Technische Universität Berlin
School IV - Electrical Engineering and Computer Science
Department of Computer Engineering and Microelectronics
Robotics and Biology Laboratory

Bachelor Thesis

Convolutional Neural Networks for the prediction of breaking contacts during protein motion

presented by

Lukas Sebastian Hönig
Matr.-Nr.: 355541
l.hoenig@campus.tu-berlin.de
Informatik

Date of submission: **14.10.2021**
Examiners: **Prof. Dr. Oliver Brock, Prof. Dr. Klaus-Robert Müller**
Advisors: **Prof. Dr. Oliver Brock, Kolja Stahl, Dr. Ines Putz**

Declaration

I hereby declare in lieu of an oath that I have produced this work by myself. All used sources are listed in the bibliography and content taken directly or indirectly from other sources is marked as such.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den

Unterschrift

Abstract

The diverse functions of proteins are closely related to their motions. Revealing and analyzing protein dynamics based on protein structure is therefore a central interest of proteomics. Elastic Network Models (ENMs) are computationally inexpensive harmonic models of coarse-grained protein dynamics. Despite their simplicity, the protein motions they predict are of surprising biological relevance. A recently introduced ENM variant called *lmc*ENM aims to improve ENMs by allowing them to capture noncollective, local motions related to ligand binding that were previously inaccessible to them. These motions are called localized functional transitions. This is achieved by identifying and removing certain connections called *breaking contacts* from the ENM network. To this end, a Support Vector Machine (SVM) is used, but its classification performance is still low.

In this thesis, I construct and evaluate a novel Deep Learning-based breaking contact predictor to be used as a replacement for the SVM in *lmc*ENM. The model is a Convolutional Neural Network, a common Deep Learning architecture that operates on image data. I show that the model mostly reaches a level of performance comparable to the SVM, but without depending on complex hand-crafted graph features. Even though it does not work equally well for all motion types, the model clearly surpasses the SVM in terms of classification performance.

I also propose and implement various other improvements to the *lmc*ENM method, for example extending the method to multichain proteins.

I close with a discussion on future applications of the combination of Deep Learning and ENMs, of which this thesis is the first exploration.

Zusammenfassung

Titel:
Faltende neuronale Netzwerke zur Vorhersage von brechenden Kontakten bei Proteinbewegungen

Die diversen Funktionen von Proteinen hängen eng mit ihren Bewegungen zusammen. Das Aufdecken und Analysieren von Proteindynamik ist daher ein zentrales Interesse der Proteomik. Elastische Netzwerkmodelle (ENMs) sind rechnerisch günstige harmonische Modelle grobaufgelöster Proteindynamik. Trotz ihrer Einfachheit haben die von ihnen vorhergesagten Proteinbewegungen überraschende biologische Relevanz. Eine kürzlich eingeführte ENM-Variante namens *lmc*ENM versucht ENMs dadurch zu verbessern, ihnen zu ermöglichen bislang unzugängliche nichtkollektive, lokalisierte Bewegungen im Zusammenhang mit der Bindung von Liganden abzubilden. Diese Bewegungen heißen lokalisierte funktionale Transitionen. Dies wird erreicht, indem bestimmte Verbindungen, genannt *brechende Kontakte*, identifiziert und aus dem ENM-Netzwerk entfernt werden. Zu diesem Zweck wird eine Stützvektormaschine (SVM) eingesetzt, deren Klassifikationsgenauigkeit allerdings noch gering ist.

In dieser Arbeit konstruiere und evaluiere ich ein neuartiges Deep Learning-basiertes Vorhersagemodell für brechende Kontakte, welches als Ersatz für die SVM in *lmc*ENM eingesetzt werden kann. Das Modell ist ein faltendes neuronales Netzwerk (CNN), eine häufige Deep Learning-Architektur, die auf Bilddaten operiert. Ich zeige, dass das Modell meist ein Leistungsniveau erreicht, welches mit der SVM vergleichbar ist, allerdings ohne auf komplexe, handgemachte Graph-Eigenschaften angewiesen zu sein. Auch wenn es nicht für alle Bewegungstypen gleich gut funktioniert, übertrifft es die Klassifikationsleistung der SVM klar.

Ausserdem schlage ich verschiedene Verbesserungen der *lmc*ENM-Methode allgemein vor und implementiere sie, zum Beispiel die Erweiterung auf mehrkettige Proteine.

Ich schliesse mit einer Diskussion von zukünftigen Anwendungen der Kombination von Deep Learning und ENMs, von welchen diese Arbeit die erste ist.

Contents

List of Figures

List of Tables

List of Acronyms

_lmc_ENM . . . . . ENM of learned maintained contacts
_mc_ENM . . . . . ENM of observed maintained contacts
AA . . . . . Amino Acid
ANM . . . . . Anisotropic Network Model
ANN . . . . . Artificial Neural Network
APC . . . . . Average Product Correction
ASAM . . . . . Adaptive Sharpness-Aware Minimization
AUROC . . . . . Area under the ROC curve
BC . . . . . Breaking Contact
BCP . . . . . Breaking Contact Prediction
BLM . . . . . Burying Ligand Motion
CDM . . . . . Coupled Domain Motion
CLM . . . . . Coupled Local Motion
CNN . . . . . Convolutional Neural Network
CV . . . . . Cross Validation
DI . . . . . Direct Information
DL . . . . . Deep Learning
ELU . . . . . Exponential Linear Unit
ENM . . . . . Elastic Network Model
ERF . . . . . Effective Receptive Field
GDT . . . . . Global Distance Test
GNM . . . . . Gaussian Network Model
GPCR . . . . . G Protein-Coupled Receptor
IDM . . . . . Independent Domain Motion
ILM . . . . . Independent Local Motion
ING . . . . . Immediate Neighbourhood Graph
LOOCV . . . . . Leave-one-out Cross Validation

MD . . . . . Molecular Dynamics

MI . . . . . Mutual Information

ML . . . . . Machine Learning

MLP . . . . . Multilayer Perceptron

MSA . . . . . Multiple Sequence Alignment

NMA . . . . . Normal Mode Analysis

NMR . . . . . Nuclear Magnetic Resonance

NSM . . . . . No Significant Motion

OTM . . . . . Other Types of Motion

PDB . . . . . Protein Data Bank

PSCDB . . . . . Protein Structural Change Database

rASA . . . . . Relative Solvent Accessible Surface Area

RMSD . . . . . Root-mean-square Deviation

ROC . . . . . Receiver Operating Characteristic

SASA . . . . . Solvent Accessible Surface Area

SGD . . . . . Stochastic Gradient Descent

SS . . . . . Secondary Structure

SSE . . . . . Secondary Structure Element

SVM . . . . . Support Vector Machine

1 Introduction

Proteins are among the most important biomolecules. Just like the genetic material encoding them, they are essential to life. They are found in every cell and perform a great variety of functions, from giving structural support, catalyzing very specific chemical reactions and detecting antigens to allowing muscles to contract and information to propagate in the brain. A great variety of functions can be encoded using sequences of the 20 proteinogenic amino acids [3]. This one-dimensional amino acid sequence determines a proteins' native three-dimensional shape to a great extent, a principle known as Anfinsens' dogma [6].

But imagining proteins as existing in a single native conformation is not completely realistic. At reasonable temperatures, proteins vibrate, jiggle and move around constantly, and so the native state of a protein is more accurately described as a *conformational ensemble* around the free energy minimum [33].

Folded proteins not only vibrate randomly around the energy minimum, but have characteristic motions associated, local and global, that are integral to their function [10, 59]. In a large majority of cases, these motions are encoded intrinsically in their structure [114]. Binding of interaction partners is also central for proteins to carry out their functions. Binding usually results in conformational changes by shifting the energetically optimal region of conformational space ("conformational selection mechanism") [38, 120]. Only rarely does ligand presence actually change the accessible conformational space itself ("induced fit mechanism") [66, 114].

The motions associated with binding are called functional transitions. Examples are receptors changing shape upon small molecule binding [72] (see Figure 1.2), transporters pumping ions across the cellular membrane [111], and individual steps of the sequence of motions allowing biomolecular machines like kinesin to walk along microtubules [15]. Another example involving the therapeutically important HIV protease enzyme is shown in Figure 1.1.

Protein motion is normally studied using experimental methods like X-ray crystallography or NMR (nuclear magnetic resonance), but these have problems attached: the insight into protein dynamics they can provide is limited, they are costly, time-consuming, and may not work for some proteins of interest [27, 67, 89, 91, 127].

A different approach altogether is to perform *in silico* experiments. The most common way to study protein dynamics computationally are molecular dynamics (MD) simulations [18, 48, 60, 90, 101]. The small time steps required for sufficient accuracy in protein settings combined with the size of macromolecular systems and

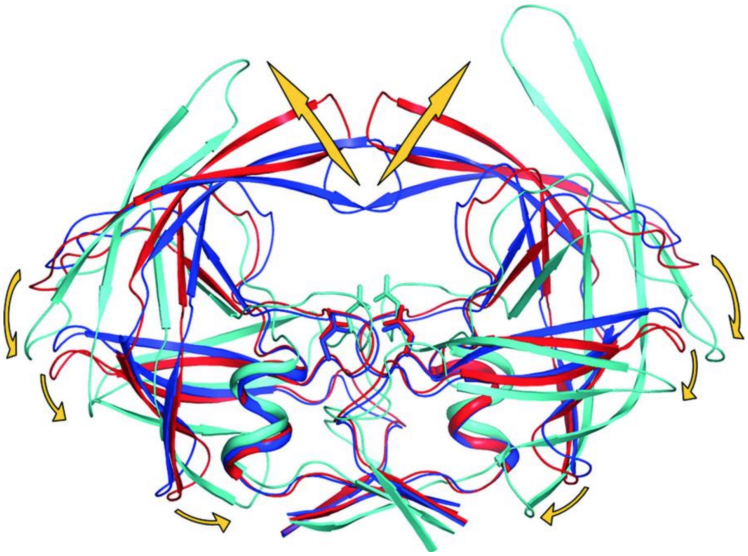

Figure 1.1: First example of a functional transition: Flap opening and closing is essential for the function of HIV protease. Three major forms are known: *closed*, *semiopen*, and *open*. In an unrestrained MD simulation devoid of any ligand, the semiopen conformation was most commonly observed, with the open and closed state also being part of the ensemble. A ligand stabilized the closed form. This provides *in silico* evidence for conformational selection theory. Blue: closed, red: semiopen, cyan: open conformation, Figure from [49]

their comparatively long relevant timescales lead to an enormous computational demand that can rarely be satisfied.

An alternative to MD simulations are coarse-grained methods, with the Elastic Network Model (ENM) being the prime example. ENMs model proteins as a network of point masses representing residues and distance-bounded spring connections between them. Protein dynamics are then given by Normal Mode Analysis (NMA) of this system, a technique from physics yielding patterns of motions inherent to oscillating systems. Despite being highly reduced models, ENMs predict intrinsic protein dynamics of surprising biological relevance [2, 10, 29, 68, 118].

On the other hand, owing to their simplicity and focus on coarse topology, ENMs only really work well for global, highly collective motions. They fail to capture localized or uncorrelated functional transitions, caused by their built-in assumption that the contact topology does not change significantly between functional conformations [16, 85, 97, 118, 133].

This limitation of ENMs was recently addressed by Putz and Brock in "Leveraging Novel Information for Coarse-Grained Prediction of Protein Motion" [99, 100]. They found that capturing of localized functional motions by ENMs can be greatly improved by removing springs in certain parts of the protein that are especially

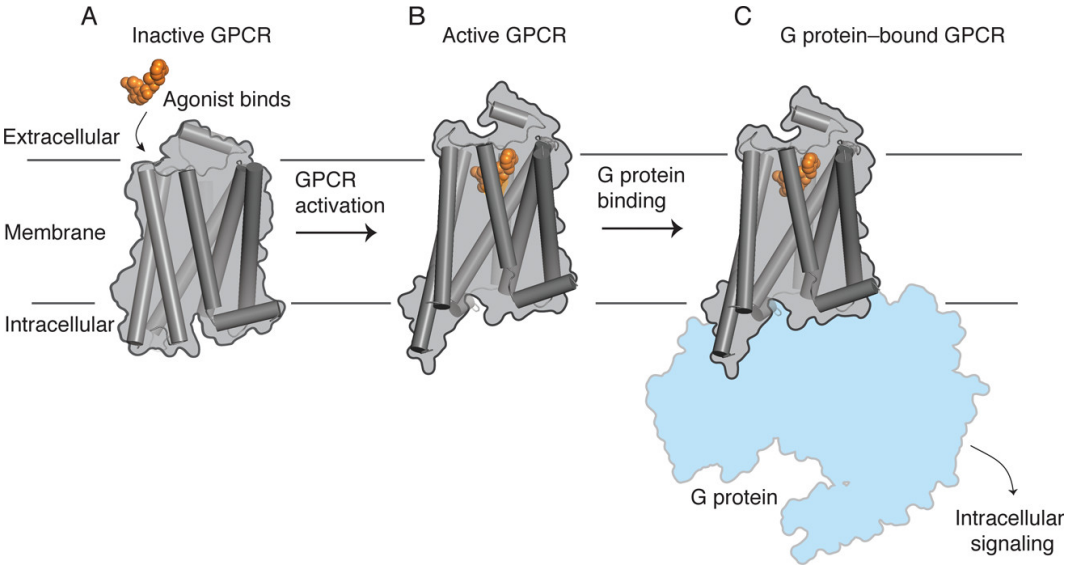

Figure 1.2: Second example of a functional transition: a ligand binds to a G protein-coupled receptor (GPCR), causing the receptor to adopt a new conformation. This in turn allows G protein binding and subsequently intracellular signaling to take place. Figure from [72]

mobile during functional transitions, so called *breaking contacts*. This results in a network that is locally more flexible in these important regions.

Because in many cases there is only an unbound structure available but not a bound one, the paper is also about building a machine learning based classifier to predict breaking contacts from only the unbound structure. This classifier is based on a Support Vector Machine (SVM). By removing predicted breaking contacts from the ENM, they arrive at a new ENM variant, called *lmc*ENM (Elastic Network Model of learned maintained contacts). While the SVM succeeds in predicting breaking contacts that lead to improved ENMs, there is still much room for improvement compared to the ground truth: the SVM reaches about half of the improvement given by the ground truth, which itself is also not a hard limit on possible ENM performance.

1.1 Contributions

In this thesis, I present a novel classifier to predict protein breaking contacts based on a ResNet, which is a type of Convolutional Neural Network model (CNN). This classifier can serve as a replacement for the SVM used in *lmc*ENM. CNNs are the standard deep learning model in computer vision and are also often used in computational structural biology [1, 77, 80, 125, 131]. I model the breaking contact

prediction problem accordingly and argue that the inductive biases of CNNs make them a good fit. Using a CNN especially allows to forgo complex hand-crafted graph features used in *lmc*ENM and to enlarge the sequential context available for prediction. Additionally, I increase the dataset size, partly by expanding my method to multichain proteins. Furthermore, I introduce some novel features, showing that they are useful for breaking contact prediction. I also introduce the concept of breaking contact relevance heuristics and apply it to the *lmc*ENM pipeline, leading to improved ENM results. To the best of my knowledge, the present work is the first application of Deep Learning to Elastic Network Models.

1.2 Thesis structure

Chapter 2 explains the theory of ENMs and machine learning necessary to understand the model developed in this thesis and its context. Chapter 3 explains the details of how the reference method *lmc*ENM [99] works, which my model is based on. Additionally, I discuss some related variants of the Elastic Network Model and how they relate to *lmc*ENM and my method.

In chapter 4, I present the model developed in this thesis along with the new dataset used to train it. I discuss the experiments performed while building the model, their results and implications in chapter 5. I draw a conclusion in chapter 6, and in chapter 7, I present ideas and directions for future work.

2 Background

2.1 Elastic Network Models

Elastic Network Models (ENMs) are coarse-grained methods to study and predict intrinsic protein dynamics. They predict thermal fluctuations of proteins around their equilibrium conformation. ENMs represent proteins by a network of point masses and spring connections between them ("mass-and-spring network"). Only the C_α-atoms of amino acids are considered as point masses and each point mass is connected to all its spatial neighbours within a fixed cutoff radius.

After constructing the network, **Normal Mode Analysis (NMA)** is applied, yielding the *normal modes* of the system. Normal modes are special intrinsic motions of systems. They are orthogonal to each other and can move independently. In a normal mode, all parts of a system move sinusoidal. Physical systems have exactly as many normal modes as they have degrees of freedom. The normal modes of a system are the eigenvectors of the system of equations formed by its equations of motion. Therefore, to perform NMA of a protein, one has to solve an eigenvalue problem, which in turn depends on the potential function used (the "force field"). Each normal mode is an eigenvector and the associated eigenvalue represents the energy or frequency of the motion. These eigenvalues are called *eigenfrequencies*. The lower the eigenfrequency, the more dominant, energetically accessible and global the mode is. All normal modes together span the entire deformation space of the protein, but often, a few low-frequency modes are enough to account for functionally relevant motions of the protein [2, 9, 29, 68, 70, 104, 106, 118]. NMA as used in ENMs is based on the **harmonic hypothesis**: the potential energy landscape of a protein - despite being locally highly rugged - can be approximated by a parabola.

Tirion introduced the first ENM in 1996, replacing the complex semiempirical MD force fields used in protein NMA up to this point with a single-parameter Hookean potential [119]. She then showed that it still reproduced the slow dynamics predicted by standard NMA well. With this new potential, no initial energy minimization was needed anymore, as the experimentally determined starting conformation is just defined to be at a local energy minimum. The computational demand of solving the NMA eigenvalue problem was also greatly reduced. This made protein NMA tractable, even for large proteins.

The assumption that a simple quadratic potential, like the one introduced by Tirion, suffices to usefully approximate intrinsic dynamics of proteins, is central to

ENMs. The second assumption made by ENMs is that these dynamics are robustly encoded by the coarse-grained geometry of the protein [7, 8, 42, 47]. ENMs are even robust to variations in formalism [76].

While many functionally relevant motions can be predicted using ENMs, their simplicity is also the source of their most important limitation: ENMs fail to capture localized functional transitions, that is, functional transitions with a low degree of collectivity [85, 133]. The uniformity of springs and indiscriminate connectivity leads to a network that is overconstrained, preventing the capturing of local motions.

Shortly after Tirion's groundbreaking paper, the **Gaussian Network Model (GNM)** variant of ENMs was introduced [8, 42], which reduced the resolution of the network from all-atom to residue level, using only the C_α-atoms. Furthermore, the GNM assumes the thermal fluctuations to be normally distributed.

In 2001, the important **Anisotropic Network Model (ANM)** was first published [7, 118]. In the case of the GNM, the normal modes are vectors containing for each point mass a scalar displacement value, analogous to experimental B-factors (see Subsection 5.5.2). ANM normal modes on the other hand consist of a displacement *vector* for each point mass. GNM modes only give magnitudes of motion for each residue, whereas ANM modes additionally give a *direction*, see Figure 2.1.

ANMs are widely used today. An overview of their applications is given in [84]. These include, among others: exploring intrinsic dynamics of large assemblies of biomolecules [86], interpolating between conformations, structural refinement, docking, evolutionary conservation, and guiding MD simulations. Additionally, *lmc*ENM, which my method is based on, is essentially a topologically sparsified ANM. Therefore, I will now discuss some of the details of how ANMs work. For a more in-depth mathematical derivation, see [7, 21, 99, 119].

2.1.1 Anisotropic Network Model

The ANM combines the residue-level resolution of the GNM with Tirion's original quadratic harmonic potential [7, 118]. The network nodes of an ANM are formed by the protein's C_α-atoms. Edges exist between all nodes whose spatial distance is within a predetermined cutoff value d_c. This parameter can be chosen on a per-protein level. Smaller cutoff values are generally desirable as they lead to a more flexible network, improving capturing of non-collective motions. Care has to be taken however, as a cutoff value that is too small will destabilize the network [55].

Edges represent springs with a uniform stiffness value. More sophisticated ANM variants exist that vary the stiffness value based on different types of additional information (see Section 3.2).

The total network potential is defined as the sum of all pairwise potentials between residues i and j:

$$V_{\mathrm{ANM}} = \sum_{i,j}^{L} \frac{K_{ij}}{2}(d_{ij}^1 - d_{ij}^0)^2 \tag{2.1}$$

where L is the number of residues of the protein, d_{ij}^1 and d_{ij}^0 are the instantaneous and equilibrium (i.e. initial) distances, and K_{ij} is an entry of the stiffness matrix, which is defined for residues i, j as

$$K_{ij} = \begin{cases} \gamma, & d_{ij}^0 \leq d_c \\ 0, & \text{otherwise} \end{cases}. \tag{2.2}$$

Here, γ is the spring stiffness constant. Typically, it is chosen as $\gamma = 1$.

The force constants of the system are captured in the Hessian H of the potential, the matrix of second partial derivatives. Reduced to residue-level, a protein in three-dimensional space has $3L$ degrees of freedom, x, y, z for each atom. Therefore the Hessian is in total a $3L \times 3L$ matrix organized in L^2 submatrices of size 3×3 ("superelements"), each describing the interaction between two C_α-atoms:

$$H = \begin{bmatrix} H_{1,1} & H_{1,2} & \dots & H_{1,L} \\ H_{2,1} & H_{2,2} & \dots & H_{2,L} \\ \vdots & \vdots & \ddots & \vdots \\ H_{L,1} & H_{L,2} & \dots & H_{L,L} \end{bmatrix} \tag{2.3}$$

with the superelements given by

$$H_{ij} = \begin{bmatrix} \frac{\partial^2 V}{\partial x_i \partial x_j} & \frac{\partial^2 V}{\partial x_i \partial y_j} & \frac{\partial^2 V}{\partial x_i \partial z_j} \\ \frac{\partial^2 V}{\partial y_i \partial x_j} & \frac{\partial^2 V}{\partial y_i \partial y_j} & \frac{\partial^2 V}{\partial y_i \partial z_j} \\ \frac{\partial^2 V}{\partial z_i \partial x_j} & \frac{\partial^2 V}{\partial z_i \partial y_j} & \frac{\partial^2 V}{\partial z_i \partial z_j} \end{bmatrix}. \tag{2.4}$$

Normal Mode Analysis then needs to invert the Hessian to yield the covariance matrix of a $3L$-variate Gaussian distribution [21]. This distribution contains the desired information about the thermal fluctuations around the experimentally defined equilibrium. But H does not have full rank: six variables correspond to rigid body motions of the protein (three translations, three rotations). These modes have an eigenfrequency of zero. Instead, a pseudoinverse is obtained by eigendecomposition, resulting in $3L - 6$ nontrivial eigenvectors and eigenvalues representing the normal modes and their frequencies.

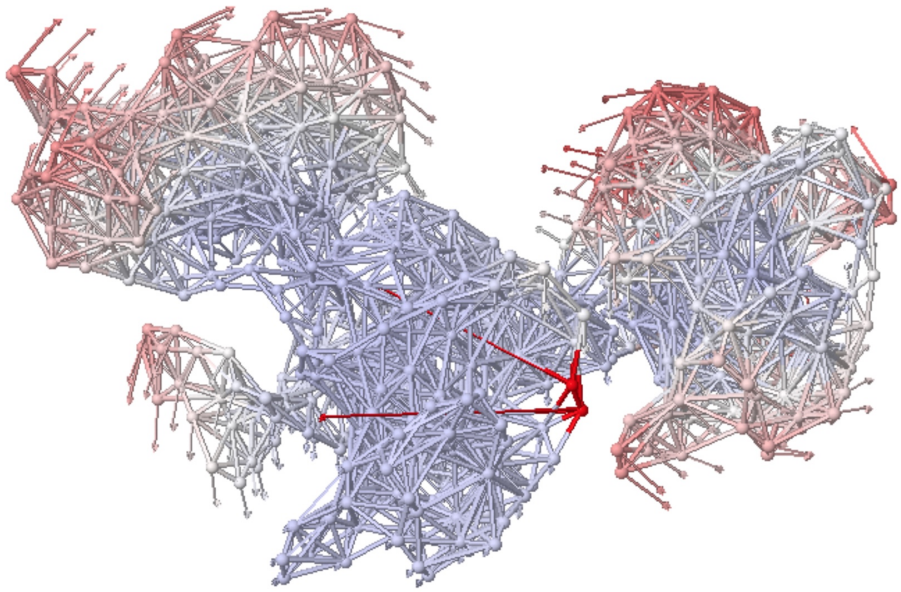

Figure 2.1: Visualization of ANM network of PDB entry 1tup showing nodes, edges and displacement vectors for the lowest-frequency normal mode. Cutoff distance is 8Å. Color indicates norm of displacement vectors, from blue over white to red. Image generated using the ANM webserver at `http://anm.csb.pitt.edu/`

2.2 Machine Learning

The model discussed in this thesis is using a Deep Learning architecture. Deep Learning (DL) is a branch of Machine Learning (ML), which itself is part of the wider field of Artificial Intelligence (AI). Before introducing DL, I will provide a brief overview of ML in general and the algorithm used in the reference method *lmc*ENM, the Support Vector Machine (SVM), to be able to compare them.

Machine learning is concerned with the development and study of algorithms that allow computers to automatically learn from experiences, progressing towards a defined goal. This learning process is applied to a set of data points known as "training data". ML algorithms can be said to find and learn patterns in the data they are applied to. Common categories of ML algorithms are supervised, unsupervised, and reinforcement learning, as well as dimensionality reduction:

- **Supervised learning** algorithms use data points that have *labels* attached to them to build a *predictive model*. Data points for an object recognition dataset could be images of objects and the labels what an image shows, e.g. "cat" or "desk". Typical use cases are classification and regression problems, i.e. categorical and continuous prediction ("Will it rain tomorrow?" vs. "How much is this house worth?"). A successful supervised learning algorithm is

said to *generalize well*, meaning that the patterns the model learned are sufficiently relevant to result in correct predictions when the model is applied to unseen data *from the same distribution*. There are many supervised learning algorithms of varying complexity, from fitting a linear function to data using the least-squares method to highly elaborate deep learning architectures with millions or even billions of learnable parameters.
Besides gathering and preparing data, a substantial fraction of the effort involved in applying a supervised learning algorithm to a specific problem is directed towards reducing **bias and variance**. Bias is the error resulting from wrong assumptions that the model makes. This can be caused by learning the wrong or too little facts from the training data. The variance error is related to sensitivity to small variations in training data. A model with high variance will pick up on unimportant patterns in the training data. High bias or variance will both result in poor generalization. Bias and variance are closely related to the problems of *underfitting* and *overfitting* to training data. Because from a certain point onwards reducing one will increase the other, the situation is also called the bias-variance tradeoff.

- **Unsupervised learning** on the other hand involves unlabeled data. The task is to learn an efficient internal representation of structures in the data. This representation can be seen as a learned *a priori* probability distribution on a specific domain. Examples are language models and clustering algorithms like k-means. Sometimes, mimicry of the input data is the goal.

- **Reinforcement learning** deals with agents in a dynamic environment. RL algorithms allow the agent to learn behaviours based on the current state of the environment (as measured by the agent's inputs) that maximize a *reward function*. A common topic in RL is the exploration-exploitation tradeoff.

- **Dimensionality reduction** algorithms reduce the number of dimensions of a dataset while trying to retaining as much useful information as possible. The motivation for this is that lower-dimensional data is easier to handle in a lot of ways (e.g. the **curse of dimensionality**). An important dimensionality reduction algorithm is Principle Component Analysis (PCA).

The algorithms and methods explained in the rest of this chapter all fall in the category of supervised learning. While both Support Vector Machines and Deep Learning methods can be used for both classification and regression, breaking contact prediction as understood in this thesis is a classification task.

2.2.1 Support Vector Machines (SVMs)

The Support Vector Machine [13] is a common supervised learning algorithm. To perform binary classification, it learns a *maximum margin*, a class boundary that is as far away as possible from points of either class. This is illustrated in Figure 2.2. The data points closest to the boundary are called support vectors. The idea behind maximizing the margin size is to achieve robust generalisation, as the larger the class gap, the more unlikely it is for an unseen data point to cross it. SVMs generally learn a linear class boundary, but arbitrary data cannot be assumed to be linearly separable. To address this, SVMs use the *kernel trick*, an efficient, implicit mapping of the input data to a higher- (possibly infinite-) dimensional space where it becomes linearly separable. Advantages of SVMs are their interpretability (at least for linear kernels), robustness, performance on datasets of limited size and strong theoretical foundation in the form of Vapnik–Chervonenkis statistical learning theory (VC theory) [123]. They also have few hyperparameters that need to be optimized. The main disadvantage of SVMs is their high training cost, which is generally cubic in the number of examples. While VC theory guarantees optimality of a training result, this only holds for one specific set of hyperparameters, and non-convex grid search still needs to be performed. Another disadvantage is their inflexibility with regards to possible outputs as well as their heavy reliance on manual feature engineering.

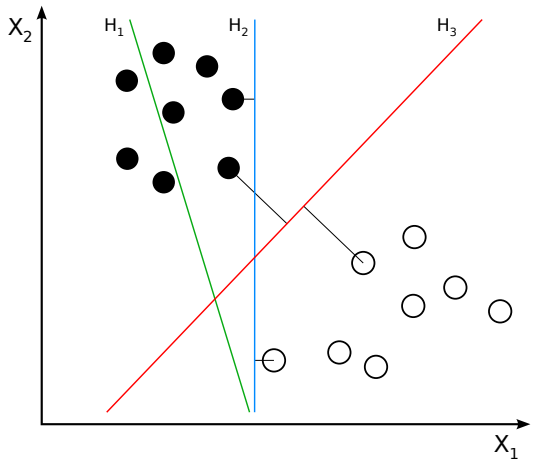

Figure 2.2: Separating and non-separating hyperplanes of a simple, linearly separable dataset. While H_1 does not separate the data, H_2 and H_3 do. But only H_3 is a maximum-margin boundary that would be learned by a SVM: data points are maximally far away from it.[1]

[1] Image by Wikimedia user *ZackWeinberg*, based on PNG version by *Cyc*. License: CC BY-SA 3.0. `https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg`

2.2.2 Deep Learning

Deep Learning [74] has taken the world by storm in the last decade, and continues to be a highly active research area. New architectures are being invented weekly and benchmark results pushed ever higher. DL-based systems are permeating society and are applied to a myriad of problems, from music recommendations [93] to quantum chemistry [122]. DL is used to create human-like text [14] and practically solve protein structure prediction [57]. Today, DL research is dominating the field of ML and has displaced most other long-established methods. The remarkable abilities of modern deep learning architectures are advancing the very notion of what computers can do, sometimes already showing superhuman performance in specific tasks [45, 95, 110] and even raising new ethical questions in the process [82].

The foundation of modern Deep Learning was laid already in 1958, with Rosenblatt's **perceptron algorithm** [103]. The perceptron is an abstraction of a biological neuron, adjusting its binary output depending on the input it "perceives" according to a *learning rule*. The perceptron's parameters define a linear decision boundary. This shows its limitations, and means that the algorithm will only converge for linearly separable data. The perceptron was conceived at a time of great optimism in the field of AI, a period known as the *golden years*. The perceptron in particular was the object of high expectations, as seen in this *New York Times* article:

> *"The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence. Later perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech and writing in another language, it was predicted."*
>
> [The New York Times, 8 July 1958, quote from [96]]

While the early AI optimism soon subsided, the modest perceptron was improved into the **multilayer perceptron (MLP)**, the first and most simple artificial neural network (ANN). Also called feedforward neural networks, ANNs are hierarchical collections of artifical neurons organized in layers: an input layer, whose dimensions are called *features*, at least one so-called hidden layer and an output layer. Neurons are typically connected to all neurons in the next layer, and the connections have weights and biases ("fully connected" layers). After the weights have been randomly initialized, ANNs are trained using **backpropagation** [62, 79, 105, 108], an efficient algorithm that recursively applies the chain rule to determine how to change the weights and biases of all neurons in the network to nudge the final output closer

to a desired one. The desired output is given by the labels of the training example and the mistake the network currently makes is quantified using a *loss function*, which must be differentiable (in fact, all parts of a neural network need to be). The average loss over the training set is minimized by **(stochastic) gradient descent (SGD)**, which backpropagation made possible to use in neural networks. Training takes place in a number of rounds called *epochs*, where in each epoch every example from the training set is shown to the network once, in randomized order.

Another important component of ANNs is the use of **nonlinear activation functions**, which are typically applied to the output of neurons. Nonlinearities are critical for the expressiveness of neural networks: without them, they collapse to linear classifiers.

Feedforward ANNs with nonlinearities are powerful learners: the **universal approximation theorems** state that ANNs can approximate any well-behaved function to arbitrary precision, using just one hidden layer of unbounded width [50] or an unbounded number of layers with bounded width [83]. This raises the question of, in light of the aforementioned result, why anyone would expend resources on more than one hidden layer.

First, notice that the universal approximation theorems are nonconstructive, i.e. they only guarantee the existence of an approximating shallow neural network for a function, not give instructions of how to arrive at it.

Second, the answer leads to the premise of deep learning: learning useful intermediate representations in **multiple hidden layers**. The classic (and admittedly strained) example is handwritten digit recognition: suppose the input is an $N \times N$ pixel grayscale image (i.e. pixels only have a single channel, the brightness value) depicting a digit from $0 - 9$, which is flattened into a vector of size N^2 and fed into the network. Then the neurons in the first hidden layer may learn very abstract shapes such as edges and ridges. The following layers learn increasingly concrete and complex concepts based on the concepts below, such as points, lines, circles, arcs etc. and combinations of those, at different locations in the input. Finally, the most high-level concepts are combined to construct flexible prototypes of the numbers itself. For example, the number 8 consists of two circles of approximately the same size stacked approximately vertically.

An illustration of the feedforward ANN architecture is given in Figure 2.3. This specific network would be considered *shallow*, as it possesses only a single hidden layer. The large number of parameters still becomes apparent.

This hierarchical representation learning has been shown to happen in deep neural networks, although the intermediate representations are not always as readily inter-

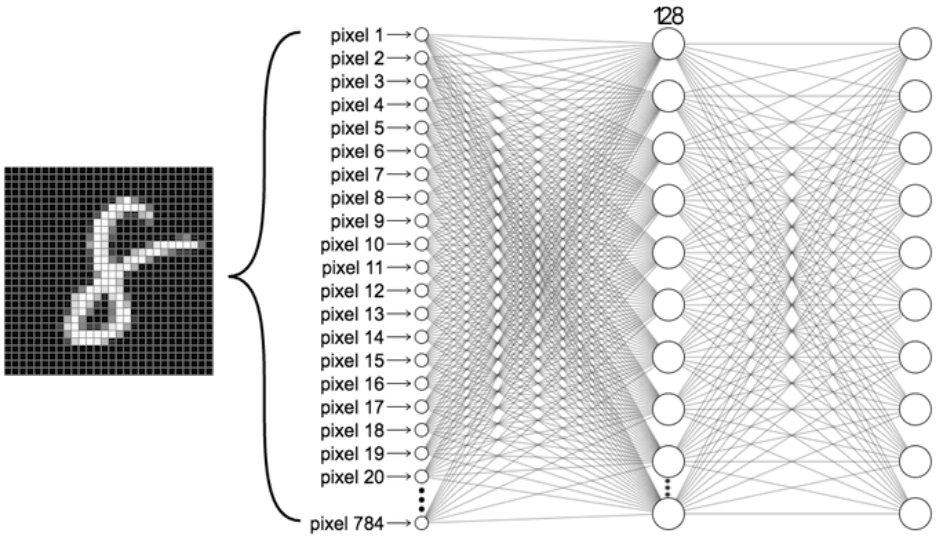

Figure 2.3: A simple, fully connected neural network for handwritten digit recognition with an example input. The network has an input layer taking a flattened 28×28 pixel grayscale image, one hidden layer of width 128, and a 10 neuron output layer representing a probability distribution over the digits from 0-9. Image from *"MIT 6.S191: Introduction to Deep Learning"*, `https://raw.githubusercontent.com/aamini/introtodeeplearning/master/lab2/img/mnist_2layers_arch.png`, MIT License

pretable as in this example. The resulting networks have, most of the time, a greater capability to generalize than shallow ones, sometimes drastically so. Depending on the problem, very deep architectures may have hundreds of hidden layers.

After neural networks fell out of favor multiple times in the history of AI, the appearance of powerful enough GPUs in the 2010s combined with breakthroughs largely solving the problems of **vanishing and exploding gradients** [37, 44, 53] led to a *deep learning renaissance* that is showing no signs of ending soon. DL is the favored machine learning technique today not least because of two facts: (a) huge datasets have become available for many practically important problems, and (b) deep neural networks are able to utilize these massive amounts of data *efficiently* and *effectively*. In other words, they scale well. Additionally, they are extremely flexible in terms of input, output, and the problem statement itself. The general idea of neural networks can be applied to unsupervised, semisupervised and supervised problems, binary- and multiclass classification, regression, learning on graphs (graph neural networks (GNNs)), images (convolutional neural networks (CNNs, see Subsection 2.2.3), Transformers), sequence data (recurrent neural networks (RNNs) and long short term memory networks (LSTMs), Transformers), generative models (GANs), and more.

Deep neural networks are not without problems, however. Owing to their enormous capacity, they are prone to overfitting and care needs to be taken to address this tendency. Many approaches to limit overfitting have been developed. Interestingly, even heavily overparametrized networks allowed to perfectly fit their training sets generalize somewhat: a testament to their sometimes counterintuitive nature [4, 136]. Even in these extreme cases, more than simple memoization seems to be happening. In practice, a combination of techniques such as **regularization**, **normalization** and **early stopping** is often used to combat overfitting.

Another theoretically predicted problem that has effectively disappeared is that of minimizing the loss function: it was thought that a simple first-order methods like gradient descent would quickly and invariably get stuck in local minima, which would largely render neural networks useless in practice [74]. However, this rarely if ever seems to matter. While proper initialization is important [45], testing a small number of random starting seeds is sufficient to account for model variability in most cases. If not, this is more likely to indicate a different underlying problem. The inexactness of gradient approximation by SGD also helps.

Furthermore, neural networks have a large number of "moving parts" that can be individually perturbed and thus many hyperparameters, optimization of which can require a certain amount of experience and resources.

Lastly, neural networks work best when there are large amounts of data available, and the required dataset size is hard to estimate beforehand. This relationship of performance and amount of data available is illustrated in Figure 2.4.

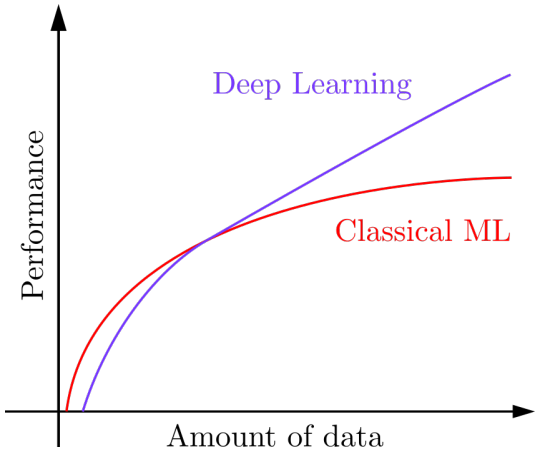

Figure 2.4: Scalability of ML algorithms: classical ML algorithms are often the best choice when moderate amounts of data are available, but can eventually fail to scale or learn from more data. Often, this is the point where deep learning becomes superior. Figure adapted from [94]

2.2.3 Convolutional Neural Networks (CNNs)

Fully connected neural networks quickly grow to have a truly enormous amount of parameters. This is especially the case with large input vectors, such as images.

The number of parameters (weights and biases) of a MLP with input dimension D_i, n_h hidden layers with the dimension of the ith hidden layer given by $D_h(i)$, and output dimension D_o is given by

$$|P_{fc}| = \underbrace{D_i D_h + \sum_{i=1}^{n_h-1} D_h(i) D_h(i+1) + D_h(n_h) D_o}_{\text{weights}} + \underbrace{\sum_{i=1}^{n_h} D_h(i) + D_o}_{\text{biases}}. \tag{2.5}$$

Consider a fully connected network for object classification (100 classes) in images. It takes as input quadratic RGB images of the modest dimension 100×100. This results in a flattened input dimension of $100 \cdot 100 \cdot 3 = 30000$. Adding 5 hidden layers of constant width 1000 already results in ~34 million parameters to be learned. The same example with 1000×1000 images results in ~3 billion parameters, practically almost intractable. It is mostly the input dimension and the hidden layer width that leads to a large number of parameters, and many real-life problems would result in even higher numbers for those. To handle such large inputs, clearly a more efficient approach is required. The full connectivity of simple MLPs makes them also very prone to overfitting.

Convolutional Neural Networks [34, 44, 75] solve these problems using sparse, local connectivity. Instead of a large number of neurons looking at individual pixels, **convolutional layers** train a number of small matrices, so called *kernels*, that are slid or *convolved* over the whole image. Common kernel sizes are for example 3×3 and 5×5. Instead of flattening an image into a one-dimensional vector, the 2D structure is retained, but a third dimension, the depth, is added. CNNs also work for primarily 1D (sequence) and 3D (for example video data) inputs, with an arbitrary depth dimension. Convolutional kernels are always applied to the whole depth of the input, but only consider the local context in width and height. To compute the activation of a kernel by an input, the kernel is for each input pixel positioned so that the pixel in question is in the middle of the kernel. Then the **dot product** of the kernel matrix and the local context of the pixel including itself is calculated (through the whole depth), which is the activation for the kernel at this pixel. For all pixels together, this yields the *activation map* of the kernel. The output depth of a convolutional layer is given by its number of kernels. The output width and height is controlled by the amount of zero-padding (extending the edges

with zeros to allow applying kernels there) and stride (how many pixels to move the kernel at a time, a setting > 1 will skip pixels). These can be adjusted so that the output dimensions are equal to or smaller than the input dimension.

Formally, the output of a 2D convolutional layer with C_{out} kernels and an input of size $W \times H \times C_{in}$ (width, height, depth) can be compactly described as

$$O_i = b_i + \sum_{j=1}^{C_{in}} \mathcal{K}(i,j) \star I_j \tag{2.6}$$

for $1 \leq i \leq C_{out}$ and where O_i is the output of the ith kernel, b_i is the scalar bias of the ith kernel, $\star$ is the valid 2D cross-correlation operator (mathematically, convolutional layers actually perform cross-correlation, not convolution), $\mathcal{K}(i,j)$ is the weight matrix of the ith kernel at depth j, and I_j is the jth input channel. Stacking all O_i along the channel dimension completes the output of the layer, resulting in an output of size $W_{out} \times H_{out} \times C_{out}$. The concrete values of W_{out} and H_{out} depend on the settings for zero-padding and stride.[2]

The basic assumption of CNNs is this: if a feature is useful at one position (x, y) of the input, it will also be useful at another position (x', y'). The kernels learn to look for one specific intermediate representation, as discussed above, over the whole input image, crucially *using the same weights* for all positions at the same depth. These principles are called **translation invariance** and **parameter sharing** and are the most important inductive biases of CNNs. Using parameter sharing, the number of parameters to be learned is drastically reduced, and for a single 2D convolutional kernel of size $F \times F$ and an input with D channels amounts to $F^2D + 1$. This low number of parameters makes it possible to have many convolutional kernels in a layer, analogous to the width of a hidden layer, as well as numerous hidden layers.

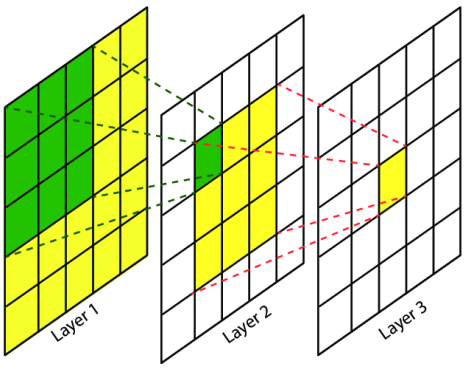

Figure 2.5: Linear increase of the effective receptive field (ERF) in CNNs through multiple convolutional layers. Figure from [78]

[2] For an in-depth calculation, see `https://cs231n.github.io/convolutional-networks/`

Each activation map is influenced by an area $F \times F$ around each pixel. This is called the **receptive field** of the kernel. Stacking convolutional layers leads to a linear increase in the **effective receptive field**: each pixel in the $F \times F$ receptive field of a second layer depends on the $F \times F$ area around itself, resulting for two equal layers in an effective receptive field of $(2F-1) \times (2F-1)$. See Figure 2.5 for an illustration of this effect. Using **dilated convolutions** [135], the effective receptive field can even be increased exponentially with only a linear increase in parameters. See Figure 2.7 for an illustration of how they work.

Typical CNN architectures also contain pooling layers, especially max-pooling, whose output is the maximum over a small, e.g. 2×2 patch of input. The purpose of pooling is to decrease the input size periodically and counteract overfitting. Pooling layers are recently becoming less important, as the same downscaling effect can also be achieved through convolutional stride. An example of a traditional CNN architecture involving convolutional, pooling and fully connected layers at the end is shown in Figure 2.6.

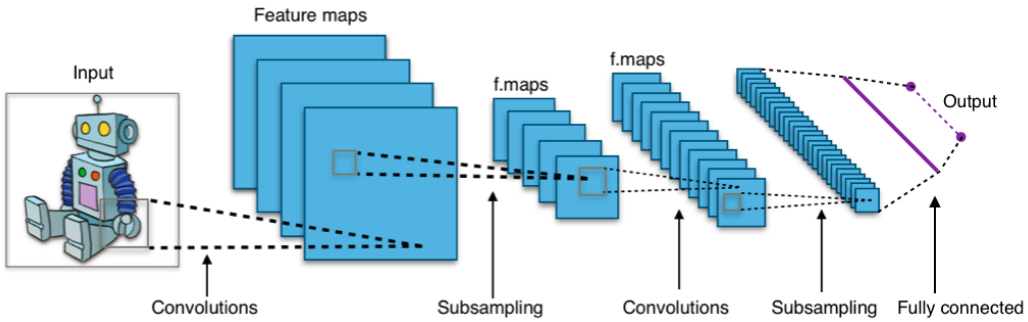

Figure 2.6: Typical CNN architecture for object recognition, showing convolutional, pooling, and fully connected layers. Translation-invariant features are learned by the convolutional layers, image size reduced by pooling layers and the final classification done using fully connected layers. Notice that input image size has to be fixed. Image by Wikimedia user *Aphex34*, License: CC BY-SA 4.0, `https://commons.wikimedia.org/wiki/File:Typical_cnn.png`

Fully convolutional architectures are also possible. They are able to handle inputs of arbitrary width and height and don't change the output size in these dimensions. I will use this property for my breaking contact prediction model. A typical task for such a network is semantic segmentation, also called *pixel-level classification*. This task involves labeling objects in an input image. It is illustrated in Figure 2.8 due to its significance to the breaking contact prediction problem, which, when approached using CNNs, can be seen as a two-class version of semantic segmentation.

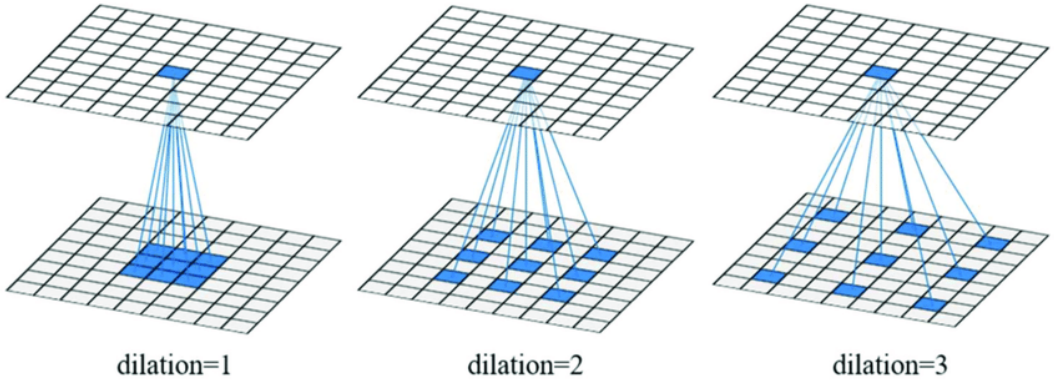

Figure 2.7: Receptive field of a dilated convolution. When the dilation is equal to one, the receptive field is that of a regular convolution. A dilation larger than one skips pixels, but over all activations no information is lost. This allows the receptive field to grow exponentially with only a linear increase in parameters. Image from [22]

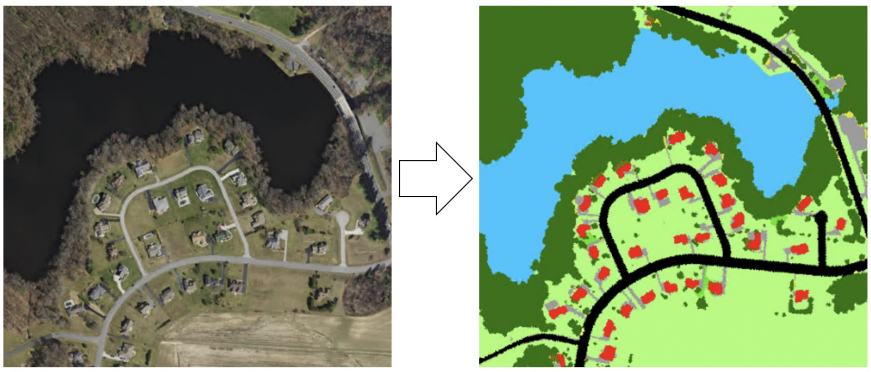

Figure 2.8: Example of semantic segmentation, also called pixel-level classification. The 2D output dimensions are equal to the input dimensions. Each pixel is assigned one of multiple classes representing a type of object. Image from `https://developers.arcgis.com/python/guide/how-unet-works/`

3 Related Work

The most important work related to my method is arguably Putz and Brock's *lmc*ENM, which I introduce in this chapter at length. Besides this, it was difficult to find closely related work, as the combination of Elastic Network Models and machine learning seems to be largely unexplored. Nevertheless, I close with a discussion of notable ENM variants in general and try to determine which is closest to *lmc*ENM and my method.

3.1 ENM of learned maintained contacts (*lmc*ENM)

The Elastic Network Model of **l**earned **m**aintained **c**ontacts, or short *lmc*ENM [99, 100], is the reference method my model is based on. Its main idea is to sparsify the connectivity of a standard ANM, making it locally more flexible, in order to allow it to capture localized function-related motion better than a simple cutoff-based ANM. To predict protein motion of an unbound structure, *lmc*ENM begins with a standard ANM. Then, using a SVM classifier (see Subsection 2.2.1) trained on known *apo-holo* pairs, the unbound structure's breaking contacts during a functional transition are predicted and subsequently removed from the ANM.

While *lmc*ENM can in principle be combined with other ENM variants that use nonuniform spring stiffness (see Section 3.2), its focus is primarily on investigating the effects of altered contact topology.

3.1.1 Baseline network $\text{ANM}_{\text{minDeg4}}$

Springs in ANMs are usually distance-cutoff based with a uniform spring stiffness (see Subsection 2.1.1). The lower the cutoff, the smaller the number of springs and the more flexible the network becomes. A more flexible network allows the ANM to better capture localized motions at the expense of motion magnitude accuracy. As the goal of *lmc*ENM is improved prediction of localized functional motions, a low cutoff of 10Å is used. ANMs with low cutoffs tend to become unstable, which manifests itself as more than six trivial (i.e. zero) eigenvalues. To support network stability, Putz and Brock additionally devised an algorithm to stabilize the contact graph. It is implementing minimum connectivity rules identified by Jeong in [55], specifically a minimum of $3L - 6$ edges (equal to the number of degrees of freedom) and a minimum node degree of 4, where L is the length of the protein (i.e. its number

of amino acid residues). Notice that the latter includes the former. The algorithm (1) connects nodes not fulfilling the minimum degree rule to close sequential and/or minimum distance non-neighbours. The stabilized ANM with cutoff 10Å is called $\text{ANM}_{\text{minDeg4}}$.

Algorithm 1: Contact Graph Stabilization

Data: Contact graph $G = (V, E)$, $V = \{v_1, v_2, \ldots, v_L\}$ of protein of length L with Euclidian distance function $\mathcal{D} : V \times V \to \mathbb{R}^+$
Result: Jeong-stable contact graph $G' = (V, E')$, $E \subseteq E'$

for $v_i \in \{v \in V \mid \deg_G(v) < 4\}$ **do**
  while $\deg_G(v_i) < 4$ **do**
    `// Check for close sequential non-neighbour`
    if $\exists\, v_j \in V \setminus N_G(v_i)$ **where** $|i - j| \leq 4$ **then**
      $E \longleftarrow E \cup \{v_i,\ \text{argmin}_{v_j \in V \setminus N_G(v_i)} |i - j|\}$
    `// Fall back to minimum distance non-neighbour`
    else
      $v_j \longleftarrow \text{argmin}_{v \in V \setminus N_G(v_i)} \mathcal{D}(v, v_i)$
      $E \longleftarrow E \cup \{v_i, v_j\}$
return (V, E)

3.1.2 Breaking contacts and other dynamic contact changes

Breaking contacts are pairs of residues in a protein whose distance is initially below a certain maximum and then changes by a minimum percentage during a functional transition. *Forming* and *maintained* contacts are defined using similar ideas, and illustrated in Figure 3.1.

Formally, this results in the following transition matrix, where i and j are residues, d^0_{ij} and d^1_{ij} are their distances in the *apo* and *holo* structures, d_c is the ANM distance cutoff, and $\Delta d_{ij} := d^1_{ij} - d^0_{ij}$:

$$T_{ij} = \begin{cases} \text{maintained contact}, & \text{if} \quad d^0_{ij} \leq d_c \quad \text{and} \quad |\frac{\Delta d_{ij}}{d^0_{ij}}| \leq e_c \\ \text{breaking contact}, & \text{if} \quad d^0_{ij} \leq d_c \quad \text{and} \quad |\frac{\Delta d_{ij}}{d^0_{ij}}| > e_c \\ \text{forming contact}, & \text{if} \quad d^0_{ij} > d_c \quad \text{and} \quad d^1_{ij} \leq d_c \\ \text{no contact}, & \text{otherwise.} \end{cases} \tag{3.1}$$

It can be seen that e_c is the minimum relative distance change of two residues in a functional transition for a contact to be considered breaking. For *lmc*ENM and the dataset used, the optimal e_c was empirically determined to be 9%. While

forming contacts are conceptually interesting, only breaking contacts were shown to be useful for improving capturing of localized motions. Adding forming contacts resulted in performance below the baseline ENM.

Given both *apo* and *holo* structures, the breaking contacts can easily be computed. Removing those from $\text{ANM}_{\text{minDeg4}}$ results in the ENM of observed maintained contacts (*mc*ENM), a ground truth network to compare *lmc*ENM to.

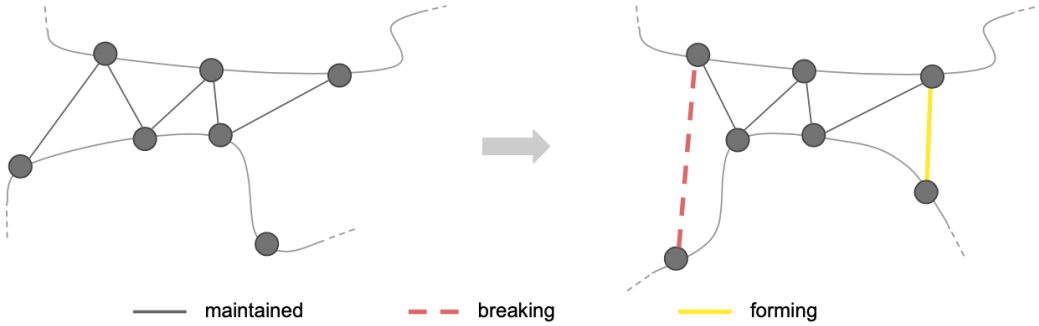

Figure 3.1: Illustration of types of dynamic contact changes: maintained, breaking and forming contacts. Figure from [99]

3.1.3 SVM dataset, features, and training

For each *apo*-contact with a minimum sequence separation of 4, i.e. for each pair of residues i, j with $d_{ij}^0 \leq d_c = 10$ and $|i - j| \geq 4$, a SVM is used to predict the probability of it breaking during a functional transition. To obtain probabilities instead of a binary value, Platt scaling [129] is used. The SVM is trained (and validated using leave-one-out cross validation) on a dataset of 90 apo-holo pairs from the Protein Structural Change Database (PSCDB) [5].

The PSCDB contains 839 pairs of unbound and ligand-bound protein structures divided into seven categories: coupled domain motion (59 pairs), independent domain motion (70), coupled local motion (125), independent local motion (135), burying ligand motion (104), no significant motion (311), and other type motion (35). While *lmc*ENM aims to specifically improve local motion types, having pairs from the full spectrum of high level protein motions allows to see how performance changes for all of them. Using several filters, Putz and Brock arrived at a set of 90 single-chain protein pairs. Note that there is a strong class imbalance due to the fact that maintained contacts are far more common than breaking contacts, which has to be taken care of during training.

The performance of the SVM directly depends on the quality and importance

of the features used. Two kinds of features are used in *lmc*ENM: (a) whole protein features, and (b) physicochemical, structural and topological properties of the structural context of a contact and its embedding in the protein's structure. The latter are derived from the **immediate neighbourhood graph (ING)** of the contact [99, 109] and the **secondary structure element (SSE) graph**.

The immediate neighbourhood graph IN_{ij} of two contacting residues i, j consists of nodes (residues) and edges (contacts between them). It includes nodes i, j, as well as their first shell neighbours, i.e. residues in direct contact with either i or j. An example IN_{ij} is shown in Figure 3.2. Notice how the graph contains sequentially contiguous and noncontiguous residues.

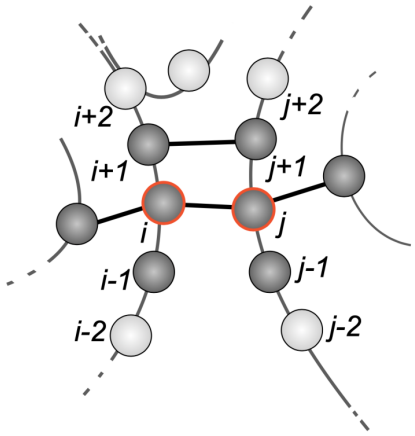

Figure 3.2: The immediate contact neighbourhood graph of residues i and j, IN_{ij}. Nodes in dark grey are part of IN_{ij}. Figure adapted from [99]

The SSE graph is an abstraction of the protein with nodes on the secondary structure (α-helices, β-strands, loop regions) level, connected by edges if any of their residues are in contact. Node labels characterize individual SSEs and edge labels describe an interface.

In total, a 170-dimensional feature vector is constructed from these data sources, with the vast majority being derived from the ING and SSE graph. Features derived from the local contact neighbourhood can be grouped into six categories (examples in parentheses):

- Pairwise (centroid distance, SSE contact type and hydrogen bonding, mutual information)
- Graph topology (number of nodes and edges, average closeness centrality, radius, diameter)
- Graph spectrum (largest and second largest eigenvalue, energy)

- Single node (closeness and betweenness centrality, sequence separation from N/C-terminus, sequence conservation)
- Node label statistics (residue depth, chemical type, SS distribution, neighbourhood impurity degree, distance to centroid)
- Edge label statistics (link impurity, mutual information distribution)

The rationale for the heavy emphasis on the local context is the hypothesis that the direct neighbourhood of a contact contains the most important information about its propensity to break. It can further be seen that to encode the graph to a constant-length feature vector, statistics *about* the graph are used, including many hand-engineered graph-theoretic features from the field of network analysis.

The top 16% of contacts by SVM score are removed from $\text{ANM}_{\text{minDeg4}}$ by setting their spring stiffness to 0. Some predicted breaking contacts may be non-removable, as doing so would violate Jeong stability. Removal of predicted breaking contacts is attempted in decreasing order of SVM score. The constant value of 16% was again chosen empirically over a number of other removal strategies, although in reality the fraction of breaking contacts is protein-specific.

3.1.4 Results

For the purposes of this thesis, the results of *lmc*ENM can be evaluated on two levels: the performance of the binary classifier given by the SVM and the impact of removing predicted breaking contacts from $\text{ANM}_{\text{minDeg4}}$. While the absolute performance of the SVM as measured by precision and coverage is rather low (see Table 3.1), it succeeds in predicting relevant and sufficiently correct breaking contacts. Removing SVM-predicted breaking contacts in most cases greatly improves capturing of local motions that were only poorly captured before. In these cases, relevant modes can be said to be shifted more to the front of the list, reducing the dimensionality of the essential deformation space (how many modes are needed to represent e.g. 70% of the deformation space). ANM performance for already well-captured motion classes does not significantly deteriorate and is sometimes lightly improved. While in some single cases *lmc*ENM even exceeds the ground truth performance of *mc*ENM, on average there is still quite some room for improved performance, which could be attained by improved prediction of breaking contacts.

A useful measure to assess the accuracy of ANMs, given two conformations, is the **mode overlap**. It describes the fraction of conformational change explained by a single mode j based on the angle between conformational displacement vector and mode direction vector M_j, and is defined as

Motion Type	Precision	Coverage	AUROC
Coupled Local Motions (28)	0.24/0.27	0.41/0.44	0.62/0.61
Independent Local Motions (18)	0.22/0.25	0.40/0.41	0.56/0.58
Coupled Domain Motions (20)	0.18/0.18	0.43/0.44	0.64/0.62
Independent Domain Motions (14)	0.15/0.16	0.39/0.37	0.62/0.63
Burying Ligand Motions (4)	0.13/0.19	0.32/0.30	0.49/0.51
Other Types of Motions (9)	0.20/0.30	0.25/0.28	0.55/0.55
All (90)	0.19/0.23	0.41/0.41	0.61/0.60

Table 3.1: Performance of the SVM classifier used in *lmc*ENM as measured by precision, coverage, and Area under the Receiver Operating Characteristic curve (AUROC) of the top 16% predicted breaking contacts. Mean and median are reported and results are grouped by motion types. The SVM performs best for local motion types. Table data from [99]

$$O_j = \frac{\left| \sum^{3L} M_j \Delta r_i \right|}{\left[\sum^{3L} M_j^2 \cdot \sum^{3L} \Delta r_i^2 \right]^{1/2}} , \tag{3.2}$$

where $\Delta r_i = (r_i^E - r_i^S)$ is the displacement vector from start to end conformation at residue i and L is the number of residues of the protein. Summation of the first k mode overlaps yields the **cumulative mode overlap**

$$CO(k) = \left[\sum_{j=1}^{k} O_j^2 \right]^{1/2} . \tag{3.3}$$

The cumulative mode overlap of the first ten modes $CO(10)$ is the most important ANM accuracy measure in [99] as well as this thesis. The following figure summarizes the impact *lmc*ENM has on the relevance of the first 10 normal modes, as well as showing the improvement relative to the ground truth *mc*ENM that is still possible.

What analysis also showed is that *lmc*ENM works best for coupled local and independent local motions, categories for which standard ENMs perform poorly, while other motion classes are relatively unaffected. Even though a constant top 16% of predicted breaking contacts are removed, the domain movers that are already well captured using standard ENMs are quite stable against false positives. The effect of *lmc*ENM and *mc*ENM on the $CO(10)$ is visible in Figure 3.3.

Putz and Brock also analyzed *lmc*ENM in terms of a number of other ENM metrics like *fluctuation profile correlation*, *fraction of variance*, and *degree of collectivity*, analysis of which falls outside the scope of this thesis.

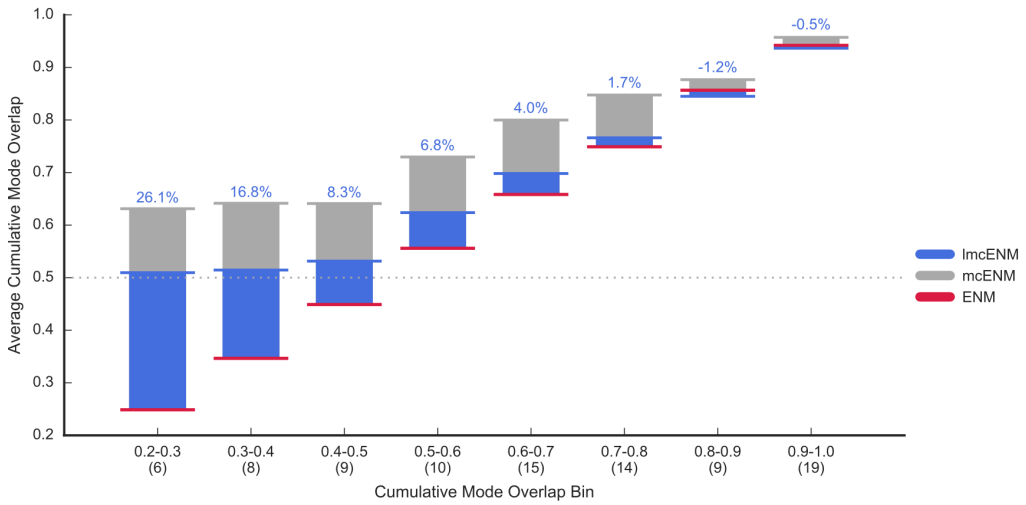

Figure 3.3: Improvement of *lmc*ENM (blue) and *mc*ENM (grey) in average 10-cumulative mode overlap, $CO(10)$, over baseline ENM. The data is binned according to how well the baseline ENM captured the motion as measured by $CO(10)$. The greatest improvement can be seen for proteins where the baseline ENM performed worst. Figure from [99].

3.2 Other ENM variants

There are a lot of other ENM variants incorporating different additional sources of information. In [99], Putz and Brock categorize variants into those using physicochemical, structural and dynamical knowledge. Common themes in ENM variants are cutoff-free spring functions based for example on spatial distance [47, 134], sequential distance [55], or bonding type [65], more involved potentials [112, 115], mixed resolution models [69], and combining ENMs with information from short MD trajectories [36, 97]. Many models trade physical accuracy for significant additional computational cost. Very recent developments are an improvement [40] of Xia's multiscale ENM [130] and a residue-specific ENM interestingly involving ligands [61].

Few if any variants choose a pure topological modification/reduction approach similar to *lmc*ENM: [25] adds springs between buried residues as well as between hydrogen-bonded residues, [36] comes conceptually closest to *lmc*ENM by only retaining springs largely maintained throughout MD simulations.

Besides fitting spring parameters to MD trajectories, none of the surveyed methods involve elaborate machine learning, and none use deep learning.

4 CNN for breaking contact prediction

In this chapter, I present the main contribution of the thesis, a novel Convolutional Neural Network (see Subsection 2.2.3) for breaking contact prediction (BCP). To do so, I first argue for the appropriateness of a CNN as a machine learning model for breaking contact prediction and the implications of CNN inductive biases for the BCP problem in contrast to *lmc*ENM's original SVM approach.

Then, I discuss the construction of the dataset used to train it and the validation approach. Next, I present the final model architecture including a list of features used. At the end of the chapter, I describe the metrics used to evaluate the model.

From here on, *lmc*ENM$_{\text{CNN}}$ is used to refer to the method presented in this thesis and *lmc*ENM$_{\text{SVM}}$ to refer to the original implementation as described in [99].

4.1 Applying CNNs to BCP

The SVM used in *lmc*ENM$_{\text{SVM}}$ classifies each *apo*-contact individually, using some global features and many features derived from graph representations of its local context. Using a CNN, it is possible to approach this problem differently, imposing fewer assumptions about the optimal structure of features. At the same time, the inductive biases of a CNN are quite suitable for breaking contact prediction.

To be able to apply CNNs to the problem, I structure input data in a **contact map**-type format. For a protein with L residues, this is a tensor M with shape $L \times L \times 96$, where $M_{i,j}$ contains the 96-dimensional feature vector of the residues at sequence positions i and j. These two residues may or may not be in contact in the apo structure, which is simply encoded in the form of a feature. Features at position i, j directly represent residue-level and pairwise features of the respective residues. Examples of residue-level features are chemical type, secondary structure at this position, and half-sphere exposure. Examples of pairwise features are three-dimensional distance, whether the contact distance is less than or equal to d_c, and mutual information (MI) in the multiple sequence alignment (MSA). The features are concatenated to a feature vector of length 96 and form the depth dimension of the input data.

Using this approach, there is no need to specify *a priori* what statistics to extract from the raw data anymore, as was the case with immediate neighbourhood graph (ING) derived features in *lmc*ENM$_{\text{SVM}}$. This simplification of features is in line with the common DL philosophy of letting the network learn efficient high-level

features on its own, avoiding error-prone and tedious manual feature engineering. The locality assumption behind *lmc*ENM$_{\text{SVM}}$'s neighbourhood graph is preserved through the locality property of convolutional kernels, but is in contrast not limited in the size of the context it can use to make a prediction: while it is a hyperparameter that has to be fixed eventually, the size of the prediction context can be easily extended by adding more convolutional layers, resulting in a larger effective receptive field, whereas the ING (ignoring the SSE graph and global feature contributions) by definition only takes into account direct neighbours.

Furthermore, the translation invariance property of CNNs applies to the breaking contact prediction problem, as patterns of breaking contacts should largely be independent of their position along the amino acid chain. Exceptions may be given by the N- and C-termini, which are often found to be rather disordered loop regions. I address this by removing proteins from my dataset whose motion almost exclusively consists of these terminal loop movements and including a "percentage of L at this position" feature, which should allow the CNN to use any remaining relevant position-specific information. More evidence for this assumption about the BCP problem is provided by the observation that sequence separation as a feature did not seem to be useful, even resulting in a higher loss than without it.

CNNs also implement the "corroborating evidence" proposed in [99]: the inputs to the final layer of all other residue pairs in the effective receptive field (ERF) are available to the pair being predicted.

The sequence-based locality of the contact map creates one downside of the CNN approach: in case they are sequentially far away, it is possible for a contact of a given residue to fall outside the ERF. This would lead to the loss of potentially valuable information. The chance of this happening decreases with a growing ERF. As the ERF can be made really large through dilated convolutions and contacts are seldom more than a few hundred residues apart sequentially, the chance of this happening could be reduced to zero by incorporating the complete contact map. On the other hand, using this data formulation, it is hard to isolate the effects of enlarging the sequential context and capturing a larger fraction of the 3D context of a contact in this sequence range, as changing the ERF affects both.

The main characteristics of modeling the BCP problem in this way are therefore:

- Input to the classifier for one protein is a $L \times L \times 96$ tensor with the features in the depth dimension.

- No high level features, instead residue-level and pairwise features are computed for all residue pairs i, j, contact or not. High level, graph-like features are

learned by the network. Noncontact positions are masked out of the loss (see Subsection 4.6.1), but can be used by kernels for prediction.

- Easily scalable prediction context through convolutional layers. In contrast, *lmc*ENM$_{\text{SVM}}$ takes into account only the first-shell neighbours in the ING (very short range), including a slightly larger context via the SSE graph (short-medium range) and some whole protein features. A deep CNN can potentially uniformly cover the complete range up to several hundred residues away. No global features are used in the CNN.
- The assumption of locality importance is preserved, but locality in the CNN data formulation is primarily sequence-based instead of spatial.

4.2 Constructing a larger dataset

To leverage the full potential of the CNN, as much training data as possible is needed. While the 90 proteins used in *lmc*ENM$_{\text{SVM}}$ contain in total ~14000 breaking and ~156000 non-breaking contacts, this is still at the low end of DL dataset sizes. Consequently, I construct my own maximally large dataset from the PSCDB. Using the PSCDB again retains comparability to *lmc*ENM$_{\text{SVM}}$.

The increased size of my dataset is the result of (a) using fewer and more lenient filters than Putz and Brock, and (b) the fact I extend my method to be able to work with **multichain proteins**, which *lmc*ENM$_{\text{SVM}}$ is unable to use. This resulted in a much more elaborate preprocessing stage, but also a broad applicability of the method. In this preprocessing stage, besides cleaning and renumbering structures, I perform three global sequence alignments using the Gotoh algorithm (weights: BLOSUM62, gap start/extension penalties: -20, -2): both conformer PDBs to their FASTA sequences and both FASTA sequences to each other. This has to be done because even though both conformers describe essentially the same protein, due to experimental difficulties not all residues are always modeled, and chain identifiers or numberings are often completely different. The alignment process results in a reliable numbering, alignment, and handling of gaps, for both single- and multichain proteins.

After downloading and parsing the complete PSCDB, I applied the following filters, besides removing obsolete PDB entries:

- Aligned sequence length must be ≥ 70.
- Experimental method used for structure determination must be X-ray diffraction and resolution at least 3Å.

- Minimum aligned RMSD of the two conformers of 0.8Å in ProDy [12]. This excludes very small movements with a negligible fraction of breaking contacts. Putz and Brock used 1Å.
- Maximum gap length of 20 in all three alignments, ignoring gaps at the termini.
- No symmetrical assemblies due to questionable biological relevance (the pipeline could theoretically handle them, and code to generate the symmetrical structures is in place).

All resulting apo-holo pairs were visually reviewed and some were removed. Reasons for removal were: highly disordered proteins, broken models or PDB bugs, exclusively terminal loop motion, nonglobular topology, and unstable baseline ENM, i.e. more than six zero eigenvalues after Jeong stabilization.[1] Proteins from the "no significant motion" category were not excluded, but had to pass the minimum RMSD filter. Hypothetical proteins were not categorically excluded, either. Finally, the largest three proteins were excluded due to their extreme size ($L > 1000$).

These steps resulted in a new set of **273 apo-holo pairs**, including 56 multichain proteins (see Table 4.1). The new dataset contains about 42600 breaking and 527000 nonbreaking contacts with sequence separation of at least four, more than three times as many breaking contacts as before. These numbers correspond to a mean breaking contact fraction of all contacts of ~8%, versus ~9% for the original *lmc*ENM$_{\mathrm{SVM}}$ dataset. Figure 4.1 illustrates the wide range of protein sizes in the dataset.

Motion Type	n	%	n_{MC}	$\%_{MC}$
Coupled Local Motion (CLM)	66	24.2	14	25.0
Independent Local Motion (ILM)	56	20.5	9	16.1
Coupled Domain Motion (CDM)	42	15.4	8	14.3
Independent Domain Motion (IDM)	44	16.1	11	19.6
Burying Ligand Motion (BLM)	11	4.0	4	7.1
Other Types of Motion (OTM)	25	9.2	3	5.4
No Significant Motion (NSM)	29	10.6	7	12.5
All	273	100	56	100

Table 4.1: Distribution of motion types in the newly constructed dataset, including number and total percentage of multichain (MC) proteins.

[1] Putz and Brock also remarked that Jeong stabilization is not always sufficient to stabilize ENMs at low cutoffs. The d_c value of 10Å worked for all of their 90 proteins, but failed for some in my extended dataset.

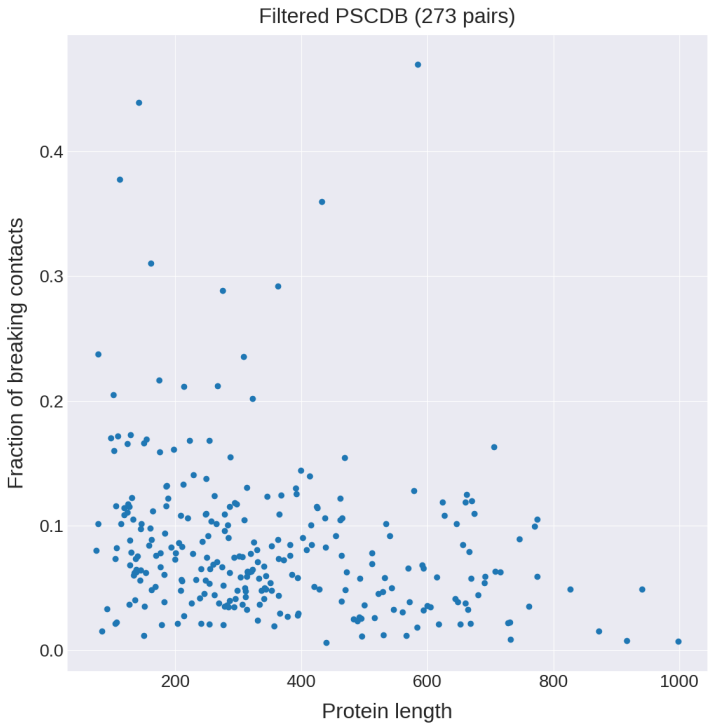

Figure 4.1: Scatter plot of protein length vs. fraction of observed breaking contacts in the new filtered PSCDB dataset. Each point represents an apo-holo pair. A wide range of protein lengths is represented in the dataset, with a concentration in the range of 100-400. There are large differences in the overall mobility between proteins as well. Plot created using matplotlib [52]

4.3 Splitting the dataset

For supervised learning algorithms, available data needs to be split at least into two sets, namely *training* and *test* data points. The parameters of the model are fitted to the training data, and generalization capability is judged on the test data. Because of the increased size of my dataset and extensive number of experiments needed, leave-one-out cross validation (LOOCV) as used in *lmc*ENM$_{\text{SVM}}$ is no longer practical and not as strongly needed anymore. Instead of cross validation, I opted for a **65-25-10 split** into training, *validation* and *test/holdout* sets. The validation set is used during model development. The test or holdout set is a portion of the data that the model is only evaluated on after hyperparameter optimization has concluded, in order to avoid overfitting (it is a common pitfall in ML to do model selection and model evaluation on the same dataset). Initially, I used a more conventional 80-10-10 split, but this lead to unacceptable instability of the validation ENM metrics from epoch to epoch, making them almost useless for model selection (see Subsection 4.6.3).

The much larger validation set in the 65-25-10 split (see Table 4.2) improved the stability of ENM metrics to a significantly more useful level.

Splitting was performed using stratified sampling over the motion types in all sets. The holdout set was constrained to contain exclusively targets present in the dataset from [99], and therefore no multichain targets. To avoid overoptimistic results, homology detection was performed on all targets. This was first attempted with HHsearch [116], but this proved too sensitive for my application, producing many false positives, even with a low e-value. In the end, homology detection was performed via pairwise threading using DeepThreader [138], like Putz and Brock. Homology was defined as a pairwise GDT_TS > 0.6 (Global Distance Test, Total Score). After the sets were generated, they were permuted until no clusters of homologous targets were split across different sets and the sets were still consistent with the rules laid out above, as well as all clusters present in one of the three sets.

Set	All	CLM	ILM	CDM	IDM	BLM	OTM	NSM
Training	179/65.5	43/24.0	36/20.1	28/15.6	30/16.8	7/3.9	16/8.9	19/10.6
Validation	68/25.0	16/23.5	15/22.1	10/14.7	10/14.7	3/4.4	7/10.3	7/10.3
Holdout	26/9.5	7/26.9	5/19.2	4/15.4	4/15.4	1/3.8	2/7.7	3/11.5

Table 4.2: Motion category distribution of the dataset after splitting into training, validation and holdout sets, aiming for a 65-25-10 split. Numbers are given as absolute and percentage of all in set (CLM: Coupled Local Motion, ILM: Independent Local Motion, CDM: Coupled Domain Motion, IDM: Independent Domain Motion, BLM: Burying Ligand Motion, OTM: Other Types of Motion, NSM: No Significant Motion).

4.4 Model architecture

The final model architecture is depicted in Figure 4.2. The model is a fully convolutional neural network (FCNN, see Subsection 2.2.3) and also a residual neural network (ResNet) [44]. ResNets are inspired by pyramidal cells in the cerebral cortex and make use of so called *skip connections*, shortcuts that jump over the internal parts of a layer and promote the flow of gradients in deep networks.

Feature maps are initially compressed from 96 to 64 and then passed through five residual blocks (ResBlocks) containing two 3×3 convolutional layers each, surrounded by auxiliary layer types. The identity input is added to the output at the end of each ResBlock and in this way passed through the network. The blocks make use of dilated convolutions [135], instance normalization [121], dropout [113], and the exponential linear unit (ELU) activation function [19]. The block architecture and dilation values of $1, 2, 4, 8, 16$ are reused from trRosetta [132], a successful protein

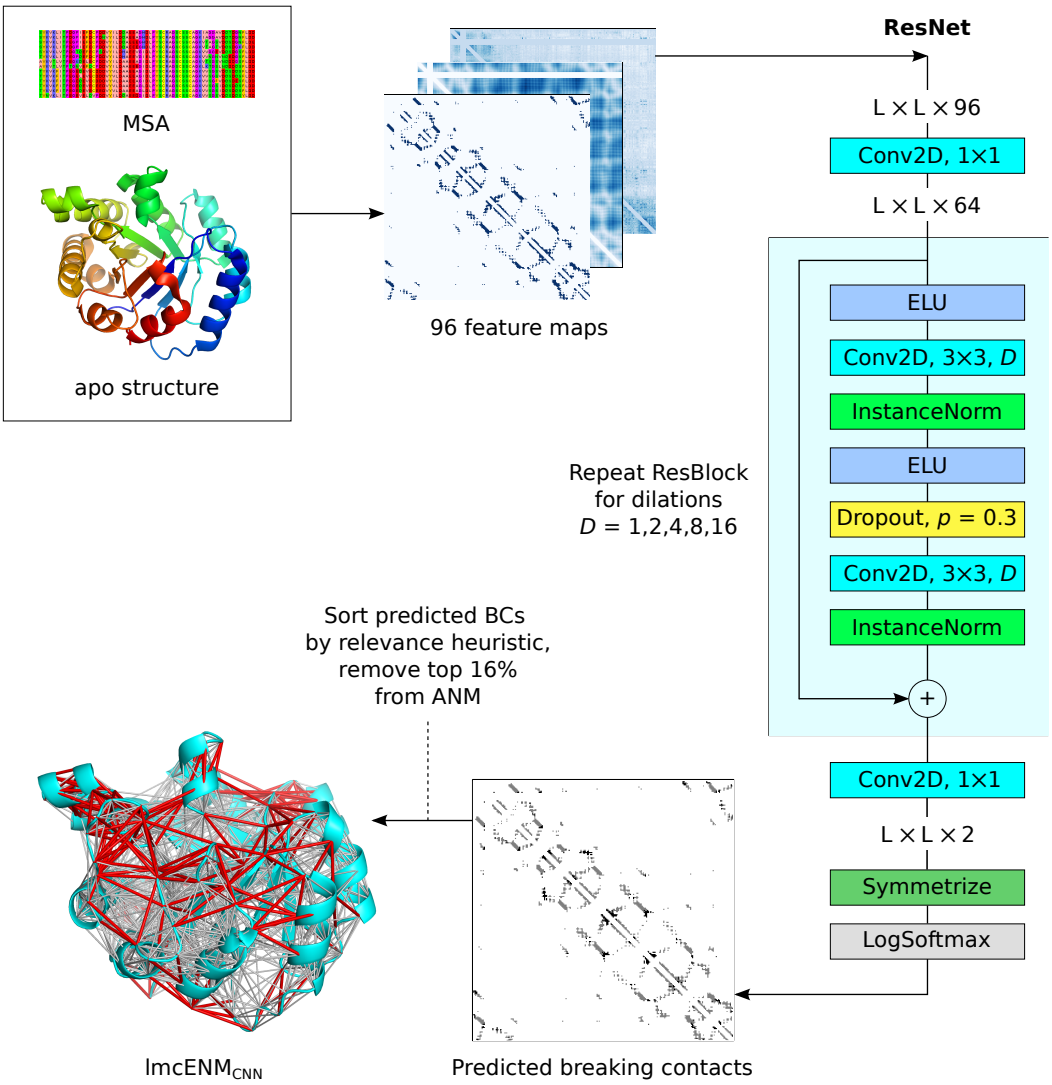

Figure 4.2: Final convolutional ResNet architecture. 96 feature maps are generated from the apo structure and its multiple sequence alignment (MSA). The ResNet first compresses them to 64 feature maps and then passes them through five residual blocks with two convolutional layers each and increasing dilation values. The outputs of the model are sorted by a special scoring function called relevance heuristic (see Section 5.4) and the top 16% predicted breaking contacts removed from the ANM, resulting in *lmc*ENM$_{CNN}$. Finally, standard NMA is performed on *lmc*ENM$_{CNN}$. Removed breaking contacts are shown in red, maintained contacts in grey. Protein renderings created using PyMOL [23], MSA illustration from `https://bioinf.comav.upv.es/courses/biotech3/theory/multiple.html`

structure prediction method. This block architecture was more stable and reliable than any tested variation of it.

Also similar to trRosetta, symmetry is enforced at the end of the network by adding the transpose in the width and height dimension to the model output.

To handle class imbalance, different weights are applied to the two classes (breaking and maintained) in the loss function (see Section 4.6), a standard technique. A weight of $1/0.15 \cong 6.67$ for the breaking contact class relative to the non-breaking contact class produces balanced results. It is of note that this is approximately the inverse of the top percent of contacts selected.

Because examples vary in their width and height, mini-batching or SGD cannot be used and samples are processed individually.

The Adam optimizer [64] with standard learning rate (0.001) is used. I combined Adam with Adaptive Sharpness-Aware Minimization (ASAM) [31, 71] with neighbourhood size $\rho = 2.0$, addition of which moderately improved classifier metrics and greatly improved validation ENM metrics, indicating significantly improved generalization. Especially the coupled local motion (CLM) class benefited: maximum $CO(10)$ values (see Subsection 4.6.3) increased about 40%. The value of $\rho = 2.0$ performed best among $\{1.0, 2.0, 4.0\}$. While without ASAM, not compressing the features didn't have a noticeable effect, in combination with ASAM this resulted in a tenfold increase in validation loss. I hypothesize that this is connected to the high ρ, which might be incompatible with the increased dimensionality. A test with $\rho = 0.2$ showed the loss increase when not compressing features to be much less. Due to time constraints, I didn't investigate this further.

The model and the full pipeline is implemented independently of the original *lmc*ENM$_{\text{SVM}}$ pipeline and is written completely in Python 3. PyTorch [98] is used for deep learning, ProDy [12] is used for ENM calculations, and NetworkX [41] for graph operations. Preprocessing steps use BioPython [20], ProDy, atomium [54] and NumPy [43]. Feature generation involves numerous external programs, including DSSP [58], STRIDE [46], POPS [32], MSMS [107], SymD [117], PyRosetta [17], and fpocket [73].

For ENM calculations, ProDy needs protein chains to be continuously numbered and without gaps. The full alignment is mapped onto such a numbering and predicted BCs in this new numbering filtered again so that they have a minimum sequence separation of four.

Predicted BCs are sorted using a relevance heuristic based on model score and apo distance (see Section 5.4) and in this order attempted to be removed from the ANM$_{\text{minDeg4}}$ of the apo structure (see Subsection 3.1.1). Predicted BCs are only removed if doing so will not violate Jeong stability (see Subsection 3.1.1). This results in *lmc*ENM$_{\text{CNN}}$, on which NMA can then be performed.

4.5 Features

The features used in the final model are listed in Table 4.3. For each feature, I specify whether it is residue-level (R) or pairwise (P), the type of its values, its size in the feature vector and a description, if necessary. Categorical features are one-hot encoded. Most fundamental features from *lmc*ENM$_{\text{SVM}}$ are reused, others added or modified. Sometimes, for features that were one-hot encoded into different ranges in *lmc*ENM$_{\text{SVM}}$, I used the raw discrete or continuous value instead. Results of adding new features are presented in Section 5.5. DSSP is used instead of STRIDE for secondary structure assignment. Owing to the fact that multichain proteins are now present in the dataset, I include a binary interchain contact feature. Besides these, the largest fundamental feature differences are in evolutionary features, which I discuss in the next subsection.

4.5.1 MSA-based features

Sequence-evolutionary information correlates with protein dynamics, especially sequence conservation [81]. Coevolutionary phenomena have also been shown to contain information about protein flexibility [11]. Evolutionary information on the sequence level is commonly captured in multiple sequence alignments (MSAs). From the MSA, many position-specific features can be computed, among others. In the case of *lmc*ENM$_{\text{SVM}}$, the features used to capture evolutionary information are mutual information (MI) and sequence conservation as measured by Jensen-Shannon Divergence (JSD) [30].

For *lmc*ENM$_{\text{CNN}}$, I used different MSA-derived features, closer to [81]: Shannon entropy [30], the MI after applying average product correction (APC) [28], and the MI after applying both minimum entropy normalization [88] and APC. Unmodified MI was not included.

Additionally, I introduce Direct Information (DI) [128] as a novel feature, which has proven to be superior to MI for protein contact prediction [87]. I hypothesized that the statistically clearer DI might be superior to MI for protein BCP as well. As I didn't implement JSD, the impact of the sum of these changes cannot be tested, but the impact of leaving out DI is tested in Section 5.5.

MSAs were generated using HHblits 3.3.0 against the `uniclust30_2018_08` database using the parameters `-mact 0 -n 3 -diff inf -cov 60 -maxfilt 500000`. Shannon entropy, MI variants, and DI were calculated using ProDy.

Feature	Cat.	Values	Size	Description
Chemical type	R	categorical	2×4	AAs can be nonpolar, polar, basic, or acidic
Secondary structure type	R	categorical	2×8	8-state SS type, calc. using DSSP
3D C_α distance	P	continuous	1	
Residues in contact	P	binary	1	3D C_α distance $\leq d_c$
Φ dihedral angle	R	continuous	2×1	
Ψ dihedral angle	R	continuous	2×1	
ω angle	R	continuous	2×1	Dihedral angle over the peptide bond
χ_1 angle	R	continuous	2×1	First sidechain dihedral angle, 0 for ALA, GLY
Normalized sequence position	R	continuous	2×1	Defined as i/L and j/L
Interchain pair	P	binary	1	Residues part of different chains
Inter-SSE pair	P	binary	1	Residues part of different SSEs
Relative solvent accessible surface area (rASA)	R	continuous	2×1	Calc. using DSSP
Solvent accessible surface area (SASA)	R	continuous	2×1	Calc. using POPS
Normalized SASA	R	continuous	2×1	Calc. using POPS
Solvation free energy (SFE)	R	continuous	2×1	Calc. using POPS
C_α half-sphere exposure (HSE)	R	continuous	2×3	HSE based on the approximate C_α-C_β vectors using three consecutive C_α positions[2]

[2] See `https://biopython.org/docs/latest/api/Bio.PDB.HSExposure.html#Bio.PDB.HSExposure.HSExposureCA`

Feature	Cat.	Values	Size	Description
C_n half-sphere exposure (HSE)	R	continuous	2×1	Residue exposure as number of C_α-atoms around its C_α-atom[3]
Residue depth	R	continuous	2×1	Calc. using BioPython
Avg. B-factor of all residue atoms	R	continuous	2×1	
Avg. B-factor of side chain atoms	R	continuous	2×1	-1 for GLY
B-factor of C_α-atom	R	continuous	2×1	
Corrected mutual information (MI)	P	continuous	1	Calc. using ProDy, average product correction[4]
Normalized and corrected MI	P	continuous	1	Calc. using ProDy, minimum entropy normalization[5]
Shannon entropy	R	continuous	2×1	Calc. using ProDy[6]
Direct information (DI)	P	continuous	1	Calc. using ProDy
Hydrogen bonding count	R	discrete	2×2	Number of donor and acceptor hydrogen bonds
Pairwise hydrogen bonding	P	binary	1	Hydrogen bond between both residues exists
Residue in symmetric part	R	binary	2×1	Calc. using SymD, see [99] for details
Distance to symmetry plane	R	continuous	2×1	Calc. using SymD, see [99] for details

[3] See `https://biopython.org/docs/latest/api/Bio.PDB.HSExposure.html#Bio.PDB.HSExposure.ExposureCN`
[4] See `http://prody.csb.pitt.edu/manual/reference/sequence/analysis.html#prody.sequence.analysis.applyMutinfoCorr`
[5] See `http://prody.csb.pitt.edu/manual/reference/sequence/analysis.html#prody.sequence.analysis.applyMutinfoNorm`
[6] See `http://prody.csb.pitt.edu/manual/reference/sequence/analysis.html#prody.sequence.analysis.calcShannonEntropy`

Feature	Cat.	Values	Size	Description
Contact with any pocket	R	binary	2×1	All pocket features calc. using fpocket. Top 20 pockets are considered.
Contact with top-ranking pocket	R	binary	2×1	
Number of contacts with pocket	R	discrete	2×1	
Score of pocket	R	continuous	2×1	0 if no contact with any pocket
Druggability of pocket	R	continuous	2×1	0 if no contact with any pocket
Avg. B-factor of pocket	R	continuous	2×1	0 if no contact with any pocket
Hydrophobicity of pocket	R	continuous	2×1	0 if no contact with any pocket
Volume of pocket	R	continuous	2×1	0 if no contact with any pocket
Polarity of pocket	R	continuous	2×1	0 if no contact with any pocket
Charge of pocket	R	continuous	2×1	0 if no contact with any pocket

Table 4.3: Features used in the final model. The feature vector has in total 96 dimensions. Features are either defined on the level of single residues (R), or pairs of residues (P). Their values are categorized as continuous, discrete, categorical, or binary. Their size as number of dimensions in the feature vector is also listed, along with a brief description. More details on most features can be found in [99].

4.6 Evaluation metrics

To assess the performance of the model, several metrics are calculated during training. After each epoch, training and validation losses, classifier metrics, as well as ENM metrics are computed. ENM metrics are calculated starting with epoch ten.

4.6.1 Loss function

The **cross-entropy loss** is used, a combination of the softmax function and negative log-likelihood loss. Let y be the model output with shape $L \times L \times 2$ and y_1 and y_0 the output describing the breaking and maintained contact classes, then the loss $\mathcal{L}(y, c)$ of class c is given by

$$\mathcal{L}(y, c) = -\ln\left(\frac{e^{y_c}}{\sum_j e^{y_j}}\right) = -\ln\left(\frac{e^{y_c}}{e^{y_0} + e^{y_1}}\right). \tag{4.1}$$

Then, class weighting and masking is applied and the mean calculated. Let w_1 and w_0 be the weights of the respective classes and $M \in \{0,1\}^{L\times L}$ be the masking matrix, then the following equation yields the final weighted loss scalar:

$$\bar{\mathcal{L}}_w(y) = \text{mean}\Big(\text{flatten}\Big(\left(\begin{bmatrix} w_0 & w_1 \end{bmatrix} \begin{bmatrix} \mathcal{L}(y,0) \\ \mathcal{L}(y,1) \end{bmatrix}\right) \cdot M\Big)\Big). \tag{4.2}$$

The masking matrix masks positions with sequence separation less than four, gaps in the sequence alignment, and positions that are not apo contacts, as noncontact positions are irrelevant for the loss.

4.6.2 Classifier metrics

Binary classifier metrics are calculated on the basis of absolute numbers of TP (true positives), FP (false positives), TN (true negatives), and FN (false negatives). I calculate the base metrics **accuracy**, **precision**, and **recall**:

$$\textsc{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad \textsc{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \textsc{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{4.3}$$

Accuracy, as defined above, is useful in problems where the classes are equally distributed, but the BCP problem exhibits strong class imbalance. Instead, the harmonic mean of precision and recall, called the $\mathbf{F_1}$ score, is used as the primary measure of classification performance:

$$\text{F}_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} = \frac{2 \cdot \textsc{Prec} \cdot \textsc{Rec}}{\textsc{Prec} + \textsc{Rec}}. \tag{4.4}$$

Additionally, I calculate the area under the Receiver Operating Characteristic (ROC) curve, called **AUROC**, as a second summary metric. It is also used in the evaluation of *lmc*ENM$_{\text{SVM}}$. For a concise description of the AUROC, refer to [99].

4.6.3 ENM metrics

After each epoch, I calculate two ENM metrics to judge the current effect the classifier has on ENMs. They are calculated based on a given set of predicted breaking contacts, and involve creating several modified ENMs for all proteins in the validation set, computing their normal modes, and then comparing them to their respective baseline $\text{ANM}_{\text{minDeg4}}$. This frequent evaluation of ENM performance is necessary because the metrics can fluctuate quite widely from epoch to epoch, and so no step can be ignored. It also allows to observe the evolution of ENM performance as training progresses, and its relationships to other metrics.

First, a set of predicted breaking contacts needs to be derived from the classifier probabilities. In developing $lmc\text{ENM}_{\text{SVM}}$, a constant number of top scoring breaking contacts, threshold values, and constant top scoring percentages of breaking contacts were tested, and eventually the top 16% of predicted breaking contacts were chosen.

For $lmc\text{ENM}_{\text{CNN}}$, I evaluate two strategies for choosing breaking contacts:

- choosing the **argmax** over the two classes (equivalent to a threshold of > 0.5). This allows for a flexible percentage of predicted BCs per protein.
- choosing the **top scoring percent** $p_{\text{top}} \in \{14, 15, 16\}$. As the average fraction of breaking contacts is slightly lower in my new dataset, no higher percentages were tested. Note that at the end of the evaluation, I realized that the number of removed BC had been calculated based on all initial contacts, i.e. including those with sequence separation less than four. As the number of apo contacts with sequence separation less than four is quite substantial, this means that actually, about 20-25% of BC with sequence separation ≥ 4 had been removed. The exact number varies by protein. I repeated the final run with the fixed calculation but no definite differences were noticed, hence the evaluation was left as is. The F_1 values were correctly calculated, however.

The first ENM metric I calculate is the $\boldsymbol{\overline{\Delta} CO(10)}$, which I define as the average change in $CO(10)$ after removing the top 16% of predicted BCs from $\text{ANM}_{\text{minDeg4}}$ for all validation set proteins.

Second, I calculate the average difference in $CO(10)$ between $lmc\text{ENM}_{\text{CNN}}$ and $lmc\text{ENM}_{\text{SVM}}$ for all proteins in the intersection of my validation (or holdout) set and the $lmc\text{ENM}_{\text{SVM}}$ 90 protein dataset. I call this metric the $\boldsymbol{\overline{\text{vs}} CO(10)}$, "vs" for "versus". A value of zero would indicate that in terms of ENM end results, on average, $lmc\text{ENM}_{\text{CNN}}$ performs on par with $lmc\text{ENM}_{\text{SVM}}$ for the shared proteins in the datasets of both methods.

Especially at the beginning of training, it can happen that the model predicts a set of BCs that result in an unstable ENM, i.e more than six zero eigenvalues. It can even happen that the contact graph breaks into multiple components, unable to be stabilized by Algorithm 1. In case of instability, I iteratively lower the percentage of top BCs that are attempted to be removed by 0.5% until the ENM is stable. The final successful percentage is reported.

Even though I don't evaluate any further ENM metrics in this thesis, all single ENM metrics from [99] are already implemented.

5 Results and Discussion

In this chapter, I present and analyze the results of the experiments performed, show how the final model parameters were chosen and what the effects of different design choices are. From the insights gained in these experiments, I try to formulate general hypotheses about the breaking contact prediction (BCP) problem.

I start by discussing the generalization performance of the model in Section 5.2, analyzed in terms of classifier and ENM performance. Next, in Section 5.3, I investigate the impact of varying the depth of the network and therefore the effective receptive field. During experimentation, I developed the idea of relevance heuristics, simple functions used to rescore the predicted weighting contacts. I investigate their impact in Section 5.4. Finally, I describe some novel classifier features and show the results of an ablation study performed on the model, in Section 5.5.

5.1 Methodology

Before going into the results, a few more words about the methodology of the experiments. Hyperparameter search for a CNN is substantially harder than for a SVM. The first reason for this is that SVMs have hardly any hyperparameters compared to a neural network. And second, SVM training yields exactly one final model for one set of hyperparameters, whereas neural network training in general is an iterative process without a defined end point, yielding a new model after each epoch. Thus, a stopping criterion has to be defined. The usual rule of thumb is to stop when either the (averaged) validation loss or accuracy reached its minimum and begins increasing again ("early stopping"). But in the case of protein BCP, the most relevant metrics are arguably the resulting ENM metrics (see Subsection 4.6.3), which I found to be only loosely correlated to loss and classifier metrics. While ENM metrics are clearly not changing randomly throughout the training process, the model with, for example, the highest F_1, is not in general the model with the highest $\overline{\Delta}CO(10)$. ENM metrics also fluctuate more than other metrics. In evaluating the results, I therefore place special attention on the relationships between the different classes of metrics, focusing on ENM metrics in particular.

Three runs of 120 epochs with fixed seeds were performed for each configuration to probe and account for inter-run model variability. Care was taken to ensure complete reproducibility when using the same seed. Experiments were performed on HPC nodes with NVIDIA Tesla V100 GPUs (32 GB VRAM).

5.2 Model performance on validation and holdout sets

The final model was not selected based on highest F_1, AUROC or maximization of a single ENM metric. Instead, I sought out from all experiments the most balanced performance over the different motion types, with a special emphasis on the local motions, as this was also the goal in the construction of *lmc*ENM$_{\text{SVM}}$. To this end, I filtered the results so that as many motion types as possible showed a positive $\overline{\Delta}CO(10)$. Then I sorted by CLM $\overline{\Delta}CO(10)$ and chose the strongest model.

This revealed the first insight, namely that it is very hard for the model to perform equally good for all types. It seemed to be especially hard to simultaneously achieve positive $\overline{\Delta}CO(10)$ on the CLM and CDM types. Only a small handful out of thousands of epochs showed this combination of metrics, and they performed poorly on other motion types. I assume this happens due to very different patterns and amounts of breaking contacts between the two types. The selected model has a positive $\overline{\Delta}CO(10)$ in four out of the seven motion types.

Set	Type	F_1 CNN	F_1 SVM	AUROC$_{\text{CNN}}$	AUROC$_{\text{SVM}}$
Validation	CLM	**0.38/0.35**	0.30/0.33	**0.81/0.82**	0.62/0.61
	ILM	**0.34/0.33**	0.28/0.31	**0.77/0.77**	0.56/0.58
	CDM	0.24/0.25	**0.25/0.26**	**0.74/0.73**	0.64/0.62
	IDM	**0.34/0.33**	0.22/0.22	**0.80/0.79**	0.62/0.63
	BLM	**0.25/0.22**	0.18/0.23	**0.59/0.58**	0.49/0.51
	OTM	**0.25/0.22**	0.22/0.29	**0.75/0.73**	0.55/0.55
	NSM	**0.29/0.26**	n.a.	**0.79/0.79**	n.a.
	All	**0.31/0.29**	0.26/0.30	**0.77/0.77**	0.61/0.60
Holdout	CLM	**0.42/0.31**	0.31/0.34	**0.82/0.82**	0.57/0.57
	ILM	**0.37/0.33**	0.31/0.29	**0.76/0.77**	0.51/0.50
	CDM	**0.21/0.22**	0.20/0.22	**0.67/0.67**	0.54/0.49
	IDM	**0.37/0.33**	0.19/0.22	**0.75/0.80**	0.65/0.68
	BLM	**0.18/0.18**	0.08/0.08	**0.79/0.79**	0.33/0.33
	OTM	**0.28/0.28**	0.17/0.17	**0.72/0.72**	0.42/0.42
	NSM	**0.34/0.38**	n.a.	**0.79/0.79**	n.a.
	All	**0.33/0.28**	0.25/0.22	**0.76/0.78**	0.54/0.51

Table 5.1: Classifier metrics of the predictions of the final model on validation and holdout sets. Micro-averaged mean and median of the values are listed. Validation values for *lmc*ENM$_{\text{SVM}}$ (F_1 SVM, AUROC$_{\text{SVM}}$) refer to the 90 protein dataset from [99] evaluated using Leave-one-out-cross validation (LOOCV), see Subsection 3.1.3. The column with higher mean is highlighted.

5.2.1 *lmc*ENM$_{\text{CNN}}$ shows improved classifier performance

The overall classifier metrics of the final model are listed in Table 5.1. The CNN clearly outperforms the SVM in terms of F_1 and AUROC. Especially the AUROC is dramatically improved, and now sits well above 0.70 for all motion types except BLM, which is also improved from a random score to almost 0.60 (0.79 in the holdout set). I observed improved classifier metrics across all experiments.

The second insight, strongly supported by the classifier metrics of *lmc*ENM$_{\text{CNN}}$, is that the probabilities of the CNN and thus the model in argmax mode, are much more accurate than those of *lmc*ENM$_{\text{SVM}}$. Having a strong argmax-based classifier makes it possible to predict a flexible percentage of breaking contacts per protein, better accounting for the different motion types and their different amounts of breaking contacts. In all the top% modes combined, only a dozen epochs out of all experiments had positive CLM and CDM $\overline{\Delta}CO(10)$, while almost one hundred argmax-based models had. While the model selected in the end is based on the top 16% mode, the decision was close. The argmax mode was also competitive in all experiments and possibly performed better for domain motion types. The average

Figure 5.1: General training plots of the final model configuration. Epoch 65, which was chosen as the final model, is marked with a dotted vertical line. The selected model has the best all-protein AUROC and second best all-protein F_1, but not the best loss.

percentage of predicted breaking contacts in argmax mode fluctuated strongly in the beginning, but then converged to 10-15%.

To give the reader an overview of the training process, Figure 5.1 shows general training plots, and additionally splits metrics by whether examples are singlechain or multichain proteins. The training plots look stable and show no indication of issues. The techniques used to combat overfitting (ASAM, dropout) seem to be working well, with slight overfitting as indicated by increasing validation loss only visible after 80 epochs. The loss for multichain proteins is quite a bit lower, but this could be due to singlechain outliers, see next paragraph.

5.2.2 *lmc*ENM$_{\text{CNN}}$ shows mixed ENM performance

The next Figure, 5.2, shows the same metrics, but this time split by motion type. Additionally, the $\overline{\Delta}CO(10)$ by motion type over the entire run is shown.

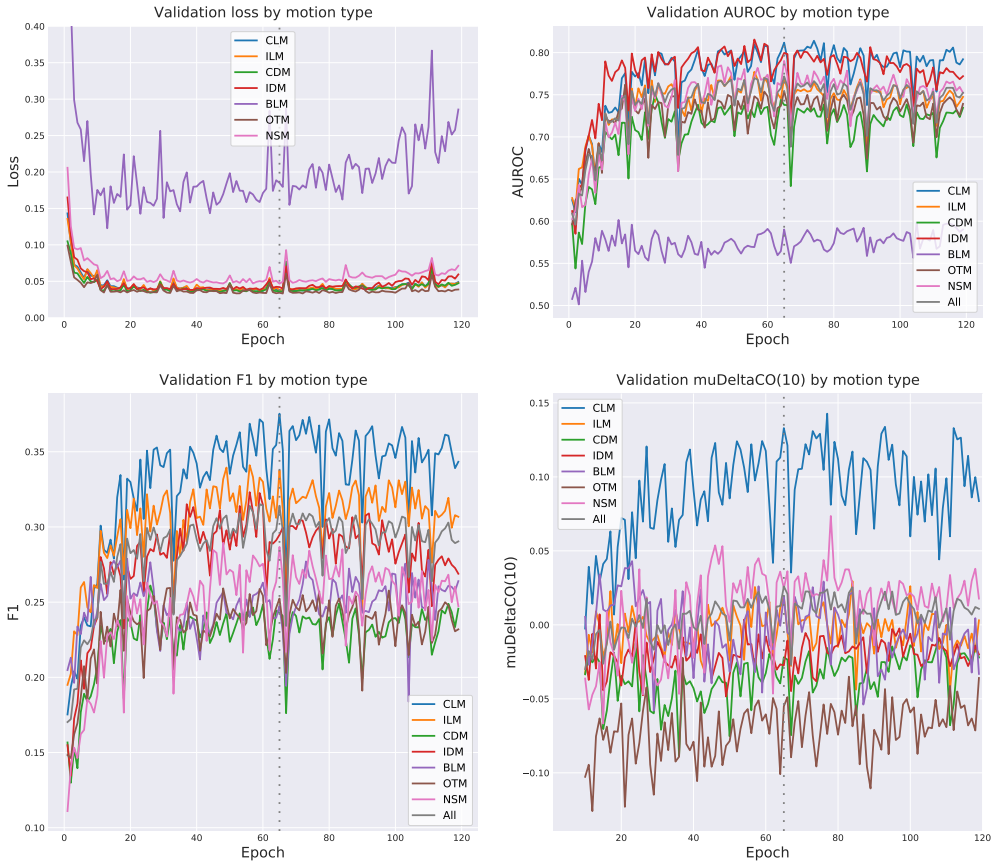

Figure 5.2: Training plots of the final model configuration by motion type. Epoch 65 is marked with a dotted vertical line.

The plots reveal two more things. First, BLM is an outlier in terms of loss and AUROC, but the same effect is not visible in the F_1 plot. I suspect this effect is not very significant, as there are only three BLM proteins in the validation set and therefore not much can be concluded. Still, the effect was consistent over all runs. Visual inspection of the BLM proteins in the training and validation sets was without findings.

Second, the $\overline{\Delta}CO(10)$ plot is quite interesting. While it has a lot of curves, it manages to show the general area where the motion types live. What is also visible, is that clearly, the CLM type performs the best, followed by NSM and ILM. Interestingly, a few epochs later, CLM and NSM show a further increase in $\overline{\Delta}CO(10)$, with NSM greatly improved. As this was at the cost of other motion types, the epoch was not chosen to limit overfitting to the local motion types.

Set	Type	ANM	*lmc*ENM$_{\text{CNN}}$	*lmc*ENM$_{\text{SVM}}$	*mc*ENM	$\overline{\text{vs}}CO(10)$
V	CLM	0.50/0.51	0.63/0.63 (16)	0.67/0.67 (5)	0.69/0.71	**0.02/0.01**
	ILM	0.54/0.52	0.56/0.56 (15)	0.52/0.46 (7)	0.72/0.72	**0.02/-0.02**
	CDM	0.86/0.90	0.84/0.86 (10)	0.86/0.85 (6)	0.91/0.92	0.0/-0.01
	IDM	0.84/0.84	0.81/0.84 (10)	0.87/0.90 (4)	0.88/0.90	**0.02/0.01**
	BLM	0.47/0.51	0.49/0.55 (3)	n.a. (0)	0.62/0.59	n.a.
	OTM	0.66/0.70	0.60/0.66 (7)	0.24/0.24 (1)	0.74/0.72	**0.14/0.14**
	NSM	0.67/0.68	0.70/0.72 (7)	n.a. (0)	0.80/0.84	n.a.
	All	0.64/0.69	0.67/0.70 (68)	0.69/0.72 (22)	0.77/0.81	**0.02/-0.00**
H	CLM	0.46/0.35	0.59/0.63 (7)	0.55/0.61 (7)	0.64/0.65	**0.03/0.06**
	ILM	0.53/0.47	0.51/0.46 (5)	0.53/0.53 (5)	0.66/0.67	-0.03/-0.01
	CDM	0.90/0.96	0.88/0.93 (4)	0.92/0.96 (4)	0.93/0.96	-0.04/-0.04
	IDM	0.87/0.87	0.84/0.87 (4)	0.85/0.87 (4)	0.89/0.88	-0.02/-0.00
	BLM	0.79/0.79	0.72/0.72 (1)	0.76/0.76 (1)	0.87/0.87	-0.04/-0.04
	OTM	0.69/0.69	0.67/0.67 (2)	0.68/0.68 (2)	0.82/0.82	-0.01/-0.01
	NSM	0.57/0.63	0.57/0.56 (3)	n.a. (0)	0.74/0.73	n.a.
	All	0.65/0.65	0.66/0.69 (26)	0.68/0.72 (23)	0.76/0.80	-0.01/-0.01

Table 5.2: $CO(10)$ metrics of the predictions of the final model on validation and holdout sets, compared to ANM$_{\text{minDeg4}}$ (listed as ANM), *lmc*ENM$_{\text{SVM}}$, and *mc*ENM. Mean and median of the values is listed. Columns ANM$_{\text{minDeg4}}$, *lmc*ENM$_{\text{CNN}}$, and *mc*ENM are calculated on the full validation and holdout sets. Columns *lmc*ENM$_{\text{SVM}}$ and $\overline{\text{vs}}CO(10)$ are calculated on the intersection of V/H sets and the 90 protein dataset from [99], see Section 4.3 and Subsection 4.6.3. This intersection didn't contain any NSM proteins (add. no BLM in the case of validation), so their comparison metrics cannot be calculated. Rows where $\overline{\text{vs}}CO(10) > 0$ are marked in bold, indicating superior performance of *lmc*ENM$_{\text{CNN}}$.

The ENM metrics react strongly to changes in classifier metrics. Classifier and ENM metrics seem to be well correlated in this run, which was not always the case.

This leads me to Table 5.2, showing the ENM metrics of $lmc\text{ENM}_{\text{CNN}}$ as measured by $CO(10)$ and compared to $lmc\text{ENM}_{\text{SVM}}$, the baseline $\text{ANM}_{\text{minDeg4}}$, and the ground truth mcENM, split by motion type.

Compared to the classifier metrics, the ENM metrics results are more mixed. While $lmc\text{ENM}_{\text{CNN}}$ outperforms $lmc\text{ENM}_{\text{SVM}}$ on the validation set, it struggles on the holdout set (although $lmc\text{ENM}_{\text{SVM}}$ seems to struggle, too). What is positive, is that $lmc\text{ENM}_{\text{CNN}}$ performs well for the local motions across the board, with the exception of holdout ILM. As shown in Figure 5.2, a later epoch would have increased CLM even further. The mixed performance on validation and holdout sets reflects the difficulty of finding a $lmc\text{ENM}_{\text{CNN}}$ model that robustly performs well across all motion types. The data forming the columns ANM, $lmc\text{ENM}_{\text{CNN}}$, $lmc\text{ENM}_{\text{SVM}}$, and mcENM is visualized in the form of boxplots in Figure 5.3.

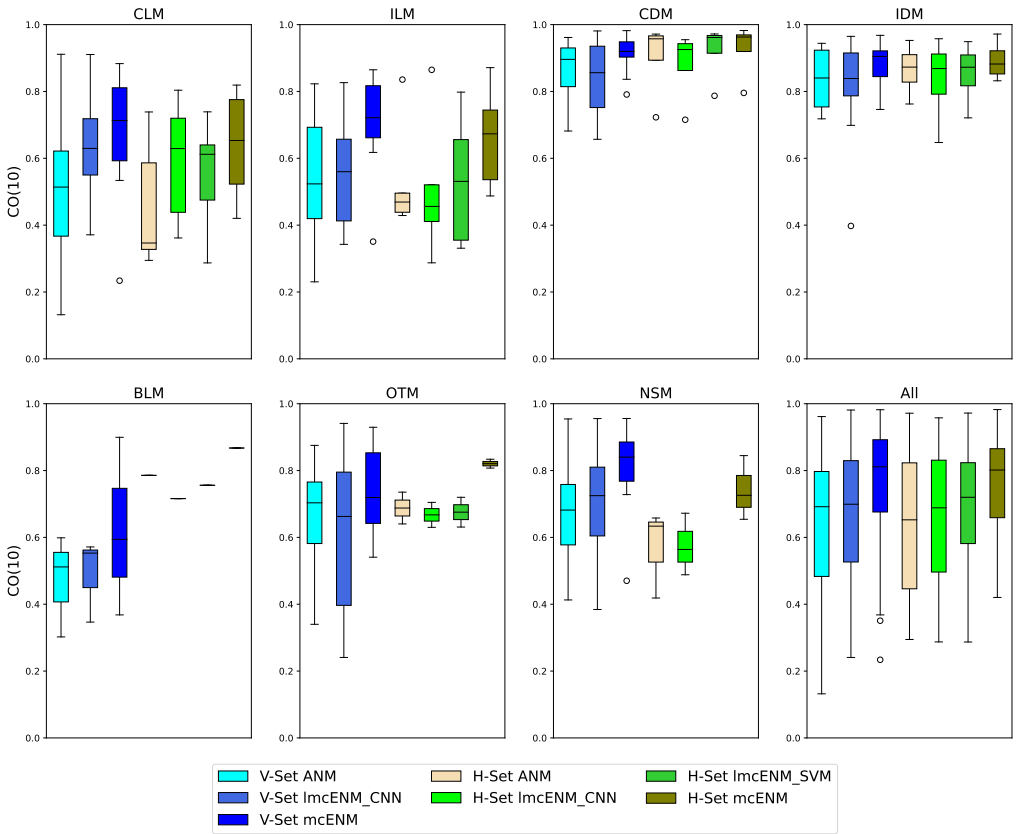

Figure 5.3: Boxplots of ENM performance as measured by $CO(10)$ by motion type, for validation and holdout sets. For the holdout set, additionally $lmc\text{ENM}_{\text{SVM}}$ is shown.

What remains unclear, is why *lmc*ENM$_{\text{CNN}}$ does not translate the considerable and consistent increases in raw classification performance into equally large improvements on the ENM metrics. Fundamentally, this means that the breaking contacts predicted by *lmc*ENM$_{\text{CNN}}$ are on average more correct, but less relevant than those of *lmc*ENM$_{\text{SVM}}$. There are a number of possible contributions to this:

- Fundamental feature differences. There could be differences introduced by the reimplementation of the fundamental features that are not beneficial. Forgoing one-hot encoding of many features in favor of raw values could play a role. The sidechain contact feature was not implemented. Furthermore, no whole protein features are used in *lmc*ENM$_{\text{CNN}}$, and the receptive field most of the time does not cover the whole contact map.

- High-level feature differences. The high-level features learned by the CNN, while clearly impressively effective, are likely still not as balanced and expressive as the manually engineered graph features used in *lmc*ENM$_{\text{SVM}}$.

- Sequential locality as induced by the 2D data format necessary for the CNN could limit performance by making propagation of relevant data difficult (see also next section).

- The filters not applied to construct the larger dataset could have led to unintended side effects. There was especially no filter for the type and size of ligand in the pairs.

- A larger dataset is still desirable for deep learning. The more data, the better. More data can help further increase generalization and stabilize ENM metrics to facilitate better model selection. The impact of limited dataset size however would have to be carefully analyzed to identify the reducible parts of bias and variance [94].

For three proteins in the validation set and two in the holdout set, the percent of removed BCs had to be iteratively lowered for them to become stable. Their metrics are listed in Table 5.3. With the exception of 1ms3A and 3eugA, who have poor F_1 and AUROC values, the model metrics are in the normal range. In the case of 1ms3A, the loss is surprisingly low for the F_1 value (which was calulated based on argmax mode). The OTM scores are likely dragged down in general by 1ms3A. Interestingly, for 2o0kA, better results than *lmc*ENM$_{\text{SVM}}$ are achieved, even though (or because) only 6.5% predicted BCs were removed.

Apo PDB ID	2vclA	1xtiA	1ms3A	2o0kA	3eugA
Set	V	V	V	H	H
Motion Type	CLM	ILM	OTM	CLM	IDM
Stable %	15.0	10.5	10.5	6.5	4.0
Loss	0.023	0.031	0.014	0.016	0.093
F_1	0.36	0.40	0.14	0.30	0.17
AUROC	0.84	0.77	0.67	0.87	0.54
ANM $CO(10)$	0.54	0.42	0.74	0.65	0.76
$\Delta CO(10)$	0.19	-0.00	-0.50	0.16	-0.12
vs$CO(10)$	n.a.	n.a.	n.a.	0.06	-0.07

Table 5.3: Proteins whose amount of removed predicted BCs had to be reduced to achieve a stable $\text{ANM}_{\text{minDeg4}}$. The amount of removed predicted BCs was lowered in steps of 0.5% until the number of zero eigenvalues was equal to six.

5.2.3 Case Studies

I want to close this discussion on general model performance with case studies showing success and failure cases of breaking contact prediction by *lmc*ENM_{CNN}.

A target where *lmc*ENM_{CNN} works really well is 2p52A from the holdout set. It is a 187 amino acid protein with motion classified as CLM. For this protein, *lmc*ENM_{SVM} only identifies 8 true positive breaking contacts, corresponding to a precision of 0.07 and recall of 0.14. Due to the large amount of false positives, *lmc*ENM_{SVM} does not manage to improve the $CO(10)$ value of this protein over $\text{ANM}_{\text{minDeg4}}$, reducing it by -0.8%. On the other hand, *lmc*ENM_{CNN} achieves a precision of 0.24 and recall of 0.50, managing to increase $CO(10)$ by 19%. Other examples where *lmc*ENM_{CNN} achieves an improvement while *lmc*ENM_{SVM} deteriorates ENM performance are 1dx9C (OTM, 168 AA, precision/recall: 0.15/0.31) and 1a8dA (ILM, 451 AA, precision/recall: 0.16/0.41), both from the validation set. Here, *lmc*ENM_{SVM} had $\Delta CO(10)$'s of -10.2% and -8.3% while *lmc*ENM_{CNN} achieves +4% and +9% respectively, aided by stronger classification performance (precision/recall: 0.24/0.51 and 0.26/0.66). Figure 5.4 shows the breaking contact networks of targets 2p52A and 1a8dA.

One target for which *lmc*ENM_{CNN} does not work at all is the already mentioned 1ms3A. It is a 632 AA long, predominantly β protein with OTM motion that can be described as domain-like. The $CO(10)$ is drastically reduced by *lmc*ENM_{CNN}, from 0.74 to 0.24. The amount of removed predicted BC had to be lowered to 10.5% to achieve a stable ENM. Above 10.5%, there were seven zero eigenvalues. The classification performance is low for 1ms3A, with an F_1 of only 0.14. It is by a

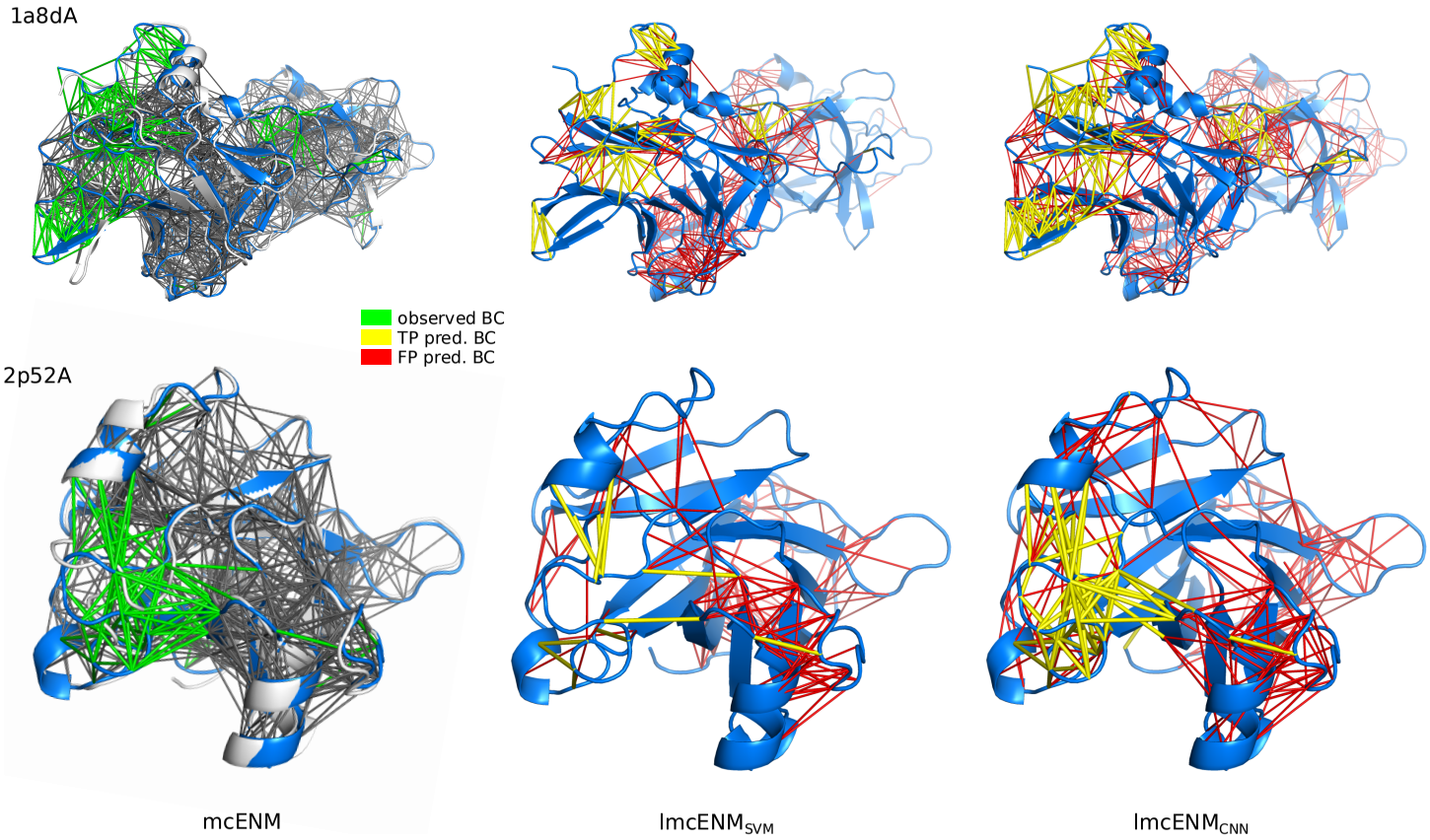

Figure 5.4: Breaking contact networks of targets 1a8dA/1fv3B (top) and 2p52A/2iomA (bottom) produced by *mc*ENM (left), *lmc*ENM$_{\mathrm{SVM}}$ (middle), and *lmc*ENM$_{\mathrm{CNN}}$ (right). For *mc*ENM, the apo conformation is shown in blue, the holo conformation in white, ANM contacts with a minimum sequence separation of four in grey, and observed breaking contacts in red. True positive (TP) predicted breaking contacts are shown in yellow and false positives (FP) in red. The two depicted targets are examples where *lmc*ENM$_{\mathrm{CNN}}$ finds significantly more true positives than *lmc*ENM$_{\mathrm{SVM}}$, resulting in higher $CO(10)$ values. It is possible that *lmc*ENM$_{\mathrm{CNN}}$ predicts clusters of breaking contacts especially well.

large margin the worst performing protein in the dataset. Interestingly, the argmax mode performs even worse for this protein ($\Delta CO(10)$ -0.51), even though only 5% breaking contacts are predicted and removed. This protein however is also one of the few where even the ground truth, *mc*ENM, worsened ENM performance, albeit only by 1.5%. There is no data available for *lmc*ENM$_{\text{SVM}}$. To further understand why this protein performs so bad, I performed a sensitivity analysis, iteratively removing more predicted breaking contacts. The results of this are shown in Figure 5.5.

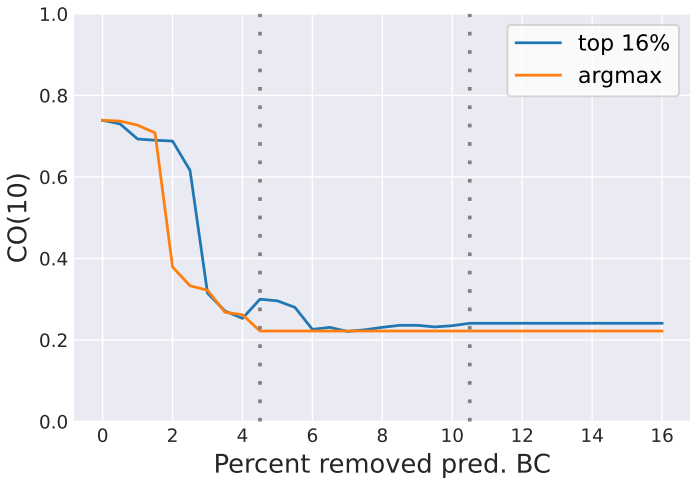

Figure 5.5: Sensitivity analysis of validation set target 1ms3A/1ms0B. The horizontal axis represents the amount of predicted breaking contacts that are removed, and the vertical axis shows the resulting $CO(10)$. The stability limits of argmax (4.5%) and top 16% mode (10.5%) are marked with vertical lines. The $CO(10)$ rapidly decreases, even when only small amounts of predicted breaking contacts are removed.

The rapid deterioration of ENM performance is something that was not seen in the sensitivity analysis in [99]. I visually inspected the different contact networks for 2.5% and 3.0%, a difference that resulted in a loss of more than 30% $CO(10)$. The protein consists of two domains connected by an α-helix, and it seems like part of the contacts that were removed in the 3% network but not in the 2.5% network were on both ends of the helix, as well as at loop regions connecting the two domains. This situation is depicted in Figure 5.6. I hypothesize that removal of these contacts could have destabilized the connection between the two domains and led to large, unwanted domain hinge motions. This hypothesis would need to be investigated further by looking at the resulting normal modes.

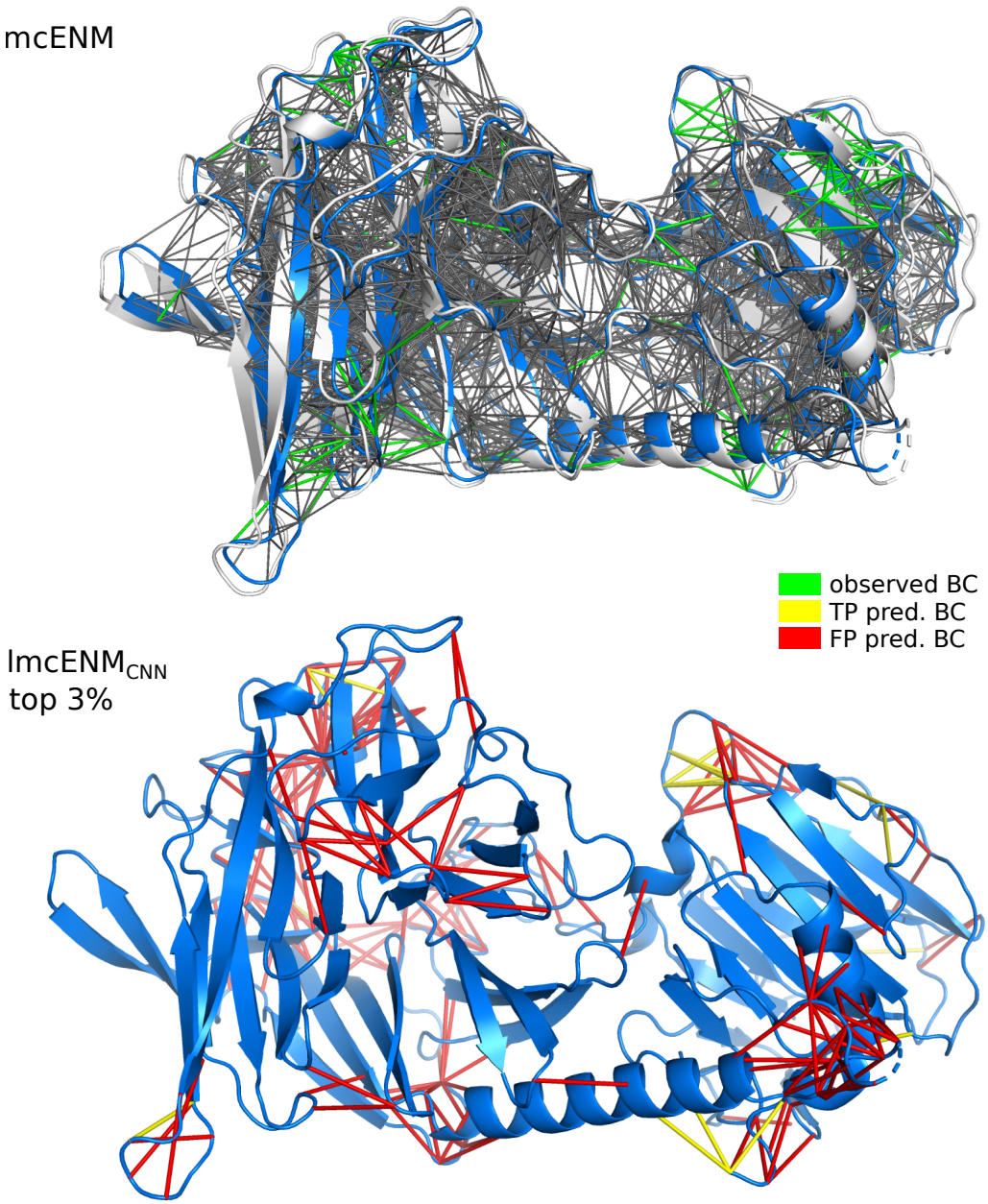

Figure 5.6: Breaking contact networks for target 1ms3A/1ms0B produced by *mc*ENM (top) and *lmc*ENM$_{\text{CNN}}$ (bottom). For *mc*ENM, the apo conformation is shown in blue, the holo conformation in white, ANM contacts with a minimum sequence separation of four in grey, and observed breaking contacts in red. For *lmc*ENM$_{\text{CNN}}$, the top 3% predicted BC are shown, with true positive predicted breaking contacts in yellow and false positives in red. The predicted BCs at the ends of the α-helix connecting both domains could be responsible for the fast deterioration of $CO(10)$ seen in the sensitivity analysis (see Figure 5.5), as well as the instability of the *lmc*ENM$_{\text{CNN}}$ network for this target when using higher percentages of predicted BCs.

5.3 Prediction context experiment

As discussed in Subsection 2.2.3, the network's effective receptive field (ERF, see Figure 2.5) and therefore the sequential context available to classify an apo contact can easily be varied by adding or removing layers. An important experiment is therefore to find the optimal network depth and more generally investigate the influence of different ERFs.

Table 5.4 shows the layerwise ERF of one "dilation block", that is, a sequence of residual blocks containing two convolutions each (see Figure 4.2), whose dilation D is doubled in each residual block, up to $D = 16$ in the fifth ResBlock. The ERF values were calculated using the recurrence relation

$$\mathrm{rf}_i = \begin{cases} 1 & \text{if } i = 0 \\ \mathrm{rf}_{i-1} + (F_i - 1) \cdot \prod_{j=1}^{i-1} S_j \cdot D_i & \text{otherwise,} \end{cases} \tag{5.1}$$

where rf_1 is the receptive field of the first layer and F_i, S_i, and D_i are the filter size side length, stride, and dilation values of the ith convolutional layer.

ResBlock	1		2		3		4		5	
ConvLayer	1	2	1	2	1	2	1	2	1	2
Dilation D	1	1	2	2	4	4	8	8	16	16
Output ERF	3	5	9	13	21	29	45	61	93	125

Table 5.4: Layer-wise effective receptive field (ERF) of one dilation block with maximum dilation of $D = 16$, corresponding to five ResBlocks.

Therefore, one full dilation block amounts to an ERF of 125×125. As this ERF is sequential and not spatial, it cannot be directly compared to the neighbourhood graph and its constituents (see Subsection 3.1.3).

I tested the following network depths: two, three and four ResBlocks (resulting in dilation sequences $D_{\mathrm{seq}} \in \{(1, 2), (1, 2, 4), (1, 2, 4, 8)\}$), as well as one, two, and three full dilation blocks. These configurations correspond to ERFs of 13, 29, 61, 125, 250, and 375, respectively. Table 5.5 shows the effects of changing ERF on the best values of various metrics.

I found that with increasing depth, the network fits the training set faster, and consequently also begins overfitting earlier. This means that best values are attained in progressively earlier epochs, but also the values are worse in general. This is especially visible for loss and AUROC values. The ENM metrics reach their optimum with a bit more depth. All best values were achieved with networks with ERF ≤ 61.

ERF	Loss ↓		AUROC ↑		F_1 ↑		$\overline{\Delta}CO(10)$ ↑		$\overline{\text{vs}}CO(10)$ ↑	
13	**0.0424**	86.3	0.7774	97.7	0.3142	96.7	0.0182	89.7	0.0115	96.3
29	0.0425	75.7	**0.7779**	93.0	**0.3200**	91.7	0.0181	96.7	**0.0167**	84.0
61	0.0425	73.3	0.7770	73.0	0.3194	92.3	**0.0192**	83.7	0.0142	73.3
125	0.0431	41.7	0.7694	64.7	0.3134	72.0	0.0183	91.0	0.0157	61.3
250	0.0440	35.3	0.7629	38.7	0.3064	40.3	0.0129	76.7	0.0056	59.3
375	0.0449	28.7	0.7556	25.3	0.3012	36.7	0.0153	47.7	0.0068	41.7

Table 5.5: Effect of increasing ERF on various model metrics. The average best value over three seeds is listed along with the average epoch where it occurred. Predicted breaking contacts were sorted by the baseline relevance heuristic (model score only), see Section 5.4. Arrows indicate whether best is defined as min or max.

While the configuration with one full dilation block, i.e. ERF 125, wasn't the best in any single metric, it clearly came out on top when searching for a balanced model that was still strong for the local motion types, as described in Section 5.2.

It is interesting that increasing the ERF does not lead to improvements visible in single metrics, more so, they are slightly worsened by it. I suspect that the relevant information gained by including more spatially close neighbours is drowned out by many sequential neighbours that have little relevance for classifying the contact, therefore making prediction actually more difficult. The fact that the most useful model was found at the middle of the scale is encouraging, however.

It is important to point out that from this result, one cannot conclude that the relevant prediction context for BCP is severely limited in general. This could only be found out with either a graph-based model (see Section 7.2) or a model with full attention over the contact map (see Section 7.3).

5.4 Relevance heuristics

To protect the stability of the ANM, predicted breaking contacts (BCs) are only removed from $\text{ANM}_{\text{minDeg4}}$ if doing so does not violate Jeong stability, that is, after Algorithm 1 was applied to the contact graph (see Subsection 3.1.1). As implemented in *lmc*ENM_{SVM}, predicted BCs are attempted to be removed one by one. It is easy to see that this procedure is sensitive to the order in which removal of BCs is attempted. A different ordering of the BCs can result in a different set of BCs that can successfully be removed, leading to a different ENM.

Breaking contacts differ in their ability to influence normal modes, as shown by a sensitivity analysis in [99]. Sometimes, a "critical mass" has to be reached before a

significant change takes place, hinting at, for example, clustering of BCs. To build the most expressive ENMs, it is not only important to predict which contacts will break, but also what characteristics make them important and which BCs are most important, in order to prioritize their removal. This is only very indirectly reflected in the SVM and CNN probabilities (see also Section 7.4).

In *lmc*ENM$_{\text{SVM}}$, predicted BCs are sorted by decreasing SVM score. I hypothesized that a different scoring function to sort predicted BCs *not only* based on the model score could improve the resulting ENMs and maybe lead to more balanced results over the motion types. This function should be simple, but at the same time express something relevant for BCP contained in the apo structure. I call these functions **relevance heuristics**.

5.4.1 Implementation and tested heuristics

For the two BC selection modes (argmax and top%), slightly different approaches have to be implement. For top% mode, a $L \times L$ *relevance matrix* is computed and element-wise multiplied to the model output to yield the final scores. From these, a top percentage is selected, 16% for the final model. For argmax mode, the contacts are, by definition, chosen solely on which output class score is higher. The resulting contacts are then re-ranked according to the relevance heuristic. Even though both modes use the same relevance heuristic, the effects are very different. While the top% ENM results can be dramatically changed, the argmax results are affected only to a small degree. This is because in top% mode, a completely different set of predicted BCs can be selected, while, in argmax mode, only the internal order of already selected BCs is changed, resulting in smaller changes. This means that although the isolated effect of changing the internal order of BCs might on average be small, it very much supports the idea of more deeply and explicitly integrating relevance into the learning process, as proposed in Section 7.4. Fully leveraging relevance heuristics in an "argmax"-like mode would be possible by means of a custom threshold, sensibly defined per relevance heuristic.

The relevance heuristics I tested are listed in the following table, 5.6, along with the idea of relevance they reflect.

5.4.2 Relevance heuristics can improve ENM metrics

Validating runs with relevance heuristics enabled partly lead to results very different from the baseline $\mathcal{R}_0$, and the individual heuristics all behaved differently from each other. I describe the observed effects in the following paragraphs.

n	$\mathcal{R}_n(i,j) :=$	Idea
0	$S(i,j)$	Baseline, model score only.
1	$S(i,j) \cdot \max(\lvert i-j \rvert - 3, 0)$	Higher sequence separation is more relevant. The first considered sequence separation, four, has relevance one.
2	$S(i,j) \cdot \mathcal{D}(i,j)$	Higher apo distance is more relevant.
3	$S(i,j) \cdot (d_c + 1 - \mathcal{D}(i,j))$	Lower apo distance is more relevant. The highest possible apo distance, 10Å, has relevance one.

Table 5.6: List of tested relevance heuristics, where i, j are sequence positions, $S(i,j)$ is the model score, $\mathcal{D}(i,j)$ is the Euclidean apo distance of residues i, j, and $\mathcal{R}_n(i,j)$ the relevance function.

Relevance heuristic $\mathcal{R}_1$ strongly improved NSM and IDM, lightly improved BLM pairs, did not significantly alter CDM, ILM, and OTM pairs, and deteriorated the $\overline{\Delta}CO(10)$ of CLM pairs to a heavy degree. CLM, NSM, and IDM curves are shown in Figure 5.7. Interestingly, CLM was improving over the whole 120 epochs and was still doing so at the end of the experiment.

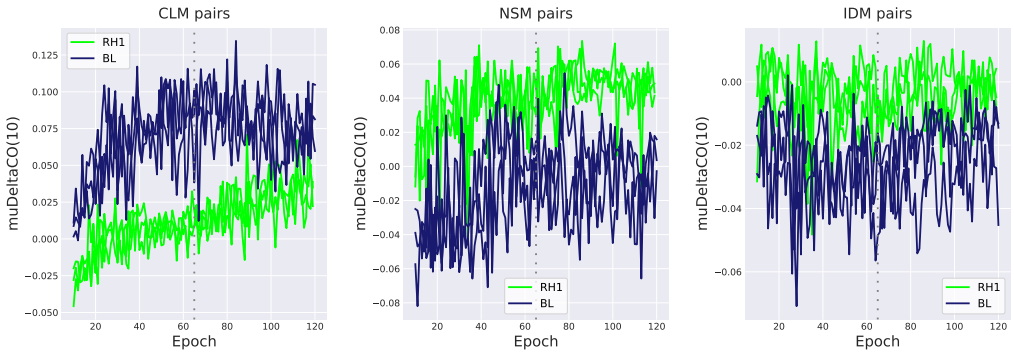

Figure 5.7: Notable ENM effects of the sequence separation based relevance heuristic $\mathcal{R}_1(i,j) := S(i,j) \cdot \max(\lvert i-j \rvert - 3, 0)$. The epochs of all three runs with and without the relevance heuristic are displayed. Epoch 65 is marked with a dotted vertical line. The top 16% of predicted breaking contacts are used.

Relevance heuristic $\mathcal{R}_2$ improved the maximum $\overline{\Delta}CO(10)$ of all motion types and the average of most, leading to a higher overall $\overline{\Delta}CO(10)$. The CLM type saw the most consistent improvement, including producing an epoch with $\overline{\Delta}CO(10) > 0.14$, the highest seen for this motion type. Due to the uniform success of this relevance

heuristic, it was used in the final model. The effects of $\mathcal{R}_2$ on the various motion types are shown in Figure 5.9 in detail.

Relevance heuristic $\mathcal{R}_3$ had another curious effect on BCP, shown in Figure 5.8. While $\mathcal{R}_2$ seemed to shift the $\overline{\Delta}CO(10)$ curves almost uniformly upwards, its "inverse" $\mathcal{R}_3$ more or less compressed them. Learning seemed to slow almost to a halt, and the range of values visited in the $\overline{\Delta}CO(10)$ plots was reduced for all motion types, for some drastically so. For the domain motion, OTM, and NSM types, this was actually greatly beneficial, but the CLM and to a lesser extent ILM types were again impacted negatively.

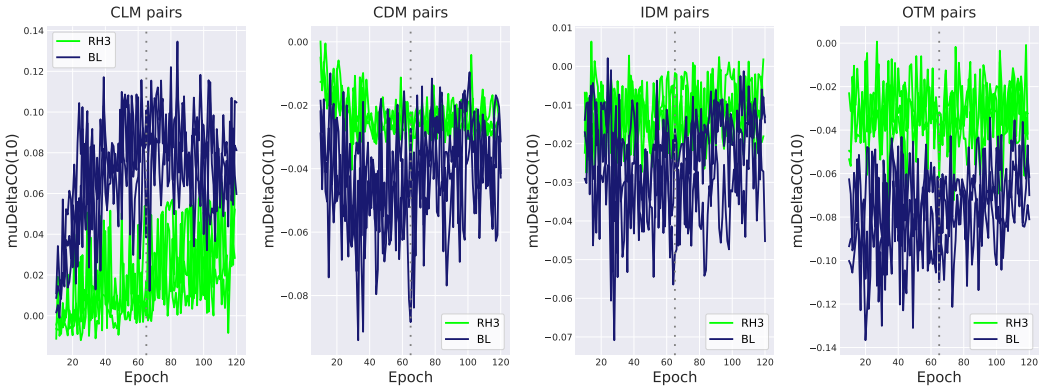

Figure 5.8: Notable ENM effects of the apo distance based relevance heuristic $\mathcal{R}_3(i,j) := S(i,j) \cdot (d_c + 1 - \mathcal{D}(i,j))$ (lower apo distance ranks higher). The epochs of all three runs with and without the relevance heuristic are displayed. Epoch 65 is marked with a dotted vertical line. The top 16% of predicted breaking contacts are used.

The observed drastic effects of the relevance heuristics are in dire need of an explanation. The universally positive effects of $\mathcal{R}_2$ seems to indicate that contacts with higher distance in the apo structure have a higher probability of breaking, and seems to amplify an effect already partly learned by the network. This hypothesis could easily be tested. I want to remind the reader of the relatively narrow range of possible apo distances that influence the heuristic, encompassing about 3.5Å up to $d_c = 10$Å (pairs with apo distance $> d_c$ are ignored anyways). As the involved distances are small, a reasonable hypothesis could be that the effect is related to weakening electrostatic potential, increasing the chance of the distance changing as the interactions are not very strong. Maybe a background distance distribution like the one calculated in [102] could be of help. The background probability of a contact breaking based on its apo distance could be used to build a more refined heuristic.

I can also see how $\mathcal{R}_1$ works against local motions, promoting a kind of sequential "anti-locality" incompatible with the local motions and their clustered BCs. If $\mathcal{R}_2$

amplifies a relationship learned by the network, then $\mathcal{R}_3$ diminishes it, leading to the observed compression of $\overline{\Delta}CO(10)$ curves and their slower change.

Further study of the motion types and their exact definitions is needed. Theoretically as well as empirically grounded explanations for the observed effects could be greatly beneficial for improving *lmc*ENM$_{\text{CNN}}$. An "anti-$\mathcal{R}_1$", i.e. "lower sequence separation is more relevant", would also be possible, for example by inversion, and would be an interesting investigation. The effects of relevance heuristics on classifier metrics also remain to be studied, and would also need an adapted cutoff.

Adding yet another dimension to the idea, relevance matrices can also be multiplied element-wise to the loss matrix prior to its mean reduction, effectively reweighting it. This would promote learning of relevance in the sense of the heuristic, although in quite a crude way.

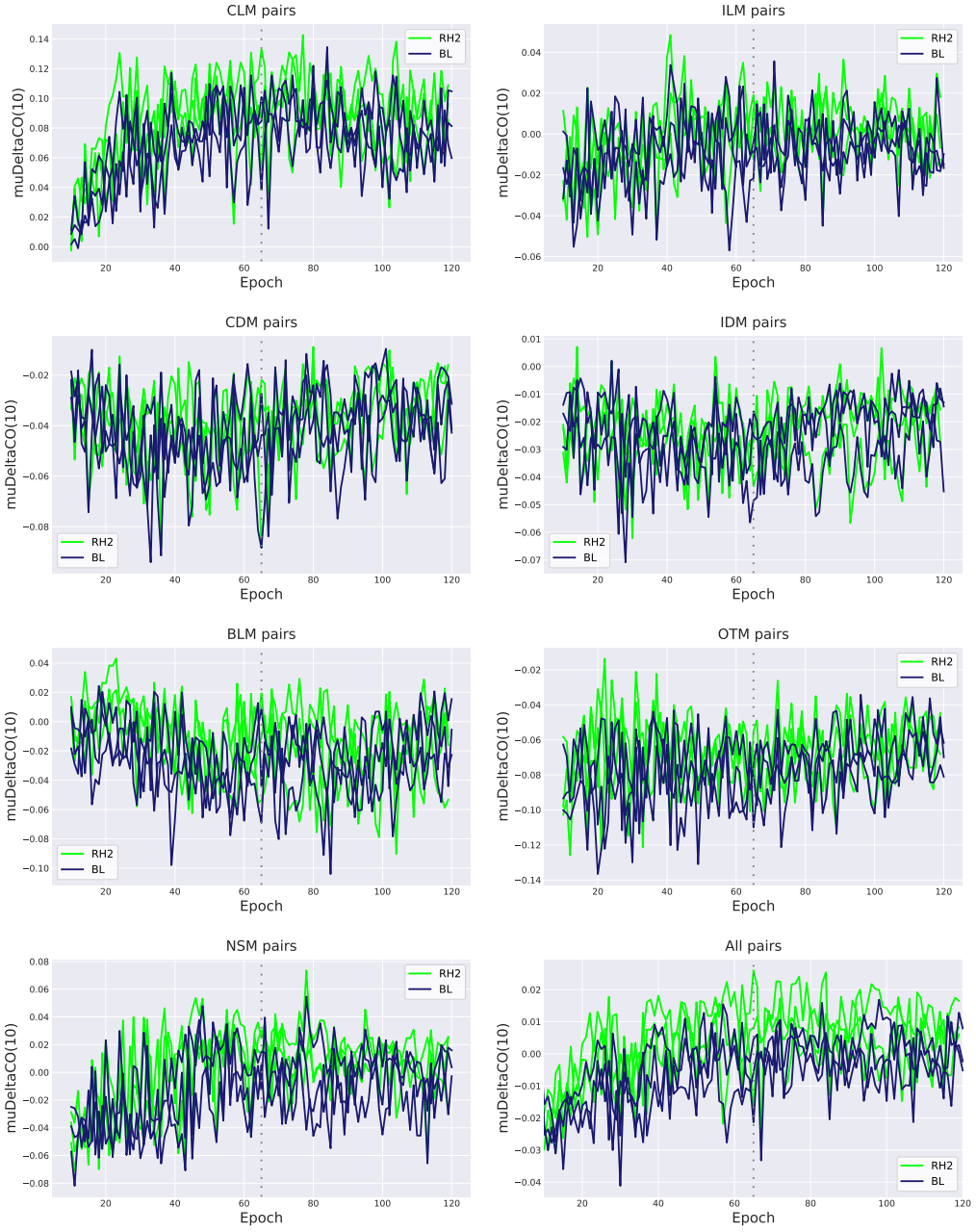

Figure 5.9: Effect of applying an apo distance based relevance heuristic on ENM metrics of motion types. The epochs of all three runs with and without the relevance heuristic are displayed. Using $\mathcal{R}_2(i,j) := S(i,j) \cdot \mathcal{D}(i,j)$ as the relevance heuristic improves $\overline{\Delta}CO(10)$ values for all motion classes and leads to a significant increase in overall $\overline{\Delta}CO(10)$, over the complete training process. Epoch 65 is marked with a dotted vertical line. The top 16% of predicted breaking contacts are used.

5.5 Evaluation of novel features

I added some new fundamental features to *lmc*ENM$_{\text{CNN}}$ that were not used in *lmc*ENM$_{\text{SVM}}$. I investigated the impact of including them by performing an ablation study of the model. While not all novel features clearly improved metrics of the model, their impact on model balance seems to be positive.

Before presenting the results of the ablation study, I give a short description of the novel features and why it might be useful to include them. Direct Information (DI) was already introduced in Subsection 4.5.1.

5.5.1 Residue dihedral angles

Besides the Cartesian coordinates of atoms, their *internal coordinates*, i.e. dihedral angles, are also listed in a PDB file. Including internal coordinates and therefore relative orientation of residues could be a valuable additional data source for breaking contact prediction. I chose to include the three backbone dihedral angles Φ, Ψ, and ω, as well as the first side chain dihedral angle χ_1 as residue-level features.

5.5.2 Atomic B-factors

In protein crystallography, the B-factor (also called Debye–Waller factor or temperature factor) measures the magnitude of vibrational motion of single atoms in the resolved structure. Concerned about certain crystallographic biases, Putz and Brock decided not to use the B-factors stored in PDB files as classification features [99]. Nevertheless, they proved to significantly improve model accuracy in my case. The general acceptance of B-factors as relevant indicators of local mobility is also reflected in the fact that ENM variants are often judged on their ability to predict them (for example [8, 65, 112]). I chose to use the average side chain, average all-atom and C_α B-factors of residues as novel features.

5.5.3 Results of ablation study

I performed an ablation study of the novel features on the configuration with a dilation sequence of $D = 1, 2, 4$ (ERF 29) and baseline relevance heuristic ($\mathcal{R}_0$). The results are shown in Table 5.7.

Based on the results of the ablation study, only the B-factors seem to provide a clear und unambiguous advantage. Removing DI or dihedrals resulted in jumps in the ENM metrics, as well as AUROC and F_1 in the case of dihedrals. The maximum $\overline{\text{vs}}CO(10)$ value of 0.0359 reached in the model without DI is quite impressive, although it may be caused by lucky outliers.

Still, none of the models with any of the novel features removed were included in the top ranked epochs when looking for a balanced model. While the ablation study would need to be repeated with the final model configuration (ERF 125, relevance heuristic $\mathcal{R}_2$), I assume a situation similar to the investigation of varied ERF in Section 5.3: slightly increased classifier accuracy and maximum ENM metrics, as seen in the ablation study, are not clearly associated with increased model performance over all motion types. This decoupling, while not completely comparable, seems to also be the case with the results of *lmc*ENM$_{\mathrm{SVM}}$, which achieves balanced ENM results with quite low classifier metrics. Ultimately, this too leads back to the central concept of relevance of predicted breaking contacts.

Values	Features	Loss ↓	AUROC ↑	F_1 ↑	$\overline{\Delta}CO(10)$ ↑	$\overline{\mathrm{vs}}CO(10)$ ↑
Avg. best	Baseline	0.0425	0.7779	0.3200	0.0165	0.0133
	w/o DI	0.0427	0.7762	0.3181	**0.0199**	**0.0236**
	w/o Dihedrals	**0.0421**	**0.7823**	**0.3258**	0.0125	0.0205
	w/o B-factors	0.0433	0.7587	0.3009	0.0152	-0.0067
Best	Baseline	0.0423	0.7796	0.3218	0.0189	0.0147
	w/o DI	0.0425	0.7782	0.3200	**0.0206**	**0.0359**
	w/o Dihedrals	**0.0419**	**0.784**	**0.3290**	0.0159	0.0276
	w/o B-factors	0.0429	0.7612	0.3039	0.0169	-0.0019

Table 5.7: Ablation study of novel features DI, dihedral angles, and B-factors. The used model configuration has an ERF of 29 and uses the baseline relevance heuristic. The average best value over three seeds and the best overall is listed. Arrows indicate whether best is defined as min or max.

6 Conclusion

In this thesis, I have built a novel Deep Learning-based classifier to predict breaking contacts during functional protein motions, called *lmc*ENM$_{\text{CNN}}$. It builds on, expands, and can act as a replacement for *lmc*ENM$_{\text{SVM}}$ [99, 100], a recently introduced Elastic Network Model (ENM) variant that promises to allow local functional motions to be successfully captured by ENMs. Protein breaking contacts were introduced along with *lmc*ENM$_{\text{SVM}}$ and represent pairs of residues that are especially flexible during functional transitions, changing their relative distance more than a predefined percentage. The predicted breaking contacts are then removed from a standard Anisotropic Network Model (ANM).

I have shown that *lmc*ENM$_{\text{CNN}}$ reaches performance on a level comparable to *lmc*ENM$_{\text{SVM}}$, especially for the local motion types. The Convolutional Neural Network (CNN) at the core of *lmc*ENM$_{\text{CNN}}$ shows greatly improved AUROC and F_1 scores compared to the SVM used in *lmc*ENM$_{\text{SVM}}$. The end results in evaluating the ENM normal modes reveal a more mixed picture. On the validation set, *lmc*ENM$_{\text{CNN}}$ surpassed the performance of *lmc*ENM$_{\text{SVM}}$, while it didn't do so on the holdout set and especially struggled with nonlocal motion types. I highlighted the difficulty of reaching balanced performance over all motion types and discussed possible contributors to this difficulty. I believe the concept of relevance to be central for the breaking contact prediction (BCP) problem and related back to this multiple times in the thesis. This should also provide guidance in improving *lmc*ENM$_{\text{CNN}}$.

Besides building a novel classifier, I also proposed and tested multiple enhancements of the *lmc*ENM method itself, including the concept of relevance heuristics, the addition of some novel fundamental features, and the inclusion of multichain proteins, achieving promising results that invite further research.

The significant engineering effort that went into building *lmc*ENM$_{\text{CNN}}$ raises the question if it was justified. All the more so as, at the moment, *lmc*ENM$_{\text{CNN}}$ is not an uncomplicated replacement for *lmc*ENM$_{\text{SVM}}$. While it works satisfyingly for the local motion types, *lmc*ENM$_{\text{CNN}}$ is not yet as balanced and reliable in its results as *lmc*ENM$_{\text{SVM}}$.

However, *lmc*ENM$_{\text{CNN}}$ is only the first step in Deep Learning-based approaches to BCP. Its value is further visible in the following achievements: first, opening up the field of ENMs to Deep Learning. Second, providing a well-built and future-proof platform to quickly iterate on and build improved and possibly much more general and integrated ENM/DL models (see Chapter 7). Third, the improvements

to the *lmc*ENM method itself proposed and implemented, including the extension to multichain proteins and the introduction of relevance heuristics (see Section 5.4), which could also be backported to *lmc*ENM$_{\mathrm{SVM}}$. And last, the insights gained from the model itself, regarding usable depth, improved classifier metrics (possibly through better prediction of clustered breaking contacts) and novel features as well as the usefulness of Adaptive Sharpness-Aware Minimization (ASAM) in this case.

The hand-engineered graph features used in *lmc*ENM$_{\mathrm{SVM}}$ have shown great effectiveness and expressiveness, leading to highly relevant predicted breaking contacts even in the context of the low classification performance of the SVM. The implicitly learned features of the CNN also work, but likely emphasize other breaking contact patterns, including clustering. A combination of the strengths of both methods would be desirable. The relevance heuristics discussed in Section 5.4 can already be seen as some domain knowledge-derived "nurture" added to the freely learned CNN features.

With many threads to follow in order to further improve *lmc*ENM$_{\mathrm{CNN}}$, the next iteration of DL-based BCP, likely to be a graph-based model, should be more clearly superior not only in terms of classifier performance, but ENM end results as well. I discuss some of the possible directions in the following chapter.

7 Future Work

7.1 Larger dataset

Deep Learning generally benefits from more data (see Figure 2.4). A larger validation set would further help to stabilize its associated ENM metrics. An even bigger dataset than the one built in this thesis (see Section 4.2) could likely be achieved by filtering the complete PDB or using an existing database like CoDNaS [92], Binding MOAD [51], or MolMovDB [35]. Experimental databases could be built that include conformational change not involving ligands. It is unknown how many more nonredundant apo-holo pairs could currently be extracted from the PDB or other databases.

7.2 Graph-based model

A graph-based deep learning model would allow a more natural, 3D-centered representation of contact neighbourhood and thus a more direct translation of the contact neighbourhood graph into a DL model. This solves a limitation of the contact-map approach used in the 2D convolutional model, which has a primarily sequential view of the protein (see Section 4.1). Like in the ING, nodes would represent residues and edges contacts between them. The graph could either be made sparse, with edges only existing for apo contacts, or a full graph of order L, making even more neighbourhood information available. Both node attributes (residue-level features) and edge attributes (pairwise features) would need to be taken into account for classification. The task is therefore one of **edge classification** [137]. In the world of graph neural networks (GNNs), this seems to be a rather niche application, with most existing models and graph convolution operators focusing on node classification [137]. Edge attributes are most of the time not being sufficiently taken into account, even for node classification [39]. Therefore, significant customization of existing graph neural network models might be necessary for BCP using GNNs. Some promising resources were found on the topic of edge classification [24, 39, 56, 63], including an article on edge classification with neighbourhood sampling using the DGL deep graph learning library.[1] The line graph of the protein could also be used as input to the GNN, resulting again in a node classification task. One issue to keep in mind is

[1] See `https://docs.dgl.ai/en/latest/guide/minibatch-edge.html`

that most GNNs tend to become unstable with more than a few layers. This could nonetheless be completely sufficient in a spatially-centered representation.

7.3 Attention-based model

Deep Learning architectures involving attention have become very popular recently, starting with the introduction of the transformer [124]. Attention mechanisms would be interesting for BCP for two reasons: they could allow for better performance than a CNN and be used in either a sequential protein view (transfomer) or in the context of a graph-based model (graph attention). A full transformer would probably even substitute for a graph model, as it can effectively attend over the complete contact map. On the other hand, transformers are very data-hungry, and a larger dataset (an order of magnitude at least) would realistically be needed to successfully apply a transformer architecture. Using an attention-based model would also allow the direct instead of only indirect investigation of the parts of the protein most relevant to the prediction of breaking contacts.

Transformers are very expensive for vision tasks, with the cost of self-attention growing quadratically in the number of pixels. Sub-quadratic transformer variants like the Vision Transformer (ViT) [26] may provide an avenue for further research.

7.4 End-to-end learning of protein motion

Using the original *lmc*ENM$_{\mathrm{SVM}}$ approach as well as the model presented in this thesis, the desired end result of improved capturing of localized functional transitions by ENMs, or more generally protein motion in ENMs, is reflected only indirectly in the learning process. This can be seen, especially in the case of a neural network, in two facts: (a) validation loss does not correlate strongly with ENM performance improvement, and (b) not even validation classifier performance directly corresponds to ENM performance. I observed epochs with an F_1 more than 0.1 smaller than another one, but with much better ENM metrics. There were also epochs with the same F_1 but vastly different $\overline{\Delta}CO(10)$. This is because the notion of *relevant* breaking contacts is missing from the learning goal. For the network, each contact is equally important, but for the ENM it is not, as shown by Putz and Brock [99]. First attempts to remediate this were made in this thesis using relevance heuristics (see Section 5.4). Besides that, it has to be manually tested which features and hyperparameters exactly bias the learning process towards breaking contacts that are especially beneficial to the modified topology.

All of this is likely the result of the fundamental gap between both parts of the method, the neural network *predicting* the topology and the ANM *using* this topology. Independent of the architecture used in the neural network model, a more elegant solution would be to learn protein motion *end-to-end*. This could be done with a wholly different protein motion formalism, for example considering recent breakthrough progress in protein structure prediction [57], an extended model becomes conceivable. This model would predict not a single conformation, but a distribution of conformations likely to be seen in the cellular milieu, with (transition) energies attached to them. Analogous to the popular distograms (discrete PDFs of a pairwise distance in a single conformation), this predicted distribution could be called a *dynagram*.

Second, I propose an alternative approach, called **differentiable ENMs**. This approach would retain the idea of representing protein motion by a small number of highly relevant normal modes, but modify ENMs in a way as to allow them to be integrated into an autograd framework. The desired end result could be described, for example, by a $CO(k)$-based loss, reflecting the goal of describing as much of the deformation space as possible with as few modes as possible. This circumvents the problem of needing examples of optimal topologies: the $CO(k)$ is bound between 0 and 1 and has the goal "built in", without needing to know what the goal looks like. If the problem is too hard, follow the gradient.

First and foremost, a working gradient-based minimization of the $CO(k)$-loss, which could for example be defined as $\mathcal{L}_{CO(k)} = -\ln CO(k)$, would enable the study of ENM topologies that optimally cover a deformation space with a desired number of normal modes. This would also answer the equivalent question of how many normal modes are needed to cover the deformation space up to a certain degree using simple ANMs, something which is principally unknown [99].

In a next step, a neural network model could be built to predict the optimal ENM stiffness matrix K of an apo conformation directly, instead of the proxy of breaking contacts. This is a much more general formulation of the *lmc*ENM insight that different motion types require different topologies to be captured well, and is also connected to the common practice of adjusting the ANM distance cutoff on a per-protein basis, or adjusting spring stiffness based on various sources of knowledge.

Two fundamental problems arise: first, binary topology as used in *lmc*ENM will hardly be possible, as the Heaviside step function $\Theta(x-1/2)$ representing the argmax over the classes that would be used to form a binary stiffness matrix is not differentiable. Either a differentiable approximation has to be used, or the idea of a binary

topology given up altogether.[2] From a learnability standpoint, the latter is likely to be more successful, due to gradient stability. A nonbinary topology also has other benefits and is the most general formulation possible. Either way, the BCP classification problem is transformed into a regression problem.

And second, normal mode analysis involves solving an eigenvalue problem. Backpropagation through an eigendecomposition is a bleeding-edge field [126], becoming numerically unstable for singular matrices like the Hessian of protein NMA. Likely, a specialized approach to work around this fact is needed here.

[2] Of course, one could train a continuous stiffness matrix K, then round the entries to 0 or 1 and use that while still calculating the loss on the continuous matrix. I don't feel confident making a prediction on the usefulness of that, however.

Bibliography

[1] Badri Adhikari. 2020. DEEPCON: protein contact prediction using dilated convolutional neural networks with dropout. *Bioinformatics* 36, 2 (January 2020), 470–477. https://doi.org/10.1093/bioinformatics/btz593

[2] Aqeel Ahmed, Saskia Villinger, and Holger Gohlke. 2010. Large-scale comparison of protein essential dynamics from molecular dynamics simulations and coarse-grained normal mode analyses. *Proteins: Structure, Function, and Bioinformatics* 78, 16 (2010), 3341–3352. https://doi.org/10.1002/prot.22841

[3] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. 2002. *Molecular Biology of the Cell* (4th ed.). Garland Science, New York, NY, United States.

[4] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. 2020. Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers. (June 2020). arXiv:1811.04918 http://arxiv.org/abs/1811.04918

[5] Takayuki Amemiya, Ryotaro Koike, Akinori Kidera, and Motonori Ota. 2012. PSCDB: a database for protein structural change upon ligand binding. *Nucleic Acids Research* 40, Database issue (January 2012), D554–D558. https://doi.org/10.1093/nar/gkr966

[6] Christian B. Anfinsen. 1973. Principles that Govern the Folding of Protein Chains. *Science* 181, 4096 (July 1973), 223–230. https://doi.org/10.1126/science.181.4096.223

[7] Ali R. Atilgan, S. R. Durell, Robert L. Jernigan, Melik C. Demirel, Ozlem Keskin, and Ivet Bahar. 2001. Anisotropy of fluctuation dynamics of proteins with an elastic network model. *Biophysical Journal* 80, 1 (January 2001), 505–515. https://doi.org/10.1016/S0006-3495(01)76033-X

[8] Ivet Bahar, Ali R. Atilgan, and Burak Erman. 1997. Direct evaluation of thermal fluctuations in proteins using a single-parameter harmonic potential. *Folding & Design* 2, 3 (1997), 173–181. https://doi.org/10.1016/S1359-0278(97)00024-2

[9] Ivet Bahar, Timothy R. Lezon, Ahmet Bakan, and Indira H. Shrivastava. 2010. Normal Mode Analysis of Biomolecular Structures: Functional Mechanisms of Membrane Proteins. *Chemical Reviews* 110, 3 (March 2010), 1463–1497. https://doi.org/10.1021/cr900095e

[10] Ivet Bahar, Timothy R. Lezon, Lee-Wei Yang, and Eran Eyal. 2010. Global dynamics of proteins: bridging between structure and function. *Annual Review of Biophysics* 39 (2010), 23–42. https://doi.org/10.1146/annurev.biophys.093008.131258

[11] Ahmet Bakan, Anindita Dutta, Wenzhi Mao, Ying Liu, Chakra Chennubhotla, Timothy R. Lezon, and Ivet Bahar. 2014. Evol and ProDy for bridging protein sequence evolution and structural dynamics. *Bioinformatics* 30, 18 (September 2014), 2681–2683. https://doi.org/10.1093/bioinformatics/btu336

[12] Ahmet Bakan, Lidio M. Meireles, and Ivet Bahar. 2011. ProDy: Protein Dynamics Inferred from Theory and Experiments. *Bioinformatics* 27, 11 (June 2011), 1575–1577. https://doi.org/10.1093/bioinformatics/btr168

[13] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. ACM Press, 144–152.

[14] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. (July 2020). arXiv:2005.14165 http://arxiv.org/abs/2005.14165

[15] N. J. Carter and Rob A. Cross. 2005. Mechanics of the kinesin step. *Nature* 435, 7040 (May 2005), 308–312. https://doi.org/10.1038/nature03528

[16] Claudio N. Cavasotto, Julio A. Kovacs, and Ruben A. Abagyan. 2005. Representing receptor flexibility in ligand docking through relevant normal modes. *Journal of the American Chemical Society* 127, 26 (July 2005), 9632–9640. https://doi.org/10.1021/ja042260c

[17] Sidhartha Chaudhury, Sergey Lyskov, and Jeffrey J. Gray. 2010. PyRosetta: a script-based interface for implementing molecular modeling algorithms using Rosetta. *Bioinformatics* 26, 5 (March 2010), 689–691. https://doi.org/10.1093/bioinformatics/btq007

[18] Stefan Chmiela, Huziel E. Sauceda, Klaus-Robert Müller, and Alexandre Tkatchenko. 2018. Towards exact molecular dynamics simulations with machine-learned force fields. *Nature Communications* 9, 1 (September 2018), 3887. https://doi.org/10.1038/s41467-018-06169-2

[19] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). (February 2016). arXiv:1511.07289 http://arxiv.org/abs/1511.07289

[20] Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski,

and Michiel J. L. de Hoon. 2009. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* 25, 11 (June 2009), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163

[21] Qiang Cui and Ivet Bahar. 2005. *Normal Mode Analysis: Theory and Applications to Biological and Chemical Systems.* CRC Press, Boca Raton, FL, United States.

[22] Ximin Cui, Ke Zheng, Lianru Gao, Dong Yang, and Jinchang Ren. 2019. Multiscale Spatial-Spectral Convolutional Network with Image-Based Framework for Hyperspectral Imagery Classification. *Remote Sensing* 11 (September 2019), 2220. https://doi.org/10.3390/rs11192220

[23] Warren L. DeLano. 2002. PyMOL: An open-source molecular graphics tool. *CCP4 Newsletter on Protein Crystallography* 40, 1 (2002), 82–92.

[24] Gage DeZoort, Savannah Thais, Isobel Ojalvo, Peter Elmer, Vesal Razavimaleki, Javier Duarte, Markus Atkinson, and Mark Neubauer. 2021. Charged particle tracking via edge-classifying interaction networks. (March 2021). arXiv:2103.16701 http://arxiv.org/abs/2103.16701

[25] Nicolas Dony, Jean Marc Crowet, Bernard Joris, Robert Brasseur, and Laurence Lins. 2013. SAHBNET, an Accessible Surface-Based Elastic Network: An Application to Membrane Protein. *International Journal of Molecular Sciences* 14, 6 (June 2013), 11510–11526. https://doi.org/10.3390/ijms140611510

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. (June 2021). arXiv:2010.11929 http://arxiv.org/abs/2010.11929

[27] Ron O. Dror, Robert M. Dirks, J.P. Grossman, Huafeng Xu, and David E. Shaw. 2012. Biomolecular Simulation: A Computational Microscope for Molecular Biology. *Annual Review of Biophysics* 41, 1 (2012), 429–452. https://doi.org/10.1146/annurev-biophys-042910-155245

[28] Stanley D. Dunn, L.M. Wahl, and Greg B. Gloor. 2008. Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. *Bioinformatics* 24, 3 (February 2008), 333–340. https://doi.org/10.1093/bioinformatics/btm604

[29] Eran Eyal, Lee-Wei Yang, and Ivet Bahar. 2006. Anisotropic network model: systematic evaluation and a new web interface. *Bioinformatics* 22, 21 (November 2006), 2619–2627. https://doi.org/10.1093/bioinformatics/btl448

[30] Julia D. Fischer, Carlos E. Mayer, and Johannes Söding. 2008. Prediction of protein functional residues from sequence by probability density estimation. *Bioinformatics* 24, 5 (March 2008), 613–620. https://doi.org/10.1093/bioinformatics/btm626

[31] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-Aware Minimization for Efficiently Improving Generalization. (April 2021). arXiv:2010.01412 http://arxiv.org/abs/2010.01412

[32] Franca Fraternali and Luigi Cavallo. 2002. Parameter optimized surfaces (POPS): analysis of key interactions and conformational changes in the ribosome. *Nucleic Acids Research* 30, 13 (July 2002), 2950–2960. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC117037/

[33] Hans Frauenfelder, Stephen G. Sligar, and Peter G. Wolynes. 1991. The energy landscapes and motions of proteins. *Science* 254, 5038 (December 1991), 1598–1603. https://doi.org/10.1126/science.1749933

[34] Kunihiko Fukushima. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* 36, 4 (April 1980), 193–202. https://doi.org/10.1007/BF00344251

[35] Mark Gerstein and Werner G. Krebs. 1998. A database of macromolecular motions. *Nucleic Acids Research* 26, 18 (September 1998), 4280–4290. https://doi.org/10.1093/nar/26.18.4280

[36] Christoph Globisch, Venkatramanan Krishnamani, Markus Deserno, and Christine Peter. 2013. Optimization of an Elastic Network Augmented Coarse Grained Model to Study CCMV Capsid Deformation. *PLOS ONE* 8, 4 (April 2013), e60582. https://doi.org/10.1371/journal.pone.0060582

[37] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 315–323. http://proceedings.mlr.press/v15/glorot11a.html

[38] Chern-Sing Goh, Duncan Milburn, and Mark Gerstein. 2004. Conformational changes associated with protein-protein interactions. *Current Opinion in Structural Biology* 14, 1 (February 2004), 104–109. https://doi.org/10.1016/j.sbi.2004.01.005

[39] Liyu Gong and Qiang Cheng. 2019. Exploiting Edge Features for Graph Neural Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Long Beach, CA, USA, 9203–9211. https://doi.org/10.1109/CVPR.2019.00943

[40] Weikang Gong, Yang Liu, Yanpeng Zhao, Shihao Wang, Zhongjie Han, and Chunhua Li. 2021. Equally Weighted Multiscale Elastic Network Model and Its Comparison

with Traditional and Parameter-Free Models. *Journal of Chemical Information and Modeling* 61, 2 (February 2021), 921–937. https://doi.org/10.1021/acs.jcim.0c01178

[41] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.

[42] Turkan Haliloglu, Ivet Bahar, and Burak Erman. 1997. Gaussian Dynamics of Folded Proteins. *Physical Review Letters* 79, 16 (October 1997), 3090–3093. https://doi.org/10.1103/PhysRevLett.79.3090

[43] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (September 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. (December 2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. (February 2015). arXiv:1502.01852 http://arxiv.org/abs/1502.01852

[46] Matthias Heinig and Dmitrij Frishman. 2004. STRIDE: a web server for secondary structure assignment from known atomic coordinates of proteins. *Nucleic Acids Research* 32, Web Server issue (July 2004), W500–W502. https://doi.org/10.1093/nar/gkh429

[47] Konrad Hinsen. 1998. Analysis of domain motions by approximate normal mode calculations. *Proteins* 33, 3 (November 1998), 417–429. https://doi.org/10.1002/(sici)1097-0134(19981115)33:3<417::aid-prot10>3.0.co;2-8

[48] Scott A. Hollingsworth and Ron O. Dror. 2018. Molecular Dynamics Simulation for All. *Neuron* 99, 6 (September 2018), 1129–1143. https://doi.org/10.1016/j.neuron.2018.08.011

[49] Viktor Hornak, Asim Okur, Robert C. Rizzo, and Carlos Simmerling. 2006. HIV-1 protease flaps spontaneously open and reclose in molecular dynamics simulations. *Proceedings of the National Academy of Sciences of the United States of America* 103, 4 (January 2006), 915–920. https://doi.org/10.1073/pnas.0508452103

[50] Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4, 2 (January 1991), 251–257. https://doi.org/10.1016/0893-6080(91)90009-T

[51] Liegi Hu, Mark L. Benson, Richard D. Smith, Michael G. Lerner, and Heather A. Carlson. 2005. Binding MOAD (Mother Of All Databases). *Proteins* 60, 3 (August 2005), 333–340. https://doi.org/10.1002/prot.20512

[52] John D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

[53] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. (March 2015). arXiv:1502.03167 http://arxiv.org/abs/1502.03167

[54] Sam M. Ireland and Andrew C. R. Martin. 2020. atomium—a Python structure parser. *Bioinformatics* 36, 9 (May 2020), 2750–2754. https://doi.org/10.1093/bioinformatics/btaa072

[55] Jay I. Jeong, Yunho Jang, and Moon K. Kim. 2006. A connection rule for α-carbon coarse-grained elastic network models using chemical bond information. *Journal of Molecular Graphics and Modelling* 24, 4 (January 2006), 296–306. https://doi.org/10.1016/j.jmgm.2005.09.006

[56] Xiaodong Jiang, Ronghang Zhu, Pengsheng Ji, and Sheng Li. 2020. Co-embedding of Nodes and Edges with Graph Neural Networks. (October 2020). arXiv:2010.13242 http://arxiv.org/abs/2010.13242

[57] 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. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* (July 2021), 1–11. https://doi.org/10.1038/s41586-021-03819-2

[58] Wolfgang Kabsch and Christian Sander. 1983. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22, 12 (December 1983), 2577–2637. https://doi.org/10.1002/bip.360221211

[59] Martin Karplus and John Kuriyan. 2005. Molecular dynamics and protein function. *Proceedings of the National Academy of Sciences* 102, 19 (May 2005), 6679–6685. https://doi.org/10.1073/pnas.0408930102

[60] Martin Karplus and Gregory A. Petsko. 1990. Molecular dynamics simulations in biology. *Nature* 347, 6294 (October 1990), 631–639. https://doi.org/10.1038/347631a0

[61] Burak T. Kaynak, Doga Findik, and Pemra Doruker. 2018. RESPEC Incorporates Residue Specificity and the Ligand Effect into the Elastic Network Model. *The Journal of Physical Chemistry B* 122, 21 (May 2018), 5347–5355. https://doi.org/10.1021/acs.jpcb.7b10325

[62] Henry J. Kelley. 1960. Gradient Theory of Optimal Flight Paths. *ARS Journal* 30, 10 (October 1960), 947–954. https://doi.org/10.2514/8.5282

[63] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. 2019. Edge-Labeling Graph Neural Network for Few-Shot Learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11–20. https://doi.org/10.1109/CVPR.2019.00010

[64] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. (January 2017). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[65] Dmitry A. Kondrashov, Qiang Cui, and George N. Phillips. 2006. Optimization and Evaluation of a Coarse-Grained Model of Protein Motion Using X-Ray Crystal Data. *Biophysical Journal* 91, 8 (October 2006), 2760–2767. https://doi.org/10.1529/biophysj.106.085894

[66] Daniel E. Koshland. 1958. Application of a Theory of Enzyme Specificity to Protein Synthesis. *Proceedings of the National Academy of Sciences of the United States of America* 44, 2 (February 1958), 98–104. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC335371/

[67] Michael Kovermann, Per Rogne, and Magnus Wolf-Watz. 2016. Protein dynamics and function from solution state NMR spectroscopy. *Quarterly Reviews of Biophysics* 49 (2016), e6. https://doi.org/10.1017/S0033583516000019

[68] Werner G. Krebs, Vadim Alexandrov, Cyrus A. Wilson, Nathaniel Echols, Haiyuan Yu, and Mark Gerstein. 2002. Normal mode analysis of macromolecular motions in a database framework: Developing mode concentration as a useful classifying statistic. *Proteins: Structure, Function, and Bioinformatics* 48, 4 (2002), 682–695. https://doi.org/10.1002/prot.10168

[69] Ozge Kurkcuoglu, Osman Teoman Turgut, Sertan Cansu, Robert L. Jernigan, and Pemra Doruker. 2009. Focused Functional Dynamics of Supramolecules by Use of a Mixed-Resolution Elastic Network Model. *Biophysical Journal* 97, 4 (August 2009), 1178–1187. https://doi.org/10.1016/j.bpj.2009.06.009

[70] Zeynep Kurkcuoglu, Ahmet Bakan, Duygu Kocaman, Ivet Bahar, and Pemra Doruker. 2012. Coupling between Catalytic Loop Motions and Enzyme Global Dynamics. *PLOS*

Computational Biology 8, 9 (September 2012), e1002705. https://doi.org/10.1371/journal.pcbi.1002705

[71] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. 2021. ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks. (June 2021). arXiv:2102.11600 http://arxiv.org/abs/2102.11600

[72] Naomi R. Latorraca, A. J. Venkatakrishnan, and Ron O. Dror. 2017. GPCR Dynamics: Structures in Motion. *Chemical Reviews* 117, 1 (January 2017), 139–155. https://doi.org/10.1021/acs.chemrev.6b00177

[73] Vincent Le Guilloux, Peter Schmidtke, and Pierre Tuffery. 2009. Fpocket: An open source platform for ligand pocket detection. *BMC Bioinformatics* 10, 1 (June 2009), 168. https://doi.org/10.1186/1471-2105-10-168

[74] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (May 2015), 436–444. https://doi.org/10.1038/nature14539

[75] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation* 1, 4 (December 1989), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

[76] Nicholas Leioatts, Tod D. Romo, and Alan Grossfield. 2012. Elastic Network Models are Robust to Variations in Formalism. *Journal of Chemical Theory and Computation* 8, 7 (July 2012), 2424–2434. https://doi.org/10.1021/ct3000316

[77] Zhong Li, Yuele Lin, Arne Elofsson, and Yuhua Yao. 2020. Protein Contact Map Prediction Based on ResNet and DenseNet. *BioMed Research International* 2020 (2020), 7584968. https://doi.org/10.1155/2020/7584968

[78] Haoning Lin, Zhenwei Shi, and Zhengxia Zou. 2017. Maritime Semantic Labeling of Optical Remote Sensing Images with Multi-Scale Fully Convolutional Network. *Remote Sensing* 9 (May 2017), 480. https://doi.org/10.3390/rs9050480

[79] Seppo Linnainmaa. 1976. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics* 16, 2 (June 1976), 146–160. https://doi.org/10.1007/BF01931367

[80] Xuyang Liu, Lei Jin, Shenghua Gao, and Suwen Zhao. 2021. Protein contact map prediction using multiple sequence alignment dropout and consistency learning for sequences with less homologs. *bioRxiv* (May 2021), 2021.05.12.443740. https://doi.org/10.1101/2021.05.12.443740

[81] Ying Liu and Ivet Bahar. 2012. Sequence Evolution Correlates with Structural Dynamics. *Molecular Biology and Evolution* 29, 9 (September 2012), 2253–2263. https://doi.org/10.1093/molbev/mss097

[82] Samuele Lo Piano. 2020. Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward. *Humanities and Social Sciences Communications* 7, 1 (June 2020), 1–7. https://doi.org/10.1057/s41599-020-0501-9

[83] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. 2017. The Expressive Power of Neural Networks: A View from the Width. *Advances in Neural Information Processing Systems* 30 (2017), 6231–6239. https://proceedings.neurips.cc/paper/2017/hash/32cbf687880eb1674a07bf717761dd3a-Abstract.html

[84] José Ramón López-Blanco and Pablo Chacón. 2016. New generation of elastic network models. *Current Opinion in Structural Biology* 37 (April 2016), 46–53. https://doi.org/10.1016/j.sbi.2015.11.013

[85] Jianpeng Ma. 2005. Usefulness and limitations of normal mode analysis in modeling dynamics of biomolecular complexes. *Structure* 13, 3 (March 2005), 373–380. https://doi.org/10.1016/j.str.2005.02.002

[86] Satyabrata Majumder, Dwaipayan Chaudhuri, Joyeeta Datta, and Kalyan Giri. 2021. Exploring the intrinsic dynamics of SARS-CoV-2, SARS-CoV and MERS-CoV spike glycoprotein through normal mode analysis using anisotropic network model. *Journal of Molecular Graphics and Modelling* 102 (January 2021), 107778. https://doi.org/10.1016/j.jmgm.2020.107778

[87] Debora S. Marks, Lucy J. Colwell, Robert Sheridan, Thomas A. Hopf, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. 2011. Protein 3D Structure Computed from Evolutionary Sequence Variation. *PLoS ONE* 6, 12 (December 2011), e28766. https://doi.org/10.1371/journal.pone.0028766

[88] Lucie C. Martin, Gregory B. Gloor, Stanley D. Dunn, and Lindi M. Wahl. 2005. Using information theory to search for co-evolving residues in proteins. *Bioinformatics* 21, 22 (November 2005), 4116–4124. https://doi.org/10.1093/bioinformatics/bti671

[89] Tatiana Maximova, Ryan Moffatt, Buyong Ma, Ruth Nussinov, and Amarda Shehu. 2016. Principles and Overview of Sampling Methods for Modeling Macromolecular Structure and Dynamics. *PLOS Computational Biology* 12, 4 (April 2016), e1004619. https://doi.org/10.1371/journal.pcbi.1004619

[90] Andrew McCammon, Bruce R. Gelin, and Martin Karplus. 1977. Dynamics of folded proteins. *Nature* 267, 5612 (June 1977), 585–590. https://doi.org/10.1038/267585a0

[91] R. J. Dwayne Miller. 2014. Femtosecond crystallography with ultrabright electrons and x-rays: capturing chemistry in action. *Science* 343, 6175 (March 2014), 1108–1116. https://doi.org/10.1126/science.1248488

[92] Alexander Miguel Monzon, Cristian Oscar Rohr, María Silvina Fornasari, and Gustavo Parisi. 2016. CoDNaS 2.0: a comprehensive database of protein conformational diversity in the native state. *Database: The Journal of Biological Databases and Curation* 2016 (2016), baw038. https://doi.org/10.1093/database/baw038

[93] Vincenzo Moscato, Antonio Picariello, and Giancarlo Sperli. 2020. An emotional recommender system for music. *IEEE Intelligent Systems* (2020), 1. https://doi.org/10.1109/MIS.2020.3026000

[94] Andrew Ng. 2018. *Machine Learning Yearning.* GitHub, eBook. https://d2wvfoqc9gyqzf.cloudfront.net/content/uploads/2018/09/Ng-MLY01-13.pdf

[95] Thai-Son Nguyen, Sebastian Stueker, and Alex Waibel. 2021. Super-Human Performance in Online Low-latency Recognition of Conversational Speech. (July 2021). arXiv:2010.03449 http://arxiv.org/abs/2010.03449

[96] Mikel Olazaran. 1996. A Sociological Study of the Official History of the Perceptrons Controversy. *Social Studies of Science* 26, 3 (1996), 611–659. https://www.jstor.org/stable/285702

[97] Laura Orellana, Manuel Rueda, Carles Ferrer-Costa, José Ramón Lopez-Blanco, Pablo Chacón, and Modesto Orozco. 2010. Approaching Elastic Network Models to Molecular Dynamics Flexibility. *Journal of Chemical Theory and Computation* 6, 9 (September 2010), 2910–2923. https://doi.org/10.1021/ct100208e

[98] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035.

[99] Ines Putz. 2018. *Leveraging Novel Information for Coarse-Grained Prediction of Protein Motion.* Ph. D. Dissertation. Berlin, Germany. Advisor(s) Brock, Oliver.

[100] Ines Putz and Oliver Brock. 2017. Elastic network model of learned maintained contacts to predict protein motion. *PLOS ONE* 12, 8 (August 2017), e0183889. https://doi.org/10.1371/journal.pone.0183889

[101] Dennis C. Rapaport. 2004. *The Art of Molecular Dynamics Simulation.* Cambridge University Press, Cambridge, UK.

[102] Martin G. Reese, Ole Lund, Jakob Bohr, Henrik Bohr, Jan E. Hansen, and Søren Brunak. 1996. Distance distributions in proteins: a six-parameter representation. *Protein Engineering, Design and Selection* 9, 9 (September 1996), 733–740. https://doi.org/10.1093/protein/9.9.733

[103] Frank Rosenblatt. 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review* 65, 6 (1958), 386–408. https://doi.org/10.1037/h0042519

[104] Manuel Rueda, Pablo Chacón, and Modesto Orozco. 2007. Thorough validation of protein normal mode analysis: a comparative study with essential dynamics. *Structure* 15, 5 (May 2007), 565–575. https://doi.org/10.1016/j.str.2007.03.013

[105] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (October 1986), 533–536. https://doi.org/10.1038/323533a0

[106] Kannan Sankar, Sambit K. Mishra, and Robert L. Jernigan. 2018. Comparisons of Protein Dynamics from Experimental Structure Ensembles, Molecular Dynamics Ensembles, and Coarse-Grained Elastic Network Models. *The Journal of Physical Chemistry B* 122, 21 (May 2018), 5409–5417. https://doi.org/10.1021/acs.jpcb.7b11668

[107] Michael F. Sanner, Arthur J. Olson, and Jean-Claude Spehner. 1996. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers* 38, 3 (March 1996), 305–320. https://doi.org/10.1002/(SICI)1097-0282(199603)38:3<305::AID-BIP4>3.0.CO;2-Y

[108] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

[109] Michael Schneider and Oliver Brock. 2014. Combining Physicochemical and Evolutionary Information for Protein Contact Prediction. *PLOS ONE* 9, 10 (October 2014), e108438. https://doi.org/10.1371/journal.pone.0108438

[110] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. (December 2017). arXiv:1712.01815 http://arxiv.org/abs/1712.01815

[111] Jens C. Skou. 1957. The influence of some cations on an adenosine triphosphatase from peripheral nerves. *Biochimica et Biophysica Acta* 23 (January 1957), 394–401. https://doi.org/10.1016/0006-3002(57)90343-8

[112] Amit Srivastava, Roee Ben Halevi, Alexander Veksler, and Rony Granek. 2012. Tensorial elastic network model for protein dynamics: Integration of the anisotropic network model with bond-bending and twist elasticities. *Proteins: Structure, Function, and Bioinformatics* 80, 12 (2012), 2692–2700. https://doi.org/10.1002/prot.24153

[113] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from

Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[114] Amelie Stein, Manuel Rueda, Alejandro Panjkovich, Modesto Orozco, and Patrick Aloy. 2011. A systematic study of the energetics involved in structural changes upon association and connectivity in protein interaction networks. *Structure* 19, 6 (June 2011), 881–889. https://doi.org/10.1016/j.str.2011.03.009

[115] Joseph N. Stember and Willy Wriggers. 2009. Bend-twist-stretch model for coarse elastic network simulation of biomolecular motion. *The Journal of Chemical Physics* 131, 7 (August 2009), 074112. https://doi.org/10.1063/1.3167410

[116] Johannes Söding. 2005. Protein homology detection by HMM–HMM comparison. *Bioinformatics* 21, 7 (April 2005), 951–960. https://doi.org/10.1093/bioinformatics/bti125

[117] Chin-Hsien Tai, Rohit Paul, K. C. Dukka, Jeffery D. Shilling, and Byungkook Lee. 2014. SymD webserver: a platform for detecting internally symmetric protein structures. *Nucleic Acids Research* 42, Web Server issue (July 2014), W296–300. https://doi.org/10.1093/nar/gku364

[118] Florence Tama and Yves-Henry Sanejouand. 2001. Conformational change of proteins arising from normal mode calculations. *Protein Engineering, Design and Selection* 14, 1 (January 2001), 1–6. https://doi.org/10.1093/protein/14.1.1

[119] Monique M. Tirion. 1996. Large Amplitude Elastic Motions in Proteins from a Single-Parameter, Atomic Analysis. *Phys. Rev. Lett.* 77, 9 (August 1996), 1905–1908. https://doi.org/10.1103/PhysRevLett.77.1905

[120] Chung-Jung Tsai, Sandeep Kumar, Buyong Ma, and Ruth Nussinov. 1999. Folding funnels, binding funnels, and protein function. *Protein Science* 8, 6 (June 1999), 1181–1190. https://doi.org/10.1110/ps.8.6.1181

[121] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2017. Instance Normalization: The Missing Ingredient for Fast Stylization. (November 2017). arXiv:1607.08022 http://arxiv.org/abs/1607.08022

[122] Oliver T. Unke, Mihail Bogojeski, Michael Gastegger, Mario Geiger, Tess Smidt, and Klaus-Robert Müller. 2021. SE(3)-equivariant prediction of molecular wavefunctions and electronic densities. (June 2021). arXiv:2106.02347 http://arxiv.org/abs/2106.02347

[123] Vladimir Vapnik. 2013. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, Luxemburg, Luxemburg.

[124] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. (December 2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[125] Sheng Wang, Siqi Sun, Zhen Li, Renyu Zhang, and Jinbo Xu. 2017. Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model. *PLOS Computational Biology* 13, 1 (January 2017), e1005324. https://doi.org/10.1371/journal.pcbi.1005324

[126] Wei Wang, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann. 2019. Backpropagation-Friendly Eigendecomposition. (June 2019). arXiv:1906.09023 http://arxiv.org/abs/1906.09023

[127] Yongmei Wang, Andrew J. Rader, Ivet Bahar, and Robert L. Jernigan. 2004. Global ribosome motions revealed with elastic network model. *Journal of Structural Biology* 147, 3 (September 2004), 302–314. https://doi.org/10.1016/j.jsb.2004.01.005

[128] Martin Weigt, Robert A. White, Hendrik Szurmant, James A. Hoch, and Terence Hwa. 2009. Identification of direct residue contacts in protein–protein interaction by message passing. *Proceedings of the National Academy of Sciences* 106, 1 (January 2009), 67–72. https://doi.org/10.1073/pnas.0805923106

[129] Ting-fan Wu, Chih-jen Lin, and Ruby Weng. 2004. Probability Estimates for Multi-Class Classification by Pairwise Coupling. In *Advances in Neural Information Processing Systems*, Vol. 16. MIT Press.

[130] Kelin Xia. 2018. Multiscale virtual particle based elastic network model (MVP-ENM) for normal mode analysis of large-sized biomolecules. *Physical Chemistry Chemical Physics* 20, 1 (2018), 658–669. https://doi.org/10.1039/C7CP07177A

[131] Jinbo Xu, Matthew McPartlon, and Jin Li. 2021. Improved protein structure prediction by deep learning irrespective of co-evolution information. *Nature Machine Intelligence* (May 2021), 1–9. https://doi.org/10.1038/s42256-021-00348-5

[132] Jianyi Yang, Ivan Anishchenko, Hahnbeom Park, Zhenling Peng, Sergey Ovchinnikov, and David Baker. 2020. Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences* 117, 3 (January 2020), 1496–1503. https://doi.org/10.1073/pnas.1914677117

[133] Lei Yang, Guang Song, and Robert L. Jernigan. 2007. How well can we understand large-scale protein motions using normal modes of elastic network models? *Biophysical Journal* 93, 3 (August 2007), 920–929. https://doi.org/10.1529/biophysj.106.095927

[134] Lei Yang, Guang Song, and Robert L. Jernigan. 2009. Protein elastic network models and the ranges of cooperativity. *Proceedings of the National Academy of Sciences* 106, 30 (July 2009), 12347–12352. https://doi.org/10.1073/pnas.0902159106

[135] Fisher Yu and Vladlen Koltun. 2016. Multi-Scale Context Aggregation by Dilated Convolutions. (April 2016). arXiv:1511.07122 http://arxiv.org/abs/1511.07122

[136] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. (February 2017). arXiv:1611.03530 http://arxiv.org/abs/1611.03530

[137] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001

[138] Jianwei Zhu, Sheng Wang, Dongbo Bu, and Jinbo Xu. 2018. Protein threading using residue co-variation and deep learning. *Bioinformatics* 34, 13 (July 2018), i263–i273. https://doi.org/10.1093/bioinformatics/bty278