Volume segmentation of protein electron density maps with 3D convolutional neural networks

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2023

Ákos GODÓ

List of Publications

Residue Assignment in Crystallographic Protein Electron Density Maps With 3D Convolutional Networks

Ákos Godó, Kota Aoki, Atsushi Nakagawa and Yasushi Yagi,

2022, IEEE Access vol. 10. Pp. 28760-28772 DOI: 10.1109/ACCESS.2022.3156108

Single Shot Residue Localization and Classification in Crystallographic Electron Density Maps

Ákos Godó, Kota Aoki, Atsushi Nakagawa and Yasushi Yagi,

2022, IEEE Access vol. 10. Pp. 108354-108365 DOI: 10.1109/ACCESS.2022.3213670

Abstract

Can state-of-the-art computer vision methods be applied for protein structure determination tasks? This thesis answers the above question by defining the problem of locating and classifying amino acid (AA) residues in protein electron density (ED) maps as an image segmentation problem.

Proteins are organic macro-molecules whose function depends on their 3D structure. By determining their structure, the information may be used in computational simulations for drug discovery, bioinformatics and computational biology.

ED maps are three-dimensional, volumetric data acquired via an imaging method called X-ray Diffraction (XRD). ED maps can be considered a 3D image of the protein molecule's shape.

Two limitations of protein structure determination are the dependency on sample resolution and processing time. Automatic model building toolkits have difficulties below 3 Å resolution which necessitates the use of high-resolution ED maps to obtain reliable results. Unfortunately, obtaining high-resolution ED maps is difficult, contributing to a low, approx. 5% success rate of model building efforts.

Model building toolkits create structural models by mapping the AA sequence into the ED map by matching and merging chain fragments. This is computationally intensive: the process may take hours to days. At lower resolutions AAs are less discernible, leading to incomplete models or the outright inability to handle low resolution ED maps.

The thesis proposes two 3D convolutional neural network (CNN) architectures to overcome the above limitations. CNNs excel at parallel computations and effectively utilize the parallel computing capabilities of modern GPUs, which by itself leads to improved computational times. The proposed methods assign AA labels directly to ED maps by crop-stitching fixed size windows, only observing the densities without relying on the AA sequence. Thus, the computational time scales linearly with the volume of the ED map instead of the number of residues in the AA chain, resulting in further reduced computational time.

A custom data set with annotations for supervised semantic and instance segmentation is created to train and evaluate network performance. It contains high-, mid- and low-resolution ED maps generated from the same approximately 7000 proteins, allowing the CNNs to specifically target each resolution. With low-resolution data in the challenging sub-3 Å range, the CNNs can be trained to be functional even where state-of-the-art toolkits fail.

A real-life data set from experimental XRD observations of over 500 protein samples is also established. These samples are used to fine-tune and measure the practical, real-life performance of the proposed CNNs.

The first proposed CNN architecture, 3D FC-DenseNet matches the performance of state-of-the-art toolkits at high resolutions, and beats them at medium and especially at low resolutions, which are a failure case for toolkits. Its upgrade, MT-StackNet is capable of simultaneous semantic and instance segmentation of ED maps and, due its improved performance, it outperforms model building toolkits even at high resolutions. It can locate and classify all AA instances with a single forward pass using a novel offset vector regression technique.

Both methods process ED maps with better performance and much faster than model building toolkits, showing that computer vision methods and CNNs are applicable for protein structure determination tasks.

Contents

Abstract **v**

Acknowledgements **xiii**

List of Figures **xv**

List of Tables **xix**

List of Abbreviations & Acronyms **xxi**

1 Introduction **1**

1.1 Background & Motivation 1

1.2 Structure Determination Difficulties 9

2 Related Work **17**

2.1 Structure Determination & Automatic Model Building Toolkits 17

2.2 Neural Networks for Volumetric Data Segmentation 18

2.3 Neural Networks in Structure Determination 19

2.3.1 Ab Initio Structure Prediction 19

2.3.2 Semantic Segmentation of ED Maps 20

2.3.3 Instance Segmentation of ED Maps 21

3 Problem Setting **23**

3.1 Semantic Segmentation 24

3.2 Instance Segmentation 24

3.3 Residue Classification 26

4 Data Set **27**

4.1 Data Acquisition 27

4.2 Data Preprocessing 31

4.3 Data Augmentation 34

4.4 Ground Truth Annotation 37

5 Methods – Semantic Segmentation **41**

5.1 3D FC-DenseNet Network Architecture 41

5.2 Objective Functions 45

5.2.1 Semantic Segmentation - IoU Loss 45

5.2.2 Fragmented Residues & Positional Weighting 46

5.3 Experiments 47

5.3.1 Ablation Tests 47

Concatenations & Gradient Flow 47

Adaptive Zero Padding & Size Mismatches 48

5.3.2 Inference & Performance Metrics 48

5.3.3 Training Configuration & Objective 52

5.3.4 Finetuning & Experimental Data 52

5.3.5 Comparison to Seqqy & ARP/wARP 53

5.4 Results & Discussion 54

5.4.1 Ablation Test Results 54

Concatenations & Gradient Flow 54

Adaptive Zero Padding & Size Mismatches 54

5.4.2 Training & Validation Results 56

5.4.3 Experimental Samples 60

6 Methods – Simultaneous Semantic & Instance Segmentation **65**

6.1 MT-StackNet Network Architecture 65

6.2 Objective Functions 70

6.2.1 Instance Segmentation – Dice Loss 70

6.2.2 Instance Segmentation – MSE Loss 73
6.3 Combined Loss Function 73
6.4 Experiments 74
6.4.1 Network Configuration 74
6.4.2 Ablation Tests 75
6.4.3 Finetuning & Experimental Samples 75
6.4.4 Inference & Performance Metrics 76
Global Predictions 76
Instance Predictions & Matching 76
Performance Metrics 78
6.5 Results & Discussion 79
6.5.1 Ablation Tests 79
6.5.2 Training & Validation 80
6.5.3 Experimental Results 80

7 Conclusion **91**

Bibliography **97**

Acknowledgements

I would like to thank my supervisors, Professors Yasushi Yagi, Kota Aoki and Atsushi Nakagawa for their guidance, patience and faith in my work. Without their advice and the research environment they helped create this work would not have been possible.

To Mika and Ammar: thank you for always putting up with me and being my rubber duckies by letting me run my ramblings by you from the beginning to the end. To Andrey: although you were my supervisor only in the early stages, your support allowed me to build the strong foundations all these results were built upon.

And last but not least, to any and all readers: thank you for taking the time to read my thesis and I hope you will come to enjoy it as much as I did researching and writing it.

List of Figures

1.1 Ribbon model of a protein 2
1.2 Structure of an amino acid 3
1.3 ED maps of AA residues. 4
1.4 Illustration of protein structure and folding. 5
1.5 Diagram of an X-ray Diffraction experiment. 5
1.6 XRD rotational instrument 6
1.7 Illustration of a diffraction pattern and ED map 7
1.8 EMDB entry resolution in shells per year 8
1.9 Comparison of XRD observations 10
1.10 Protein sequences in the EMBL 11
1.11 Structural models in the PDB 12
1.12 Number of structures in the PDB per resolution 14
1.13 Comparison of ED map resolutions 15

3.1 Example input ED map at medium resolution 23
3.2 Multi-channel semantic output 24
3.3 Instance segmentation offset vector field 25
3.4 Bounding region vs. POR-based instance proposals 26

4.1 Overview of the deposition pipeline 27
4.2 Excerpt from a .pdb structural model file 28
4.3 Synthetic data generation pipeline 30
4.4 Comparison of experimental and generated ED maps 31
4.5 Crystallographic unit cells 31

4.6 Effects of pre-processing on input ED maps 32
4.7 Equal sized ED map window crops 33
4.8 Comparison of AA side chain visibility at high and low resolution 34
4.9 L/D-enantiomers of a chiral AA 35
4.10 Effects of augmentation on AA distributions 36
4.11 Overview of the GT annotation pipeline 38
4.12 Structural model guided annotation 38
4.13 Residue pair classes 39

5.1 Diagram of the 3D-FCDenseNet architecture 42
5.2 Diagram of the 3D FC-DenseNet Down Block 43
5.3 Diagram of the 3D FC-DenseNet Up Block 44
5.4 Residue fragment examples. 46
5.5 1D and 2D examples of the positional weighting scheme . . . 46
5.6 Diagram of the crop-stitching inference process 49
5.7 3D FC-DenseNet ablation test results 55
5.8 3D FC-DenseNet confusion matrices 58
5.9 Global class prediction output of 3D FC-DenseNet 59
5.10 Qualitative results in windows 59
5.11 3D FC-DenseNet experimental sample performance graphs . . 62

6.1 MT-StackNet architecture diagram 66
6.2 3D ResU-Net backbone of MT-StackNet 67
6.3 Residual Block used in MT-StackNet 67
6.4 MT-StackNet pipeline overview 68
6.5 Detailed flowchart of MT-StackNet inference pipeline 77
6.6 MT-StackNet and FC-DenseNet per-residue performance comparison 82
6.7 MT-StackNet loss graphs 87

6.8 MT-StackNet experimental performance graphs 88
6.9 Comparison of MT-StackNet and Seqqy Mean Rank-1 Accuracies and Standard Deviations 89
6.10 Scatter plot of MT-StackNet and Seqqy Rank-1 Accuracies . . 89

7.1 SR pipeline concept overview 93
7.2 A Cryo-EM Coulomb map of a protein. 94

List of Tables

4.1 Pre- and post-augmentation window counts and CSD 37

5.1 3D FC-DenseNet configuration overview 52
5.2 3D FC-DenseNet ablation test results 54
5.3 3D FC-DenseNet validation results 57
5.4 3D FC-DenseNet fine-tuning performance comparison 60
5.5 3D FC-DenseNet performance comparison to Seqqy 60

6.1 MT-StackNet ablation test results 79
6.2 MT-StackNet validation performance comparison 81
6.3 MT-StackNet experimental performance comparison 83

List of Abbreviations & Acronyms

AA	Amino Acid
AI	Artificial Intelligence
Å	Ångstrom ($1\,\text{Å} = 10^{-10}m$)
BG	Background
Cat	Concatenation
CB	Concatenation Block
CCD	Charge Coupled Device
CNN	Convolutional Neural Network
Concat	Concatenation
Conv	Convolution
Cryo-EM	Cryogenic Electron Microscopy
CSD	Chi-Square Distance
DB	Down Block
DL	Dense Layer
ED	Electron Density
EM	Electron Microscopy
EMBL	European Molecular Biology Laboratory
EMDB	Electron Microscopy Data Bank
FC	Fully Convolutional
FCDN	FC-DenseNet (see **FC**)
FN	False Negative
FOV	Field of View

FP	False Positive
FT	Fine-tuning
GPU	Graphics Processing Unit
Grad	Gradient
IoU	Intersection over Union
ML	Machine Learning
MSE	Mean Squared Error
MT	Multi-Task
MTSN	MT-StackNet (see **MT**)
NMR	Nuclear Magnetic Resonance
NN	Neural Network
PDB	Protein Data Bank
POR	Pixel Offset Regression
R1	Rank-1
R3	Rank-3
RoI	Region of Interest
RPN	Region Proposal Network
SB	Skip Block
SBDD	Structure Based Drug Design
SR	Super-Resolution
seq	Sequence
SVM	Support Vector Machine
TD	Transition Down
TN	True Negative
TP	True Positive
Trans	Transpose
Trn	Training
TU	Transition Up

UB Up Block

Val Validation

w/ with

w/o without

XRD X-ray diffraction

AA Residue 3 Letter Codes

ALA Alanine

ARG Arginine

ASN Asparagine

ASP Aspartic Acid

CYS Cysteine

GLN Glutamic Acid

GLU Glutamine

GLY Glycine

HIS Histidine

ILE Isoleucine

LEU Leucine

LYS Lysine

MET Metionine

PHE Phenylalanine

PRO Proline

SER Serine

THR Threonine

TRP Triptophan

TYR Tyrosine

VAL Valine

Chapter 1

Introduction

1.1 Background & Motivation

Machine learning (ML) and artificial intelligence (AI) may well be the greatest contributors to scientific and industrial progress in the early 21st century. Leveraging the immense volume of data accumulated during the information age, ML algorithms excel at identifying patterns and features in sets of data and offer the same or better performance than a human.

Currently, the pinnacle of ML and AI is represented by deep learning and neural networks (NN). NNs learn to perform various tasks by optimizing an objective and identifying the relevant features without human intervention being necessary. Their generalization capability is so wide that one may choose from a number of freely available algorithms [1][2][3] and apply them for natural language processing [4], object detection [5] or image synthesis [6] with good results.

The use of NNs also extends into creating so-called expert systems, computer programs which, with the use of AI, can simulate the knowledge and decisions of experts in particular fields such as healthcare [7] or biology [8]. Applications include early detection of diseases in novel, non-invasive ways [9], [10] and the development of drug compounds in silico [11], [12].

While general architectures may perform well out of the box, NNs designed for specific tasks can outperform them. To do so, one needs to consider the constraints of the task they're being applied to. Designing such expert systems is multidisciplinary by nature, as it requires familiarity with both the field of application to define the problem tractably for ML, and NNs themselves to design an NN architecture capable of leveraging the specifics of the data.

This work deals with applying AI and ML methods to the field of structural biology by attempting to create NN-based methods for determining the structure of proteins based on experimental crystallographic observations.

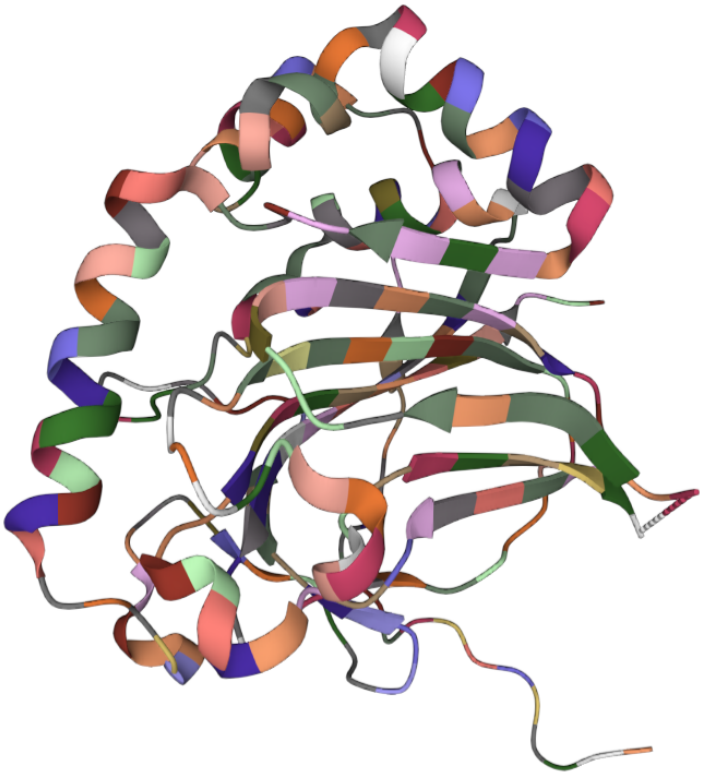


FIGURE 1.1: Ribbon model of a protein (PDB ID 6E0T [13]). Individual colors represent the different AA residues in the chain.

Proteins (Figure 1.1) are ubiquitous organic macromolecules often referred to as the 'building blocks of life' due to their participation in almost all biological processes. Their role and function is defined by their three-dimensional shape. If their structure is determined, the information can be incorporated

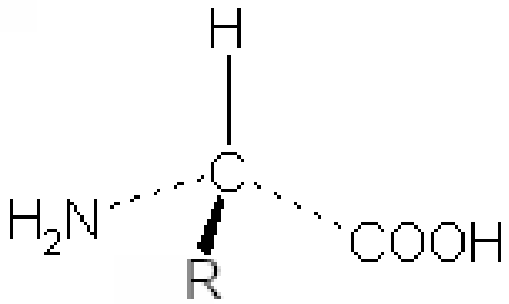


FIGURE 1.2: Structure of an amino acid. The side chain denoted by R between the basic amino (–NH2) and an acidic carboxyl (–COOH) functional groups determines the residue's type and properties.

into computer simulations and further research e.g. in pharmaceuticals, bioinformatics or structural biology. To demonstrate the impact of in silico chemistry, with structure-based drug design (SBDD) it is possible to rapidly trial drug candidate molecules and their compatibility with protein binding sites using computater simulations [14]. Since no actual laboratory experiment is needed to calculate various criteria for the drug, SBDD can cut drug discovery costs significantly.

Proteins are chains of amino acid (AA) molecules whose order ultimately decides a protein's structure. AAs are small, organic molecules (Figure 1.2) containing both a basic amino (–NH2) and an acidic carboxyl functional (–COOH) group [15]. Inbetween these groups, unique side chains can be found, providing the residue with its physical and chemical properties. Through the amino and carboxyl groups the AA residues can form a so-called peptide bond with each other and create a polypeptide chain, a protein. Naturally occurring proteins are chains of the 20 standard AA residues (Figure 1.3).

Protein structure is organized into levels. Primary structure refers to the order of AAs in the constituting linear residue chain. A protein's secondary structure arises when the residue chain assumes energetically optimal formations which can be sheets, helices or loops. Tertiary structure is the organization of secondary structure elements so that they too achieve functionally or

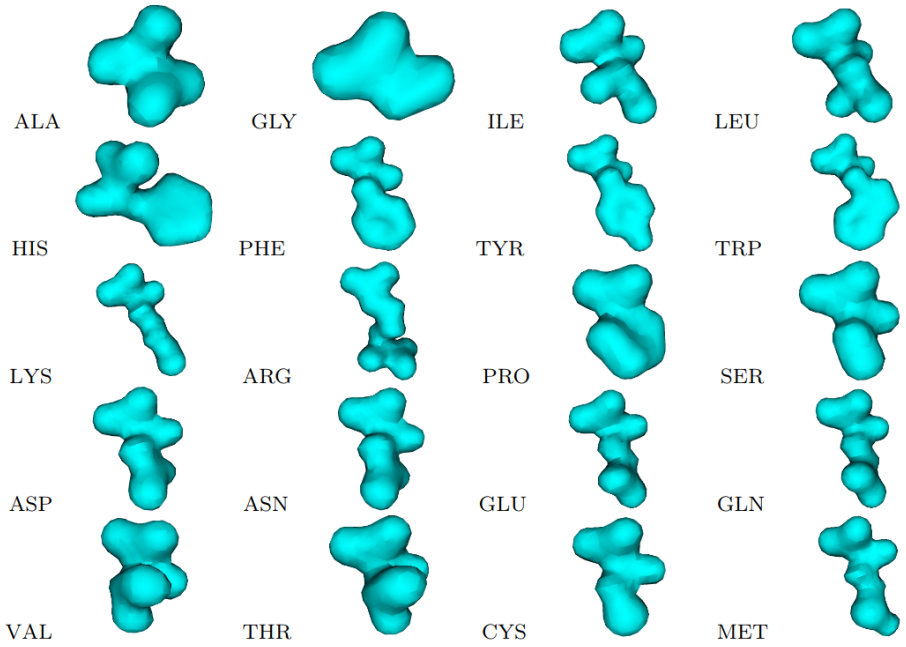


FIGURE 1.3: High-resolution electron density maps of the 20 standard AA residues commonly constituting naturally occurring proteins.

energetically optimal states (Figure 1.4).

The process of a protein assuming its 3D shape is referred to as folding. It is this folded structure that determines what biological processes proteins can participate in and with what ligand molecules they may interact.

The final structure is dependent on the order of residues in the chain. The AA sequence can be obtained rapidly and reliably with modern genome sequencing techniques [17], but the underlying mechanisms of the folding process are not fully understood. Due to this, assigning higher order structures to the residue chain and predicting the shape of the folded protein remains difficult even with the recent NN and AI-based breakthroughs [18][19].

Alternatively, one may choose to observe the naturally folded state of proteins. In this case, structure determination refers to re-assigning the primary structure to the end result of the folding process, inferring the location and conformation of each residue. To facilitate this, various imaging methods exist to observe the 3D shape of proteins.

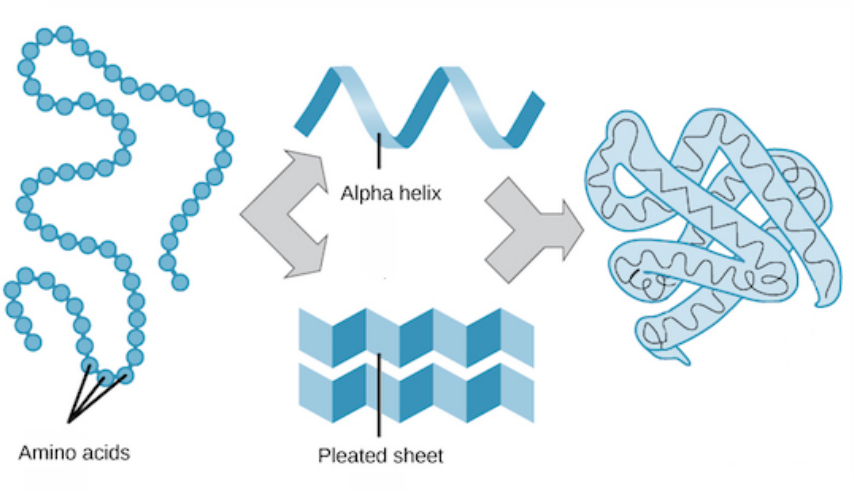


FIGURE 1.4: Illustration of protein structure and folding. The primary structure is a chain of AA residues (left). The chain folds into helices and sheets, forming the secondary structure elements (middle). The tertiary structure (right) arises from further folding of secondary structure elements. Source: [16]

One of the popular imaging methods used for protein structure determination is X-ray Diffraction (XRD). X-rays are high-energy electromagnetic waves with wavelengths $\lambda_{xray} \approx 10^{-8}m$ to $10^{-12}m$. Using particle accelerators or X-ray generators, the X-ray wavelength can be fine-tuned to be as close to a single wavelength as possible, resulting in a so-called monochromatic beam.

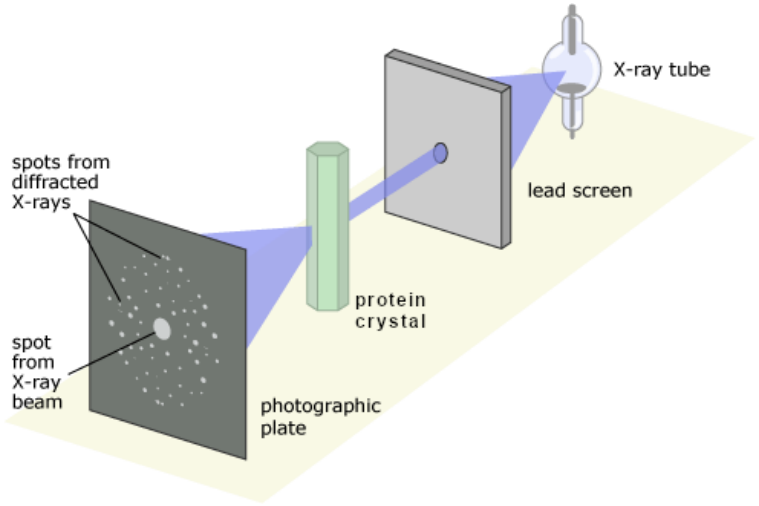


FIGURE 1.5: Diagram of an X-ray Diffraction experiment. The crystal is illuminated by powerful X-ray beams, producing a diffraction pattern on a photographic detector plate. Source: [22]

In an XRD experiment (Figure 1.5), a target protein with a known primary structure is selected, obtained in large quantities and at high concentrations crystallized. The crystals are mounted in a high-precision rotational instrument (Figure 1.6) and illuminated by X-ray beams from multiple angles. Since the wavelength of the electrons λ_{e^-} and λ_{xray} are comparable, the electrons interact with the electromagnetic waves of the X-rays.

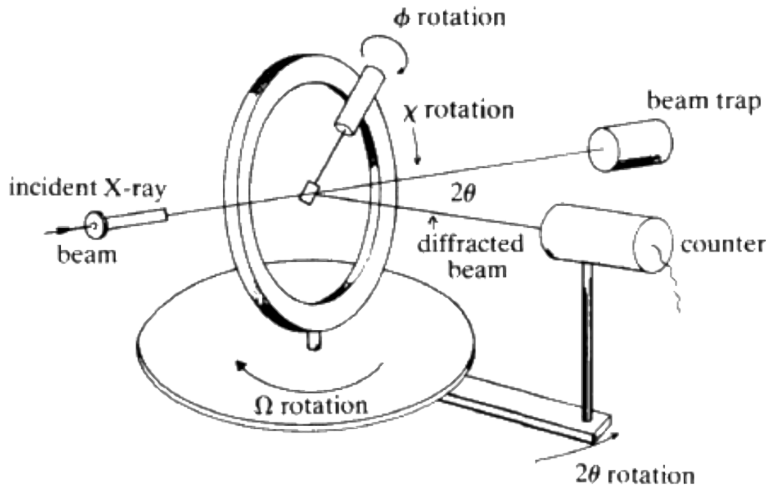


FIGURE 1.6: A high precision, 3-degrees-of-freedom rotational instrument used in XRD experiments to capture diffraction patterns at various angles. Source: [20]

This interaction scatters the X-ray beams, creating diffraction patterns recorded on a detector plate such as a CCD sensor or a pixel array detector (Figure 1.5). By rotating the crystal in various angles, the observed set of diffraction patterns are then reconstructed into three-dimensional data, representing the phase and amplitude of the observations (Figure 1.7).

In a classical camera or microscopy setup where the electromagnetic light wavelength falls into the visible spectrum, optical lenses can be used to focus the scattered beams. For very short wavelengths in the X-ray spectrum, no such lenses exist: they are replaced by Fourier optics. The inverse discrete Fourier-transform of the observed intensities acts a mathematical lens to acquire the electron density (ED) map.

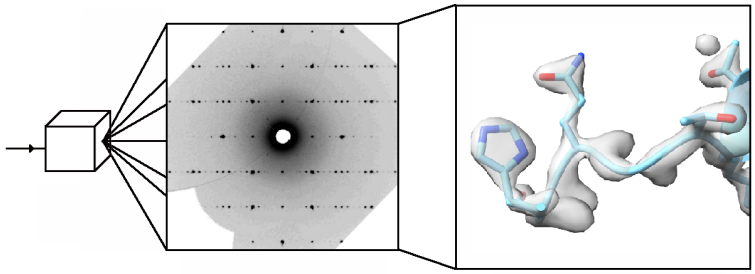


FIGURE 1.7: Illustration of a diffraction pattern and the reconstructed ED map with the molecular structure inside. Diffraction pattern source: [21]

Another imaging method rising in popularity is cryogenic electron microscopy (Cryo-EM). One of the main advantages of Cryo-EM is that it captures data directly in real space instead of having to process the diffraction patterns and intensities of XRD from reciprocal space. Furthermore, it has recently experienced what is referred to as a 'resolution revolution' [23]. Before, the method had difficulties producing high-resolution samples but nowadays this barrier seems broken with more and more high resolution models being deposited (Figure 1.8) to the EM Data Bank (EMDB) [24]. Unfortunately, due to the recency of the 'resolution revolution', the amount of high resolution Coulomb maps is not enough to reliably train a NN model, so this work limits its scope to XRD ED maps.

Both the ED maps from XRD experiments and Coulomb maps from Cryo-EM can be considered a volumetric, three-dimensional snapshot of a molecule's shape. This poses the question: can state-of-the-art computer vision methods be applied or repurposed to process ED maps? This work proposes to answer the question by

(1) defining locating and classifying AA residues in ED maps as an image segmentation problem,

(2) establishing the to-date largest data set built for ML tasks with XRD ED maps,

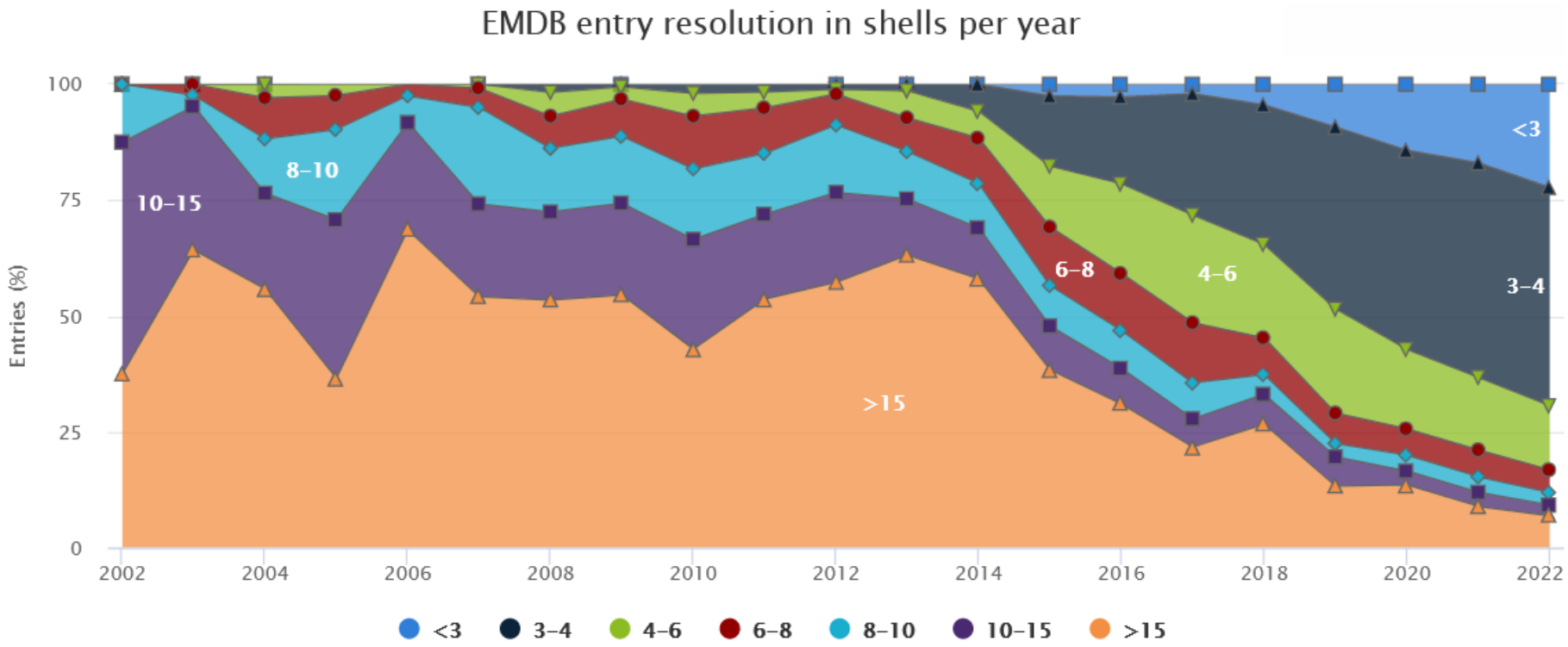


FIGURE 1.8: EM Data Bank (EMDB) entry resolution in shells per year. Due to the recent 'resolution revolution', the resolution of Cryo-EM Coulomb maps has increased drastically. Source: [25]

(3) designing novel 3D convolutional neural network (CNN) architectures designed for semantic and instance segmentation of XRD ED maps,

(4) demonstrating the capabilities, performance and evolution of said methods, showing that it is possible to transfer image segmentation knowledge from 2D images to protein structure determination tasks,

(5) showing that the methods have practical use as they can outperform current toolkits used by experts.

1.2 Structure Determination Difficulties

Broadly speaking, protein structure determination can be split into two categories:

(1) model building into experimental observations where the goal is to find and label each AA in maps acquired either via XRD or Cryo-EM; and

(2) ab initio structure prediction, where the goal is to predict the folded protein structure from the AA sequence.

This work concerns itself with the former category. Although automated toolkits for model building into experimental observations do exist, the process remains difficult and time consuming.

The difficulties begin at sample preparation: crystallizing proteins is unfortunately a trial-and-error process. Even if a large quantity of the target protein can be obtained, there is no guarantee that it will crystallize properly (or at all) for it to be usable in an XRD experiment. Successful crystallization depends on a myriad of precise environmental factors (such as temperature, concentration, pH, etc.) and the resulting protein crystal's quality can only be determined after the fact.

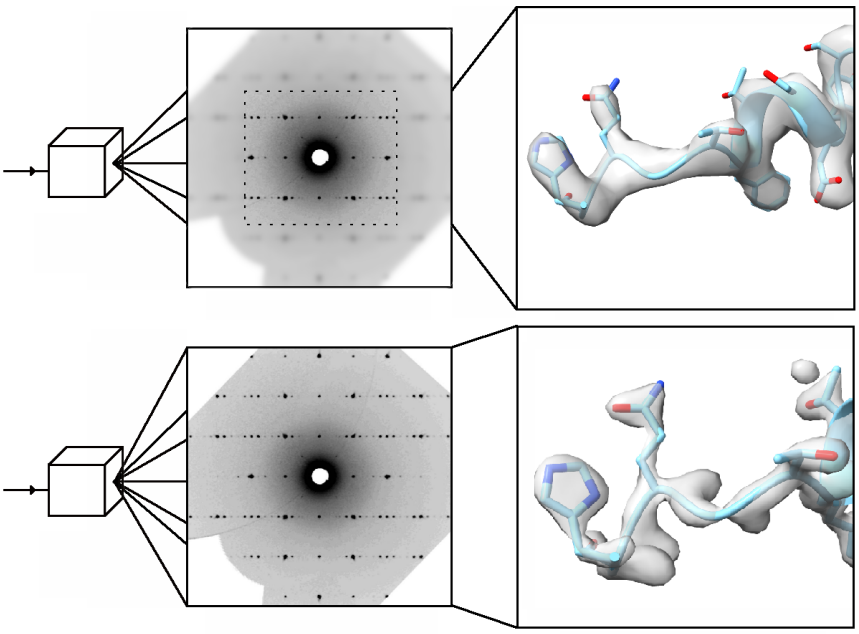


FIGURE 1.9: Comparison of a low-resolution (top) and high-resolution (bottom) XRD observation. The quality of the diffraction pattern and consequently the ED map depends on the quality of the crystal. Diffraction pattern source: [21]

Imperfections in the crystals may reduce the quality and resolution of the observed images (Figure 1.9). Even if the XRD experiment is a success, there is still no guarantee that the observed ED map can be used for automated structure determination.

As of 2022, over 230 million known protein sequences [26] have been determined and deposited to the European Molecular Biology Laboratory (EMBL) repository (Figure 1.10). In a stark contrast, only about 200,000 solved structures have been deposited in databases [27] such as the Protein Data Bank (PDB) [28] (Figure 1.11). The gap between the number of proteins with known sequences and of those with solved structures grows at an increasing pace which is illustrated by the staggeringly low success rate of model building efforts: it is estimated to be around 5.2% [29].

One of the factors limiting the success of automated model building is the resolution of observed ED maps, measured in Ångstroms ($1\,\text{Å} = 10^{-10}m$). XRD ED maps with a resolution above $3\,\text{Å}$ are considered high resolution

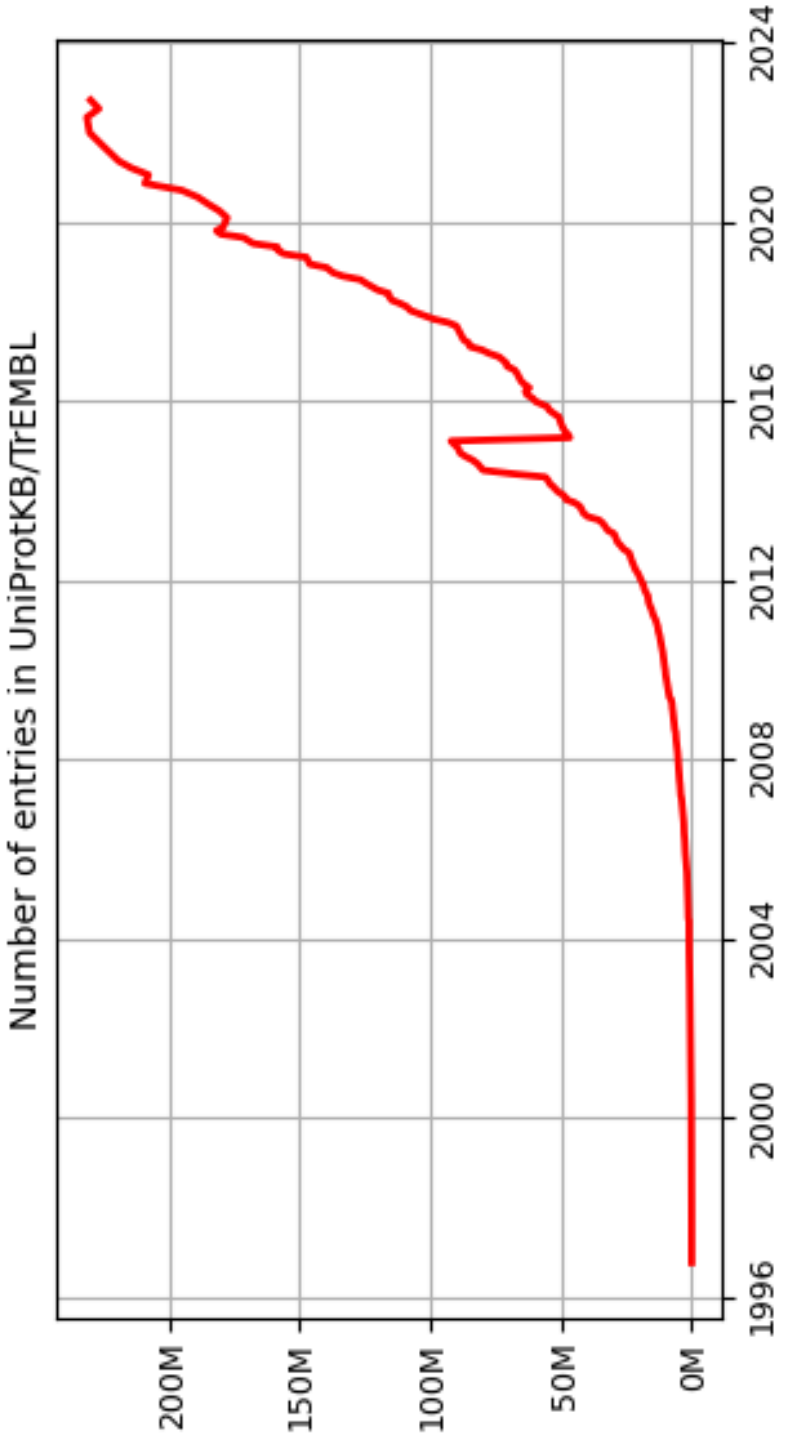


FIGURE 1.10: Growth of the number of protein sequences deposited to EMBL. Source: [26]

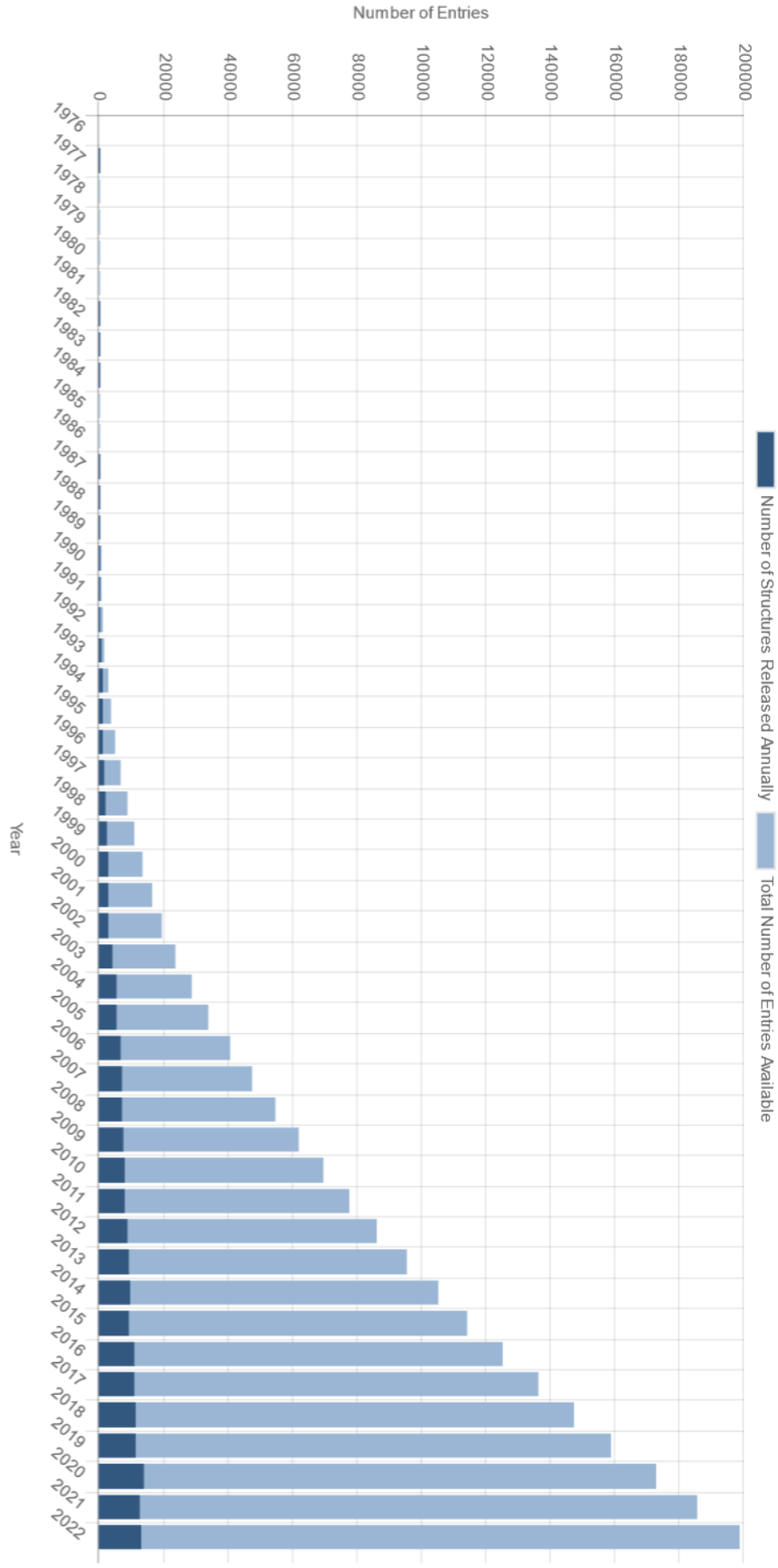


FIGURE 1.11: Growth of the number of deposited structural models in the PDB. Source: [27]

and are difficult to obtain. Over 150,000 structural models, an overwhelming majority, in the PDB were built into ED maps of at least 3.5 Å resolution (Figure 1.12). Since only successfully built models are deposited, the large gap between solved structures and known protein sequences highlights just how much of a limiting factor ED map resolution is.

The main reason for the reliance on higher resolution samples is that as resolution decreases (Figure 1.13), the characteristic side chains making residues distinguishable become less pronounced (Figure 1.9). Human experts and model building toolkits both rely on the characteristic shapes and magnitudes of residue electron densities to assign AA labels to the ED maps.

Low resolution samples tend to make automatic methods less reliable or even non-functional. This may necessitate the manual assignment of residues which, depending on the human interpretability of the ED map, may take days and still does not guarantee that a complete or accurate enough model can be constructed at all.

Literature [30] advises against attempting automated model building into ED maps worse than 3 Å resolution. For SBDD purposes even higher precision structural models obtained from ED maps with at least 2.5 Å resolution are recommended [31].

Summarizing the above, it can be concluded that resolution and computational time are two of the largest bottlenecks in structure determination. This necessitates the creation of methods which remain functional with low resolution ED maps and are capable of rapidly handling large volumes of samples.

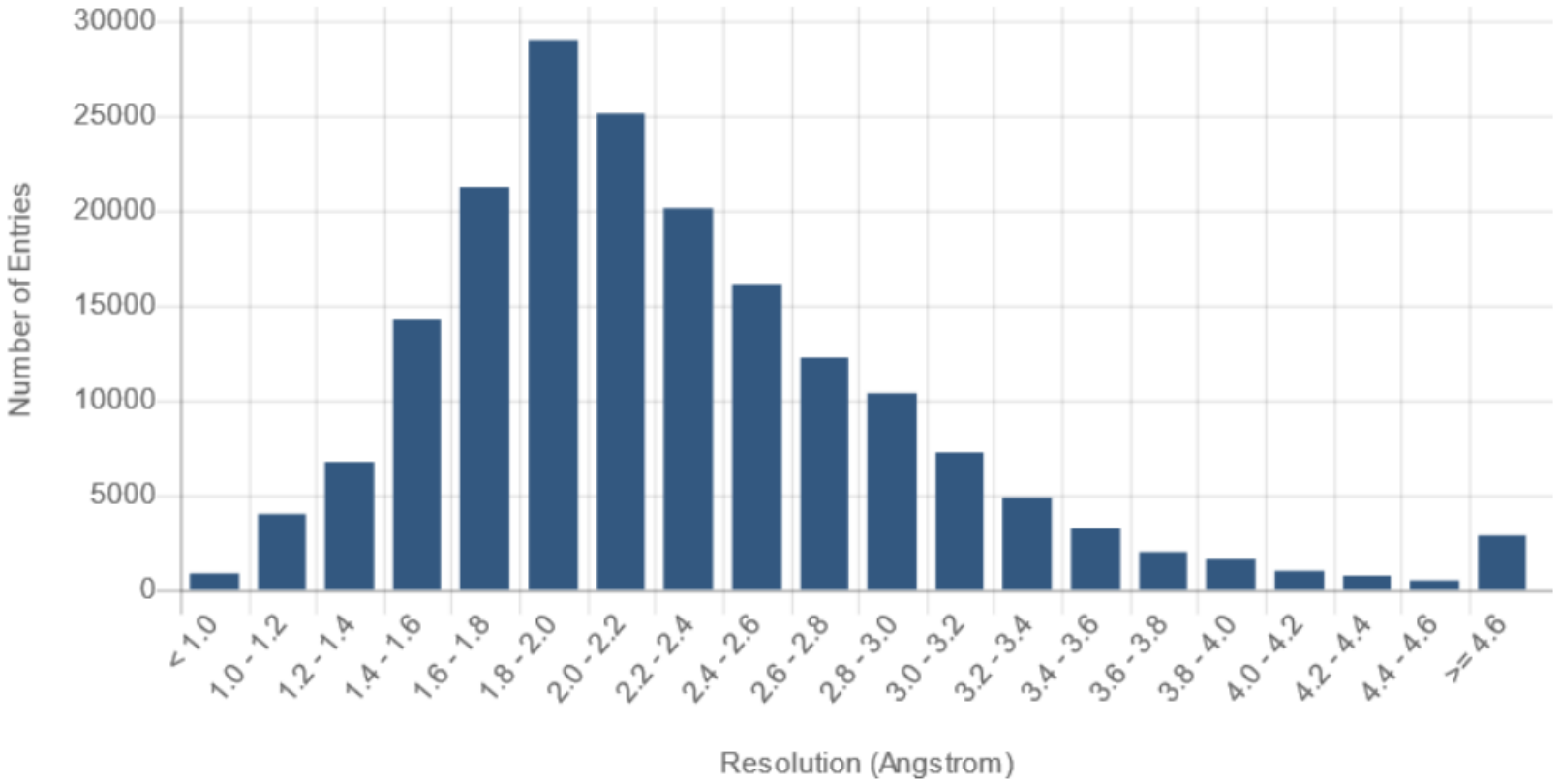


FIGURE 1.12: Histogram of the number of structural models deposited to PDB. The decrease in number of depositions below 3Å show that it is significantly harder to build models into lower resolution ED maps. Source: [27]

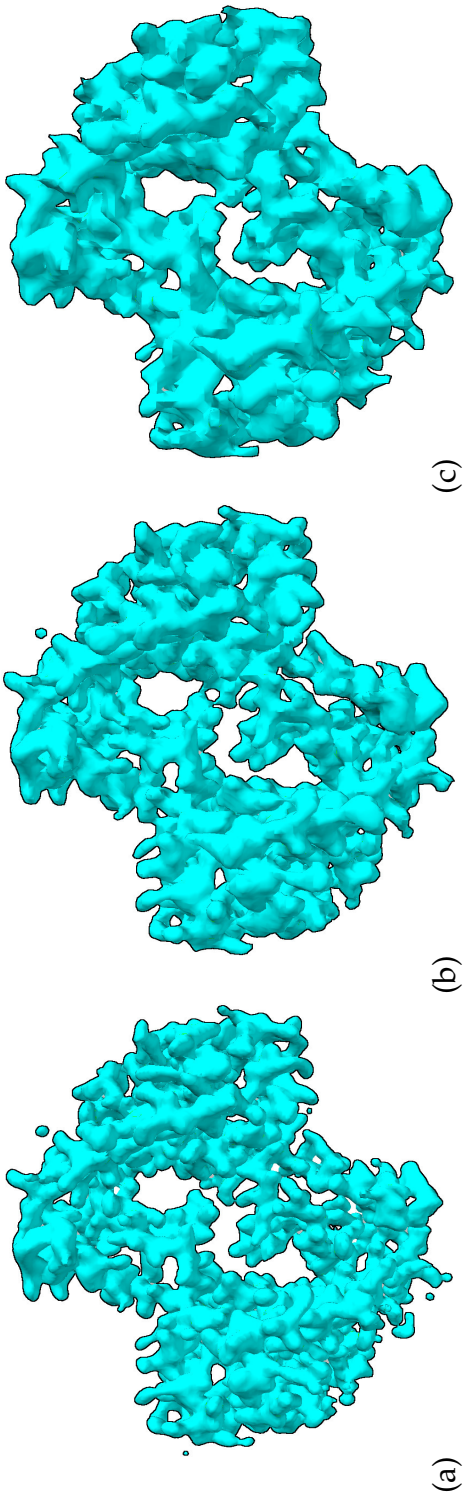


FIGURE 1.13: Pre-processed ED maps at 2 Å (a), 3 Å (b) and 4 Å(c) resolutions. It becomes harder to observe and distinguish residues as resolution decreases.

Chapter 2

Related Work

2.1 Structure Determination & Automatic Model Building Toolkits

Currently popular automatic model building toolkits take an iterative approach when building models into experimental observations such as ED maps. For samples with better than 2.7 Å resolution ARP /wARP [32] performs well when initialized from random atoms or small fragments. Auto-Build [33] [34] in *Phenix* [35] relies on RESOLVE [36] for secondary structure (helices, sheets, loops) assignment which can create 70-90% complete models for high-quality samples with better than 3 Å resolution. Buccaneer [37] handles main chain tracing using a Bayesian approach with 72% accuracy at or below 3 Å resolution and 79% for higher resolutions.

Machine learning has been used to assist the above toolkits. Earlier model building methods using machine learning relied on pattern matching [38] or feature-based approaches [39] [40]. In the toolkit *Coot* [41], NNs [42] are used to validate the built models.

The above take a more classical statistical approach rather than the computer vision approach this work presents. In [42], after a structural model is built, various features (e.g. bond angles, resolution and ED map-to-model correlation) for the protein structural model have to be manually calculated

and input into a feed-forward NN to regress to a correctness score for the built model. Although it's a NN-based approach, since its purpose is protein structural model evaluation instead of model building, its target is different from that of the proposed methods of this work.

Seqqy [43] is a Support Vector Machine (SVM) based state-of-the-art method for ED map primary structure assignment. It consists of 20 (one for each type of the 20 standard AAs) one vs. all SVM classifiers to determine each side chain's class. Seqqy is available as part of the ARP/wARP toolkit and heavily relies on its other modules to perform well: Seqqy assigns AA labels along the main chain traced by another module within ARP/wARP [44]. Seqqy can achieve 60% accuracy in sequence assignment at 2-3 Å resolution and up to 50% at lower resolutions. This work will take Seqqy as a basis of comparison, since its scope and capabilities (AA labelling) are closest to the methods proposed below.

2.2 Neural Networks for Volumetric Data Segmentation

ED maps are commonly represented as three-dimensional volumetric data [45] which is common in bio-imaging where the internal structures are just as or more important than surfaces. Still, for machine learning tasks the volumetric representation is used less often than alternatives such as point clouds [46] [47] [48] [49] or surface meshes [50] [51] [52]. Even though these representations reduce computational complexity, they would also discard essential information captured in the volumetric representation (e.g. surface meshes sacrificing internal structure).

It has been shown that directly operating on volumetric data outperforms

the 2D slice-based approach due to preserving important spatial relationships [53]. Though, until recently, directly operating on dense inputs with 3D neural networks wasn't viable due to the large processing power (particularly memory) requirements.

The 3D UNet [54] was one the first architectures to perform volumetric data segmentation and serves a basis for later methods. Xin Yang et al. [55] combined a 3D U-Net with a bi-directional recurrent neural network to segment prenatal ultrasound volumes. In brain MR segmentation, approaches such as VoxResNet [56] and various 3D U-Net based architectures [57] [58] [59] have proven successful. In [60], Lu et al. achieve results surpassing the 2D slice-based approach by using 3D convolutional neural networks to classify volumetric thyroid images.

Residual [61][62] and dense networks [63] [64] allow the construction of deeper networks with more learnable parameters while also allowing for improved gradient flow to avoid the problem of vanishing gradients. This is especially important with 3D volumetric data, which due to its large dimensionality, is prone to encounter the vanishing gradient problem. It has been shown that 3D ResUNets [65] outperform the classical 3D UNet.

2.3 Neural Networks in Structure Determination

2.3.1 Ab Initio Structure Prediction

Although not directly the scope of this work, significant progress has been made with ab initio structure prediction as well due to the introduction of NNs. The AA sequences from proteins lend themselves well for use with attention-based transformer models [67]. Interestingly, the multi-head attention in transformers with the AA sequence as its inputs can be considered a

revival of Protein Contact Networks [68], an approach preceding the prevalence of AI, which also used a contact matrix of the sequences to predict protein structure.

BERT-style models [69] [19] have recently gained traction and are the top performers for the CASP[70] structure prediction competition. For ab initio structure prediction, DeepMind's AlphaFold [19] has been a groundbreaking improvement. Although it produces high-accuracy structures in general, human supervision is necessary to validate the predictions. Furthermore, recent research [71] has revealed doubts about AlphaFold's performance for novel structures.

RoseTTA fold [18] also relies on a Transformer style architecture, but unlike AlphaFold, it relies not only on the sequence information, but also uses a 2D distance matrix as well. It uses a two-track architecture and connects the information flow of the two representations in each track. Unfortunately, they could not leverage the third track, the 3D representation due to memory limitations despite promising significant performance boosts. Still, it outperformed most methods in the CASP14 competition, losing only to AlphaFold.

Despite these advances, the validation of novel structures can be done by building a structural model into the observed ED map, which emphasizes the need for efficient methods for model building into experimental observations.

2.3.2 Semantic Segmentation of ED Maps

Cryo-EM potential maps are analogous to ED maps acquired via crystallography [72] [73] in that they are also volumetric snapshots of molecular structure. Due to this, it is important to consider methods dealing Cryo-EM potential maps as well.

For model building into experimental observations, Haruspex [74] is a semantic segmentation NN architecture which labels secondary structure elements (helices, sheets and loops) in Cryo-EM potential maps analogous to XRD ED maps using a U-Net style architecture. Moritz et al. propose a cascaded NN architecture for secondary structure segmentation of XRD ED maps [75].

2.3.3 Instance Segmentation of ED Maps

This work approaches locating individual AA residues as an instance segmentation problem. For 2D problems Mask R-CNN [76] and its offshoots [77] [78] offer fast and accurate instance segmentation. They perform a multi-stage process, by first proposing regions of interest (RoI) over the detected instances. This is why they are also referred to as Region-Proposal Networks (RPN).

RPNs then standardize the aspect ratio and size of proposed RoIs via an RoI pooling operation. This is necessary because further processing with NNs requires fixed size inputs. These RoIs are then refined into pixel-perfect instance masks.

A^2-net [79] is trained for model building into Cryo-EM potential maps using their custom A^2 (short for Amino Acid) data set. A^2-net works in manner similar to Mask R-CNN: it extracts individual AA instances and applies instance classifiers to acquire residue labels.

When dealing with AAs where size and aspect ratio are defining features, standardizing them via RoI pooling is not viable. Although it is a good fit for 2D images, by closely adhering to the Mask R-CNN, incompatibilities with the task such as this will inevitably arise. A^2-net overcomes the limitations of the Mask R-CNN RoI pooling layer by proposing an aspect-ratio preserving RoI pooling (APRoI) module.

The reliance on extra modules to patch incompatibilities instead of forethought in designing the architecture to match the problem brings with it a computational overhead. Furthermore, A^2-net relies on the AA sequence to first extract residues through region proposals. This is a single point of failure: if a residue is not extracted it will not be classified which leads to incomplete models. Other methods such as Seqqy [80] in ARP/wARP [81] also rely on such an approach and produce less complete models at lower resolutions.

An alternative to RPN-based instance segmentation is pixel offset regression (POR) [82]. Instead of generating and refining instance proposals in multiple passes, each pixel is assigned a keypoint and an offset vector pointing to their keypoint. Pixels sharing a keypoint are considered to be parts of the same instance. This approach is already being utilized for various applications [83] [84] that do not rely on instance proposals.

The methods proposed in this work overcome the limitations of both the RPN-based instance segmentation and the reliance on the AA sequence. This is achieved by:

- formalizing the AA instance segmentation problem with voxel-precision with POR in mind from the start to avoid unnecessary computational overhead when refining the AA instance masks;
- only relying on the ED map as inputs to bypass the single point of failure the mapping of AA sequence prior to semantic segmentation or instance classification represents; and
- designing architectures from the ground up to fit the problem definition instead of patching off-the-shelf methods.

Chapter 3

Problem Setting

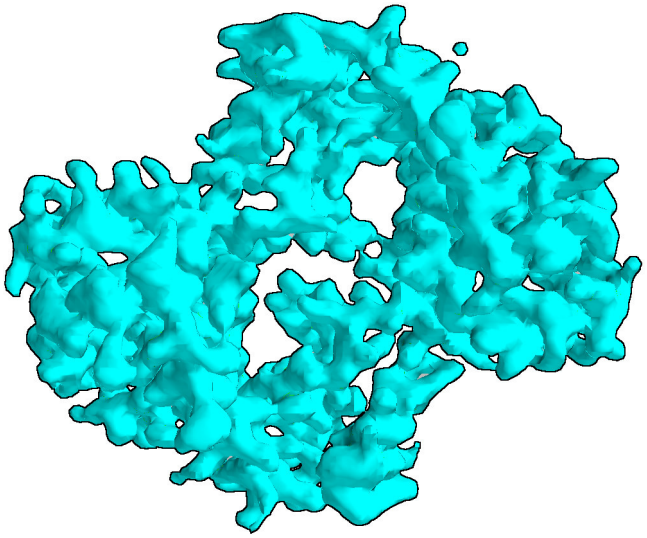


FIGURE 3.1: Medium resolution ED map represented as volumetric data. Voxel values correspond to the ρ average electron density value in the voxel's volume.

The task of finding and classifying amino acid residues in ED maps is cast as a computer vision, more specifically image segmentation, problem with three objectives:

1. assign amino acid class labels to each voxel of ED maps (semantic segmentation);

2. locate groups of voxels which belong to individual amino acid instances (instance segmentation);

3. combine the semantic prediction with the instance predictions to classify each residue in the protein.

3.1 Semantic Segmentation

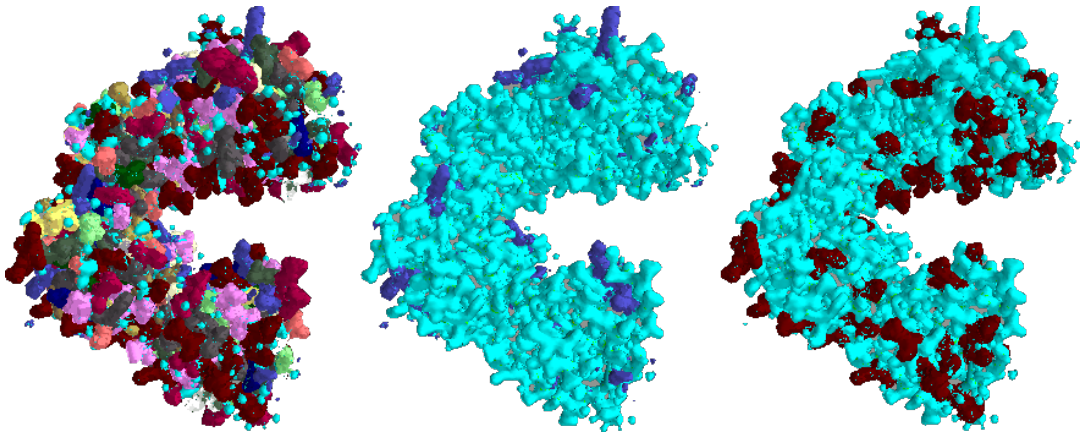


FIGURE 3.2: The semantic segmentation objective is to produce a class probability map $S(\mathbf{X})$ (left) by observing the ED map $\mathbf{X}$. $S(\mathbf{X})$ is composed of multiple channels $S(\mathbf{X})_c$ (middle, right), each corresponding to probabilities of voxels belonging to individual AA residues.

The de facto standard of protein crystallography, the ccp4 format [45], represents ED maps as three-dimensional volumetric data. In these 3D arrays $\mathbf{X} \in \Re^{W \times H \times D}$, $X[x, y, z] = \rho$ is the average electron density value for a given (x, y, z) voxel's volume (Figure 3.1).

The semantic segmentation goal is to generate a multi-channel class probability map $S(\mathbf{X}) \in \Re^{W \times H \times D \times C}$ (Figure 3.2 by only observing the ρ density values in $\mathbf{X}$. $S(\mathbf{X})[x, y, z] = \mathbf{c} \in \Re^{C}$ is the class probability vector of a voxel (x, y, z) belonging in each of the possible C classes (AA residue labels).

3.2 Instance Segmentation

The goal of instance segmentation is to locate amino acid instances $g_k \in G$ in the input ED map $\mathbf{X}$. Following the idea of POR, This is done via an

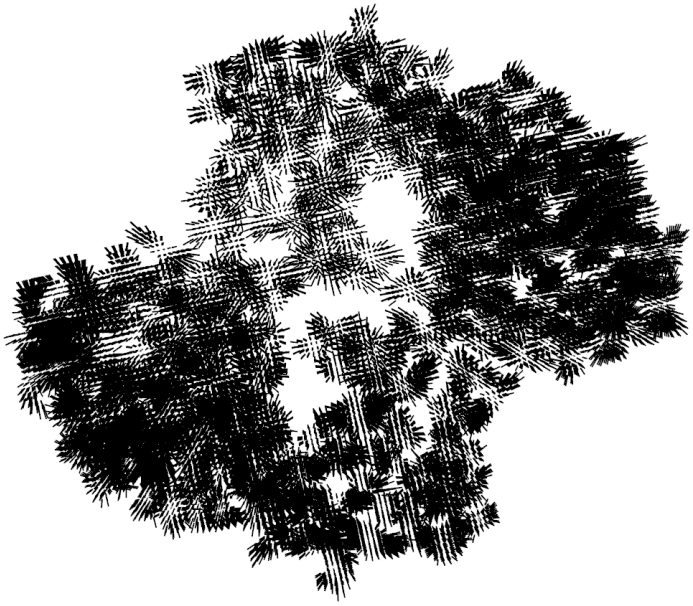


FIGURE 3.3: The instance segmentation objective is to produce a vector field $I(\mathbf{X}) \in \Re^{W \times H \times D \times 3}$. Each vector points to the instance centroid assigned to its voxel.

offset vector field $I(\mathbf{X}) \in \Re^{W \times H \times D \times 3}$. Every voxel votes for its instance $g_k = \{(x, y, z)\} \in G$ via an offset vector $I(\mathbf{X})[x, y, z]$ pointing at its instance centroid (x_k, y_k, z_k) (Figure 3.3). An instance is a group of voxels sharing the same centroid: $g_k = \{(x_i, y_i, z_i) | I(\mathbf{X})[x_i, y_i, z_i] + (x_i, y_i, z_i) = (x_k, y_k, z_k) \forall i\}$.

Since ED maps contain regions other than AA volumes, voxels not belonging to the 20 standard residues (e.g. zero-density voxels, solvent molecules, noise, etc.) are denoted by a zero-vector offset and assigned to a special background/void instance $g_{BG} = \{(x_i, y_i, z_i) | I(\mathbf{X})[x_i, y_i, z_i] = \mathbf{0} \forall i\}$.

The number of instances, K, is known from the number of residues constituting the protein's AA sequence. The NN algorithms described below take $\mathbf{X}$ as their only input and generate $S(\mathbf{X})$ and $I(\mathbf{X})$. Other than when evaluating results, they do not rely in any way whatsoever on the residue sequence (e.g. number of residues per class, residue positions in the sequence or neighbourhood information, etc.) of the protein.

3.3 Residue Classification

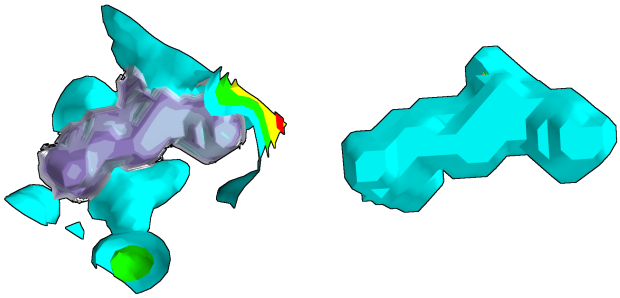


FIGURE 3.4: Bounding region based instance proposal (left, instance highlighted in blue) includes background and regions of other instances due to the dense nature of ED maps. Using POR (right) allows instances to be defined with voxel precision from the start.

Finally, instance classification is achieved by sampling the class probability map in the predicted (x_i, y_i, z_i) voxels of an instance g_k, aggregating over all voxels and taking the most probable as the instance's amino acid class label: $class(g_k) = argmax(\sum_i S[x_i, y_i, z_i])$ where $(x_i, y_i, z_i) \in g_k$.

To facilititate the use of POR, instances are defined on a voxel-by-voxel basis instead of using bounding volumes. Due to the dense packing of AA residues, instance bounding volumes would have large overlaps with other instances. Furthermore, they would also contain large volumes of background voxels (Figure 3.4).

By using a POR-based approach, a more precise more precise definition of instances is possible. This ensures a one-to-one relationship between voxels and their instances due to the lack of overlapping bounding volumes.

Chapter 4

Data Set

4.1 Data Acquisition

The proposed methods of this thesis are 3D CNN architectures trained with a supervised learning scheme. A data set consisting of ED map samples and corresponding expected outputs, the ground truth (GT) is necessary to train and evaluate the performance of NNs. As no public ED map data set with labels for supervised semantic and instance segmentation exists as of writing, a custom data had to be established from scratch.

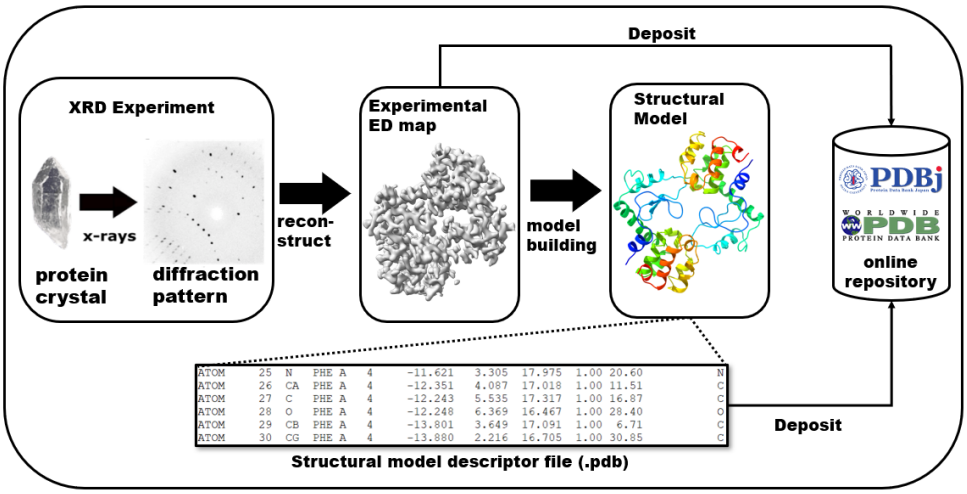


FIGURE 4.1: Overview of the deposition pipeline. The ED map observed in the XRD experiment and the structural model built into it are deposited together.

The results of XRD experiments are deposited into open data bases such as the Protein Data Bank (PDB) [85]. The experimentally acquired ED maps

```
CRYST1   26.423   58.469   63.884  90.00  90.00  90.00 P 21 21 21    4
ORIGX1      1.000000  0.000000  0.000000        0.00000
ORIGX2      0.000000  1.000000  0.000000        0.00000
ORIGX3      0.000000  0.000000  1.000000        0.00000
SCALE1      0.037846  0.000000  0.000000        0.00000
SCALE2      0.000000  0.017103  0.000000        0.00000
SCALE3      0.000000  0.000000  0.015653        0.00000
ATOM      1  N   SER A   1       3.766  13.373   6.089  1.00 63.98           N
ATOM      2  CA  SER A   1       3.576  14.832   5.815  1.00 63.70           C
ATOM      3  C   SER A   1       3.128  15.702   7.000  1.00 62.26           C
ATOM      4  O   SER A   1       1.936  15.919   7.279  1.00 62.46           O
ATOM      5  CB  SER A   1       2.748  15.090   4.536  1.00 66.44           C
ATOM      6  OG  SER A   1       3.578  15.005   3.359  1.00 69.61           O
```

FIGURE 4.2: Excerpt from a .pdb structural model file. CRYST, ORIGX, and SCALE records contain information about the unit cell. ATOM records contain information about each atom's type, parent residue, co-ordinates and element.

(in the .ccp4 format) and the structural models (in the .pdb format [86]) built into them are deposited together (Figure 4.1).

The structural model descriptor file contains detailed information including the experimental configuration and the type and location of residues. Records pertaining to each residue define both their position of in the residue chain and also the co-ordinates of all their constituting atoms (Figure 4.2).

First, to determine if the problem can be solved by NNs, a clean ED map data set is created. This allows the direct observation of network performance without noise and artifacts affecting performance measurements. Already solved structural models from PDB are acquired and fixed resolution ED maps are generated based on them.

Structural models meeting the following criteria were considered:

1. chain type is protein;
2. no ligands;
3. oligomeric state is 1 (monomer);
4. experimental method is X-ray diffraction.

Training and validation data sets are created from 6990 structural models. For each structural model, three ED maps are generated with *Phenix* [35]: at 2 Å, 3 Å, and 4 Å resolution. The ED maps are in the *.ccp4* map format [45].

The result is three data sets consisting of the same proteins, but with each having ED map samples at a different, fixed resolution. Obtaining three data sets like so allows the NNs to directly target high, medium or low resolution samples. Furthermore, this also enables contrasting and comparing the effects of decreasing ED map resolution (Figure 1.13).

Real-life ED maps have varying resolution and quality, on top of the .pdb structural model built into them possibly not being completely accurate. The reliability of structural models is measured via their R-factor values [87]:

$$R = \frac{||F_{obs}| - k|F_{calc}||}{|F_{obs}|}$$

where $|F_{obs}|$ is the observed sample intensities, $|F_{calc}|$ are the intensities calculated based on the built structural model and k is a scale factor. The R-free value describes the discrepancy between experimental observations and the ideal, calculated values [88] based on the .pdb model. If the R-factor of an observation is too high, the observation may be closer to noise rather than actual diffraction data. $R = 0.59$ corresponds to randomly distributed atoms.

$|F_{calc}|$ corresponds to intensities based on the structural model, which can be reconstructed into a synthetic ED map which looks as if the determined structural model could be built into it. Structure building toolkits such as *Phenix* contain modules to create such synthetic data which closely resemble real-life observations so the R-Factor calculation is precise. This pipeline (Figure 4.3) is utilized to create the synthetic data set's ED maps. This ensures that noise is minimal and that the ED maps on the structural models are as close to real data as possible (Figure 4.4).

Although the above is an efficient way to determine if the problem can be solved by the proposed methods, any results achieved on the fixed resolution training and validation sets may not reflect performance on actual experimental data, although the domain gap is minimal. To measure that, a

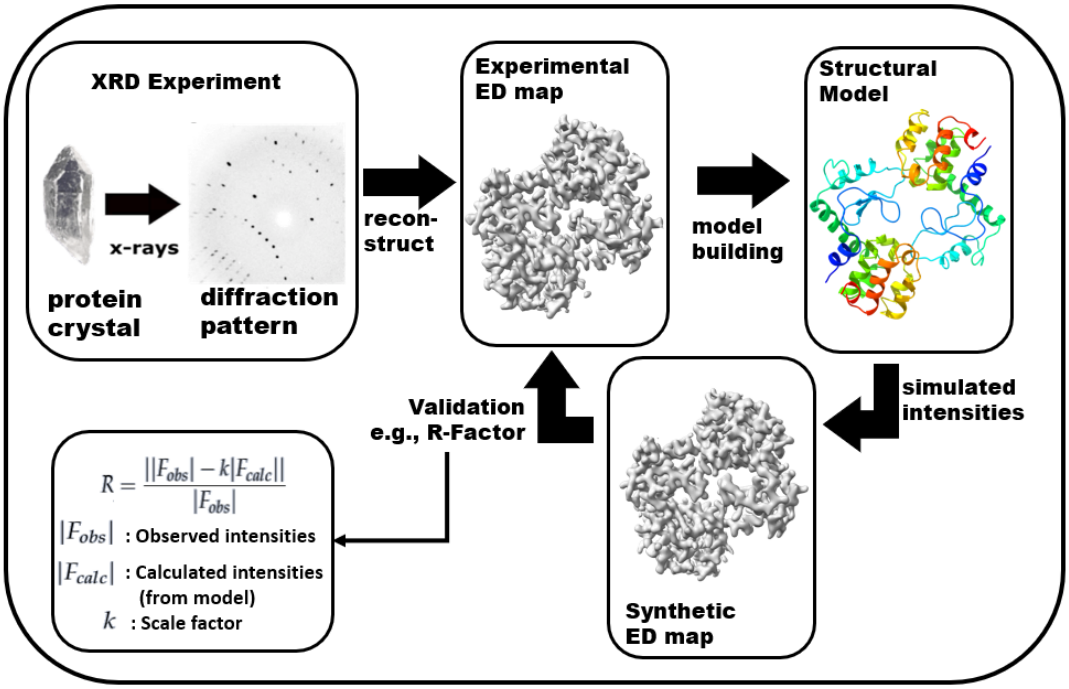


FIGURE 4.3: The validation pipeline used to create synthetic samples closely resembling experimental ED maps in *Phenix* for calculating the R-Factor is used to create the fixed resolution ED maps of the synthetic data set.

data set of ED maps converted to the .ccp4 format directly from experimentally observed reflection data, as opposed to being based on the structural models (.pdb files).

To ensure the fidelity of the GT (if the R-factor is too high, the GT can't be trusted), this dataset is sourced from PDB-REDO [90] instead of the PDB. PDB-REDO is a repository for structural models refined further after their initial deposition to PDB, ensuring that the highest quality structural models are used for GT.

The experimental test set contains 594 proteins between 2–4 Å resolution with an R-free (the R-factor calculated on a separate, validation set of intensities) value under 0.35. 50 proteins each from over 2.5 Å, 2.5–3.5 Å, and below 3.5 Å resolutions are reserved for an experimental test set used to measure performance on real-life data. The rest are reserved for fine-tuning the networks after training with the fixed resolution data.

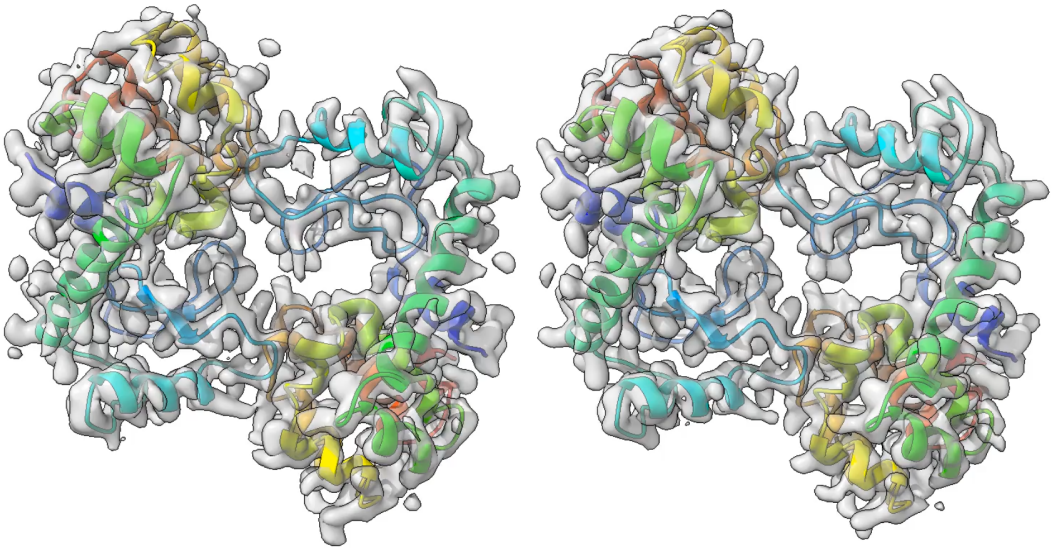


FIGURE 4.4: Experimentally observed (left) and generated (right) of a protein (PDB ID 104L [89]) with its structure overlaid. *Phenix* generates ED maps very close to real-life data.

4.2 Data Preprocessing

By default, the contents of a .ccp4 format ED map correspond to the so-called unit cell of the protein crystal. The unit cell represents the smallest unique part of the protein crystal which repeats infinitely. They organize into a lattice based on their symmetry space group. It is important to note that due to the infinite tiling, a unit cell may not enclose an entire ED map of the protein by only containing fragments which form the ED map once the unit cells are tiled properly.

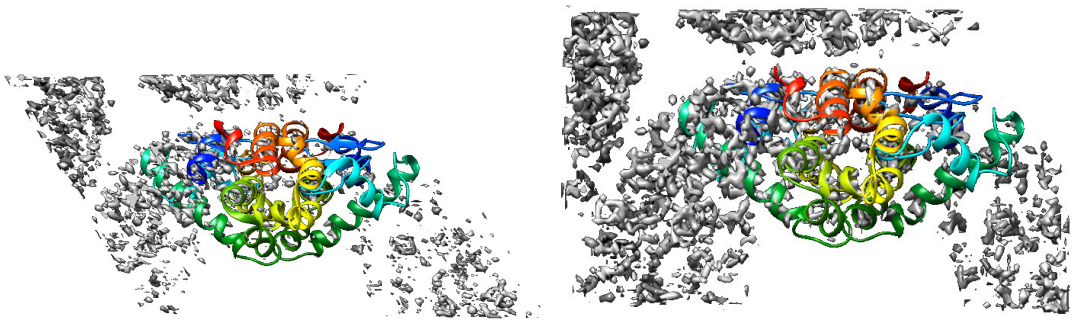


FIGURE 4.5: A unit cell in a non-orthonormal basis (left) and transformed to an orthonormal basis (right). The ribbon model shows the protein structure, highlighting how more than one complete ED may be present in the unit cells.

The unit cell base vectors depend on the space group and may not be orthonormal. To ensure that AA shapes are consistent across all samples, each ED map is transformed to the P1 space group, with an orthonormal basis (Figure 4.5).

This is done by relaying the information from the CRYST, ORIGX and SCALE records of the *.pdb* (Figure 4.2) to Phenix. CRYST records contain information about the unit cell's parameters and space group. ORIGX records present the the transformation from orthogonal co-ordinates to the space group of the crystal. SCALE records contain the transformation from orthogonal co-ordinates to crystallographic co-ordinates. During the process, the ED maps are sampled so that a distance of 2 voxels corresponds to the physical resolution of the data.

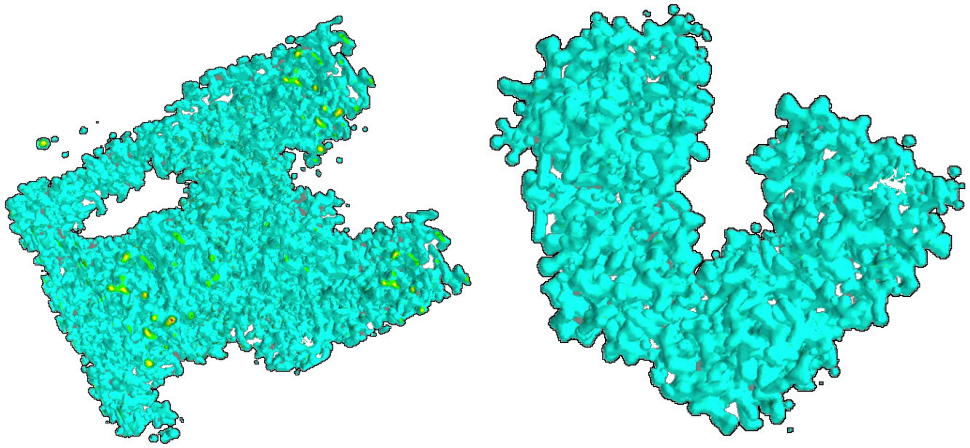


FIGURE 4.6: Unit cells before (left) and after (right) centering and isolating a single, complete map.

To remove repeats stemming from the repeating crystal nature of the samples so-called virtual unit cells are centered a complete ED map of each protein ensuring that the result contains a full ED map of the protein. All regions outside the 3 Å radii of atom positions are culled from the ED maps, leaving a volume containing only a single complete ED map (Figure 4.6).

ED map magnitudes are normalized by applying a procedure similar to the one in [75]. Setting voxels below a threshold intensity value of 1.5 to

zero creates a uniform background (BG) and clearly separates it from voxels with actual density information. The minimum of the remaining non-zero voxels is shifted to zero. To deal with outliers, the intensities are divided by their median and any voxels with outlying intensity values are clamped to the 98th percentile.

Due to the varying sizes of ED maps, it is not possible to perform mini-batch training with them as-is, since that would require equal sized inputs. Resizing the ED maps is not feasible because that would alter the resolution of the samples and also undesirably alter the shapes of residue side chains. Instead, equally sized windows are cropped from the normalized ED maps (Figure 4.7).

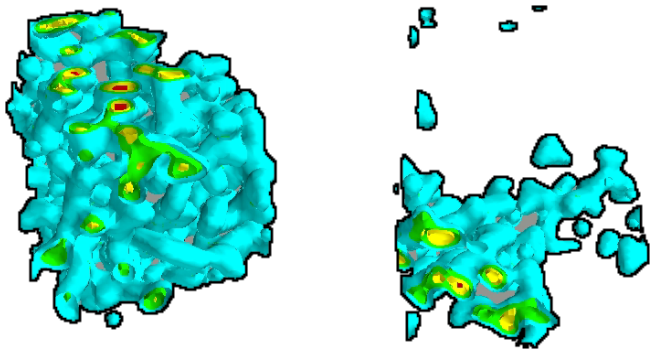


FIGURE 4.7: Equal sized windows with the internals visible. Cropping windows from the varying size ED maps allows for mini-batch training. Note how cropping windows may cut residues in half around the edges.

As the resolution decreases, each voxel corresponds to longer distances, leading to the field-of-view (FOV) widening if window sizes stay constant. Combined with residue side chains becoming less defined (Figure 4.8), extra complexity is present at lower resolutions. To alleviate this, window sizes are $48 \times 48 \times 48$ voxels for the 2 and 3 Å data sets, and $36 \times 36 \times 36$ voxels for the 4 Å data set. The reduction in size keeps the FOV consistent at lower resolutions.

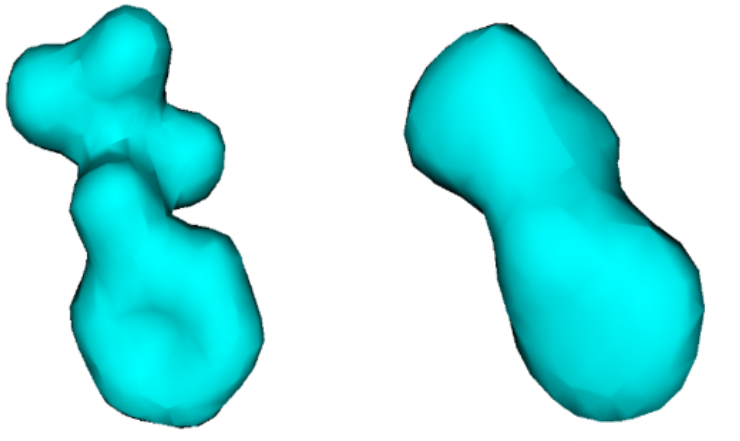


FIGURE 4.8: As the ED map resolution decreases AA side chains become less discernible. At high resolutions (left), the individual atoms and bonds of the PHE residue are easy to identify, while at low resolutions (right) human interpretation becomes difficult.

4.3 Data Augmentation

The aim of data augmentation is not specifically to increase the number of input samples, since there are already over 350,000 input windows across the three data sets (Table 4.1). Rather, the objective is to balance out the individual AA residue observation probabilities. This is measured via the Chi-Square Distance (CSD), defined as:

$$CSD(H,U) = \sum_{c} \frac{(H(c) - U(c))^2}{H(I)},$$

where H is the normalized distribution of observation probabilities, and U is an ideal uniform observation probability distribution. $c \in [0, C)$ is a class in the distributions with $H(c)$ (in the normalized, observed case) and $U(c)$ (in the ideal case) its observation probability.

Using a breadth-first search algorithm, the ideal number of occurences for each window in the data set is determined so that they minimize the CSD. Then, using 3D rotation and selection operations the data sets are augmented.

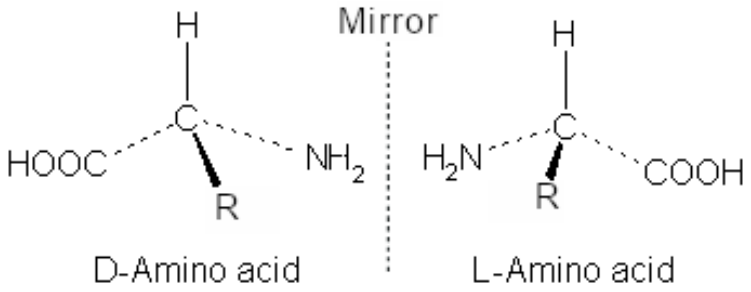


FIGURE 4.9: L/D-enantiomers of a chiral AA. The central carbon atom is the center of chirality. Since natural proteins only consist of L-enantiomers and the D-enantiomers are produces by mirroring or flipping, only rotations are legal augmentation operations.

Training set augmentation operations are rotations around the primary axes in multiples of 90° . During the augmentation process, the optimal number of occurrences (between 1 and 10: the original and 9 possible rotations) is assigned to each window so that the overall CSD of the fold is minimized.

Inversions and flips are not legal augmentation operations. This is due to the fact that AA residues are chiral with the exception of the non-chiral glycine. Chirality, or handedness, is a stereochemical property of tetrahedral molecules arising from asymmetry around an atom which becomes the center of chirality [91]. In tetrahedral molecules such as AAs, if all substituents are different to a central atom, that atom becomes a chiral center (Figure 4.9). In AAs, the center of chirality is the α-carbon to which the functional groups and the side chain is attached. Due to this asymmetry, the transformation capable of permuting the order of substituents short of reassembling the molecule is mirroring.

The non-superimposable mirrored pairs are referred to as enantiomer pairs. Although their chemical properties are identical, their physical properties differ. One enantiomer rotates polarized light left or counterclockwise hence being named levorotatory enantiomer (L-enantiomer), the other rotates polarized light right or clockwise and is called the dextrotatory enantiomer (D-enantiomer) of the chiral molecule. Due to a natural phenomenon

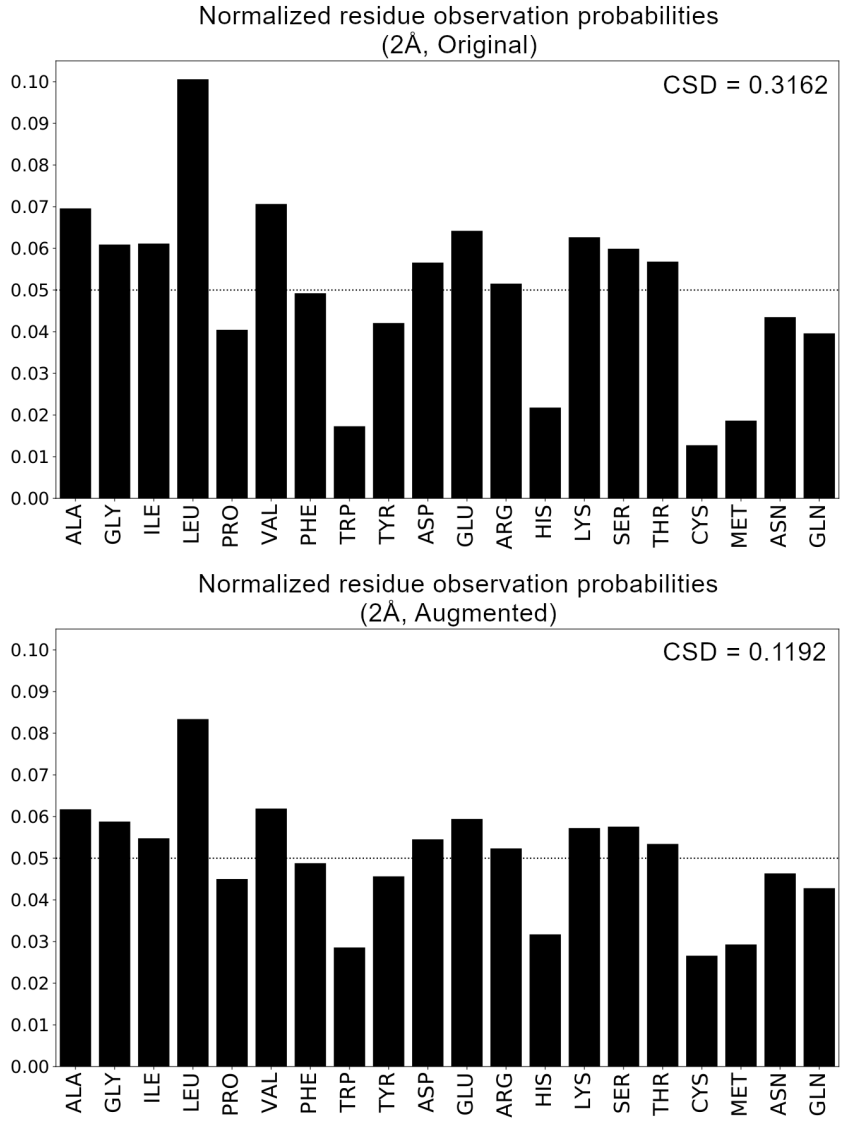


FIGURE 4.10: Normalized residue observation probabilities of the 2Åata set windows. The dashed line represents the 20-class uniform distribution with a 0.05 probability for each class.

called homochirality, only the L-enantiomers occur in natural proteins and as such no D-enantiomers should appear in the data sets either.

Validation sets are augmented via selections: augmented validation sets consist of 80% of the original windows with the lowest possible CSD. Selections are preferred over rotations to promote the uniqueness of validation set windows and reduce redundancy from repeating windows which could affect performance measurements. The effects of augmentation on the number of windows and CSD are detailed in Figure 4.10 and Table 4.1.

TABLE 4.1: Comparison of window counts (N) and CSD pre- and post-data augmentation of the Training (Trn.) and Validation (Val.) sets

Trn.	N Pre	N Post	Val.	N Pre	N Post
2 Å	148849	305162	2 Å	37212	29770
3 Å	76403	159601	3 Å	19100	15281
4 Å	90064	188288	4 Å	22516	18013
Trn.	**CSD Pre**	**CSD Post**	**Val.**	**CSD Pre**	**CSD Post**
2 Å	0.3162	0.1192	2 Å	0.3166	0.2555
3 Å	0.3267	0.1185	3 Å	0.3207	0.2735
4 Å	0.3230	0.1494	4 Å	0.3234	0.2702

4.4 Ground Truth Annotation

When a structural model is deposited into repositories such as the PDB, both the experimental observations and the structural model built into those observations are deposited together. The GT is based on the structural model description .pdb file. When building the experimental data set both the experimentally observed ED map and the .pdb file are downloaded and the ED map is annotated according to the structural model. For the synthetic data set only the .pdb file is acquired and the ED maps are generated from it at fixed resolutions before annotating (Figure 4.11).

By reading the ATOM records of the .pdb file (Figure 4.2), every constituting atom's co-ordinates and parent residue type can be obtained. Annotations take the form of Gaussian spheres placed at each atom's co-ordinates (Figure 4.12). Sphere radii depend on the resolution: $R_2 = 3$, $R_3 = 2$ and $R_4 = 1$. The standard deviation is half the radius for all resolutions: $\sigma = R/2$.

The semantic segmentation GT, $\hat{S}(\mathbf{X}) \in \Re^{W \times H \times D \times C}$, is a multi-channel probability mask describing the expected $S(\mathbf{X})$ semantic output of the network. The Gaussian annotations are placed at the atom co-ordinates in the GT channel $S(\mathbf{X})_c$ corresponding to the parent residue's assigned class label $c \in [0, C)$.

From each (x, y, z) voxel's perspective, the end result is a $\hat{S}(\mathbf{X})[x, y, z] =$

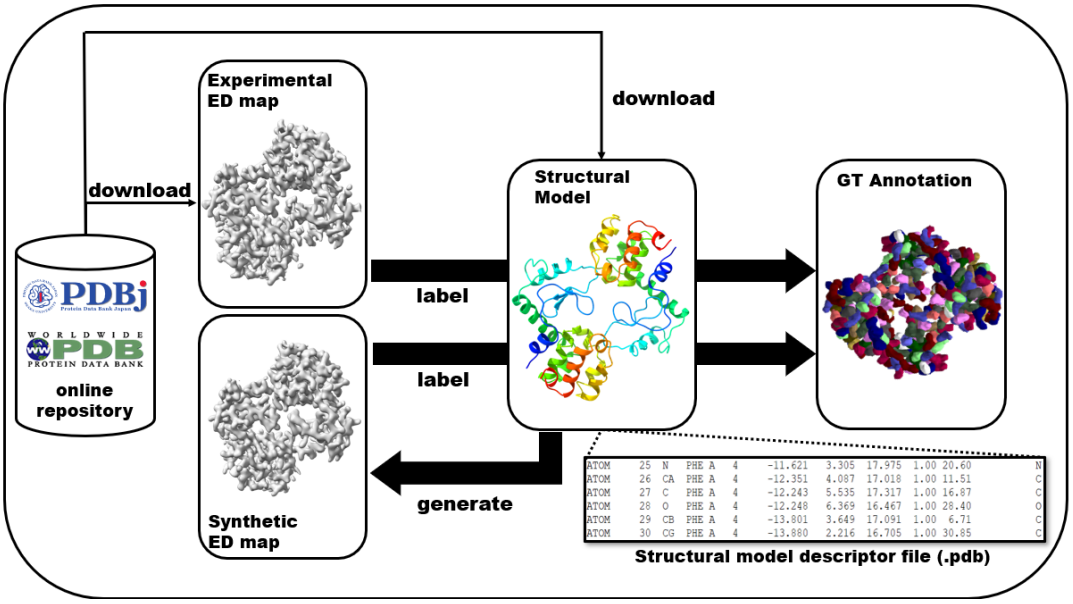


FIGURE 4.11: Since both the synthetic and experimental ED maps correspond to the respective .pdb structural model descriptor file, the annotation process is the same for both. For experimental samples the ED maps are directly annotated using the .pdb file, for synthetic samples the ED map is first generated from the .pdb at a fixed resolution.

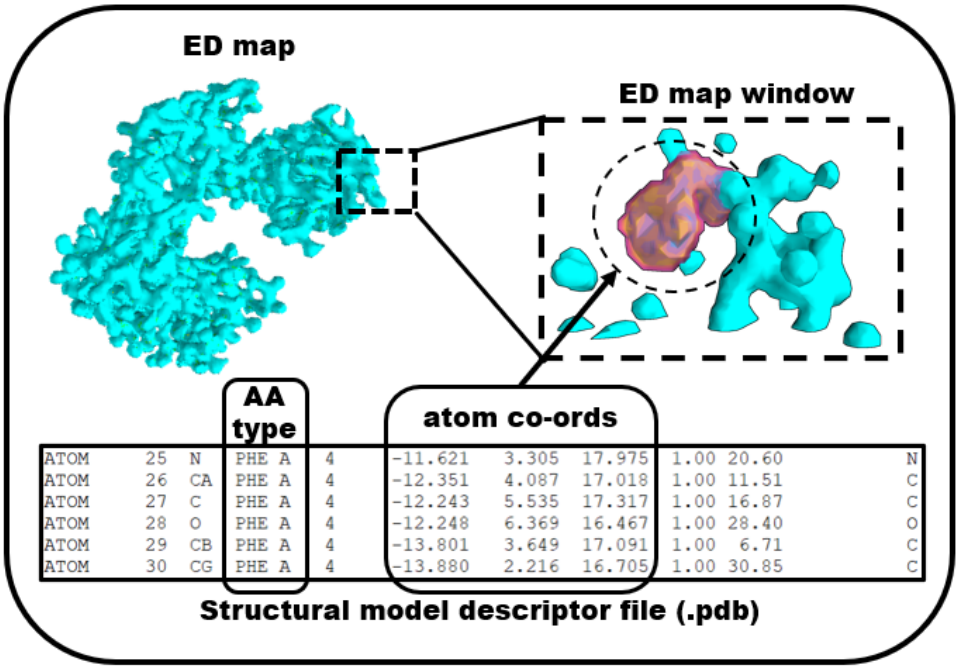


FIGURE 4.12: GT annotations are guided by the .pdb structural model descriptor file. Using the ATOM records, annotations according to the AA type can be placed in the atom co-ordinates.

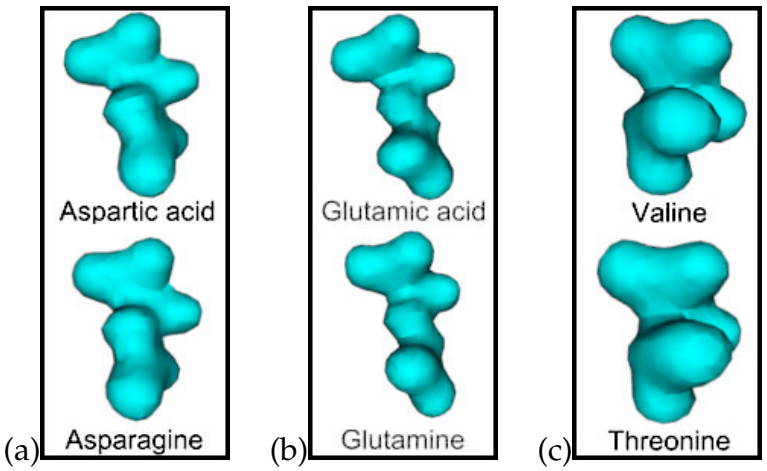


FIGURE 4.13: ASN & ASP (a), GLN & GLU (b) and VAL & THR (c) were combined into single classes in the semantic segmentation class. This is due to their near identical electron densities even at 2 Å resolution.

$\hat{\mathbf{c}} \in \Re^C$ vector describing the likelihood of (x, y, z) belonging in class $c \in [0, C)$. To guide the NNs towards increased confidence when rendering predictions, $\hat{S}(\mathbf{X})$ is reduced to a single-channel expected class label mask by applying the arg max operation over the channels. This results in a $W \times H \times D$ shaped GT mask where each value is the expected class label $c \in [0, C)$.

The residue pairs of ASP/ASN, GLU/GLN and VAL/THR are easily confused by human experts even at high resolutions due to their similar side chains. These pairs (Figure 4.13) are grouped into a single class each.

Voxels where $\mathbf{X}[x, y, z] > 0$ but are not parts of the 20 standard AA residues (solvents, noise, etc.) and those where $\mathbf{X}[x, y, z] = 0$ are assigned to an extra BG or void class. Voxels that would not receive a semantic class label for whatever reason by the end of this labeling process also get assigned to the BG class. This results in $C = 18$, with 17 AA classes and the additional BG class.

The instance segmentation GT, $\hat{I}(\mathbf{X}) \in \Re^{W \times H \times D \times 3}$ uses the same Gaussian spheres as above. While creating the semantic annotations for a residue, its centroid is calculated as the mean of the residue's atom co-ordinates. All

voxels' offset vectors inside the annotation spheres are set to point at the parent residue's centroid.

Similarly to $\hat{S}(\mathbf{X})$, voxels where $\mathbf{X}[x, y, z] = 0$ (non-density points) or $\hat{S}(\mathbf{X})[x, y, z, c] = 0, c \in [0, 17)$ (voxels part of the BG) are assigned a zero-vector offset: $\hat{I}(\mathbf{X})[x, y, z] = (0, 0, 0)$.

Using offset vectors instead of the absolute positions in the GT meshes well with the use of POR and, more importantly, allows the networks to learn a position-independent encoding of AA locations when performing instance segmentation.

Chapter 5

Methods – Semantic Segmentation

This work proposes 3D convolutional NNs as the solution to the problem definition. First, a custom semantic segmentation NN, the 3DFC-DenseNet (Figure 5.1) is presented. The motivation of of this network is to show that the problem of AA label assignment can be defined as a semantic segmentation problem and that NNs are capable of solving it.

Inspired by the two-dimensional FC-DenseNet [64] family of architectures, it is designed to handle volumetric data as inputs. FC-DenseNets perform particularly well in two-dimensional semantic segmentation tasks and can be considered an upgrade to the U-Net [92].

5.1 3D FC-DenseNet Network Architecture

DenseNet-based architectures [63] consist of so-called Dense Blocks (DB). Within each DB are multiple Dense Layers containing a convolutional layer, a ReLU non-linearity, dropout and batch normalization (Figures 5.2, 5.3). The output of each Dense Layer is concatenated and passed along to the final Dense Layer to form the output, creating a more dense feature map (hence the name) and allowing for feature reuse within DBs. The gradient can flow along these concatenations during back-propagation preventing vanishing gradients. This is especially beneficial for the large input dimensions of volumetric data and an advantage over 3D U-Net [93] architectures.

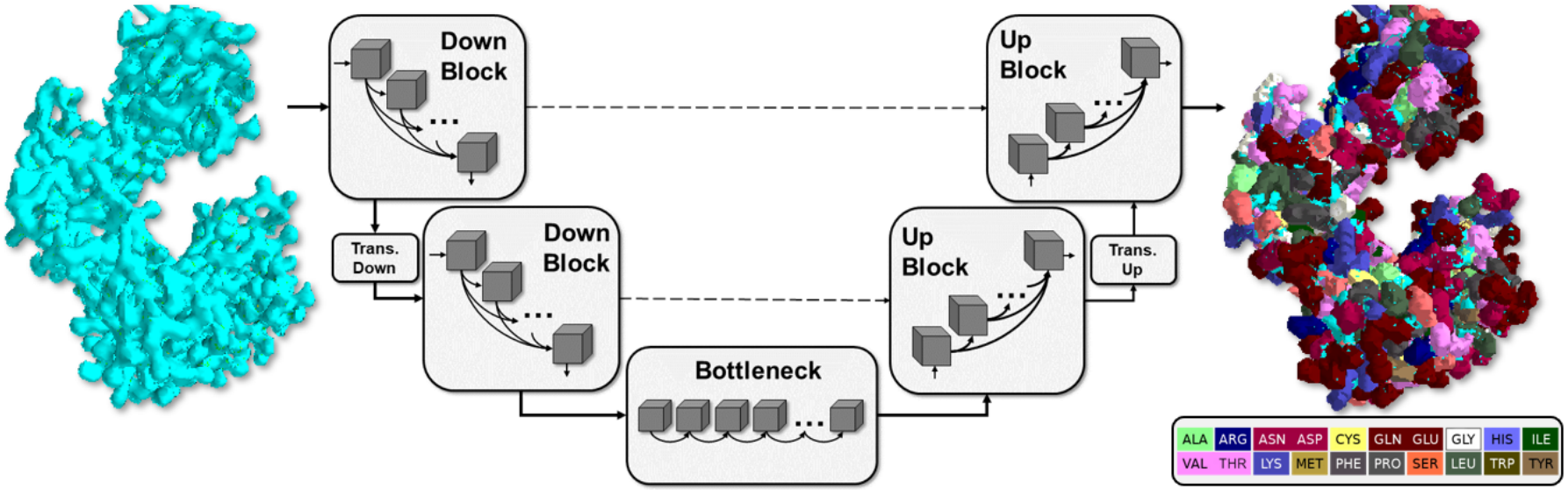


FIGURE 5.1: The proposed deep neural network architecture, 3DFC-DenseNet, can assign AA labels directly to protein electron density maps without relying on their residue sequence information.

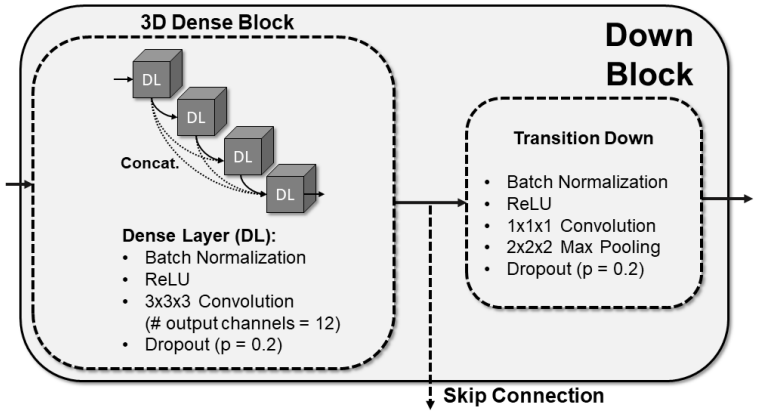


FIGURE 5.2: Diagram of the 3DFC-DenseNet architecture's Down Block. Dense blocks consist of four dense layers each, the concatenations along Dense Layers (DL) enable better gradient flow.

The network architectures are constructed using three-dimensional DBs to handle volumetric data. Each DB contains four dense layers (a $3 \times 3 \times 3$ convolution followed by batch normalization and a ReLU non-linearity), with a growth rate (the number of outputs of each Dense Layer) of 12 and a dropout probability of 0.2.

Transition Down (TD, Figure 5.2) and Up (TU, Figure 5.3) blocks manipulate feature map sizes between DBs. TD blocks halve feature map sizes towards the bottleneck via $2 \times 2 \times 2$ max pooling, preceded by a batch normalization layer, a ReLU non-linearity and $1 \times 1 \times 1$ depth convolution. TUs double feature map sizes from the bottleneck towards the output using a 3D transpose convolution layer of kernel size $3 \times 3 \times 3$ with a stride of 2 to increase feature map sizes.

The final $1 \times 1 \times 1$ depth convolution reduces the number of output channels to correspond to the number of classes. A softmax layer is applied over the channels to acquire the per-voxel class probabilities as a sparse segmentation mask as the output.

Concatenations along skip connections between TU and TD allows the

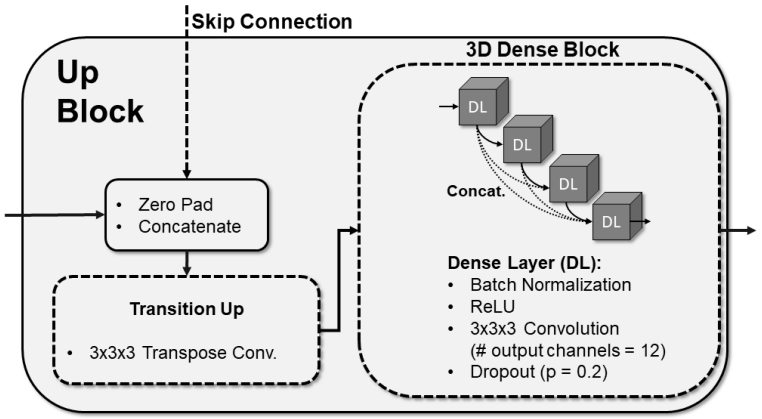


FIGURE 5.3: Diagram of the 3DFC-DenseNet architecture's Up Block. Adaptive Zero Padding activates in case of size mismatches and solves such problems arising from the use of strided transpose convolutions.

use of features from before the bottleneck when rendering the output. The spatial dimensions of the feature maps between corresponding TUs and TDs must be the same to be able to concatenate them.

Strided transpose convolutions can map multiple input sizes to the same output size [94]. Transition Ups are used to double feature map dimensions halved by the $2\times2\times2$ max poolings in Transition Downs meaning they can't produce feature maps of odd dimensions. The size mismatches that can arise due to these can prevent the architecture from using input sizes which produce odd sized feature maps after max pooling.

As a solution, an adaptive zero-padding is implemented at the connected TU and TD blocks to resolve size mismatches by zero-padding the feature map sizes at the TU blocks when needed. The use of ReLU non-linearities ensure that the minimum activation for each layer is zero meaning the zero-padding operation will not interfere with the range of values in the layer outputs. In conjunction with the fully convolutional nature of the architecture this allows any input size to be accepted and avoids having to manually change network parameters just to fit certain input sizes.

5.2 Objective Functions

5.2.1 Semantic Segmentation - IoU Loss

To solve the semantic segmentation task, the goal of the network is to maximize the Intersection over Union (IoU). The IoU defined between two sets A and B as:

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

measures the overlap between two sets. It assigns a score in the range of $[0, 1]$: 0 corresponding to no overlap, and 1 corresponding to a perfect overlap of the two sets.

In terms of the class probability predictions $S(\mathbf{X})$ and GT $\hat{S}(\mathbf{X})$ over for a class c:

$$IoU(S(\mathbf{X}), \hat{S}(\mathbf{X}))_c = \frac{S(\mathbf{X})_c \cap \hat{S}(\mathbf{X})_c}{S(\mathbf{X})_c \cup \hat{S}(\mathbf{X})_c}$$

where $\hat{S}(\mathbf{X})_c$ and $S(\mathbf{X})_c$ are the GT and predicted output channels corresponding to class $c \in [0, C)$.

It would be simple to define the semantic segmentation objective function for a class as $1 - IoU(S(\mathbf{X}), \hat{S}(\mathbf{X}))_c$. But to calculate if a (x, y, z) voxel belongs to class c, one must apply the arg max operation: if $\arg\max(S(\mathbf{X})[x, y, z]) = c$, then (x, y, z) belongs to c. Since the arg max operator is not differentiable, the IoU can not directly be optimized in this form.

Instead, a surrogate for the IoU, The Lovasz Softmax Loss [95] is the training objective L_{Lovasz}. It approximates the IoU and similarly to the above, it is calculated per-channel over semantic class probability maps. Although the implementation in [95] is for 2D data, this work extends the Lovász-Softmax Loss to handle three-dimensional volumetric data.

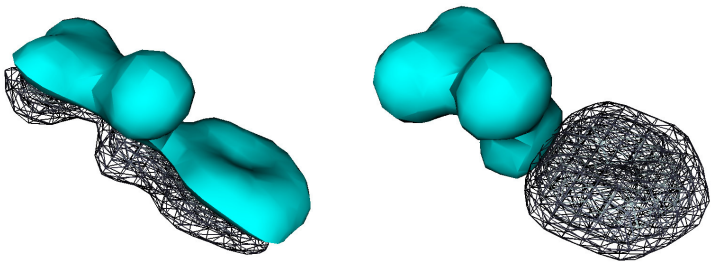


FIGURE 5.4: Residue fragment examples. If the portions rendered in wireframe are cropped, the fragments remaining may be insufficient for proper segmentation or classification

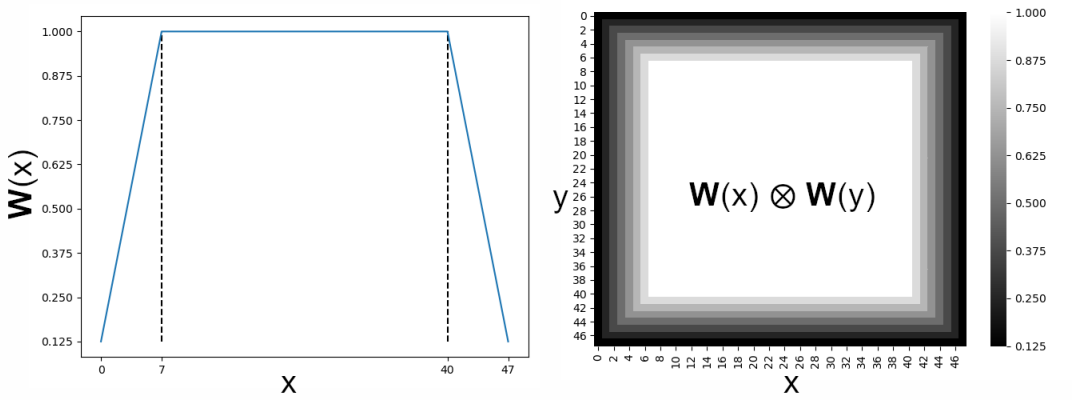


FIGURE 5.5: One- (left) and two-dimensional (right) positional weights **W** in a window of size 36×36×36 with $l_{frame} = 8$. By using positional weighting, the impact of mislabeling residue fragments around window edges is diminished.

5.2.2 Fragmented Residues & Positional Weighting

Cropping windows introduces the problem of fragmented residues close to window edges (Figure 4.7). The remaining fragments may not be sufficient (Figure 5.4) to properly identify a residue. To reduce the error caused by mislabeling residue fragments in the frame regions a positional weight $\mathbf{W} \in \Re^{W \times H \times D}$ (Figure 5.5) is applied to L_{Lovasz} to reduce the penalty around window borders where incomplete instances would appear.

Given a point x along a primary axis in a window of size X with frame size l_{frame}, the weighting component **W** is defined as

$$W(x) = \begin{cases} F(d(x, l_{frame})), & \text{if } x \in [x, l_{frame}] \\ F(d(x, X - l_{frame})), & \text{if } x \in [X - l_{frame}, X] \\ 1, & \text{otherwise.} \end{cases}$$

Where d is the L1 distance between two points. The weighted loss for point x is the product of the Lovász-Softmax Loss, L_{Lovasz}, and the loss weight assigned to that position: $L_{Lovasz}(x)W(x)$. In all experiments, F is a decreasing linear function $F(x) = (1 - x/l_{frame})$. The full spatial weight can be obtained as an outer product of the x, y and z components: $\mathbf{W} = W_x \otimes W_y \otimes W_z$.

The size of the frame region l_{frame} in voxels is resolution dependent: $l_{frame2\text{Å}} = 8$, $l_{frame3\text{Å}} = 6$, $l_{frame4\text{Å}} = 4$. Despite being different sizes in voxels, these all correspond to approx. 8 Å, which was selected so that the even the longest AA side chains of ARG and LYS would fit.

5.3 Experiments

5.3.1 Ablation Tests

Concatenations & Gradient Flow

The main motivator of basing the 3D FC-DenseNet architecture on the 2D FC-DenseNet was its use of concatenations within Dense Blocks (Figures 5.2, 5.3) to allow feature reuse and direct gradient flow to deeper layers [64]. An ablation test is conducted to confirm how well this property translates to a 3D case as well as for this specific problem.

In this ablation test, one configuration is trained with the Dense Block concatenations enabled, and another one without the concatenations. It is expected that due to the lack of concatenations enabling stronger gradient flow, the deeper layers (the bottleneck along with its proximate Up and Down

blocks) will see their gradient magnitude diminish. It might also affect training capability.

Each configuration is trained from scratch for 5 epochs. The metrics used to measure the effects of ablation on performance are the loss and IoU over validation set windows. To observe the effects on gradients, the gradient L2 norms normalized to have a maximum of 1 are compared.

Adaptive Zero Padding & Size Mismatches

When using the 4 Å data set's 36×36×36 voxel input size, odd feature map sizes are produced after two Transition Downs (from 18×18×18 to 9×9×9). These, due to the above described behavior of strided transpose convolutions are expected to cause a size mismatch when attempting to concatenate along the skip connections.

To confirm this and demonstrate the effects of the proposed adaptive zero padding, the 4 Å configuration is trained both with and without the adaptive zero paddings before each Transition Up block. It is expected that training will fail without the zero paddings, as it is not possible to concatenate tensors with mismatched shapes.

5.3.2 Inference & Performance Metrics

Inference is done in a patch-wise manner, by cropping ED maps into overlapping windows and stitching the per window network predictions onto the input ED map to form a global prediction (Figure 5.6).

Window sizes and l_{frame} are identical to those used during training, validation and fine-tuning: $48 \times 48 \times 48$ voxels at high and medium resolutions (2, 3 Å), and $36 \times 36 \times 36$ voxels at low resolutions (4 Å).

The stride between subsequent windows is the double of l_{frame}, ensuring that the fragmented instances of frame regions will re-appear in full in at

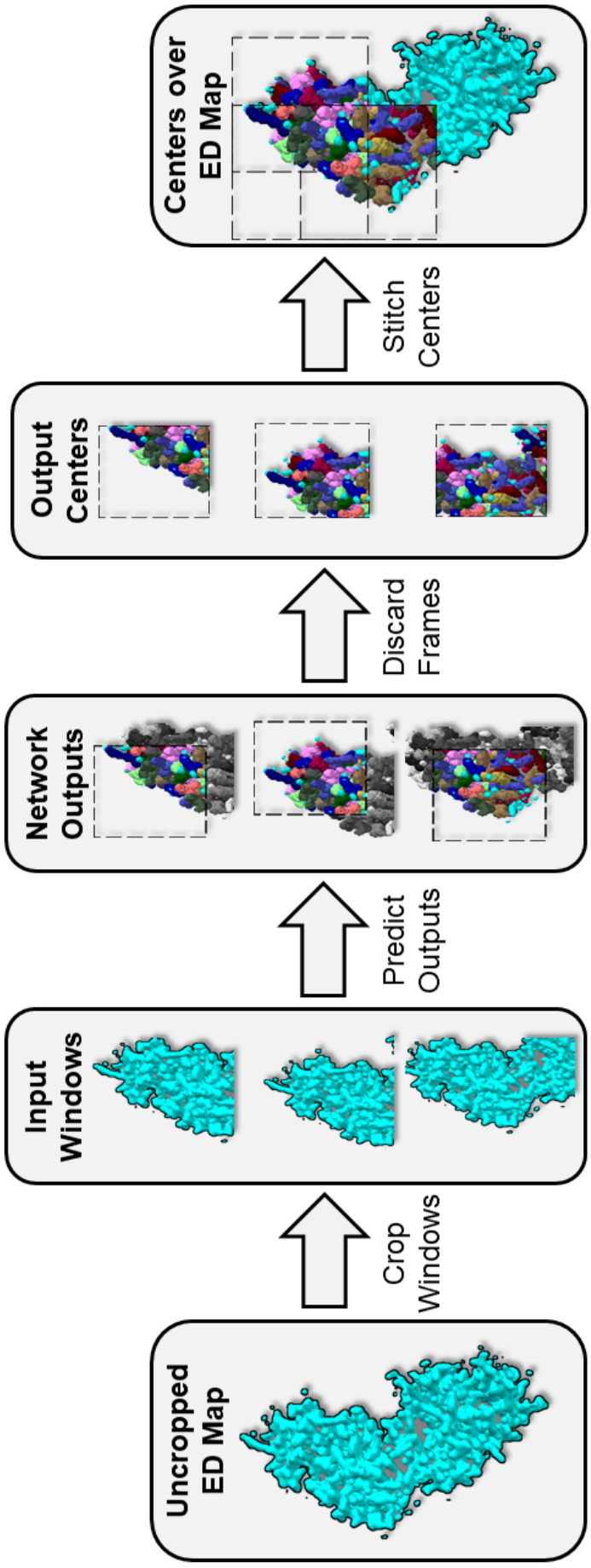


FIGURE 5.6: The 3D FC-DenseNet processes ED maps in batches of overlapping windows. To produce the global prediction for the entire ED map, the processed window centers (in color) are stitched onto the original ED map while the frame regions containing fragmented residues (in grayscale) are discarded.

least one subsequent window's center. In case the global prediction does not entirely cover the non-BG volumes, additional windows centered on the missing regions are cropped and processed as a failsafe.

Only the window centers are used to form the global prediction, as the frame regions may contain fragmented residues (Figure 5.4) the networks may be unable to segment properly. Network performance is measured over the global prediction after every window is processed, and not as a per window average.

Per-voxel semantic segmentation performance is measured by the mean IoU of the class probability predictions $S(\mathbf{X})$ and GT $\hat{S}(\mathbf{X})$ over all non-BG (where $c = 17$ is the BG/void) classes:

$$IoU(S(\mathbf{X}), \hat{S}(\mathbf{X})) = \frac{1}{C} \sum_{c=0}^{C} IoU(S(\mathbf{X}), \hat{S}(\mathbf{X}))_c$$

where the IoU of each c class is:

$$IoU(S(\mathbf{X}), \hat{S}(\mathbf{X}))_c = \frac{S(\mathbf{X})_c \cap \hat{S}(\mathbf{X})_c}{S(\mathbf{X})_c \cup \hat{S}(\mathbf{X})_c}$$

Additionally the True Positive (TP) rate, True Negative (TN) rate and F1-Score are also measured. A voxel (x, y, z) is considered a TP for class c if

$$c = \arg\max \hat{S}(\mathbf{X})[x, y, z] = \arg\max S(\mathbf{X})[x, y, z]$$

meaning that the most confident predicted and expected class for (x, y, z) are both c. If a voxel is a TP for class c it is considered to be a TN for every other class.

A voxel (x, y, z) is considered a False Positive (FP) for class c if

$$\arg\max \hat{S}(\mathbf{X})[x, y, z] \neq \arg\max S(\mathbf{X})[x, y, z] = c$$

meaning that the most confident predicted and expected class for (x, y, z) are not both c. In this case, the voxel is considered a False Negative (FN) for the $\arg\max \hat{S}(\mathbf{X})[x, y, z]$ class it should have been.

The F1-Score for class c is defined in terms of the above quantities:

$$F1_c = \frac{2TP_c}{2TP_c + FN_c + FP_c}$$

where, TP_c, FN_c, FP_c are the total TP, FN and FP for class c. Then, similarly to the IoU above, the average F1-Score is obtained as

$$F1 = \frac{1}{C} \sum_{c=0}^{C} F1_c$$

Residue class predictions are obtained by sampling the semantic output $S(\mathbf{X})$ in the union of each residue atoms' radii ($R_{2\,\text{Å}} = 3$, $R_{3\,\text{Å}} = 2$, $R_{4\,\text{Å}} = 1$). The reliance on the .pdb model file (where the atom positions are defined) is necessary as 3D FC-DenseNet is unaware of individual residue instances.

The predictions $S(\mathbf{X})[x, y, z]$ of each (x, y, z) voxel are averaged to obtain the residue's predicted class. This represents a majority voting scheme to render residue class predictions. This is more in line with what is ultimately the objective: to provide correct predictions for residue volumes and not necessarily every voxel in the ED map.

Per-residue performance is measured by Rank-1 (R1) and Rank-3 (R3) hit rates. For a Rank-1 hit, the most confident predicted class must be the same as the residue's expected class in the .pdb model file. For a Rank-3 hit, it is enough that the the expected class is among the top 3 most confident class predictions.

5.3.3 Training Configuration & Objective

3D FC-DenseNet is trained separately from scratch in three training and validation runs using the 2 Å, 3 Å, and 4 Å. This allows for direct observation and comparison of the effects of decreasing resolution on performance.

The 3DFC-DenseNet57 architecture used for the 2 Å and 3 Å samples contains 5 UBs and 5 DBs with a bottleneck of 15 dense layers for a total depth of 57 layers. To accomodate the reduced input size, the 3DFC-DenseNet49 configuration contains one less UB and DB (Table 5.1), with a total depth of 49 convolutional layers.

TABLE 5.1: Summary of the training configurations used for training the 2 Å and 3 Å models (3D FC-DenseNet 57) and the 4 Å model (3D FC-DenseNet 49). The latter is one block shallower on the up and down paths

3D FC-DenseNet 57	**3D FC-DenseNet 49**
3x3x3 Conv.	
Down Block x 5 (4 Dense Blocks ea.)	Down Block x 4 (4 Dense Blocks ea.)
Bottleneck (15 Dense Layers)	
Up Block x 5 (4 Dense Blocks ea.)	Up Block x 4 (4 Dense Blocks ea.)
1x1x1 Conv.	

Since 3D FC-DenseNet is a novel approach for a newly defined problem using a fresh data set, its training and validation is done in a 5-fold cross-validation setup. The data sets are split over proteins into five folds: each time, a different set of 1,398 ED maps are used for validation and the remaining 5592 ED maps for training. Five-fold cross-validation is necessary to show that the method is stable and that performance does not have biases towards individual training samples.

5.3.4 Finetuning & Experimental Data

After training and validation, the network models are fine-tuned using the experimental data set samples in a resolution range $\pm$ 0.5 Å around their

training resolutions. Samples under 2.5 Å resolution are processed by the 2 Å model, those in the 2.5–3.5 Å range by the 3 Å model, and those above 3.5 Å resolution by the 4 Å model.

50 protein ED maps in each resolution range are used for evaluating performance, the rest are used to finetune the models trained on the fixed resolution training sets. To demonstrate the effects of fine-tuning, performance on experimental data is evaluated both before and after the fine tuning process.

5.3.5 Comparison to Seqqy & ARP/wARP

3DFC-DenseNet's mean rank-1 and rank-3 AA labelling performance is compared to that of Seqqy [43]. Seqqy is not available stand-alone, but as a module in the ARP/wARP toolkit and runs in conjunction with other modules performing out-of-scope tasks such as loop closure [96] or refinement and modification of the input samples through REFMAC [97]. This makes direct comparison difficult and slightly favors Seqqy by allowing it to rely on additional information and modules.

By default, Seqqy relies on residue sequence information to assign AAs to the ED maps, while 3DFC-DenseNet does not. As there is an option to omit the sequence information when using Seqqy, its performance is measured both with and without the residue sequence provided.

ARP/wARP is designed for iterative model building. Seqqy is run with default settings for five cycles to build structural models for the ED maps in the experimental data set. The structural models used in the experimental data sets serve as the GT here as well. In Seqqy-built models, each predicted residue is paired to the GT residue with the highest IoU intersecting bounding box.

5.4 Results & Discussion

5.4.1 Ablation Test Results

Concatenations & Gradient Flow

With concatenations enabled, the improved gradient flow is observable via the norm of the gradients, especially at deeper layers and the bottleneck (Figure 5.7). The normalized gradient magnitudes in the bottleneck layers are an order of magnitude higher with concatenations enabled.

Performance is also affected positively by using concatenations in Dense Blocks with an average increase of 49.28% (Table 5.2) across all resolutions. The measured losses are higher and converge slower without concatenations (Figure 5.7). These results confirm that the use of concatenations in Dense Blocks are significant contributors to the performance of 3D FC-DenseNet.

TABLE 5.2: Ablation test results: semantic segmentation IoU measured on the 3Å validation set fold 1, after 5 epochs of training with (Normal) and without (No Cat) Dense Block concatenations

Resolution	**2Å**	**3Å**	**4Å**
Normal IoU	0.7014	0.5365	0.2534
No Cat IoU	0.4750	0.3303	0.1839
% Difference	47.67%	62.43%	37.75%

Adaptive Zero Padding & Size Mismatches

In line with expectations, the odd $9\times9\times9$ feature map size after the second Transition Down when using the 4 Å data set's $36\times36\times36$ voxel windows resulted in a failure case without zero paddings. No odd feature map shapes are created with the increased $48\times48\times48$ window sizes of the 2 Å and 3 Å setups.

In these cases the zero paddings are inactive even when enabled, because no mismatches occur. Zero paddings before skip connection concatenations

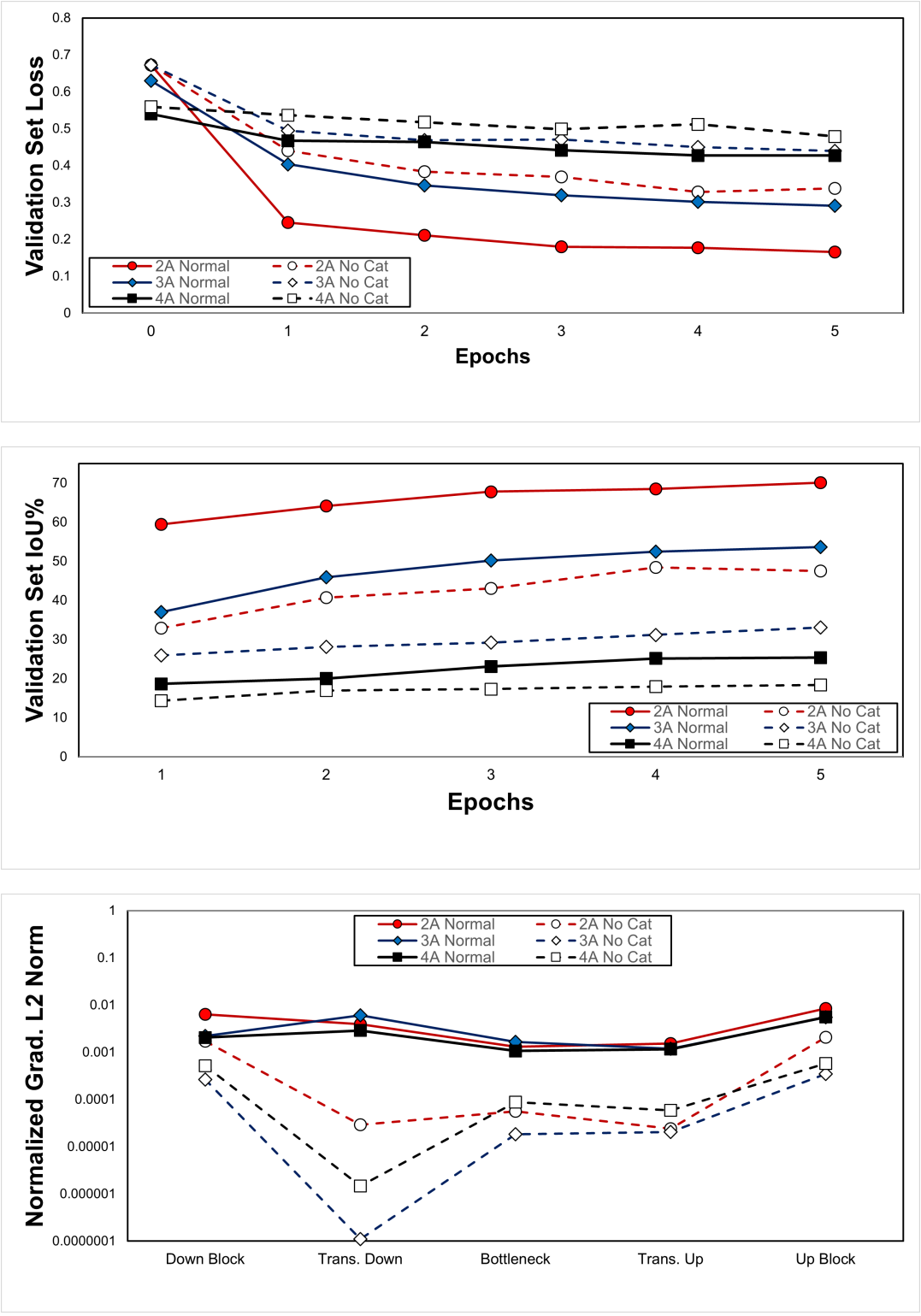


FIGURE 5.7: Ablation test results, comparing runs Runs with concatenations enabled ("Normal" runs) and disabled ("No Cat" runs). "Normal" runs optimize faster (Top) and outperform (Middle) "No Cat" runs. Normalized gradient norms (Bottom) show that more of the gradient is conserved in deeper layers with DB concatenations in "Normal" runs. Note the logarithmic scale on the bottom chart.

offer a general solution so that architecture parameters do not have to be adjusted by hand on a layer-by-layer basis to fit input sizes. This allows the implementation of problem-specific considerations such as reducing window sizes to keep the FOV consistent even at lower resolutions.

5.4.2 Training & Validation Results

The results of the 5-fold cross-validation in Table 5.3 show that the proposed method is stable across all folds of all resolutions and is a viable approach to segmenting ED maps according to AA residues.

The baseline Rank-1 Hit Rate of a random classifier for an 18-class classification problem is approximately 0.0556 (naïvely, disregarding data set imbalance). 3DFC-DenseNet achieves a mean Rank-1 Hit Rate of 0.5054 at 4 Å, a significant increase over the baseline even at the lowest resolution. The Rank-1 Hit Rate is 0.7983 at 3 Å and 0.8912 at 2 Å.

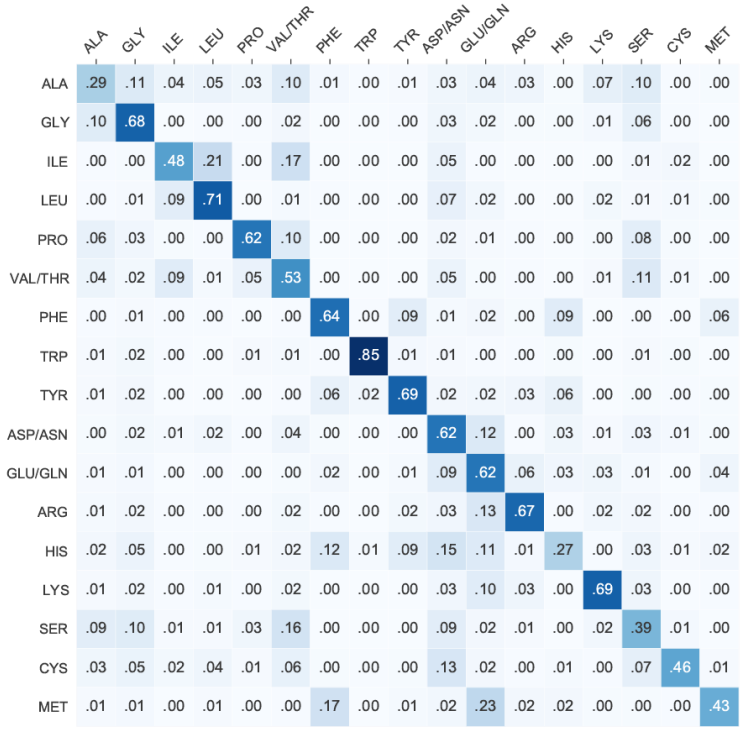
The difference between the Rank-1 and Rank-3 AA hit rates increases as the sample resolution decreases: 3.68% at 2 Å, 7.22% at 3 Å and 27.36% and 4 Å. The correct prediction becomes less likely to be the most confident towards lower resolutions. This hints at increasing confusion between residues as side chains become harder to distinguish.

The confusion matrix in Figure 5.8(a) for the Rank-1 AA hit rates at 4 Å supports this. Residues such as ALA and GLY or ILE and LEU become near-indistinguishable at 4 Å, especially without information about the AA sequence. The observed confusion is in line with experimental observations. A modified similarity matrix from the software O [98] in Figure 5.8(b) confirms that the classes confused by the network at 4 Å are ones with high similarity.

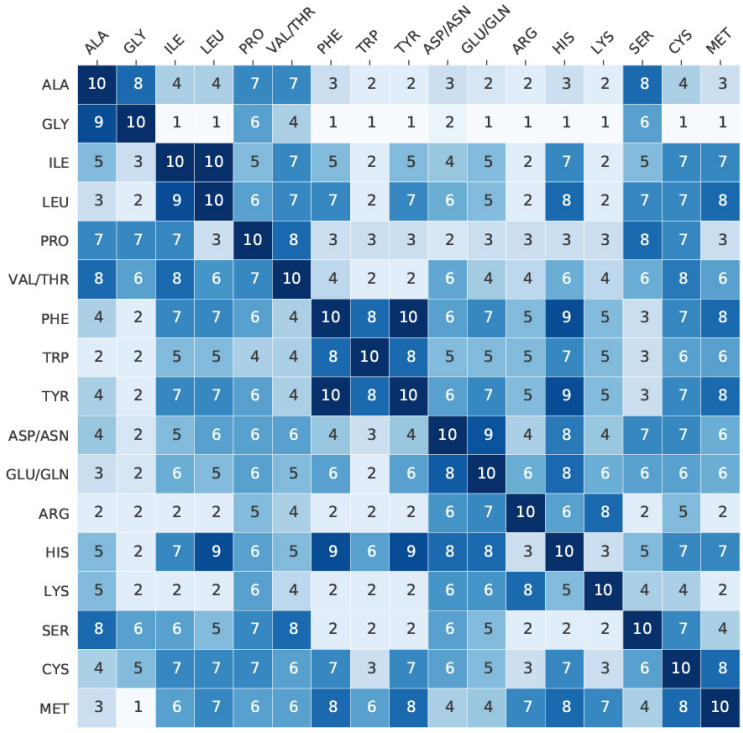
The detrimental effect of lower resolutions on side chain observability is demonstrated by the results, especially in the case of the voxel-wise metrics.

TABLE 5.3: Five-fold cross validation results: the best performing fold based on Rank-1 Hitrate is marked in bold

2Å	Fold 1	Fold 2	Fold 3	**Fold 4**	Fold 5
IoU	0.8231	0.8346	0.8256	**0.8343**	0.8229
True Pos.	0.8415	0.8402	0.8346	**0.8463**	0.8361
True Neg.	0.9998	0.9997	0.9997	**0.9998**	0.9997
F1-Score	0.8845	0.8858	0.8780	**0.8892**	0.8835
Rank-1 Hitrate	0.8814	0.8867	0.8783	**0.8911**	0.8783
Rank-3 Hitrate	0.9154	0.9182	0.9137	**0.9191**	0.9123
3Å	Fold 1	Fold 2	Fold 3	**Fold 4**	Fold 5
IoU	0.6752	0.6894	0.6663	**0.6810**	0.6788
True Pos.	0.7231	0.7331	0.7271	**0.7311**	0.7185
True Neg.	0.9996	0.9996	0.9995	**0.9996**	0.9996
F1-Score	0.7767	0.7884	0.7775	**0.7828**	0.7787
Rank-1 Hitrate	0.7906	0.7939	0.7885	**0.7983**	0.7881
Rank-3 Hitrate	0.8479	0.8498	0.8471	**0.8541**	0.8462
4Å	**Fold 1**	Fold 2	Fold 3	Fold 4	Fold 5
IoU	**0.3316**	0.3149	0.3279	0.3052	0.3316
True Pos.	**0.4694**	0.4615	0.4679	0.4376	0.4652
True Neg.	**0.9979**	0.9979	0.9979	0.9978	0.9978
F1-Score	**0.5224**	0.5311	0.5324	0.4978	0.5308
Rank-1 Hitrate	**0.5171**	0.5004	0.5101	0.4865	0.5132
Rank-3 Hitrate	**0.6563**	0.6360	0.6487	0.6302	0.6476



(a)



(b)

FIGURE 5.8: Confusion matrix of the results measured over the 4 Å validation sets (a) and the modified similarity matrix (b), *.slider_matrix* [99] from O [98] (lowest similarity score is 1, highest is 10). Residues with high similarity tend to be confused by 3D FC-DenseNet.

The majority voting scheme for per-residue metrics alleviates this: comparing the voxel-wise mean IoU to rank-1 hit rates shows increases of 6.65% at 2 Å, 16.76% at 3 Å and most significantly 56.64% at 4 Å.

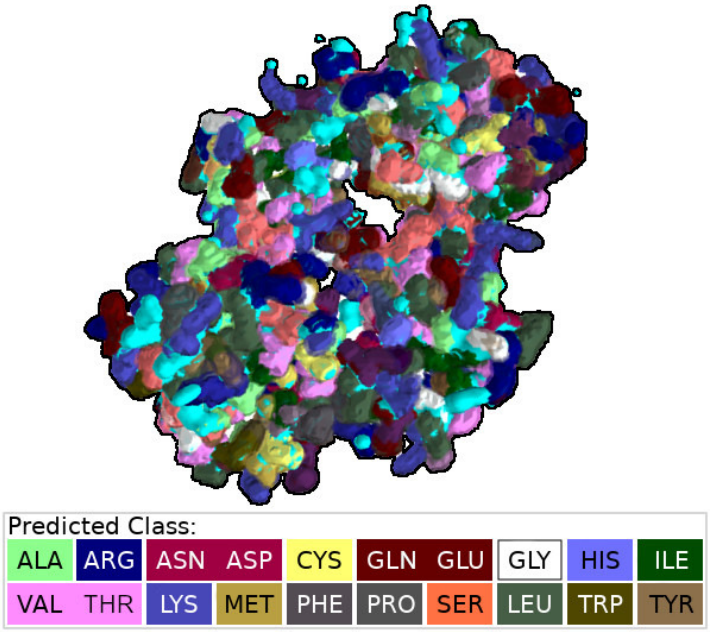


FIGURE 5.9: Global class predictions over a 3 Å ED map.

Foreground-background separation is handled well. This is shown quantitatively by the high TN rates and qualitatively as correct predictions 'snap' to non-BG voxels (Figures 5.9, 5.10).

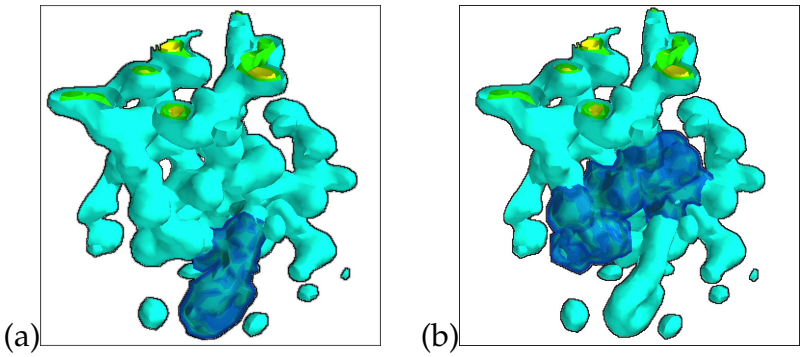


FIGURE 5.10: Voxels predicted as PHE (a) and CYS (b) in a 2 Å resolution window. The network outputs 'snap' closely to the input densities.

5.4.3 Experimental Samples

Without fine-tuning, the performance of 3D FC-DenseNet on experimentally observed ED maps is significantly worse than with the fixed resolution samples (Table 5.4). This may be partly due to the varying sample resolutions and additional artifacts due to this being real-life data. Fine-tuning increases performance for both Rank-1 and Rank-3 hit rates. The fine-tuned results are comparable to the performance of the methods in use by model building tool kits.

TABLE 5.4: Comparison of mean validation set and experimental set hit rates, without and with finetuning (FT)

Resolution	Val. R1	Val. R3	Non-FT R1	Non-FT R3	FT R1	FT R3
<2.5 Å	0.8912	0.9192	0.5397	0.6552	0.7116	0.8119
2.5–3.5 Å	0.7983	0.8541	0.3789	0.5327	0.6045	0.7495
>3.5 Å	0.5172	0.6564	0.2072	0.3500	0.3328	0.5122

3D FC-DenseNet matches the performance of *seqqy* [43] and ARP/wARP [32] with fine-tuned rank-1 hit percentages of approx. 72.07% vs. 71.16% at high resolutions. At medium resolutions 3D FC-DenseNet outperforms Seqqy's 41.29% accuracy with 60.47%. Most importantly, it remains functional at low resolutions, while Seqqy fails even with sequence information present. Table 5.5 shows that while the option is present in ARP/wARP not to use sequence information, it's not able to perform well without it.

It is important to note that 3D FC-DenseNet is a set of 3 per-resolution specialist networks, each being able to target a specific resolution range (low,

TABLE 5.5: Comparison of 3D FC-DenseNet's finetuned rank-1 (FT R1) and rank-3 (FT R3) mean performance with Seqqy (ARP/wARP) after five iterations, with (+seq) and without (-seq) sequence information

Resolution	FT R1	FT R3	Seqqy, -seq	Seqqy, +seq
<2.5 Å	0.7116	0.8119	0.0628	0.7207
2.5–3.5 Å	0.6047	0.7495	0.0204	0.4129
>3.5 Å	0.3328	0.5122	0.0068	0.0143

medium and high), while Seqqy is a set of per-residue SVM classifiers having to deal with all resolution ranges. According to [80], Seqqy was trained and tested with a bias towards higher resolution ED maps, its training set consisted of 1000 ED maps over 2.5 Å resolution and its test set contained approx. 400 ED maps in the 2–3 Å and an additional approx. 400 in the 3–4 Å range. Seqqy also heavily relies on AA chain being mapped to the ED map (main chain tracing) before classifying side chains. If pre-tracing the main chain fails (or the sequence information is not provided), Seqqy fails even at high resolutions since it attempts to classify the residues in the locations specified by the chain tracing process.

The increased performance of 3D FC-DenseNet is due to a combination of factors: the training and fine-tuning data set itself covers a wider range of resolutions and better encompasses the task, specialist networks are deployed for each resolution range, and by not relying on the AA sequence, the networks do not need the main chain pre-traced to assign AA labels, thus bypassing the single point of failure the reliance on the residue sequence would represent.

Hit rates are lower for samples with higher R-Free values (Figure 5.11). Such samples are more common at lower resolutions, leading to a doubly challenging situation of low resolution and low sample quality.

The average time taken for a 2 Å sample (the largest ones in volume) is 29 seconds using an Nvidia GTX 1080Ti GPU. A smaller protein, 6E0T, takes about 22 seconds to process by 3DFC-DenseNet while a single cycle with ARP/wARP takes over 2 minutes and over 7 minutes for 5 cycles. As a more extreme example, the protein 2Y1V takes up to 15 minutes for 1 cycle with ARP/wARP and almost 70 minutes for five. 3D FC-DenseNet segments it in 86 seconds.

At the time of writing, the Nvidia GTX 1080Ti GPU used for measuring performance is several hardware generations old, with Nvidia's 40 series

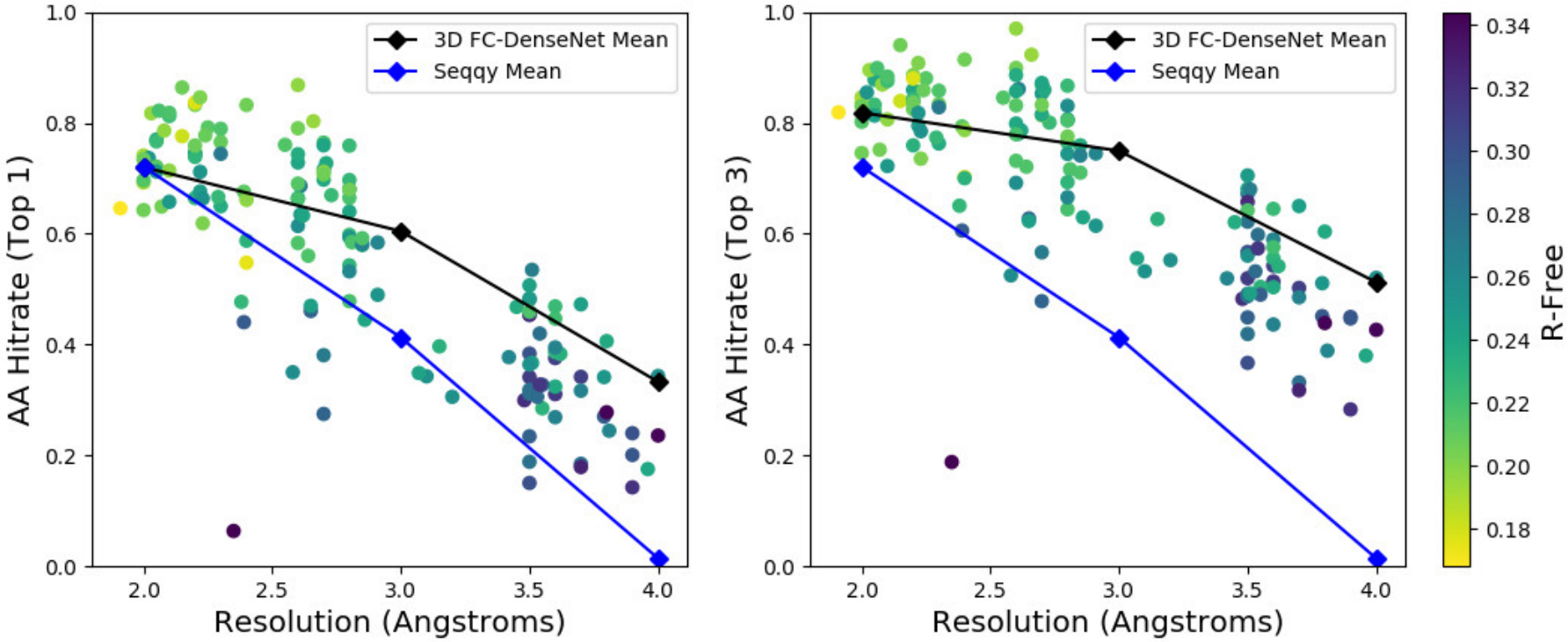


FIGURE 5.11: Peformance comparison over experimental samples. The scattered points correspond to the Rank-1 and Rank-3 performance of finetuned networks. The line points in black represent average performance of 3D FC-DenseNet (Rank-1 & Rank-3), blue line points show Seqqy's AA labelling performance at high, mid and low resolutions (Rank-1 only, Seqqy can't provide Rank-n).

GPUs having just released. This underscores how accessible the performance boosts represented by NN based methods are to the end user: no cutting edge hardware is required for their use. But even on a CPU, with no optimizations such as multithreading being implemented, 3DFC-DenseNet takes 196 seconds for 6E0T and 715 seconds for 2Y1V. While significantly slower than the GPU version, this shows that the decreased computational time is not only due to the use of GPUs.

Part of the reason why 3D FC-DenseNet is so fast compared to Seqqy is how their processing time scales. The processing time for Seqqy scales linearly with the length of the residue chain, while the processing time for 3D FC-DenseNet scales linearly with the volume of the protein's ED map.

Since a large number of residues may be found in an input volume and NNs perform batch calculations in parallel as fast as a single input, it can be seen that 3D FC-DenseNet represents a significant speedup even on readily available consumer hardware.

3D FC-DenseNet represents an important step towards NN-based automated model building. It is proof that the task can be treated as an image segmentation problem, and that the established data sets are adequate in format and breadth.

Chapter 6

Methods – Simultaneous Semantic & Instance Segmentation

While 3D FC-DenseNet shows that it is possible to tackle the task with computer vision and image segmentation methods, it is a rather rudimentary network with several shortcomings and possible improvements. For a network to be useful in practice, it should also be able to locate individual AA residues on its own, which 3D FC-DenseNet could not do.

3D FC-DenseNet requires different configurations for different window sizes, making it somewhat inconvenient to deploy. Furthermore, the NN is not aware of individual AA instances and only assigns the most probable AA label to each voxel.

An upgraded network architecture should follow up on the strength of 3D FC-DenseNet (improved gradient flow, compatibility with varying input sizes), improve on its shortcomings (necessity of different configurations, inability to detect individual residues) and also offer improved performance. The upgrade to the 3D FC-DenseNet was designed with these in mind.

6.1 MT-StackNet Network Architecture

The proposed upgrade to 3D-FCDenseNet is a multi-task stacked residual network architecture referred to as MT-StackNet (Figure 6.1). The MT-StackNet

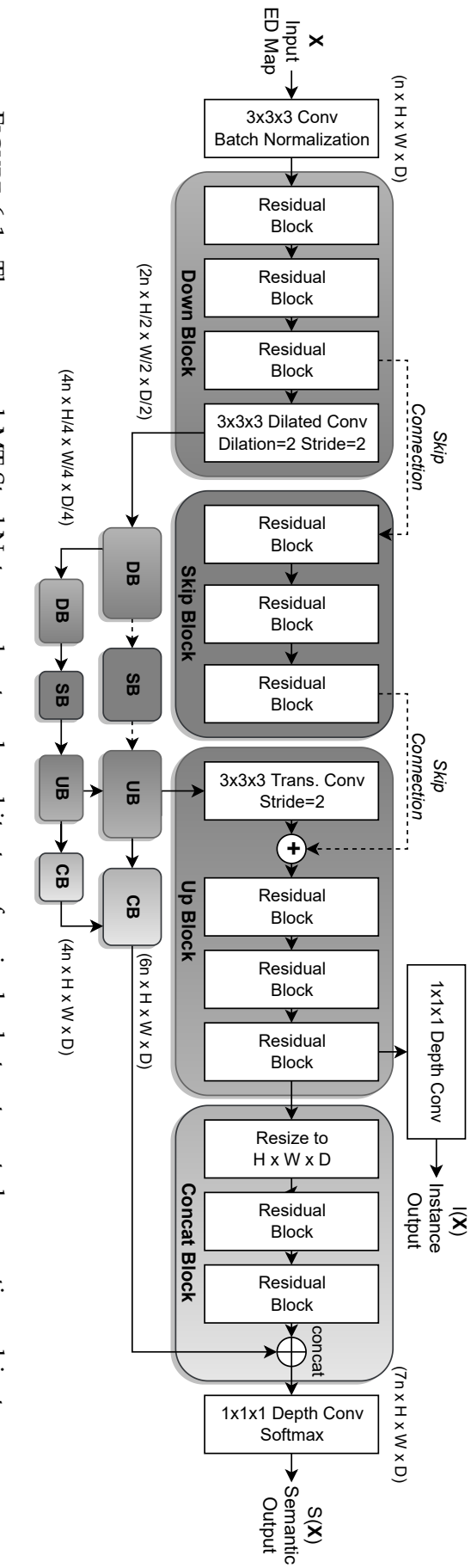


FIGURE 6.1: The proposed MT-StackNet neural network architecture for single shot automated semantic and instance segmentation of protein ED maps.

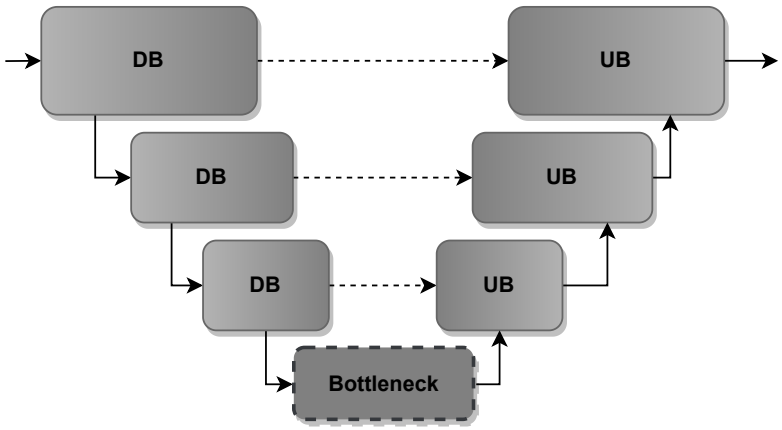


FIGURE 6.2: The backbone of the architecture is a residual U-Net. The final architecture has had the bottleneck removed and its layers re-distributed along the skip connections as Skip Blocks. Additionally, Concatenation Blocks have also been added after Up Blocks to perform the semantic segmentation task.

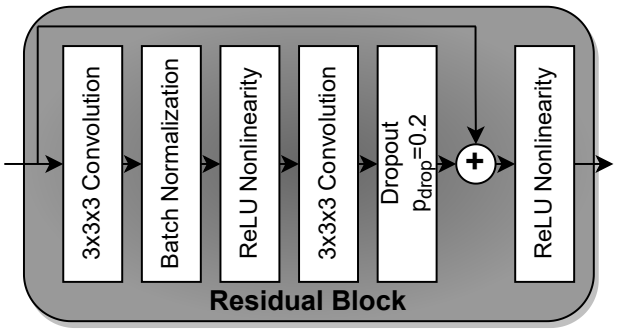


FIGURE 6.3: The internal structure of the ResNet [61] style Residual Blocks adapted to handle 3D volumetric data. Residual blocks handle the bulk of the processing in MT-StackNet.

can be considered an evolution of a residual 3D U-Net backbone (Figure 6.2). MT-StackNet uses residual connections instead of 3D FC-DenseNet's concatenations which retains the improved gradient flow but also allows additional processing of the residual (Figure 6.3).

It performs both semantic and instance segmentation using shared feature maps and is capable of outputting both $S(\mathbf{X})$ and $I(\mathbf{X})$ in a single forward pass (Figure 6.4). The most impactful upgrade over 3D FC-DenseNet is thanks to the added instance segmentation capability: MT-StackNet can locate and classify residues automatically.

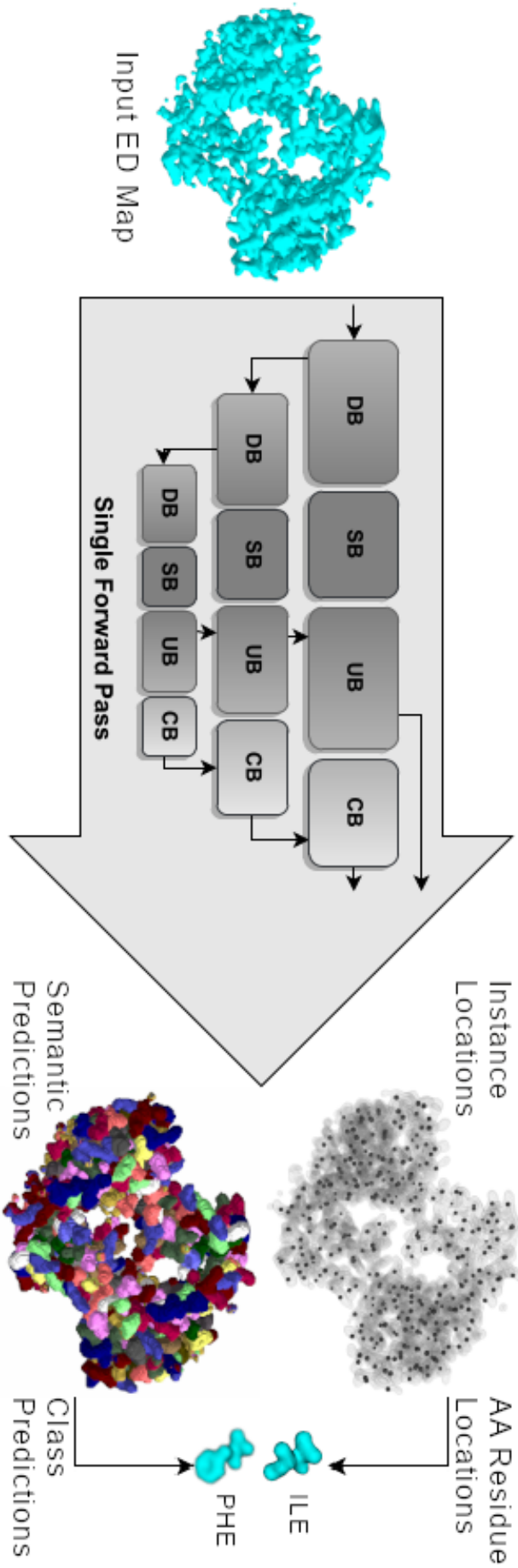


FIGURE 6.4: The MT-StackNet architecture predicts all residue locations and a global semantic class prediction in a single forward pass.

The architecture consists of four main building blocks: Down Blocks (DB), Up Blocks (UB), Skip Blocks (SB) and Concatenation Blocks (CB). Each block contains multiple residual modules [61] in addition to layers specific to the given block's task.

The residual modules (Figure 6.3) contain two $3 \times 3 \times 3$ convolutions, each followed by a batch normalization and a ReLU nonlinearity. The input of the module is added to the output of the second batch normalization layer. A dropout layer with a slight drop probability of $p_{drop} = 0.2$ is included after the second convolutional layer to reduce overfitting by preventing co-adaptation [100] while still being less likely to disturb the newly learnt features in the Residual Blocks.

DBs contain three residual blocks followed by a strided dilated convolution layer. Instead of max poolings, dilated convolutions (with a dilation and stride of 2) downsample the feature map [101]. Each DB output serves as the input of another DB one stack below.

DBs double the feature map channels and halve their spatial dimensions: $n2^i \times H/2^i \times W/2^i \times D/2^i$, where $i \in \mathbb{N}$ is the current stack depth (e.g. moving from stack 0 to stack 1 results in a feature map size of $(2n \times H/2 \times W/2 \times D/2)$).

UBs consist of a transpose convolution layer doubling feature map and three residual blocks. Their task is to restore feature maps to their size from one stack above. Since its usefulness was already demonstrated with 3D FC-DenseNet, an adaptive zero-padding is included at transpose convolutions to solve size mismatches arising from odd-sized feature maps.

SBs contain three residual layers. The input of SBs is the final residual block's output from the DB on the corresponding stack level. SB outputs are summed with the corresponding UB's transpose convolution output (the exception being the bottom-most stack where there are no features from a lower stack to add).

CBs are used to collect each stack level's feature maps and concatenate them. They consist of two residual blocks after resizing the feature maps to the first level's spatial dimensions to be able to concatenate them (with the exception again being the bottom level without a lower feature map to concatenate).

The input electron density $\mathbf{X}$ is a single channel, 3D volumetric array with dimensions of $W \times H \times D$. A $3 \times 3 \times 3$ 3D convolution followed by a batch normalization in the top level of the stack creates the initial $W \times H \times D \times n$ dimension feature map from the input.

The semantic class probability map output, $S(X)$, is generated using a $1 \times 1 \times 1$ depth convolution layer after concatenating every stack's feature maps with CBs. The instance segmentation offset vector field, $I(\mathbf{X})$, is the output of another depth convolution after the first stack's UB.

MT-StackNet produces both the semantic and the instance segmentation output from the same, shared feature maps. By aggregating the features in the CBs, the semantic segmentation task supervises every level of the feature stack while the instance segmentation task simultaneously provides class-agnostic supervision.

Due to the relatively small input sizes, the bottleneck is removed (Figure 6.2) and its layers re-distributed along the skip connections, forming SBs (Figure 6.1). This is motivated by not wanting to waste learnable parameters for feature maps with drastically reduced spatial dimensions.

6.2 Objective Functions

6.2.1 Instance Segmentation – Dice Loss

In the case of the instance segmentation task, the network is tasked to maximize the voxel-wise Dice-coefficient of corresponding predicted g_k and GT

$\hat{g}_k$ instances. The Dice-coefficient of two sets A and B is generally defined as:

$$Dice(A, B) = \frac{2(A \cap B)}{|A| + |B|}$$

Similarly to the IoU, a different formulation is necessary so that it becomes suitable for use in an objective function.

First, an embedding $\mathbf{E}$ is created from $I(\mathbf{X})$ which embeds voxels to the absolute position where their offset vector points:

$$\mathbf{E}[x, y, z] = I[x, y, z] + (x, y, z)$$

The unique $(\hat{u}, \hat{v}, \hat{w})$ co-ordinates in the GT embedding $\hat{\mathbf{E}}$ correspond to centroids of GT instances $\hat{g}$. Thus, $\hat{g}_k \in \hat{g}$ will be groups of points which embed to these unique co-ordinates in $\hat{I}(\mathbf{X})$:

$$\hat{g}_k = \{(x_i, y_i, z_i) | \hat{\mathbf{E}}[x_i, y_i, z_i] = (\hat{u_k}, \hat{v_k}, \hat{w_k})\}$$

Unfortunately, the values of the output embedding $\mathbf{E}$ do not have a one-to-one correspondence to the predicted instance centroids. Ideally, one could define $g_k = \{(x_i, y_i, z_i) | \mathbf{E}[x_i, y_i, z_i] = (\hat{u_k}, \hat{v_k}, \hat{w_k})\}$, but due to the imperfect outputs of the network this is not feasible.

Instead, for training purposes, a relaxation radius δ is defined which allows us to define predicted instances in terms of GT centroids:

$$g_k = \{(x_i, y_i, z_i) | \mathbf{E}[x_i, y_i, z_i] = (\hat{u_k} \pm \delta, \hat{v_k} \pm \delta, \hat{w_k} \pm \delta)\}$$

This means that the networks are allowed to have minor errors in the $I(\mathbf{X})$ offsets and as long as the offsets point in a δ-sized neighbourhood of an expected $\hat{g}_k$ instance centroid $(\hat{u_k}, \hat{v_k}, \hat{w_k})$, they are grouped together and represent the location of the predicted instance g_k.

Additionally, a membership probability for the voxels of predicted instances proportional to the distance from the centroid is calculated:

$$p((x_i, y_i, z_i) \in g_k) \propto 1/||\mathbf{E}[x_i, y_i, z_i] - (\hat{u_k}, \hat{v_k}, \hat{w_k})||_2$$

For each g_k instance, a membership probability mask $\mathbf{M_k} \in \Re^{W \times H \times D}$ is created:

$$\mathbf{M_k}[x, y, z] = 1 - tanh(ReLU(||\mathbf{E}[x_i, y_i, z_i] - (\hat{u_k}, \hat{v_k}, \hat{w_k})||_2 - \delta))$$

The GT membership mask $\hat{\mathbf{M}}_\mathbf{k}$ is binary:

$$\hat{\mathbf{M}}_\mathbf{k}[x, y, z] = \begin{cases} 1 & \text{if } (x, y, z) \in \hat{g}_k \\ 0 & \text{if } (x, y, z) \notin \hat{g}_k \end{cases}$$

The Dice coefficient for the predicted instance g_k and its corresponding GT instance $\hat{g}_k$ is calculated in a grayscale manner between $\mathbf{M_k}$ and $\mathbf{M'_k}$:

$$Dice(\mathbf{M_k}, \hat{\mathbf{M}}_\mathbf{k}) = \frac{2\mathbf{M_k}\hat{\mathbf{M}}_\mathbf{k}}{\sum \mathbf{M_k} + \sum \hat{\mathbf{M}}_\mathbf{k}}$$

Here, since the GT membership mask $\hat{\mathbf{M}}_\mathbf{k}$ is binary, the product $\mathbf{M_k}\hat{\mathbf{M}}_\mathbf{k}$ is the pixel-wise intersection and each mask's respective sum is analogous to its number of elements.

The reasons for using Dice instead of IoU are mostly technical: the Lovasz Softmax Loss does not scale well with the number of instances (in the hundreds or thousands per protein). The Dice-coefficient is an analogous metric which is differentiable out of the box, thus lending itself well for use when optimizing neural networks.

Each voxel is either part of an AA instance or the BG, resulting in the $\hat{\mathbf{M}}_\mathbf{k}$ GT masks fully partitioning the input density. Larger δ values correspond to more lenient assignments but would reduce L_{Dice} by g_k being larger than $\hat{g}_k$ and including voxels of other instances.

Although this already introduces an inherent regularization effect, the δ relaxation parameter is optimized directly for faster convergence. By providing δ to the optimizer, its value is also optimized along network weight updates when reducing the Dice loss.

6.2.2 Instance Segmentation – MSE Loss

Initially, the $I(\mathbf{X})$ offsets are expected to be random. To help with early training, an auxiliary Mean Squared Error (MSE) is calculated:

$$MSE(I(\mathbf{X}), \hat{I}(\mathbf{X})) = \frac{1}{n}\sum_{n}(I(\mathbf{X}_n) - \hat{I}(\mathbf{X})_n)^2$$

The MSE Loss, L_{MSE} is considered over the non-BG voxels of $I(\mathbf{X})$:

$$L_{MSE}(I(\mathbf{X}), \hat{I}(\mathbf{X})) = \begin{cases} MSE(I(\mathbf{X}), \hat{I}(\mathbf{X})) & \text{if } X[x,y,z] > 0 \\ 0 & \text{else} \end{cases}$$

L_{MSE} helps orient the $I(\mathbf{X})$ offsets in the early phases of training and guides the network towards forming the g_k instances used by L_{Dice}.

6.3 Combined Loss Function

For semantic segmentation, the goal of the network is still to maximize the Intersection over Union (IoU) between the network's semantic output $S(\mathbf{X})$ and the expected semantic output $\hat{S}(\mathbf{X})$. The Lovasz Softmax Loss [95], L_{Lovasz}, is calculated per-channel over the $S(\mathbf{X})_c$ semantic class probability maps.

This is the same objective function as the one used when training 3D FC-DenseNet, allowing for direct comparison. Similarly, the $\mathbf{W}$ positional weighting is applied with same parameters: $l_{frame2\text{Å}} = 8$, $l_{frame3\text{Å}} = 6$, $l_{frame4\text{Å}} = 4$ and $F(x) = (1 - 1/8x)$.

L_{MSE} is also not expected to perform well in the frame regions. The same $\mathbf{W}$ positional weighting described above is applied to it as well. L_{Dice} does

not have **W** applied to it. This is because L_{Dice} is derived from L_{MSE}, thus applying **W** to L_{Dice} would mean applying **W** twice to the same task.

The final, combined objective function is the sum of the above terms:

$$L = \mathbf{W}L_{Lovasz} + \mathbf{W}L_{MSE} + L_{Dice} + ||\delta||_2$$

6.4 Experiments

6.4.1 Network Configuration

For 3D FC-DenseNet, different network configurations were necessary for high-to-medium and low sample resolutions. In an improvement over the FC-DenseNet57 and FC-DenseNet49 architectures, MT-StackNet handles all resolutions and input sizes without needing different configurations depending on sample resolution.

As shown in Figure 6.1, MT-StackNet contains 3 stack levels. 3 residual blocks are in each DB, SB and UB. CBs have 2 residual blocks. The margin parameter is initialized to be 2 voxels at all resolutions. This corresponds to the physical resolution of the data for all samples since they were upsampled using Phenix when creating the data sets.

The networks are trained from scratch for 10 epochs (by which point the losses of all configurations converge) using the Adam optimizer with a batch size of 50 windows.

The initial learning rate is 0.001, reduced epoch-by-epoch with PyTorch's Cosine Annealing scheduler by a factor of 0.001 until a final learning rate value of 0.00001. These hyperparameters lead to a stable and smooth optimization of the objective functions where network weights do not explode.

6.4.2 Ablation Tests

Ablation tests measure the performance impact of the Bottleneck's removal from the 3D ResU-Net Backbone and the addition of SBs along skip connections. There are 4 possible combinations of configurations: with SBs and Bottleneck present, either the Bottleneck or the SBs present, and lastly both removed. The configuration with the Bottleneck present but no SBs corresponds to the baseline 3D ResUNet performance.

Feature maps have significantly reduced in sizes by the time they reach the Bottleneck. In the 2 and 3 Å cases the Bottleneck would receive a $6 \times 6 \times 6$ sized feature map, whilst in the 4 Å case a $4 \times 4 \times 4$ one. The learnable parameters are wasted on such small feature maps, thus it is expected to see no significant decrease in performance by removing the bottleneck.

Conversely, it would be beneficial to reallocate the Bottleneck's layers to higher stack levels, where larger feature map sizes may contain information that could still be extracted. The SBs represent the Bottleneck distributed along the skip connections, where the feature map sizes are larger. An increase in performance is expected from their inclusion.

The average, 3 Å case over the validation sets after one epoch of training is considered when measuring the performance impact of each configuration. The hyperparameters of the ablation tests are identical to those of the actual training sessions.

6.4.3 Finetuning & Experimental Samples

The performance of MT-StackNet is also measured on real-life, experimentally acquired samples.

As was the case with 3D FC-DenseNet, the models pre-trained over the fixed resolution samples are fine-tuned with resolution-appropriate ED maps. The high-resolution model pre-trained over the 2 Å data set handles samples

over 2.5 Å resolution. The medium resolution model (pre-trained at 3 Å resolution) handles the 2.5–3.5 Å range, and the ED maps below 3.5 Å are handled by the low-resolution model (pre-trained at 4 Å resolution).

50 samples in each resolution range are used as an experimental test set for acquiring the final performance measurements.

6.4.4 Inference & Performance Metrics

Global Predictions

The MT-StackNet inference pipeline follows the same steps as the one with 3D FC-DenseNet (Figure 6.5). To reiterate: input ED maps are cropped into windows in sequence and input to MT-StackNet to obtain window outputs. A global prediction covering the ED map is acquired by stitching the window outputs together. Performance metrics are calculated for the global predictions covering the entire ED map and not on a per-window basis.

Window sizes are identical to those used during training and validation. The stride between subsequent windows is the double of l_{frame} (the size of a window's frame region), ensuring that the fragmented instances of frame regions will re-appear in full in at least one subsequent window's center.

The process is identical for both the semantic output $S(\mathbf{X})$ and offset vector output $I(\mathbf{X})$, with the only difference being that $I(\mathbf{X})$ undergoes further processing to obtain instance predictions.

Instance Predictions & Matching

As opposed to training and validation, inference over the test sets assumes the real-life scenario of there being no GT to match predicted instances to. Instance localization in this case is done by using K-Means clustering to find the $g_k | k \in [0, K)$ instances in the embedding $\mathbf{E}$.

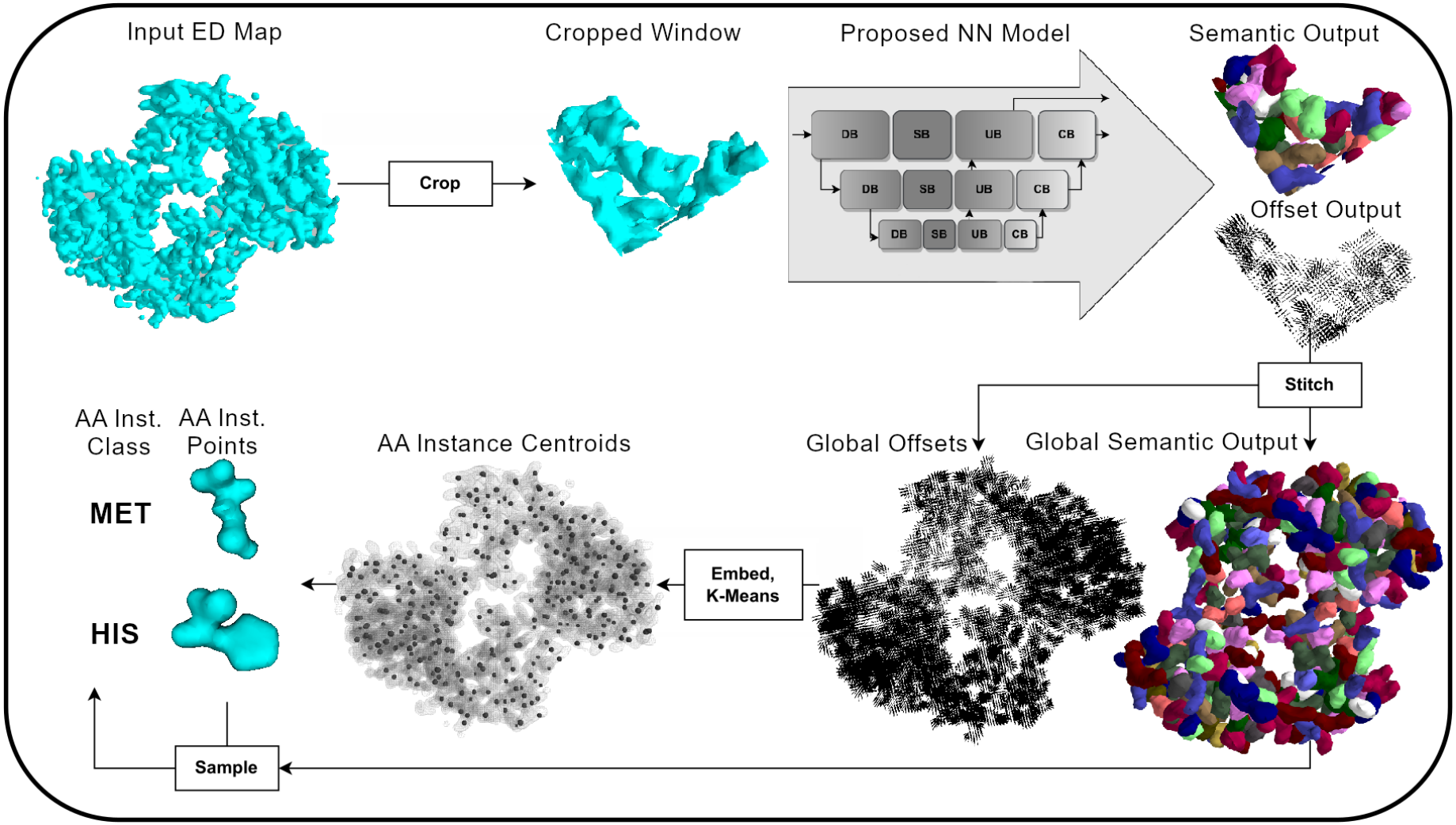


FIGURE 6.5: Flowchart of the inference pipeline. The input ED map is cropped into windows and their network outputs are stitched into global outputs. The global embeddings are calculated from the global offsets and the K instance centroids are located using K-means. The cluster points corresponding to the centroids are the AA instance points, whose class is determined by sampling the global semantic output in their location.

After instance localization, it is still necessary to match the located instances to GT instances. The goal is to match each g_k instance to the closest $\hat{g}_l$ GT instance. The distance between g_k and $\hat{g}_l$ is the Euclidean distance between the centroids of g_k, (u_k, v_k, w_k), and $\hat{g}_l$, $(\hat{u}_l, \hat{v}_l, \hat{w}_l)$:

$$d(g_k, \hat{g}_l) = ||(u_k, v_k, w_k) - (\hat{u}_l, \hat{v}_l, \hat{w}_l)||_2$$

This is approached as a linear sum assignment problem where the optimal pairings are obtained by minimizing the costs in the distance matrix D, where $D_{kl} = d(g_k, \hat{g}_l)$. The solver utilized is an openly available implementation [102] of the shortest path augmentation [103] method.

The end results of this step are paired g_k and $\hat{g}_k$ instances acquired without having to rely on the GT. Class predictions are then acquired by sampling the semantic output $S(\mathbf{X})$ in voxels corresponding to the g_k instances.

Performance Metrics

As was the case with 3D FC-DenseNet, semantic segmentation performance is measured in a per-voxel manner using the mean IoU of class probability predictions. Since MT-StackNet is an instance segmentation algorithm as well, instance segmentation performance receives more scrutiny.

The instance segmentation performance metric is the mean Dice-coefficient of the GT instances $\hat{g}_k \in \hat{G}$ and the paired, predicted instances $g_k \in G$:

$$Dice(\hat{G}, G) = \frac{1}{K} \sum_k Dice(\hat{g}_k, g_k)$$

where the corresponding instances' Dice-coefficient is calculated point-wise as

$$Dice(\hat{g}_k, g_k) = \frac{2(\hat{g}_k \cap g_k)}{|\hat{g}_k| + |g_k|}$$

Additionally, instance classification performance is measured via Rank-1 and Rank-3 Accuracy, Precision and Recall of paired instances. As before, for a prediction to be considered a Rank-1 hit, its top predicted class must match the GT class, while for a Rank-3 hit it is enough for the GT class to be among the three predictions with the highest probability.

6.5 Results & Discussion

6.5.1 Ablation Tests

The measured semantic and instance performance metrics in Table 6.1 show that removing the Bottleneck did not have a negative effect neither semantic nor instance segmentation performance. Its layers did not contribute to performance and the small feature maps by the bottleneck's stack level did not hold any further useful information.

By redistributing the Bottleneck layers to higher stack levels as SBs, network performance significantly increased. The larger feature maps still had information that could be extracted. In conclusion, the Bottleneck's layers redistributed as SBs are a better use of computational performance.

TABLE 6.1: Semantic and instance segmentation performances with (w/) and without (w/o) Skip Blocks (SB) and Bottleneck (The best performer is marked in boldface)

Semantic IoU	w/ Bottleneck	w/o Bottleneck
w/o SBs	0.4008	0.4114
w/ SBs	0.4442	**0.5091**
Instance Dice	w/ Bottleneck	w/o Bottleneck
w/o SBs	0.7569	0.7579
w/ SBs	0.7972	**0.7988**

With the bottleneck removed and SBs added, semantic segmentation performance increases from 40.07% to 50.91% and instance segmentation performance goes from 75.69% to 79.88%.

6.5.2 Training & Validation

Results (Table 6.2) show that MT-StackNet outperforms 3D FC-DenseNet on all tasks. When provided with GT instance locations for instance classification (as was the case with 3DFC-DenseNet), the actual difference in semantic segmentation performance is shown. Even when having to rely on the imperfect predicted instances, MT-StackNet outperforms 3D FC-DenseNet.

Furthermore, the increased performance of MT-StackNet is present for all AA residues at all resolutions, making MT-StackNet consistently better than 3D FC-DenseNet with no downsides (Figure 6.6).

The networks are very balanced predictors at all resolutions with no significant imbalance between precision and recall. The margin parameter δ decreases from start to finish (Figure 6.7), meaning that the networks optimize the instance segmentation losses while making the task harder as training progresses.

6.5.3 Experimental Results

The performance improvements compared to 3DFC-DenseNet carry over (Figure 6.8) to the experimental data set as well. Despite the varying resolution and quality of the experimental samples, MT-StackNet (similarly to 3D FC-DenseNet) remains functional even with low resolution ED maps (Table 6.3).

The loss graphs (Figure 6.7) show that all the networks continued learning during finetuning even after convergence on the fixed resolution data sets. The δ assigment margin increases initially during fine-tuning (Figure 6.7), but it starts to converge after the first few epochs and remains under the initial setting of 2. This ultimately means that the networks can assign voxels to instances with accuracy higher than the samples' physical resolution.

3D FC-DenseNet outperformed the accuracy of the Seqqy [80] module in ARP/wARP [81] at medium (60.47% for 3DFCDN vs. 41.29% for Seqqy)

TABLE 6.2: Fixed resolution data set results comparison between MT-StackNet (MTSN) and 3DFC-DenseNet (FCDN). GT for instance segmentation Dice-Coefficient refers to relying on GT locations instead of predicted instances

2Å	**MTSN**	**MTSN (GT Instances)**	**FCDN57**
Semantic IoU	0.9109	0.9109	0.8343
Instance Dice	0.7937	GT	GT
Rank1 Accuracy	0.8613	0.9696	0.8911
Rank3 Accuracy	0.9109	0.9799	0.9191
Precision	0.8560	0.9743	0.9591
Recall	0.8613	0.9696	0.9253
3Å	**MTSN**	**MTSN (GT Instances)**	**FCDN57**
Semantic IoU	0.7763	0.7763	0.6810
Instance Dice	0.8170	GT	GT
Rank1 Accuracy	0.8298	0.9139	0.7983
Rank3 Accuracy	0.8893	0.9468	0.8541
Precision	0.8298	0.9214	0.8877
Recall	0.8298	0.9139	0.8344
4Å	**MTSN**	**MTSN (GT Instances)**	**FCDN49**
Semantic IoU	0.4184	0.4184	0.3316
Instance Dice	0.6467	GT	GT
Rank1 Accuracy	0.6059	0.6720	0.5171
Rank3 Accuracy	0.8115	0.8710	0.6563
Precision	0.6191	0.7017	0.5540
Recall	0.6059	0.6720	0.5505

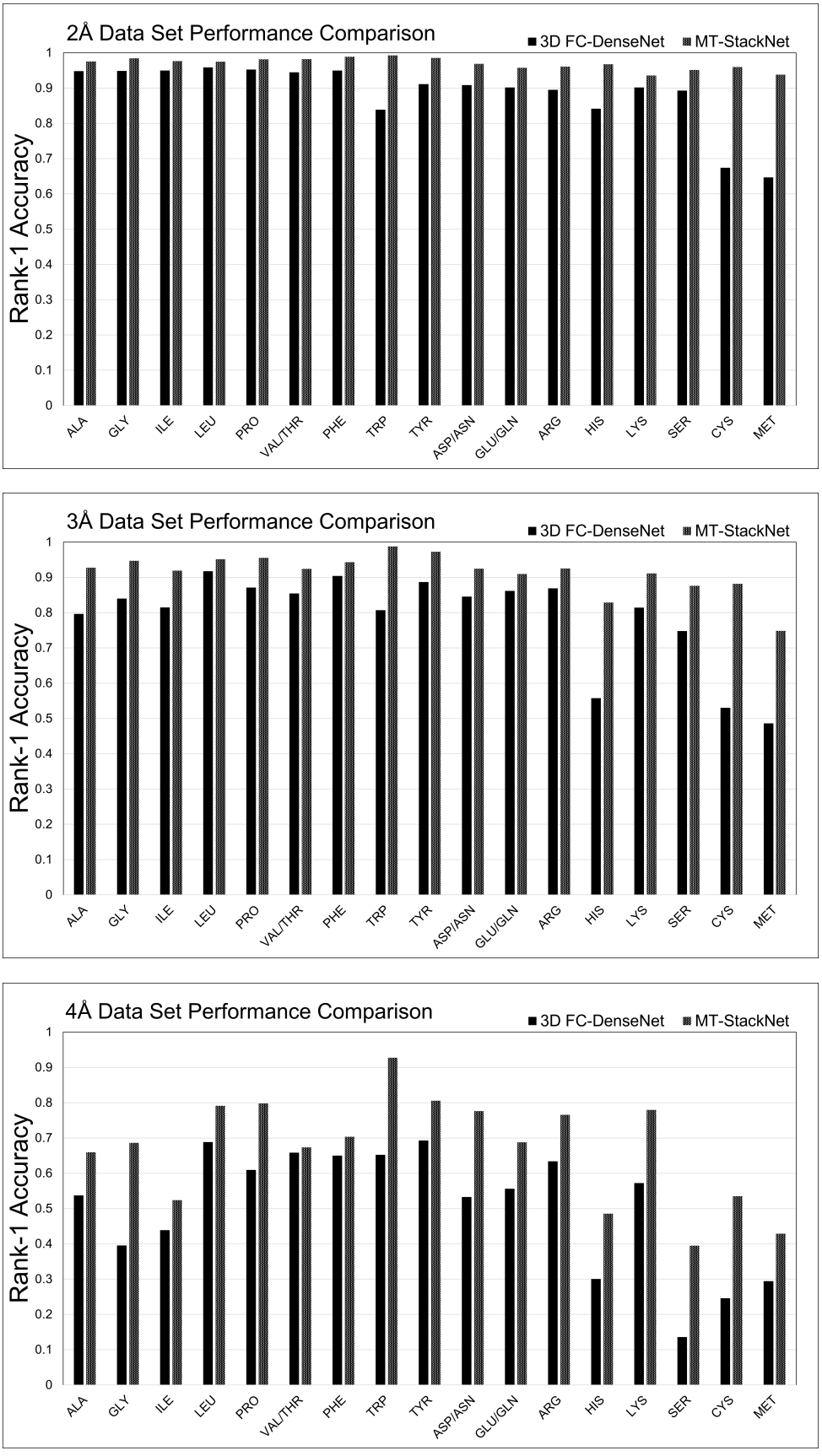


FIGURE 6.6: Per-residue Rank-1 Accuracies of 3D FC-DenseNet and MT-Stacknet at high (top), medium (middle), and low resolutions (bottom). MT-Stacknet consistently outperforms 3D FC-DenseNet at all resolutions.

TABLE 6.3: Experimental data set results comparison between MT-StackNet (MTSN) and 3DFC-DenseNet (FCDN). GT for instance segmentation Dice-Coefficient refers to relying on GT locations instead of instance predictions.

<2.5Å	**MTSN**	**MTSN (GT Instances)**	**FCDN57**
Semantic IoU	0.7575	0.7575	0.6078
Instance Dice	0.7982	GT	GT
Rank1 Accuracy	0.7716	0.8670	0.7116
Rank3 Accuracy	0.8542	0.9196	0.8119
Precision	0.7735	0.8734	0.8414
Recall	0.7716	0.8670	0.7567
2.5–3.5Å	**MTSN**	**MTSN (GT Instances)**	**FCDN57**
Semantic IoU	0.6033	0.6033	0.5108
Instance Dice	0.7231	GT	GT
Rank1 Accuracy	0.6478	0.7639	0.6047
Rank3 Accuracy	0.7636	0.8518	0.7495
Precision	0.6602	0.7857	0.8142
Recall	0.6478	0.7639	0.6873
>3.5Å	**MTSN**	**MTSN (GT Instances)**	**FCDN49**
Semantic IoU	0.2310	0.2310	0.2182
Instance Dice	0.5341	GT	GT
Rank1 Accuracy	0.3282	0.4075	0.3328
Rank3 Accuracy	0.5317	0.6247	0.5122
Precision	0.3626	0.4679	0.4620
Recall	0.3282	0.4075	0.3428

and low resolutions (33.28% for 3DFCDN vs. 1.43% for Seqqy) and nearly matched its performance at high resolutions (71.16% for 3DFCDN vs. 72.06% for Seqqy).

Due to the performance improvements, MT-StackNet outperforms the accuracy of 3D FC-DenseNet on average by 23.53% (Figure 6.9). Consequently, MT-StackNet now outperforms Seqqy at high resolutions (77.16% for MT-StackNet vs. 72.06% for Seqqy) as well.

Although Seqqy may perform better than MT-StackNet for some proteins, this is only true at high resolutions (Figure 6.10) and MT-StackNet consistently outperforms Seqqy on average. Furthermore, Seqqy's Rank-1 Accuracy standard deviations make it less reliable predictor even at high resolutions: its error is 33.54% at high, 37.98% at medium, and 4.67% at low resolutions (although accuracy here is already very low). MT-StackNet maintains a lower standard deviation (Figure 6.9) across the board: 8.09% at high, 13.22% at medium, and 9.29% at low resolutions.

It is important to note that Seqqy may produce less than complete models where not all GT AA residues have a corresponding predicted residue while MT-StackNet always outputs the requested K number of residues. The measured accuracy, precision and recall values only consider the matched residue pairs which skews the measurements in Seqqy's favour since unassigned residue labels are excluded from recall and accuracy calculations. Seqqy's completeness on the experimental data set is 70.25% over 2.5 Å, 41.27% between 2.5–3.5 Å and 01.76% under 3.5 Å resolution.

With same hardware used for inference as 3DFC-DenseNet (GTX 1080Ti GPU), MT-StackNet computes results much faster than both 3D FC-DenseNet and Seqqy. Seqqy attempts to map the AA sequence by constructing fragments and attempting to match and merge the built fragments which is a computationally intensive task. This leads to seqqy's computational time scaling with the number of residues in the protein.

In comparison, both 3D FC-DenseNet and MT-StackNet's computational time scales linearly with the volume of the ED map. Since input windows are processed in a fixed amount of time regardless of how many residues are in them, the more AAs there are in a given volume, the greater the performance gain over Seqqy. Furthermore, thanks to the parallel processing capabilities of modern GPUs, batches of input volumes take the same amount of time as a single input window, leading to drastically decreased processing times.

To demonstrate the discrepancy between Seqqy and the proposed methods' computational time, two case studies were performed with proteins chosen based on their number of residues. An average sized protein An average sized [104], 476 residues long protein, PDB ID 6E0T [13], and a considerably larger 1815 residues long protein, PDB ID 2Y1V [105], are compared. Since Seqqy does not utilize the GPU, the computational time of 3D FC-DenseNet and MT-StackNet was measured utilizing only the CPU as well as with GPU acceleration.

The shorter protein, 6E0T is processed in 10 seconds with MT-StackNet with GPU acceleration and 64 seconds on the CPU, 22 seconds with 3D FC-DenseNet with GPU acceleration and 196 seconds without. Seqqy takes approximately 7 minutes to process 6E0T.

The larger protein, 2Y1V takes 25 seconds with MT-StackNet on the GPU and 205 seconds without GPU acceleration, 86 seconds with FC-DenseNet with GPU acceleration and 715 seconds on the CPU. Seqqy requires approximately 70 minutes for 2Y1V.

The improved computational time between 3DFC-DenseNet and MT-StackNet is due to MT-StackNet being both shallower mostly thanks to the removal of the bottleneck (MT-StackNet is at most 48 and 44 layers deep for the semantic and instance outputs respectively vs. 3DFC-DenseNet's 59 layer depth) and narrower since the feature maps are not being concatenated. This leads to a more efficient use of parameters and less operations needed overall to

process a sample.

It can be concluded that MT-StackNet is significant upgrade over 3DFC-DenseNet in capabilities, performance and speed as well.

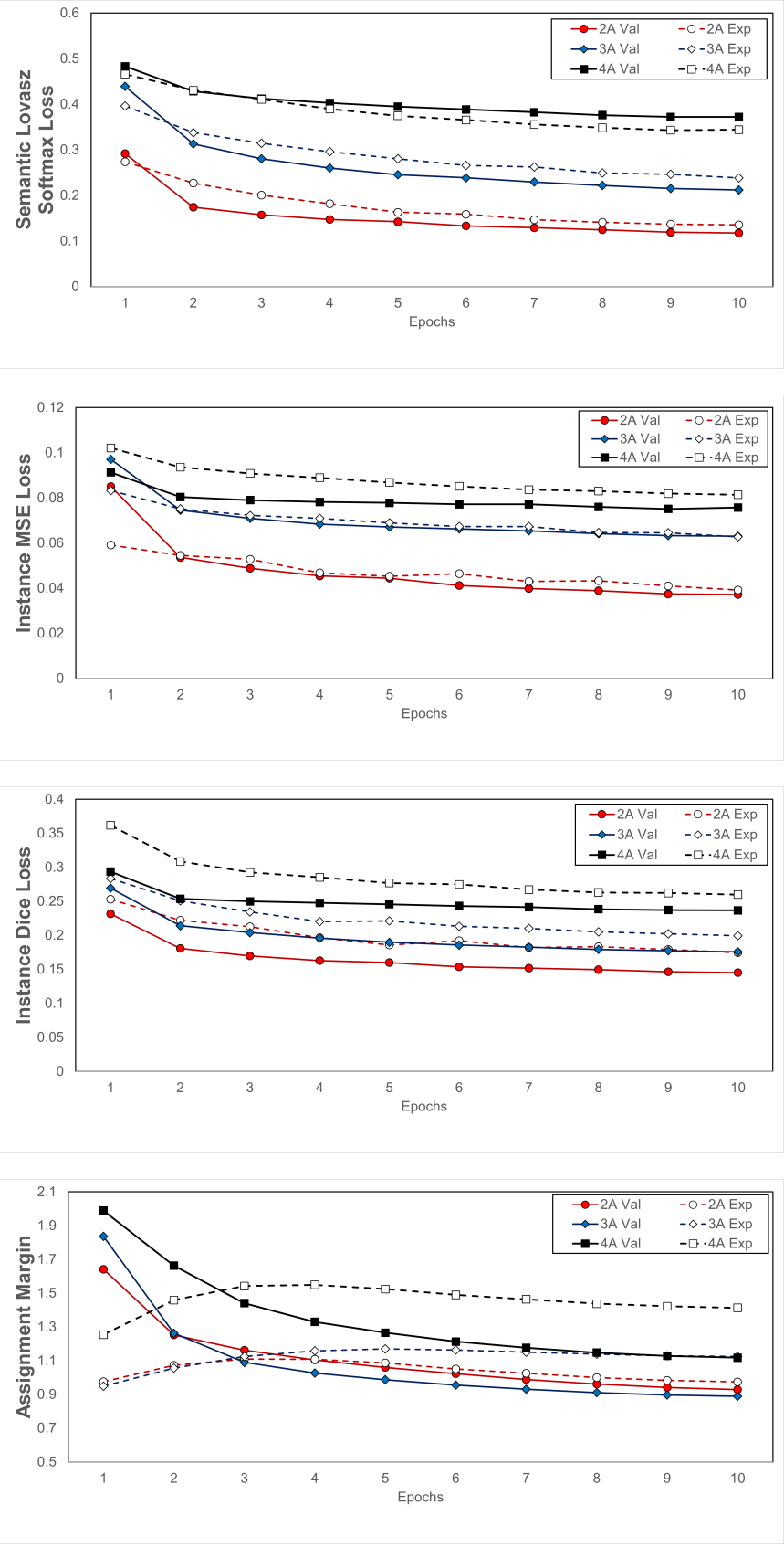


FIGURE 6.7: Fixed resolution (Val, solid lines and markers) and experimental (Exp, dashed lines and empty markers) data set loss graphs.

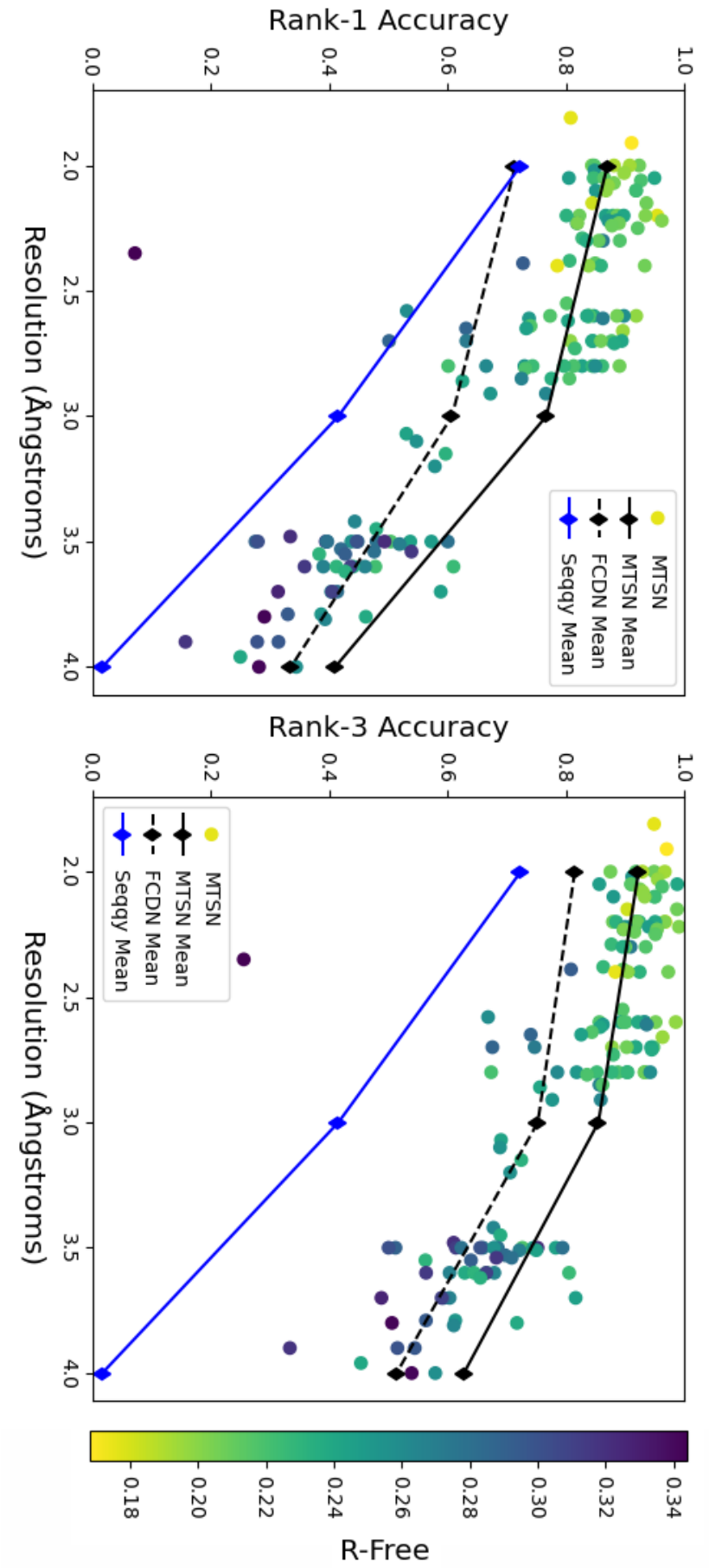


FIGURE 6.8: Experimental sample Rank-1 and Rank-3 accuracies. Lines represent the average accuracies over the experimental data sets. Compared to the baseline set by 3D FC-DenseNet (FCDN), MT-StackNet (MTSN, with GT AA locations for comparison) presents improvements even at low resolutions and with high R-Free samples as well.

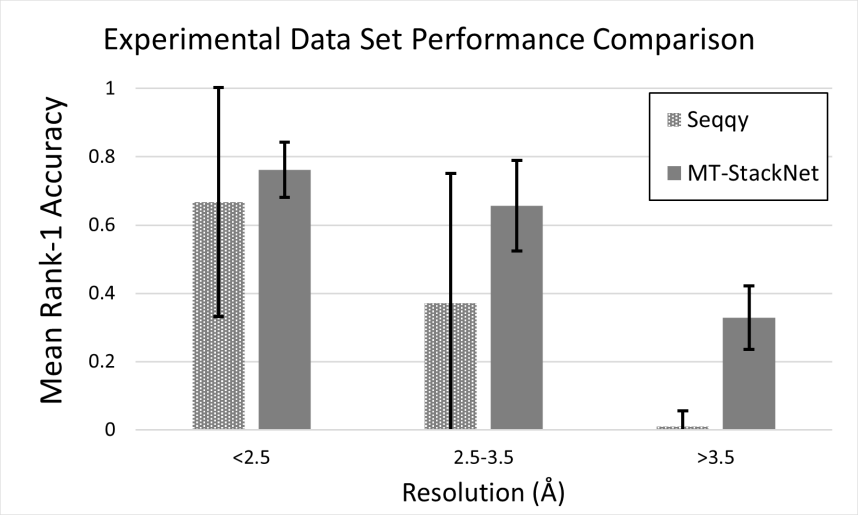


FIGURE 6.9: Mean Rank-1 Accuracies and Standard Deviations (error bars) of Seqqy and MT-StackNet on the experimental Data Set.

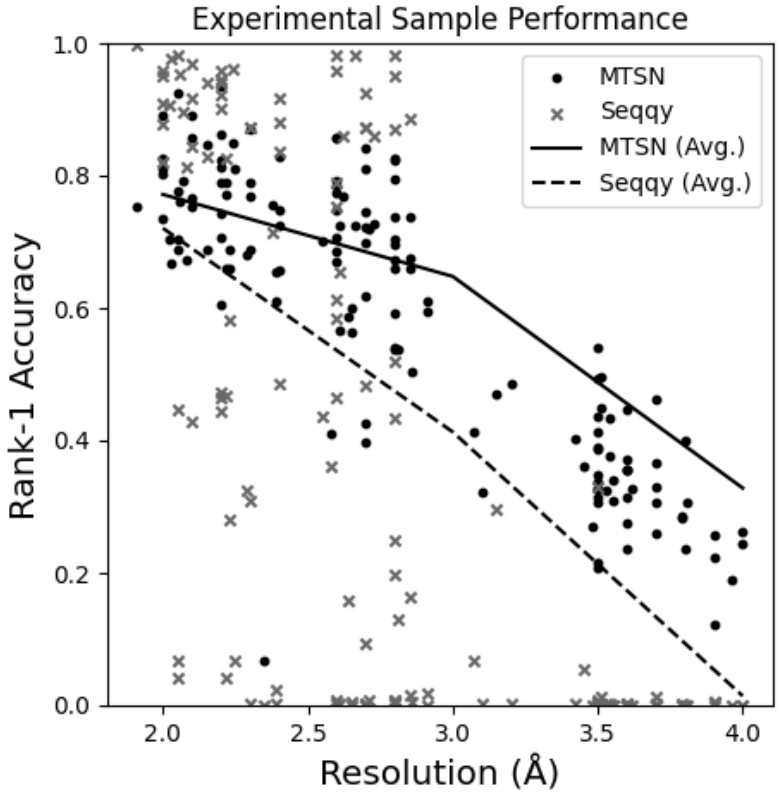


FIGURE 6.10: Scatter plot of per-sample Rank-1 Accuracies of Seqqy and MT-StackNet (MTSN) with predicted residue locations on the experimental data set. The lines represent the average accuracy over the experimental data sets.

Chapter 7

Conclusion

This work proposed a computer vision and machine learning approach to to classifying and locating AA residues in protein ED maps. The necessary background of protein structure, XRD and structure determination difficulties were presented, based on which it defined the task as semantic and instance segmentation problem. The proposed methods are 3D CNN architectures capable of directly segmenting volumetric data.

A custom data set for ML tasks involving ED maps was created meeting the problem criteria. To the author's knowledge, it is the broadest data set of its kind with over 20,000 generated ED map samples across multiple resolutions. By containing ED maps at multiple resolutions of the same protein, it allows the direct observation of the effects of decreasing resolution on performance. Furthermore, a data set consisting entirely of approx. 600 real-life, experimentally acquired ED maps was created as well to measure the proposed methods' practical performance.

The proposed 3D CNN architectures are capable of assigning AA labels by only observing the ED maps, as opposed to current state-of-the-art structure determination toolkits which also rely on the protein's AA sequence. The methods were measured agains Seqqy, a state-of-the-art structure determination algorithm part of the popular ARP/wARP toolkit.

3D FC-DenseNet matches the performance of Seqqy at high resolutions

(above 2.5 Å) and beats it at lower resolutions. It can also segment low resolution samples below 3.5 Å resolution, where Seqqy fails to assign meaningful AA labels.

The MT-StackNet architecture is an all-around upgrade to 3D FC-DenseNet in both capability and accuracy. In addition to being able to locate all residues in the input in a single forward pass, it outperforms 3D FC-DenseNet by approx. 23.53%. Both architectures can segment entire ED maps in seconds, a fraction of the time required by ARP/wARP.

By being able to reliably segment protein ED maps quickly and remaining functional at low resolutions, the architectures directly target the resolution and volume bottlenecks currently present in structure determination tasks. The results compare favorably to existing state-of-the-art methods. Both 3D FC-DenseNet and MT-StackNet only rely on observing the ED maps.

Furthermore, the presented methods are more than just applications of existing NN architectures. They have been custom built with novel elements (adaptive zero padding in 3D FC-DenseNet and POR, SBs, CBs in MT-StackNet) which improve volumetric data segmentation performance.

Despite these, there remain points for improvement. The increases in performance MT-StackNet presents over 3D FC-DenseNet are weaker with low resolution samples. A possible reason for this is the limited amount of samples in the experimental data sets.

The straightforward way to improve this would be to extend the data sets with further low resolution samples. Alternatively, the fixed resolution data sets could be made to better resemble real data. Transferring the properties of the experimental samples to the fixed resolution samples used for pretraining by simulating their varying resolution and R factor is a possibility.

The data sets themselves hold various future possibilities: by having multiple resolution ED maps of the same proteins, super-resolution (SR) or reconstruction tasks could also be directly solved using the data set. In this way,

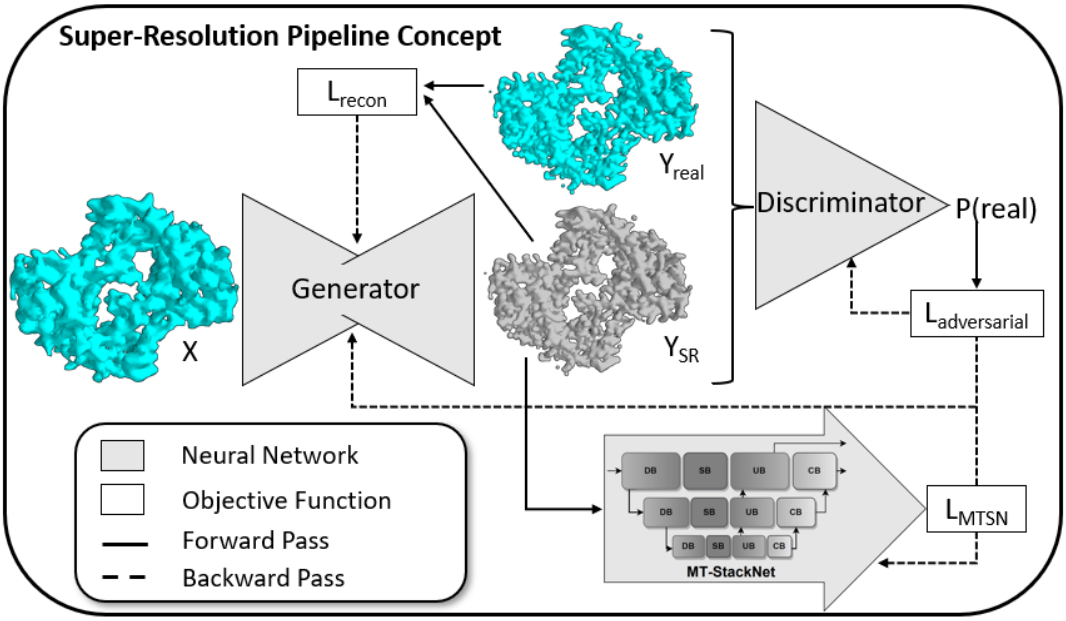


FIGURE 7.1: Overview of the possible future enhancement SR pipeline's concept. Having an auxiliary reconstruction loss L_{recon} and MT-StackNet's task losses L_{MTSN} as downstream supervision on top of the adversarial loss $L_{adversarial}$ may encourage the generator to not only create SR densities, but to also refine them to increase AA classification performance.

low and medium resolution sample quality could be increased via super-resolution to benefit from MT-StackNet's (or possibly one of its successors') performance on higher resolution samples.

This is more in line with refinement and goes beyond simple SR. By training a volumetric SR NN (such as VolumeNet [106]) as a generator in an adversarial manner the ED map resolution and detail could be increased. This is fitting for the data set since 'real' higher resolution samples exist for all proteins, so the discriminator can be tasked with discriminating SR and 'real' ED map samples. Additionally, auxiliary reconstruction losses such as the Structural Similarity Index Measure [107] coupled with L1 distance or feature similarities may be used to ensure that the generated samples align with the actual higher resolution targets.

Then, MT-StackNet could serve as downstream supervision to assist with instance and class specific guidance. In this way the generator would not

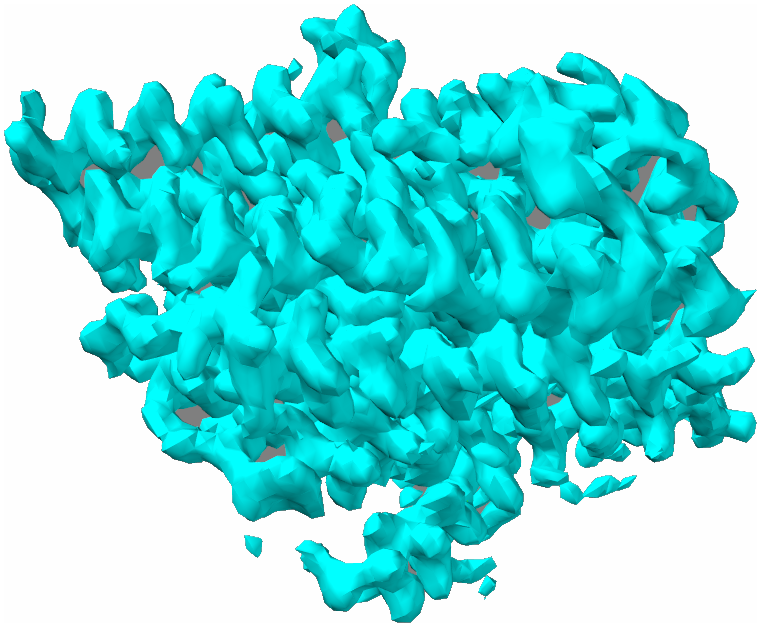


FIGURE 7.2: A Cryo-EM Coulomb map of a protein. Coulomb maps also fit the problem definition and are compatible with the methods presented. A dataset for training with Coulombs maps may be necessary due to them having different features from ED maps.

only output SR densities, but also refine them as needed to increase performance on the AA classification task. An overview of the concept can be seen in Figure 7.1.

Another popular imaging method is Cryo-EM. The resulting Coulomb maps obtained from Cryo-EM experiments also fit the problem definition (Figure 7.2). The network architectures presented in this work could be applied to Cryo-EM Coulomb maps with little to no modification. Training with Cryo-EM samples may be necessary as Coulomb maps may have different features from ED maps. Including Cryo-EM samples presents another extension opportunity for the data set.

Currently the network architectures only utilize considerations from computer vision for performance improvements. By designing network modules or loss functions with chemical considerations in mind (e.g. incorporating force fields, torsion angles, bond energies, etc.) networks specific to this task may be designed.

As the next step in automated model building, the instance segmentation GT could be enhanced to regress to atom positions in addition to AA locations. This would be a significant step towards end-to-end NN-based structure determination since the knowledge of individual atom positions and the class of their parent residue is enough to generate a .pdb model file.

All in all, as answer to the central question of the thesis, it can be summarized that computer vision methods and NNs are not only applicable to protein structure determination tasks, but that they also perform remarkably well.

Hopefully, this work represents just the first steps towards the next generation of protein structure determination methods and the approaches presented here will prove useful for not only those with backgrounds in computer vision, but for experts of structural biology and crystallography as well.

Bibliography

[1] *Papers with Code - The latest in Machine Learning*, [Online; accessed 30. Nov. 2022], Nov. 2022. [Online]. Available: https://paperswithcode.com.

[2] *Hugging Face – The AI community building the future.* [Online; accessed 30. Nov. 2022], Nov. 2022. [Online]. Available: https://huggingface.co.

[3] *PyTorch Hub*, [Online; accessed 30. Nov. 2022], Nov. 2022. [Online]. Available: https://pytorch.org/hub.

[4] L. Floridi and M. Chiriatti, "GPT-3: Its Nature, Scope, Limits, and Consequences," *Minds &. Machines*, vol. 30, no. 4, pp. 681–694, Dec. 2020, ISSN: 1572-8641. DOI: 10.1007/s11023-020-09548-1.

[5] W. Wang, J. Dai, Z. Chen, *et al.*, "Internimage: Exploring large-scale vision foundation models with deformable convolutions," *arXiv preprint arXiv:2211.05778*, 2022.

[6] *Stability.Ai - Stable Diffusion Announcement*, [Online; accessed 30. Nov. 2022], Nov. 2022. [Online]. Available: https://stability.ai/blog/stable-diffusion-announcement.

[7] K.-H. Yu, A. L. Beam, and I. S. Kohane, "Artificial intelligence in healthcare," *Nat. Biomed. Eng.*, vol. 2, no. 10, pp. 719–731, Oct. 2018, ISSN: 2157-846X. DOI: 10.1038/s41551-018-0305-z.

[8] S. Webb, "Deep learning for biology.," *Nature*, vol. 554, no. 7690, pp. 555–558, Feb. 2018, ISSN: 0028-0836. [Online]. Available: https://go.gale.com/ps/i.do?id=GALE%7CA660257863&sid=googleScholar&v=2.1%E2%81%A2=r&linkaccess=abs&issn=00280836&p=HRCA&sw=w&userGroupName=anon%7E56abcf18.

[9] K. P. Smith and J. E. Kirby, "Image analysis and artificial intelligence in infectious disease diagnostics," *Clinical Microbiology and Infection*, vol. 26, no. 10, pp. 1318–1323, Oct. 2020, ISSN: 1198-743X. DOI: 10.1016/j.cmi.2020.03.012.

[10] N. Lorenzovici, E.-H. Dulf, T. Mocan, and L. Mocan, "Artificial Intelligence in Colorectal Cancer Diagnosis Using Clinical Data: Non-Invasive Approach," *Diagnostics*, vol. 11, no. 3, p. 514, Mar. 2021, ISSN: 2075-4418. DOI: 10.3390/diagnostics11030514.

[11] S. Dutta and K. Bose, "Remodelling structure-based drug design using machine learning," *Emerging Top. Life Sci.*, vol. 5, no. 1, pp. 13–27, May 2021, ISSN: 2397-8554. DOI: 10.1042/ETLS20200253. eprint: 33825834.

[12] J. Bajorath, "Deep Machine Learning for Computer-Aided Drug Design," *Front. Drug. Discov.*, vol. 0, 2022, ISSN: 2674-0338. DOI: 10.3389/fddsv.2022.829043.

[13] T.-L. Yeh, C.-Y. S. Lee, L. M. Amzel, P. J. Espenshade, and M. A. Bianchet, "The Hypoxic Regulator of Sterol Synthesis Nro1 Is a Nuclear Import Adaptor," *Structure*, vol. 19, no. 4, pp. 503–514, Apr. 2011, ISSN: 0969-2126. DOI: 10.1016/j.str.2011.01.017.

[14] M. Batool, B. Ahmad, and S. Choi, "A Structure-Based Drug Discovery Paradigm," *Int. J. Mol. Sci.*, vol. 20, no. 11, Jun. 2019. DOI: 10.3390/ijms20112783.

[15] *IUPAC - Nomenclature and Symbolism for Amino Acids and Peptides*, [Online; accessed 30. Nov. 2022], Sep. 2021. [Online]. Available: https://iupac.qmul.ac.uk/AminoAcid/AA1n2.html.

[16] *Protein structure: Primary, secondary, tertiary & quatrenary (article) | Khan Academy*, [Online; accessed 24. Nov. 2022], Nov. 2022. [Online]. Available: https://www.khanacademy.org/science/biology/macromolecules/proteins-and-amino-acids/a/orders-of-protein-structure.

[17] S. Behjati and P. S. Tarpey, "What is next generation sequencing?" *Arch. Dis. Child. Educ. Pract. Ed.*, vol. 98, no. 6, pp. 236–238, Dec. 2013, ISSN: 1743-0585. DOI: 10.1136/archdischild-2013-304340.

[18] M. Baek, F. DiMaio, I. Anishchenko, *et al.*, "Accurate prediction of protein structures and interactions using a three-track neural network," *Science*, vol. 373, no. 6557, pp. 871–876, Aug. 2021, ISSN: 0036-8075. DOI: 10.1126/science.abj8754.

[19] *AlphaFold: Using AI for scientific discovery*, [Online; accessed 28 Nov. 2022]. [Online]. Available: https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology.

[20] *Schematic of 4-circle diffractometer*, [Online; accessed 20. Dec. 2022], Dec. 2022. [Online]. Available: https://serc.carleton.edu/details/images/8400.html.

[21] *File:Reciprocal space.png - Wikimedia Commons*, [Online; accessed 22. Dec. 2022], Sep. 2010. [Online]. Available: https://commons.wikimedia.org/wiki/File:Reciprocal_space.png.

[22] *Atoms and X-rays: Seeing inside a crystal - Understanding Science*, [Online; accessed 28. Dec. 2022], Aug. 2022. [Online]. Available: https://undsci.berkeley.edu/the-structure-of-dna-

cooperation-and-competition/atoms-and-x-rays-seeing-inside-a-crystal.

[23] T. M. de Oliveira, L. van Beek, F. Shilliday, J. É. Debreczeni, and C. Phillips, "Cryo-EM: The Resolution Revolution and Drug Discovery," *SLAS Discov.*, vol. 26, no. 1, pp. 17–31, Jan. 2021, ISSN: 2472-5560. DOI: 10.1177/2472555220960401. eprint: 33016175.

[24] C. L. Lawson, A. Patwardhan, M. L. Baker, *et al.*, "EMDataBank unified data resource for 3DEM," *Nucleic Acids Res.*, vol. 44, no. D1, pp. D396–D403, Jan. 2016, ISSN: 0305-1048. DOI: 10.1093/nar/gkv1126.

[25] Emdb, *Electron Microscopy Data Bank*, [Online; accessed 1. Dec. 2022], Dec. 2022. [Online]. Available: https://www.ebi.ac.uk/emdb/statistics/emdb_resolution_year.

[26] Embl-Ebi, *Current Release Statistics < Uniprot < EMBL-EBI*, [Online; accessed 29. Nov. 2022], 2022. [Online]. Available: https://www.ebi.ac.uk/uniprot/TrEMBLstats.

[27] wwpdb. org, *wwPDB: Deposition Statistics*, [Online; accessed 29. Nov. 2022], 2022. [Online]. Available: https://www.wwpdb.org/stats/deposition.

[28] wwPDB consortium, "Protein Data Bank: the single global archive for 3D macromolecular structure data," *Nucleic Acids Research*, vol. 47, no. D1, pp. D520–D528, Oct. 2018, ISSN: 0305-1048. DOI: 10.1093/nar/gky949. eprint: https://academic.oup.com/nar/article-pdf/47/D1/D520/27436045/gky949.pdf. [Online]. Available: https://doi.org/10.1093/nar/gky949.

[29] *Success Rates in Crystallography*, [Online; accessed 1. Dec. 2022], 2012. [Online]. Available: `https://www.umass.edu/molvis/workshop/allstruc/xrc_succ.htm`.

[30] D. M. Blow, *Outline of crystallography for biologists*. Oxford University Press, 2020.

[31] Y.-F. Huang, "Study of mining protein structural properties and its application," PhD Dissertation, National Taiwan University, Taipei, Taiwan, 2007.

[32] G. Langer, S. X. Cohen, V. S. Lamzin, and A. Perrakis, "Automated macromolecular model building for X-ray crystallography using ARP/wARP version 7," *Nat. Protoc.*, vol. 3, no. 7, pp. 1171–1179, 2008, ISSN: 1750-2799. eprint: 18600222.

[33] T. C. Terwilliger, R. W. Grosse-Kunstleve, P. V. Afonine, *et al.*, "Iterative model building, structure refinement and density modification with the *PHENIX AutoBuild* wizard," *Acta Crystallographica Section D*, vol. 64, no. 1, pp. 61–69, 2008.

[34] Phenix, *Model building (and more) in the AutoBuild GUI*, [Online; accessed 3 Dec. 2022], 2022. [Online]. Available: `http://phenix-online.org/documentation/reference/autobuild_gui.html`.

[35] P. V. A. e. a. D. Liebschner, "Macromolecular structure determination using x-rays, neutrons and electrons: Recent developments in phenix," *Acta Cryst.*, vol. D75, pp. 861–877, 2019.

[36] T. C. Terwilliger, "Automated main-chain model building by template matching and iterative fragment extension," *Acta Crystallographica Section D: Biological Crystallography*, vol. 59, no. 1, pp. 38–44, 2003.

[37] K. Cowtan, "The *Buccaneer* software for automated model building. 1. Tracing protein chains," *Acta Crystallographica Section D*, vol. 62, no. 9, pp. 1002–1011, 2006.

[38] T. Holton, T. R. Ioerger, J. A. Christopher, and J. C. Sacchettini, "Determining protein structure from electron-density maps using pattern matching," *Acta Crystallographica Section D: Biological Crystallography*, vol. 56, no. 6, pp. 722–734, 2000.

[39] J. Lyons, A. Dehzangi, R. Heffernan, *et al.*, "Predicting backbone cα angles and dihedrals from protein sequences by stacked sparse auto-encoder deep neural network," *Journal of computational chemistry*, vol. 35, no. 28, pp. 2040–2046, 2014.

[40] M. Kowiel, D. Brzezinski, P. J. Porebski, I. G. Shabalin, M. Jaskolski, and W. Minor, "Automatic recognition of ligands in electron density by machine learning," *Bioinformatics*, vol. 35, no. 3, pp. 452–461, 2019.

[41] P. Emsley, B. Lohkamp, W. G. Scott, and K. Cowtan, "Features and development of coot," *Acta Crystallographica Section D - Biological Crystallography*, vol. 66, pp. 486–501, 2010.

[42] P. S. Bond, K. S. Wilson, and K. D. Cowtan, "Predicting protein model correctness in Coot using machine learning," *Acta Cryst. D*, vol. 76, no. 8, pp. 713–723, 2020, ISSN: 2059-7983.

[43] G. Chojnowski, J. Pereira, and V. S. Lamzin, "Sequence assignment for low-resolution modelling of protein crystal structures," *Acta Cryst. D*, vol. 75, no. 8, pp. 753–763, 2019, ISSN: 2059-7983.

[44] R. J. Morris, A. Perrakis, and V. S. Lamzin, "ARP/wARP's model-building algorithms. I. The main chain," *Acta Crystallogr., Sect. D: Biol. Crystallogr.*, vol. 58, no. 6, pp. 968–975, Jun. 2002, ISSN: 0907-4449. DOI: 10.1107/S0907444902005462.

[45] CCP4, *CCP4 Program Suite: maplib*, [Online; accessed 28 Nov. 2022], 2021. [Online]. Available: `http://legacy.ccp4.ac.uk/html/maplib.html#description`.

[46] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017.

[47] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "SEMANTIC3D.NET: A new large-scale point cloud classification benchmark," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1-W1, 2017, pp. 91–98.

[48] M. Tiator, C. Geiger, and P. Grimm, "Point cloud segmentation with deep reinforcement learning," *24th European Conference on Artificial Intelligence - ECAI 2020*, pp. 2768–2775, 2020.

[49] N. Zhao, T.-S. Chua, and G. H. Lee, "Few-shot 3d point cloud semantic segmentation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8873–8882, 2021.

[50] A. X. Chang, T. Funkhouser, L. Guibas, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv:1512.03012*, 2015.

[51] A. Karambakhsh, B. Sheng, P. Li, P. Yang, Y. Jung, and D. D. Feng, "VoxRec: Hybrid Convolutional Neural Network for Active 3D Object Recognition," *IEEE Access*, vol. 8, pp. 70 969–70 980, 2020, ISSN: 2169-3536.

[52] H. Zhao, M. Tang, and H. Ding, "HoPPF: A novel local surface descriptor for 3D object recognition," *Pattern Recognit.*, vol. 103, p. 107 272, 2020, ISSN: 0031-3203.

[53] H. Lu, H. Wang, Q. Zhang, S. W. Yoon, and D. Won, "A 3D Convolutional Neural Network for Volumetric Image Semantic

Segmentation," *Procedia Manuf.*, vol. 39, pp. 422–428, Jan. 2019, ISSN: 2351-9789. DOI: 10.1016/j.promfg.2020.01.386.

[54] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," in *International conference on medical image computing and computer-assisted intervention*, Cham: Springer International Publishing, 2016, pp. 424–432.

[55] X. Yang, L. Yu, S. Li, *et al.*, "Towards automated semantic segmentation in prenatal volumetric ultrasound," *IEEE transactions on medical imaging*, vol. 38, no. 1, pp. 180–193, 2018.

[56] H. Chen, Q. Dou, L. Yu, J. Qin, and P.-A. Heng, "Voxresnet: Deep voxelwise residual networks for brain segmentation from 3d mr images," *NeuroImage*, vol. 170, pp. 446–455, 2018.

[57] L. Sun, S. Zhang, H. Chen, and L. Luo, "Brain Tumor Segmentation and Survival Prediction Using Multimodal MRI Scans With Deep Learning," *Front. Neurosci.*, vol. 13, p. 810. 2019, ISSN: 1662-4548. eprint: 31474816.

[58] W. Chen, B. Liu, S. Peng, J. Sun, and X. Qiao, "S3D-UNet: Separable 3D U-Net for Brain Tumor Segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Cham: Springer International Publishing, 2019, pp. 358–368, ISBN: 978-3-030-11725-2.

[59] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*, IEEE, 2016, pp. 565–571.

[60] H. Lu, H. Wang, Q. Zhang, S. W. Yoon, and D. Won, "A 3D Convolutional Neural Network for Volumetric Image Semantic

Segmentation," *Procedia Manuf.*, vol. 39, pp. 422–428, 2019, ISSN: 2351-9789.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[62] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[63] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[64] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017, pp. 11–19.

[65] L. Yu, X. Yang, H. Chen, J. Qin, and P. A. Heng, "Volumetric convnets with mixed residual connections for automated prostate segmentation from 3d mr images," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[66] A. Godo, K. Aoki, A. Nakagawa, and Y. Yagi, "Residue assignment in crystallographic protein electron density maps with 3d convolutional networks," *IEEE Access*, vol. 10, pp. 28 760–28 772, 2022. DOI: 10.1109/ACCESS.2022.3156108.

[67] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc.,

2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[68] M. Vassura, L. Margara, P. Di Lena, F. Medri, P. Fariselli, and R. Casadio, "Reconstruction of 3D Structures From Protein Contact Maps," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 5, no. 3, pp. 357–367, Mar. 2008, ISSN: 1557-9964. DOI: 10.1109/TCBB.2008.27.

[69] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[70] *Home - Prediction Center*, [Online; accessed 1 Dec. 2022]. [Online]. Available: https://www.predictioncenter.org/index.cgi.

[71] D. Chakravarty and L. L. Porter, "AlphaFold2 fails to predict protein fold switching," *Protein Sci.*, vol. 31, no. 6, e4353, Jun. 2022, ISSN: 0961-8368. DOI: 10.1002/pro.4353.

[72] M. A. Marques, M. D. Purdy, and M. Yeager, "CryoEM maps are full of potential," *Curr. Opin. Struct. Biol.*, vol. 58, pp. 214–223, Oct. 2019, ISSN: 0959-440X. DOI: 10.1016/j.sbi.2019.04.006.

[73] M. Beckers, D. Mann, and C. Sachse, "Structural interpretation of cryo-EM image reconstructions," *Prog. Biophys. Mol. Biol.*, vol. 160, pp. 26–36, Mar. 2021, ISSN: 0079-6107. DOI: 10.1016/j.pbiomolbio.2020.07.004.

[74] P. Mostosi, H. Schindelin, P. Kollmannsberger, and A. Thorn, "Haruspex: A Neural Network for the Automatic Identification of Oligonucleotides and Protein Secondary Structure in Cryo-Electron Microscopy Maps," *Angew. Chem.*, vol. 132, no. 35, pp. 14 898–14 905, 2020, ISSN: 0044-8249.

[75] D. Si, S. A. Moritz, J. Pfab, *et al.*, "Deep learning to predict protein backbone structure from high-resolution cryo-em density maps," *Scientific reports*, vol. 10, no. 1, pp. 1–22, 2020.

[76] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[77] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154–6162.

[78] A. Kirillov, Y. Wu, K. He, and R. Girshick, "Pointrend: Image segmentation as rendering," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9799–9808.

[79] K. Xu, Z. Wang, J. Shi, H. Li, and Q. C. Zhang, "A2-Net: Molecular Structure Estimation from Cryo-EM Density Volumes," *AAAI*, vol. 33, no. 01, pp. 1230–1237, Jul. 2019, ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33011230.

[80] G. Chojnowski, J. Pereira, and V. S. Lamzin, "Sequence assignment for low-resolution modelling of protein crystal structures," *Acta Cryst. D*, vol. 75, no. 8, pp. 753–763, 2019, ISSN: 2059-7983.

[81] *ARP/wARP Home Page*, [Online; accessed 1. Dec. 2022], Dec. 2022. [Online]. Available: https://www.embl-hamburg.de/ARP.

[82] Y. Li, X. Bian, M.-c. Chang, L. Wen, and S. Lyu, "Pixel offset regression (por) for single-shot instance segmentation," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2018, pp. 1–6. DOI: 10.1109/AVSS.2018.8639428.

[83] K. Dijkstra, J. van de Loosdrecht, L. R. Schomaker, and M. A. Wiering, "Centroidnet: A deep neural network for joint object

localization and counting," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11053 LNAI, pp. 585–601, 2019, ISSN: 16113349. DOI: 10.1007/978-3-030-10997-4_36.

[84] D. Neven, B. de Brabandere, M. Proesmans, and L. van Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," *arXiv*, 2019, ISSN: 23318422.

[85] wwPDB consortium, "Protein Data Bank: the single global archive for 3D macromolecular structure data," *Nucleic Acids Research*, vol. 47, no. D1, pp. D520–D528, Oct. 2018, ISSN: 0305-1048.

[86] wwPDB consortium, *wwPDB: File Format*, [Online; accessed 29 Nov. 2022]. [Online]. Available: http://www.wwpdb.org/documentation/file-format.

[87] J. Wang, "Estimation of the quality of refined protein crystal structures," *Protein Science : A Publication of the Protein Society*, vol. 24, no. 5, p. 661, May 2015. DOI: 10.1002/pro.2639.

[88] A. L. Morris, M. W. MacArthur, E. G. Hutchinson, and J. M. Thornton, "Stereochemical quality of protein structure coordinates," *Proteins Struct. Funct. Bioinf.*, vol. 12, no. 4, pp. 345–364, Apr. 1992, ISSN: 0887-3585. DOI: 10.1002/prot.340120407.

[89] D. W. Heinz, W. A. Baase, F. W. Dahlquist, and B. W. Matthews, "How amino-acid insertions are allowed in an α-helix of T4 lysozyme," *Nature*, vol. 361, no. 6412, pp. 561–564, Feb. 1993, ISSN: 1476-4687. DOI: 10.1038/361561a0.

[90] R. P. Joosten, F. Long, G. N. Murshudov, and A. Perrakis, "The PDB_REDO server for macromolecular structure model optimization," *IUCrJ*, vol. 1, no. 4, pp. 213–220, 2014, ISSN: 2052-2525. DOI: 10.1107/S2052252514009324.

[91] K. Mislow, "Molecular chirality," *Topics in stereochemistry*, vol. 22, pp. 1–82, 2009.

[92] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer International Publishing, Cham, 2015, pp. 234–241.

[93] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," in *International conference on medical image computing and computer-assisted intervention*, Cham: Springer International Publishing, 2016, pp. 424–432.

[94] *ConvTranspose3d - PyTorch 1.13 documentation*, [Online; accessed 26. Nov. 2022], 2022. [Online]. Available: `https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose3d.html`.

[95] M. Berman and M. B. Blaschko, "Optimization of the jaccard index for image segmentation with the lovász hinge," *CoRR, abs/1705.08790*, vol. 5, 2017.

[96] G. Chojnowski, J. Pereira, and V. S. Lamzin, *ARP/wARP 8.0*, [Online; accessed 28. Nov. 2022], 2022. [Online]. Available: `https://www.embl-hamburg.de/ARP/Manual/UserGuide8.0.html`.

[97] R. A. Nicholls, F. Long, and G. N. Murshudov, "Low-resolution refinement tools in refmac5," *Acta Crystallographica Section D: Biological Crystallography*, vol. 68, no. 4, pp. 404–417, 2012.

[98] G. J. Kleywegt, J.-Y. Zou, M. Kjeldgaard, and T. A. Jones, "Around o," in *International Tables for Crystallography Volume F*, John Wiley & Sons, Ltd, 2006, ch. 17.1, pp. 353–356, ISBN: 9780470685754. DOI: `https://doi.org/10.1107/97809553602060000691`. eprint:

https://onlinelibrary.wiley.com/doi/pdf/10.1107/97809553602060000691. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1107/97809553602060000691.

[99] J.-Y. Zou and T. A. Jones, "Towards the automatic interpretation of macromolecular electron-density maps: Qualitative and quantitative matching of protein sequence to map," *Acta Crystallographica Section D*, vol. 52, no. 4, pp. 833–841, 1996.

[100] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[101] J Springenberg, A. Dosovitskiy, T. Brox, and M Riedmiller, "Striving for simplicity: The all convolutional net," in *ICLR (workshop track)*, 2015.

[102] cheind, *py-lapsolver*, [Online; accessed 28. Nov. 2022], Nov. 2022. [Online]. Available: https://github.com/cheind/py-lapsolver.

[103] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[104] A. Tiessen, P. Pérez-Rodríguez, and L. J. Delaye-Arredondo, "Mathematical modeling and comparison of protein size distribution in different plant, animal, fungal and microbial species reveals a negative correlation between protein size and protein number, thus providing insight into the evolution of proteomes," *BMC Res. Notes*, vol. 5, no. 1, pp. 1–23, Dec. 2012, ISSN: 1756-0500. DOI: 10.1186/1756-0500-5-85.

[105] L. El Mortaji, C. Contreras-Martel, M. Moschioni, *et al.*, "The full-length Streptococcus pneumoniae major pilin RrgB crystallizes in a fibre-like structure, which presents the D1 isopeptide bond and provides details on the mechanism of pilus polymerization," *Biochem. J.*, vol. 441, no. 3, pp. 833–843, Feb. 2012, ISSN: 0264-6021. DOI: 10.1042/BJ20111397.

[106] Y. Li, Y. Iwamoto, L. Lin, R. Xu, R. Tong, and Y.-W. Chen, "VolumeNet: A Lightweight Parallel Network for Super-Resolution of MR and CT Volumetric Data," *IEEE Trans. Image Process.*, vol. 30, no. 4840-4854. Pp. ; 2021, ISSN: 1941-0042. DOI: 10.1109/TIP.2021.3076285. eprint: 33945478.

[107] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004, ISSN: 1941-0042. DOI: 10.1109/TIP.2003.819861.