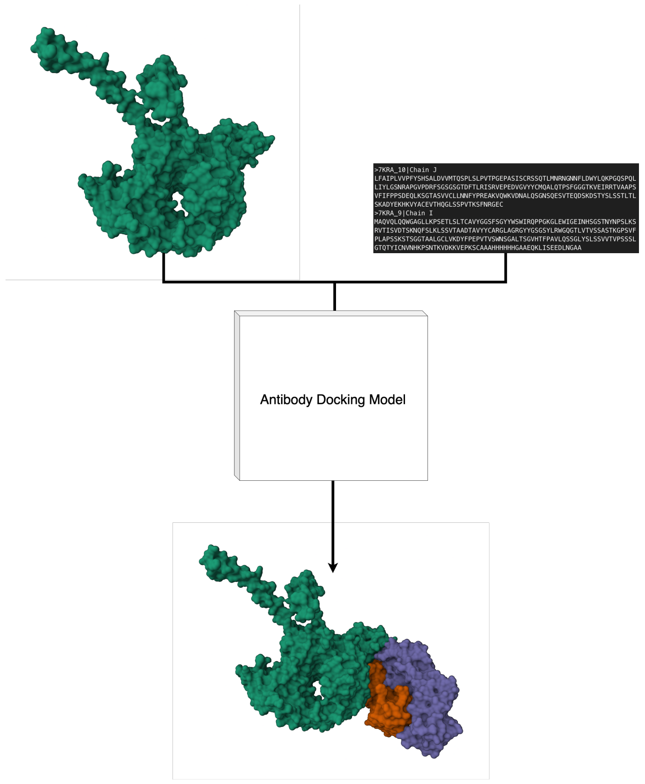
Model	**AUROC**	**AUPRC**
EpiDope	54.7	10.2
Bepipred-2.0	64.8	13.2
ElliPro	63.2	12.2
Discotope-2.0	65.5	15.4
ScanNet_WT	64.8	13.5
ScanNet_T	71.2	18.2
GraphBepi	75.1	**26.1**
Our Model	**76.1**	23.8

Our model outperformed all others in the AUROC metric, and came second in the AUPRC metric. Notice that the performance of our model is very similar to that of GraphBepi [36]. GraphBepi uses a very similar architecture to ours, and was in fact published fairly recently. The main differences between the two models are that GraphBepi uses evolutionary features (we do not), they use the 3 billion parameter ESM2 model (we use the 150 million parameter one), and we reduce the dimension of the ESM2 embeddings with a fully-connected layer before feeding them to the EGNN (they do not).

Chapter 4

Antibody Docking

Now that we have developed a solid B-cell epitope model, we will discuss how we used it to improve antibody docking. But before we go over our methods, we will briefly provide a solid formulation to the problem of antibody docking, as well as the baseline docking model we used in our experiments.



```
>7KRA_10|Chain J
LFAIPLVVPFYSHSALDVVMTQSPLSLPVTPGEPASISCRSSQTLMNRNGNNFLDWYLQKPGQSPQL
LIYLGSNRAPGVPDRFSGSGSGTDFTLRISRVEPEDVGVYYCMQALQTPSFGGGTKVEIRRTVAAPS
VFIFPPSDEQLKSGTASVVCLLNNFYPREAKVQWKVDNALQSGNSQESVTEQDSKDSTYSLSSTLTL
SKADYEKHKVYACEVTHQGLSSPVTKSFNRGEC
>7KRA_9|Chain I
MAQVQLQQWGAGLLKPSETLSLTCAVYGGSFSGYYWSWIRQPPGKGLEWIGEINHSGSTNYNPSLKS
RVTISVDTSKNQFSLKLSSVTAADTAVYYCARGLAGRGYYGSGSYLRWGQGTLVTVSSASTKGPSVF
PLAPSSKSTSGGTAALGCLVKDYFPEPVTVSWNSGALTSGVHTFPAVLQSSGLYSLSSVVTVPSSSL
GTQTYICNVNHKPSNTKVDKKVEPKSCAAAHHHHHHGAAEQKLISEEDLNGAA
```

Figure 4-1: The problem of antibody docking, illustrated.

4.1 The Problem of Docking

The problem of antibody docking can be formulated as follows: given the sequence and three-dimensional structure of an antigenic protein, and the amino-acid sequence of an antibody, we would like to predict, as accurately as possible, the three-dimensional structure of the complex that will be formed when the given antibody docks onto the given antigen. Accuracy, or, more correctly, *success rate*, here is commonly defined in terms of what percentage of the test samples very closely resemble the ground truth antigen-antibody complex.

4.2 Model

Creating a solid antibody docking model is beyond the scope of this thesis; for that reason, in our experiments, we used Wengong Jin's antibody docking model. Wengong is a Postdoctoral Associate at Eric and Wendy Schmidt Center of Broad Institute, researching novel machine learning algorithms for biology and medicine. For the remainder of this thesis, we will be referring to his model as "the docking model".

4.2.1 The Docking Model

The docking model is a part of Wengong's research that has not yet been published; for that reason, we will be using it as a black box without going into too much detail regarding its underlying architecture.

The docking model takes in three inputs; the antigen (both its three-dimensional structure and its amino-acid sequence), the antibody (just its sequence), and a seed. The seed is a residue on the antigenic protein around which the model will attempt to dock the antibody. The docking model was trained with knowledge of the optimal (in the vast majority of cases) seed, the *epitope center*. The epitope center is defined as the antigenic residue with the closest distance to any residue belonging to a Complimentarity-Determining Region (CDR) of the antibody we are docking; notice that here we are referring to a single antibody, in contrast to how we have been interpreting the notion of an epitope so far in this thesis.

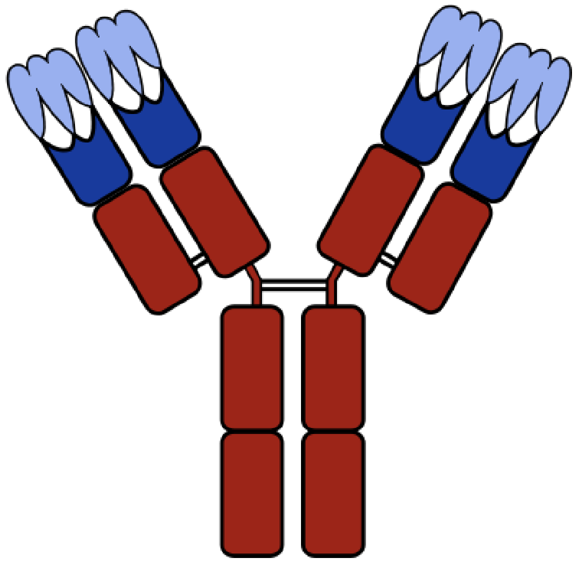


Figure 4-2: An antibody, with its CDRs colored in light blue; the CDRs will come in contact with the antigen upon docking [35].

Observe that the optimal seed will not always be known; in fact, in most practical applications, we will not know the seed at all. The quality of the predictions can therefore vary significantly based on the choice of seed. That is where our epitope model comes in; our hypothesis is that using our model's predictions, after some clever post-processing, we will be able to come up with a seed or a collection of seeds that will produce structures with quality comparable to using the optimal seed.

Namely, we would like to select a set $\mathcal{S}$ of K seeds $s_1, s_2, \ldots, s_K$ such that the following metric is minimized:

$$\begin{aligned} \hat{y}_n &= \mathbf{g}_{\text{dock}}(x_n; s_i) \\ \mu_{\text{RMSD}}(\mathcal{S}) &= \frac{1}{N} \sum_{n=1}^{N} \min_{s_i \in \mathcal{S}} \text{RMSD}\Big(y_n, \hat{y}_n\Big) \end{aligned} \tag{4.1}$$

where N is the number of test samples, y_n is the three-dimensional structure of the ground truth docking conformation, and $\mathbf{g}_{\text{dock}}$ the docking model, taking x_n as features and s_i as the seed, to produce a predicted structure, which we denote with

$\hat{y}_n$. RMSD refers to the Root Mean Square Distance metric, defined as follows:

$$\text{RMSD}(y, \hat{y}) = \sqrt{\frac{1}{L}\sum_{i=1}^{L}\|y_i - \hat{y}_i\|^2} \tag{4.2}$$

where L is the number of atoms in the ground-truth structure, indexed in the order they appear in its PDB representation.

4.2.2 Post-Processing

To keep the evaluation of our docking predictions fair, we retrained our model on Wengong's training set (training/validation charts of the best model we trained can be found in A.3). We labeled residues as described in the Data Processing section (3.3.2), and we also created a separate label for the epitope center. Furthermore, we employed the redundancy reduction technique, which was described in section 3.3.2 as well.

Since we have established the overall architecture of our epitope model, we now have the chance to get creative with some finer parts of our model, as well as with the choice of loss function for training. Because of the fact that in the context of antibody docking there are three possible labels for each residue (non-epitope, epitope, epitope-center), we had the option to change the setting of our problem from binary classification to multi-class classification, or to multi-label classification, difference being that in the latter, each sample can have multiple labels attached to it. That would, of course, imply slightly tweaking our loss function to account for the change in setting, as well as changing our evaluation metrics. We also explored a completely different training regime, in which only the top K ranking residues were backpropagated. We go over in greater detail about each model/training configuration we employed and how each performed on the test set.

We ran our models on Wengong's test set to produce per-residue logits. For models trained in a multi-class or multi-label setting, we then applied the softmax function to turn the logits into per-label probabilities (p_0, p_1, p_2), for each residue:

$$p_i = \sigma(\mathbf{z})_i = \frac{e^{\mathbf{z}_i}}{\sum_{j=0}^{2} e^{\mathbf{z}_j}} \tag{4.3}$$

For binary classification models, we would apply the sigmoid function, to obtain the probability $p = \sigma(\mathbf{z}) = \frac{1}{1+e^{-p}}$ that the given residue is either an epitope or an epitope-center.

In the remainder of this section, we will address how we leveraged our predictions to improve the predictive quality of the docking model, by selecting a set of K seeds.

Baselines

The baseline that we will be comparing our seeding method against will be simple; randomly sample K residues from the antigen sequence and use them as seeds. The docking model could also be used in a setting where a seed is given, however the area of effect of that particular seed is set to infinity, therefore virtually not using any seed at all. For completeness sake, we include this baseline in our tables of comparisons as well, even though with as little as two tries, even the model selecting random seeds mentioned above outperforms it in all the relevant metrics.

Our Three Seeding Methods

Algorithm 1 Our first seed-selection approach is to rank the antigen residues based on the probability that they are either an epitope or an epitope-center, that is, p in the single-label setting, and $p_1 + p_2$ in the multiclass/multilabel setting. Then, we pick the top-K ranked residues as our candidate seeds.

Algorithm 2 To introduce some randomness to our approach, we tried something a little more elaborate; for each antigen, we create a distribution q by normalizing the per-residue predicted probabilities. For $i = 1 \cdots R$, where R is the number of residues in the antigen, we have:

$$q_i = \frac{p_1^{(i)} + p_2^{(i)}}{\sum_{j=1}^{R}(p_1^{(j)} + p_2^{(j)})} \tag{4.4}$$

The distribution q is defined similarly for the binary classification setting. Then, we pick residues in decreasing order of q_i until we have selected at least 60% of the distribution. From this resulting set of residues, we randomly sample K seeds.

Algorithm 3 To account for residue density in certain parts of the antigenic protein, we used the following algorithm. First, we create distribution q as described above. We select top ranking residues until we have covered 75% of the distribution. Then, we employ the k-Medoids algorithm [20] to cluster the resulting set of points into 10 clusters (since in our experiments $K \leq 10$). From each cluster, we pick the representative (medoid), and, after ranking the medoids in decreasing order of q_i, we select the top-K ones as our seeds.

Algorithm 4 In order to reduce the randomness of the previous algorithm, we tried a setting where instead of picking the medoid as the representative of each cluster, we select the top ranking residue from each cluster. In that fashion we obtain a set of K residues in a more deterministic fashion (since the K-medoids algorithm is innately stochastic and the representative medoid is chosen randomly).

4.3 Results

In this section, we go over how the docking predictions of each configuration we employ (model, training regime, and post-processing algorithm) perform in various evaluation metrics. We test multiple different training settings, and evaluate each model's predictions with a wide array of metrics. We briefly mentioned each candidate training regime in the previous section; here, we go over all of them in more detail, and present the performance of each method.

Multilabel, Cross Entropy Loss

The first setting we examine is that of using multiple labels to represent the three different classes {non-epitope, epitope, epitope-center}. Specifically, we represent non-epitopes as the vector $[1, 0, 0]$, epitopes but not epitope-centers as $[0, 1, 0]$, and epitope centers as $[0, 1, 1]$. The rationale behind this choice is that we would like epitope-centers to be ranked highly both as epitope-centers and as simple epitopes; a multi-class, single-label scheme would not achieve that.

For our loss function, we merely move from Binary Cross Entropy to Multi-class Cross Entropy. To account for the fact that the vast majority of residues in our training set are non-epitope residues, and that there will be at most one or two epitope centers per antigen, we weight the three different class losses accordingly, with 0.1 for non-epitope, 0.4 for epitope, and 0.5 for epitope-center. In that way, it is ensured that true-positive predictions of non-epitopes do not dominate our loss function.

To evaluate our model, as well as the docking predictions we use a multitude of metrics; AUROC and AUPRC on the test set, labeled as descibed in section 3.3.2, μ_{RMSD}, as defined in 4.1, for a single seed and for $K = 10$ seeds. Finally, we also introduce a new metric, $\text{success}_{\delta}^{(K)}$, defined as the percentage of predictions after using K seeds, where the RMSD was within δ of the RMSD achieved when perfectly seeding the docking model with the correct epitope center.

Below is a table summarizing the results in this particular setting, using the three seeding methods described in the previous section, for the evaluation methods we just discussed:

Setting	$\text{success}_{\delta=2}^{(1)}$	$\text{success}_{\delta=1}^{(1)}$	$\text{success}_{\delta=2}^{(10)}$	$\text{success}_{\delta=1}^{(10)}$	$\mu_{\textbf{RMSD}}^{(1)}$	$\mu_{\textbf{RMSD}}^{(10)}$
Random Baseline	3	3	35	29	36	20
Seedless Baseline	6	5	-	-	**31**	-
Algorithm 1	**12**	**12**	47	44	34	20
Algorithm 2	11	7.6	44	39	34	19
Algorithm 3	7.6	7.6	**50**	44	35	**18**
Algorithm 4	7.6	6.1	**50**	**45**	34	**18**

AUROC	**AUPRC**
76.6	38.3

Observe that even though our model reports a mean RMSD score very close to the random baseline for $K = 10$ seeds, the success rate of our model is significantly higher.

Binary Classification, Binary Cross Entropy Loss

In this setting, we treat epitope-centers as mere epitopes, and modify the task back to a binary classification task. Below is a table summarizing our results in this experiment.

Setting	$\text{success}^{(1)}_{\delta=2}$	$\text{success}^{(1)}_{\delta=1}$	$\text{success}^{(10)}_{\delta=2}$	$\text{success}^{(10)}_{\delta=1}$	$\mu^{(1)}_{\mathbf{RMSD}}$	$\mu^{(10)}_{\mathbf{RMSD}}$
Random Baseline	3	3	35	29	36	20
Seedless Baseline	6	5	-	-	**31**	-
Algorithm 1	15	**15**	47	45	32	22
Algorithm 2	10	6	53	46	34	19
Algorithm 3	**18**	14	**56**	47	**31**	**18**
Algorithm 4	15	**15**	**56**	**51**	32	**18**

AUROC	**AUPRC**
74.1	15.3

There are a few conclusions to be drawn from these two experiments. Clearly, using binary classification out-performs the multi-label setting when it comes to success rate, both for a single seed and for multiple seeds. When comparing RMSDs for a single seed, the binary classification setting again comes out on top. For multiple seeds the performance is similar; however, we should prioritize success rate over RMSD when deciding which setting is the best, since a low RMSD does not necessarily imply a good predicted structure, unless said RMSD is less than 10 (the experimentally agreed-on cutoff for acceptable predicted structure).

Considering that only 9% of the predicted structures have RMSD less than 10, when perfectly seeding with the correct epitope center, we would be limiting ourselves if we used RMSD as our main metric. This low success rate of the perfectly-seeded model is also the reason why we changed the definition of success rate, which normally is defined as the percentage of predicted structures with RMSD to the ground truth of less than 10. Penalizing ourselves based on the limitations of the docking model would give us a false sense of how good our seeding model is.

Top-K Backpropagation

Based on the observations from our previous two experiments, we decided to switch to a binary classification setting for our current experiment. This brings us to our current setting, in which we introduce a second Binary Cross Entropy loss function, that will only back-propagate the top-K ranked residues per sample. To ensure that the remaining residues are also properly trained on, we will use this new loss function in tandem with the original binary classification loss.

The reasoning behind this choice is the fact that we want to maximize how many epitopes or epitope-centers will be in the top-K ranking residues. By introducing this additional loss, we are forcing our model to rank highly residues that are either epitopes, or epitope-centers; if the model were to rank a non-epitope highly, then the additional loss we introduce would further penalize this choice.

Here is how this setting performed in our docking experiments:

Setting	$\text{success}^{(1)}_{\delta=2}$	$\text{success}^{(1)}_{\delta=1}$	$\text{success}^{(10)}_{\delta=2}$	$\text{success}^{(10)}_{\delta=1}$	$\mu^{(1)}_{\textbf{RMSD}}$	$\mu^{(10)}_{\textbf{RMSD}}$
Random Baseline	3	3	35	29	36	20
Seedless Baseline	6	5	-	-	**31**	-
Algorithm 1	**8**	**8**	**48**	**44**	36	**19**
Algorithm 2	5	5	44	42	37	**19**
Algorithm 3	5	3	44	**44**	36	**19**
Algorithm 4	**8**	**8**	47	40	36	**19**

AUROC	**AUPRC**
71.1	14.0

The results of this experiment were quite underwhelming. This is likely due to a combination of factors; first of all, the fact that by introducing the top-K loss, we are incentivising the model to rank K residues on each sample higher than the rest. Secondly, since this is a binary classification model, our model does not discriminate between epitopes and epitope-centers. However, the vast majority of positives in our training and validation sets will be simple epitopes, not epitope-centers. It is therefore a considerable possibility that the majority of the top-K ranking residues, which were correctly identified as positives, will be epitopes, not epitope centers. The docking model is however very sensitive to selecting the correct epitope center, since it has only been trained in that context. Therefore, we need to add back some information that will allow the model to correctly distinguish between residues that are epitopes and residues that are epitope centers.

Multi-label loss combined with binary top-K backpropagation

To address the issue of not selecting epitope centers, we revert back to a multi-label setting; however, when backpropagating the top-K ranked residues, we do so in a binary manner. Given our predicted logits z_1, z_2, z_3, which will be backpropagated normally using the multi-label loss, we define $z_{2,3} = z_2 + z_3$. Then, for each sample, we select the K residues with the highest $z_{2,3}$ score for the additional backpropagation. In that way, our model retains the epitope-center information, while also attempting to rank highly epitopes and epitope centers.

The results of this experiment are shown below.

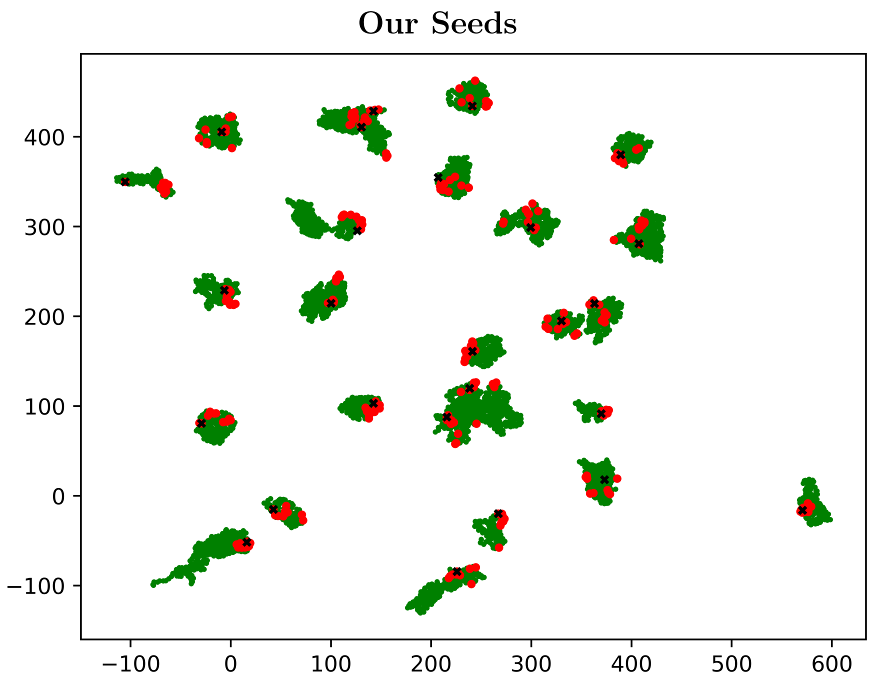
Setting	$\text{success}^{(1)}_{\delta=2}$	$\text{success}^{(1)}_{\delta=1}$	$\text{success}^{(10)}_{\delta=2}$	$\text{success}^{(10)}_{\delta=1}$	$\mu^{(1)}_{\mathbf{RMSD}}$	$\mu^{(10)}_{\mathbf{RMSD}}$
Random Baseline	3	3	35	29	36	20
Seedless Baseline	6	5	-	-	**31**	-
Algorithm 1	14	11	**59**	**53**	34	19
Algorithm 2	6	5	41	35	36	20
Algorithm 3	**21**	**21**	52	48	33	**18**
Algorithm 4	15	9	**59**	47	34	**18**

AUROC	**AUPRC**
72.9	38.0

As you can observe, this setting yields the highest success rate among all other settings we have tried, both for a single seed (21%) and for $K = 10$ seeds (59%). Compared to the random and seedless baselines, this gives us a single seed approach that achieves 3.5 times the success rate and a $K = 10$ seed approach which achieves 1.7 times the success rate. Specifically compared to the seedless approach, after 10 tries our model performs almost 10 times as high as when evaluating on success rate.

An interesting observation can be made by looking at the columns of this table. Notice that the success rate of our model at $K = 10$ seeds, when using Algorithm 1 or Algorithm 4, is 59; almost double the success rate compared to the random baseline. However, if we compare the mean RMSD scores, we notice that they do not differ significantly (19 and 18 on our end vs. 20). The only way that such disparity can be explained is if our model correctly identifies potential docking sites. Of course, our model does not always identify **the** correct docking site, which is why we do not have 100% success. After all, our model was trained only on antigenic sequences, without accounting for the particular antibody we want to dock. On the other hand since the predictions our model makes are limited by the capabilities of the original docking model, the mean RMSD is inevitably going to be relatively high. Recall that we define success rate in terms of how close we are to a perfectly seeded prediction and not the ground truth. This would also explain why the random baseline scores so well in the mean RMSD metric. Since it is randomly selecting residues uniformly, it is bound to cover more area than our model, therefore achieving a decent RMSD score, since it is likely to randomly dock the antibody in close proximity to the true docking site. However, its success rate is going to be significantly lower than that of our model, since it rarely actually locates the correct docking site; from our experiments, we have observed that Wengong's model is very sensitive to the correct epitope center, therefore an attempt to dock somewhere that isn't on the epitope center, or very close to it, will usually end in failure.

To verify these claims, we sampled a subset of 26 antigen structures from our dataset, in which both of our model as well as the random baseline reported fairly good predictions based on the RMSD metric. We specifically chose good predictions to be able to distinguish our model's behavior to that of the random baseline. We ran our seeding algorithm and the random baseline to obtain 20 seeds per antigen, 10 from our model and 10 from the random baseline. Below we have plots of the structures projected into two dimensions, with the seeds marked in red and the true epitope center in black:



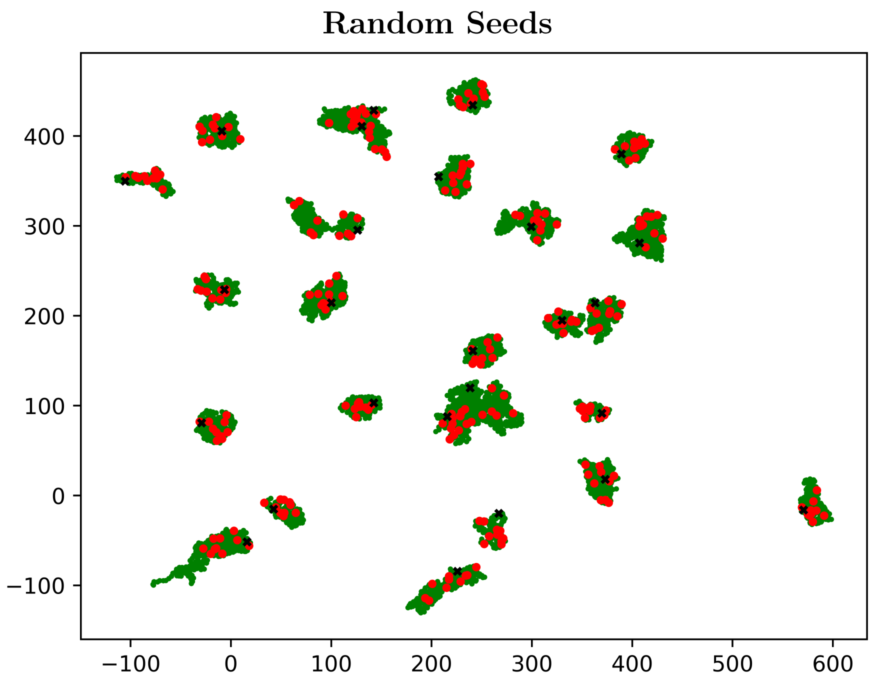


Table 4.1: Our seeds vs. randomly selected seeds in a sample of 26 structures; observe that our seeds cover less space and clump together in sites, as desired.

Observe that our seeds are significantly more localized compared to the random ones. We can therefore conclude that our model is able to identify candidate binding sites, and does not merely produce random seeds. This also explains the discrepancy we noticed between mean RMSD and success rate, when comparing our deterministic seeding method with the random one. This goes to show that, for this particular task, mean RMSD can be a very misleading metric.

4.3.1 Improved Structure Predictions

Our seeding method is not perfect, but it significantly improves the quality of certain predictions. Below we present some randomly sampled structures from the test set, where the improvement in the predicted docking conformation is evident. Note that for both the baseline and the refined prediction, only a single seed was used.

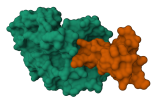
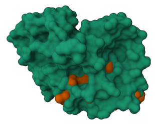
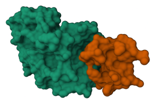
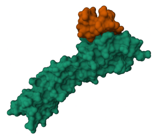
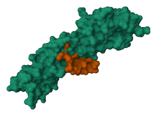
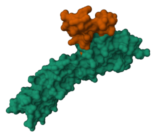
Ground Truth	Baseline Prediction	Our Refined Prediction
		
		

Table 4.2: Single-Seed Comparison

We also compared the baseline with the refined predictions for $K = 10$ seeds. For each model, out of the 10 generated structures per sample, we selected the one with the smallest RMSD with the ground truth crystal structure.

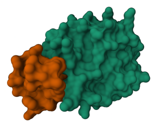
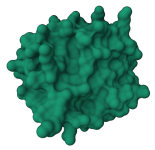
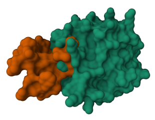
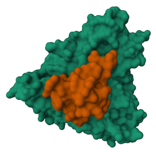
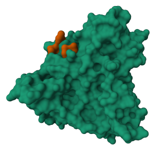
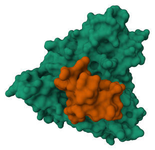
Ground Truth	Baseline Prediction	Our Refined Prediction
		
		

Table 4.3: K-Seed Comparison; notice that in some cases the baseline model places the antibody completely inside the antigen, which is why it is not visible in the visualizations.

Observe that our model correctly identifies the docking site in each case; the deviation from the ground truth, predominantly in the form of incorrect torsion angles and docking orientation, is a failure of the original docking model.

Chapter 5

Conclusions

5.1 Summary of Results

In this thesis we presented a SOTA machine learning architecture for the task of B-cell epitope prediction. Our model outperformed all other models in the current bibliography when using AUROC as our evaluation metric (76), and came a close second when evaluating using AUPRC (24). Our attempts to further increase the performance of our model using linear epitope data from the IEDB database were not successful.

We then used our model to improve the predictive quality of Wengong Jin's antibody docking model. We noticed improved quality in certain structures, and, while our initial evaluation metric, mean-RMSD, improved only slightly, we observed immense performance increase in the success rate of our predictions. Specifically, our model was able, with only 10 tries, to achieve a success rate almost 10 times as high as that of the original seedless model (59 vs. 6), and 1.7 times as high as the setting in which 10 random residues are selected as seeds. This led us to investigating, and we ended up showing that our model is able to correctly identify binding sites, something that is obviously not the case for the random baseline.

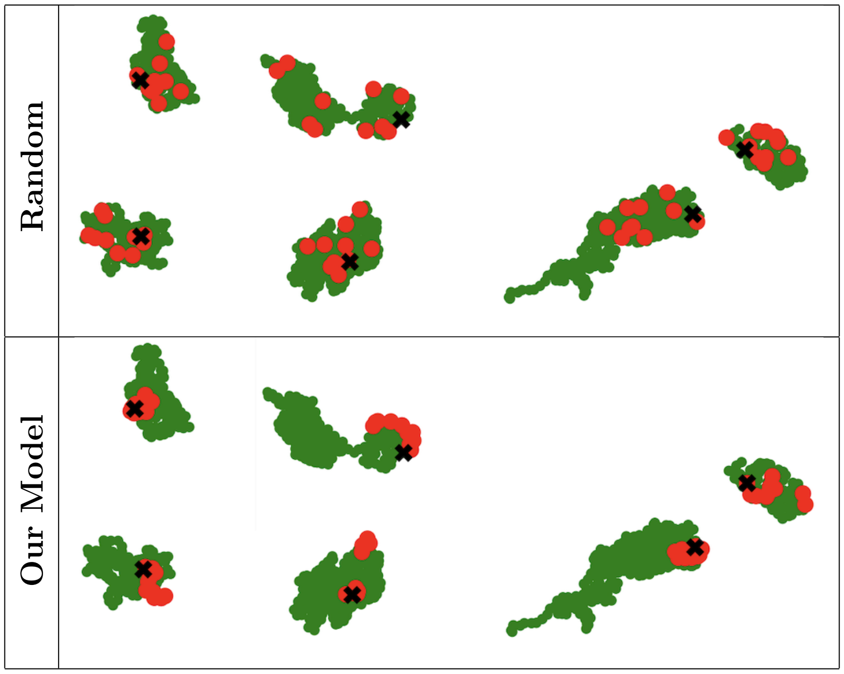


Table 5.1: Some hand-picked structures where the difference in ability to distinguish docking sites between our model and the random baseline is evident.

5.2 Future Work

There are two main areas of the proposed work presented in this thesis that we believe require further investigation.

5.2.1 Incorporating Linear Epitope Data

Our current B-cell epitope model is decently performant, however there is much room for improvement. We have come up with a few ways we can increase the quality of our predictions. In our experiments, we attempted to leverage the linear epitope data from IEDB [33]; we tried using the IEDB data for pretraining, however the distributional shift was too intense, and we ended up observing a decrease in performance. We do believe, nonetheless, that with the proper pre-processing, we should be able to utilize the IEDB data; since this task is very data-sensitive, and the availability of reliable, large-scale datasets we can use is extremely limited, we believe that managing to incorporate in a clever way the IEDB data into our training routine will certainly help with performance.

To be able to incorporate the IEDB data in our training data, however, we need to address the distributional shift between the linear epitopes in IEDB and the epitopes in our dataset. One way to do that would be to collect all the antigen sequences in IEDB and locate all antigen-antibody complexes in which they participate. This part will be tricky, since there are antigens in IEDB which form complexes that cannot be found in the publicly available datasets we have mentioned so far. To retrieve these complexes, we could look into the (publicly available) dataset of AlphaFold predicted structures, or UniProt [5].

After retrieving these structures, we can label them in the way described in section 3.3.2, by marking residues as epitopes when they are in close proximity to some antibody CDR. That should be enough to take care of the difference in distribution, and hopefully incorporating the re-labeled IEDB entries in our training set will help us increase our model's performance.

5.2.2 Using Our Model to Improve AlphaFold Predictions

During our experiments, we used a docking model that was trained to be able to accept a candidate binding site as bias for predicting the final conformation. However, the state of the art in antibody docking models, AlphaFold Multimer, does not offer this type of functionality. We would like to experiment with "manually" seeding AlphaFold Multimer using our model's predictions, by tweaking the template library AlphaFold uses. One way to do that, would be to use our seeding model, in tandem with a docking model that is trained to accept candidate docking sites, to produce a set of possible structures. Then, we will inject these structures into the AlphaFold template collection and run the model to produce a refined prediction, or set of predictions (since AlphaFold can produce multiple candidate docking and folding conformations). Then, using AlphaFold's uncertainty scoring, we will be able to select the most likely correct structure.

Such an experiment has a lot of technical challenges and requires a deep understanding of the underlying AlphaFold architecture, however it certainly has potential.

Bibliography

[1] Stephen K Burley, Charmi Bhikadiya, Chunxiao Bi, Sebastian Bittrich, Li Chen, Gregg V Crichlow, Cole H Christie, Kenneth Dalenberg, Luigi Di Costanzo, Jose M Duarte, Shuchismita Dutta, Zukang Feng, Sai Ganesan, David S Goodsell, Sutapa Ghosh, Rachel Kramer Green, Vladimir Guranović, Dmytro Guzenko, Brian P Hudson, Catherine L Lawson, Yuhe Liang, Robert Lowe, Harry Namkoong, Ezra Peisach, Irina Persikova, Chris Randle, Alexander Rose, Yana Rose, Andrej Sali, Joan Segura, Monica Sekharan, Chenghua Shao, Yi-Ping Tao, Maria Voigt, John D Westbrook, Jasmine Y Young, Christine Zardecki, and Marina Zhuravleva. RCSB Protein Data Bank: powerful new tools for exploring 3D structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic Acids Research*, 49(D1):D437–D451, 11 2020.

[2] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[3] Joakim Clifford, Magnus Haraldson Høie, Morten Nielsen, Sebastian Deleuran, Bjoern Peters, and Paolo Marcatili. Bepipred-3.0: Improved b-cell epitope prediction using protein language models. *bioRxiv*, 2022.

[4] Maximilian Collatz, Florian Mock, Emanuel Barth, Martin Hölzer, Konrad Sachse, and Manja Marz. EpiDope: a deep neural network for linear B-cell epitope prediction. *Bioinformatics*, 37(4):448–455, 09 2020.

[5] The UniProt Consortium. UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531, 11 2022.

[6] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking, 2022.

[7] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176, mar 1964.

[8] Wikimedia Foundation. Macromolecular docking. `https://en.wikipedia.org/wiki/Macromolecular_docking`, 2022.

[9] Octavian-Eugen Ganea, Xinyuan Huang, Charlotte Bunne, Yatao Bian, Regina Barzilay, Tommi S. Jaakkola, and Andreas Krause. Independent se(3)-equivariant models for end-to-end rigid protein docking. *CoRR*, abs/2111.07786, 2021.

[10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. 2017.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[12] Andrew C. Hunt, James Brett Case, Young-Jun Park, Longxing Cao, Kejia Wu, Alexandra C. Walls, Zhuoming Liu, John E. Bowen, Hsien-Wei Yeh, Shally Saini, Louisa Helms, Yan Ting Zhao, Tien-Ying Hsiang, Tyler N. Starr, Inna Goreshnik, Lisa Kozodoy, Lauren Carter, Rashmi Ravichandran, Lydia B. Green, Wadim L. Matochko, Christy A. Thomson, Bastain Vögeli, Antje Krüger-Gericke, Laura A. VanBlargan, Rita E. Chen, Baoling Ying, Adam L. Bailey, Natasha M. Kafai, Scott Boyken, Ajasja Ljubetič, Natasha Edman, George Ueda, Cameron Chow, Amin Addetia, Nuttada Panpradist, Michael Gale, Benjamin S. Freedman, Barry R. Lutz, Jesse D. Bloom, Hannele Ruohola-Baker, Sean P. J. Whelan, Lance Stewart, Michael S. Diamond, David Veesler, Michael C. Jewett, and David Baker. Multivalent designed proteins protect against sars-cov-2 variants of concern. *bioRxiv*, 2021.

[13] Martin Closter Jespersen, Bjoern Peters, Morten Nielsen, and Paolo Marcatili. BepiPred-2.0: improving sequence-based B-cell epitope prediction using conformational epitopes. *Nucleic Acids Research*, 45(W1):W24–W29, 05 2017.

[14] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Antibody-antigen docking and design via hierarchical equivariant refinement, 2022.

[15] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[16] Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference*

on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[18] Jens Vindahl Kringelum, Claus Lundegaard, Ole Lund, and Morten Nielsen. Reliable b cell epitope predictions: Impacts of method development and improved benchmarking. *PLoS Computational Biology*, 8(12), 2012.

[19] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[20] Peter J. Rousseeuw Leonard Kaufman. *Partitioning Around Medoids (Program PAM)*. John Wiley & Sons, Ltd, 1990.

[21] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

[22] Sepideh Parvizpour, Mohammad M. Pourseif, Jafar Razmara, Mohammad A. Rafi, and Yadollah Omidi. Epitope-based vaccine design: a comprehensive overview of bioinformatics approaches. *Drug Discovery Today*, 25(6):1034–1042, 2020.

[23] Julia Ponomarenko, Huynh-Hoa Bui, Wei Li, Nicholas Fusseder, Philip E. Bourne, Alessandro Sette, and Bjoern Peters. Ellipro: a new structure-based tool for the prediction of antibody epitopes. *BMC Bioinformatics*, 9(1):514, 2008.

[24] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. 2021.

[25] Constantin Schneider, Matthew I J Raybould, and Charlotte M Deane. SAbDab in the age of biotherapeutics: updates including SAbDab-nano, the nanobody structure tracker. *Nucleic Acids Research*, 50(D1):D1368–D1372, 11 2021.

[26] Thomas Shafee. *File:Alpha beta structure (full).png*. Wikipedia, the Free Encyclopedia, Feb 2017.

[27] Jamie B. Spangler, Jakub Tomala, Vincent C. Luca, Kevin M. Jude, Shen Dong, Aaron M. Ring, Petra Votavova, Marion Pepper, Marek Kovar, and K. Christopher Garcia. Antibodies to interleukin-2 elicit selective t cell subset potentiation through distinct conformational mechanisms. *Immunity*, 42(5):815–825, 2015.

[28] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, 2017.

[29] Peter D. Sun, Christine E. Foster, and Jeffrey C. Boyington. Overview of protein structural and functional folds. *Current Protocols in Protein Science*, 35(1):17.1.1–17.1.189, 2004.

[30] Jérôme Tubiana, Dina Schneidman-Duhovny, and Haim J. Wolfson. Scannet: an interpretable geometric deep learning model for structure-based protein binding site prediction. *Nature Methods*, 19(6):730–739, 2022.

[31] The Center for Molecular Life Sciences University of Basel. *Peptide torsion angles.* BIOZENTRUM.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[33] Randi Vita, James A. Overton, Jason A. Greenbaum, Julia Ponomarenko, Jason D. Clark, Jason R. Cantrell, Daniel K. Wheeler, Joseph L. Gabbard, Deborah Hix, Alessandro Sette, and Bjoern Peters. The immune epitope database (IEDB) 3.0. *Nucleic Acids Research*, 43(D1):D405–D412, 10 2014.

[34] Wikipedia. *Basic B cell function: bind to an antigen, receive help from a cognate helper T cell, and differentiate into a plasma cell that secretes large amounts of antibodies.* Wikipedia, the Free Encyclopedia, 2023.

[35] Wikipedia. *Sketch of an antibody with the variable domains shown in blue, and the CDRs (which are part of the variable domains) in light blue.* Wikipedia, the Free Encyclopedia, 2023.

[36] Yuansong Zeng, Zhuoyi Wei, Qianmu Yuan, Sheng Chen, Weijiang Yu, Yutong Lu, Jianzhao Gao, and Yuedong Yang. Identifying b-cell epitopes using alphafold2 predicted structures and pretrained language model. *bioRxiv*, 2022.

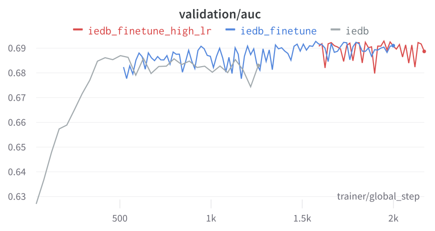
Appendix A

Implementation Details

In this appendix section, we will briefly give some implementation details about our models, and show some training/validation plots. To implement the machine learning architecture described in this thesis, we used PyTorch (version 1.11) with the PyTorch Lightning wrapper (version 1.6.2). For our language models embeddings we used ESM2 ("bleeding edge" version) by Facebook. Our models were trained on 8 NVIDIA RTX A6000 GPUs, each with 48 GB of GDDR6 memory.

A.1 IEDB Experiments

We extensively experimented with using the linear epitopes from the IEDB database for a pretraining routine. Our idea was to train a model using IEDB as the training set and evaluate it on our independent validation set. We would then pick the training checkpoint that scores highest on metrics such as AUROC and AUPRC, and use the model weights from that particular checkpoint as a starting point for a fine-tuning run on the original training set. Below we present the validation curves of this experiment. The noisiness that can be observed on the finetuning curves is due to the fact that the original training dataset was much smaller than the IEDB dataset, therefore gradient updates occurred more frequently.



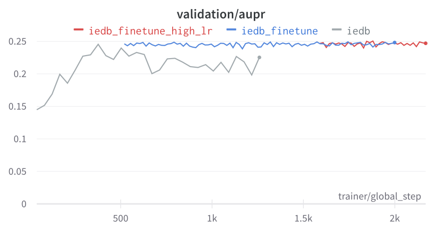


Table A.1: Training/Validation Curves

Notice that both when using a small learning rate (10^{-5}) as well as when using a larger learning rate (5×10^{-4}), the performance of the finetuned model on the validation set is quite underwhelming when compared with the performance of the original model (see next subsection of the Appendix). This is most likely due to the distributional difference between the IEDB data and the the data we used, which was obtained from SAbDab; it is likely that residues marked as epitopes on proteins within SAbDab are marked as non-epitopes in similar proteins within IEDB.

We also tried a similar setting, where instead of using the IEDB data for pre-training, we instead injected them into our training set, and flagged all the samples depending on whether they were a part of the original training set or IEDB. This flag was one-hot encoded and concatenated with the rest of the protein embeddings that were fed into the EGNN, and it was also used to determine the backpropagation learning rate for each individual sample; we wanted IEDB samples to have a lower learning rate, such that in case of conflicting information the original training set labels would prevail. This setting however, albeit creative, also underperformed compared to simply not using IEDB at all, likely due to the same reasons mentioned above.

A.2 Our Best Epitope Model

Below we present the training/validation curves of the reported 76 AUROC B-cell epitope model.

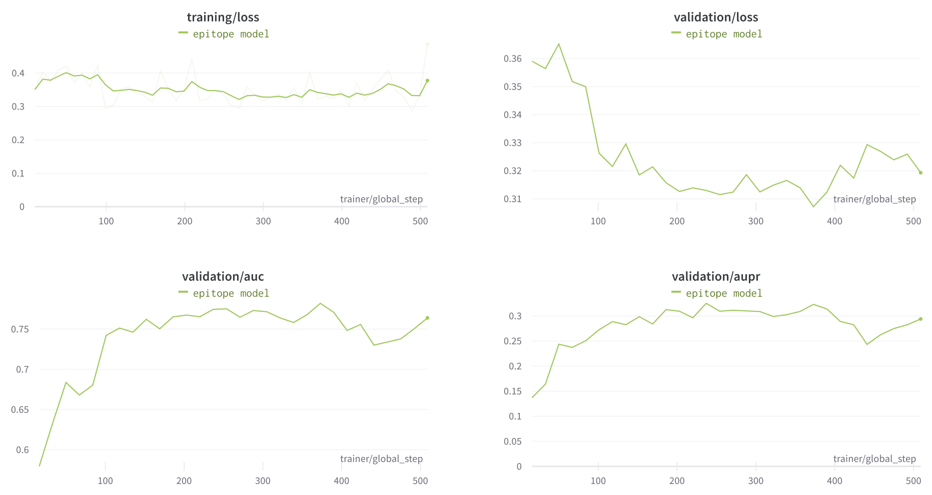


Table A.2: Training/Validation Curves

For testing our model on the independent dataset, we selected the model checkpoint with the highest AUPRC score.

A.3 Our Best Seeding Model

We also trained additional models on Wengong's training set. Here we present the training/validation curves for the most performant of these models, namely the one trained in the multi-label cross entropy loss with the additional top-K binary cross entropy loss setting (see section 4.3).

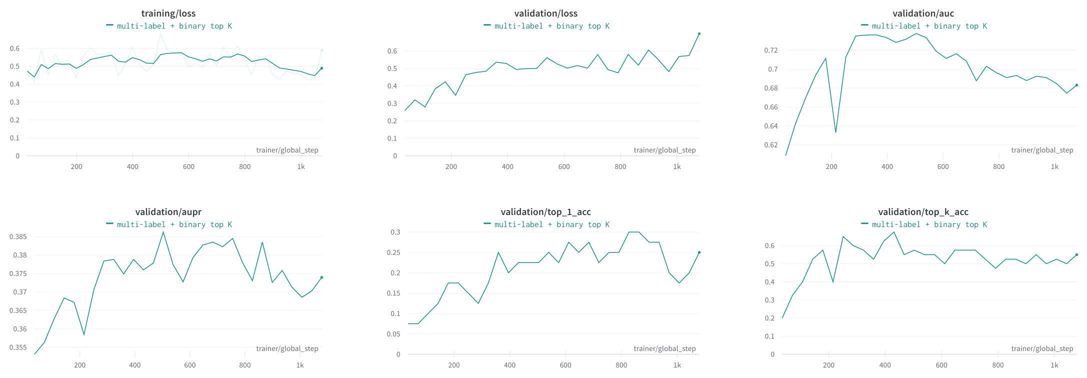


Table A.3: Training/Validation Curves

Here, we are introducing two additional metrics to help us with checkpointing. Top-1 accuracy, that is, the percentage of samples where the top-ranked residue was either an epitope or an epitope-center, and top-K accuracy, that is, the percentage of samples where at least one of the top-K ranked residues was an epitope or an epitope-center. For the test set, we ended up selecting the model checkpoint with the highest top-K accuracy reported on the validation set, since our main focus was trying to improve the performance after using multiple seeds rather than just a single one.

Observe that the validation loss plot in this setting is much more noisy compared to the simple binary classification setting. This makes sense, since the additional loss in conjunction with the limited amount of training data samples makes it difficult for the model to reach a training optimum without overfitting.