Investigating the Potential of Incorporating Protein Language Models (pLMs) into ML/DL Approaches for Enhanced Prediction of Allosteric Sites in Proteins

by

Moaaz Ur Rehman Azhar Khokhar

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

August 31, 2023

Investigating the Potential of Incorporating Protein Language Models (pLMs) into ML/DL Approaches for Enhacned Prediction of Allosteric Sites in Proteins

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Moaaz Ur Rehman Azhar Khokhar

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Dr. Attila Gürsoy (Advisor)

_____

Assoc. Prof. Dr. Özlem Keskin (Co-Advisor)

_____

Assoc. Prof. Dr. Aykut Erdem

_____

Prof. Dr. Banu Ozkan

Date: _____

To my advisors, family, and colleagues…

ABSTRACT

Investigating the Potential of Incorporating Protein Language Models (pLMs) into ML/DL Approaches for Enhanced Prediction of Allosteric Sites in Proteins

Moaaz Ur Rehman Azhar Khokhar

Master of Science in Computer Science and Engineering

August 31, 2023

Allostery, the process by which binding at one site perturbs a distant site, is being rendered as a key focus in the field of drug development with its substantial impact on protein function. Allosteric drugs activate or inhibit proteins and offer advantages over non-allosteric drugs. However, the identification of allosteric sites is a challenging task due to unavailability of huge dataset, their distance, and lack of conservation across protein structures. A variety of computational techniques have been developed in the past to predict allosteric sites, such as Normal Mode Analysis (NMA), Molecular Dynamics (MD), and Machine Learning (ML), utilizing both static pocket characteristics and the dynamics of proteins; the performance of these methods needs further improvement. This research investigates the potential of incorporating Protein Language Models (pLMs) into ML and/or DL approaches to improve prediction of allosteric residues, based on the fact that the pLMs (e.g., ProtBERT from the family of ProtTrans pLMs: based on BERT architecture) effectively capture the spatial relationship among residues, eventually contributing to identification of allosteric sites/pockets. ProtBERT-BFD (ProtTrans) was fine-tuned on the Allosteric Dataset (ASD) of protein sequences, which predicts the allosteric residues with an F1 score of 61.54% on the test dataset. Several ML and DL approaches were utilized including XGBoost, SVM, AutoML, and GNNs. With the inclusion of fine-tuned pLM features, all of the aforementioned approaches improve the prediction performance of allosteric sites over previous studies by a considerable margin. XGBoost, being the highest performing model in this study, improves the results by combining the features extracted from finetuned ProtBERT with pocket features extracted by FPocket, resulting in an F1 score of 75.76% for allosteric pockets/sites. Case studies have been performed on proteins with known allosteric sites in addition to the case study to predict novel allosteric sites on new proteins.

ÖZETÇE

Proteinlerdeki Allosterik Bölgelerin Gelişmiş Tahmini için Protein Dil Modellerini (pLM'ler) ML/DL Yaklaşımlarına Dahil Etme Potansiyelinin Araştırılması

Moaaz Ur Rehman Azhar Khokhar

Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans

31 Ağustos 2023

Allosteri, proteinin bir bölgesindeki değişikliğin, mesela başka bir moleküle bağlanmanın, proteinin uzak bir bölgesini etkilediği süreç olarak tanımlanabilir. Allosteri protein fonksiyonu üzerindeki önemli etkisi sebebiyle ilaç geliştirme alanında önemli bir odak noktasıdır. Allosterik ilaçlar proteinleri aktive veya inhibe edebilir, allosterik olmayan ilaçlara göre avantajlar sunar. Bununla birlikte, allosterik bölgelerin tanımlanması zorlu bir iştir. Geçmişte allosterik bölgeleri tahmin etmek için Normal Mod Analizi (NMA), Moleküler Dinamik (MD) ve Makine Öğrenimi (MÖ) gibi hem statik cep özelliklerini hem de proteinlerin dinamiklerini kullanan çeşitli hesaplama teknikleri geliştirilmi olmakla birlikte bu yöntemlerin performansının daha da geliştirilmesi gerekmektedir. Bu araştırmada, pDM'lerin (örneğin, ProtTrans pDM ailesinden BERT mimarisine dayalı ProtBERT'in) allosterik kalıntıların tahminini iyilrştirmek için Protein Dil Modellerini (pDM'ler), MÖ ve/veya DÖ yaklaşımlarıyla birlikte kullanılma potansiyelini araştırılıyor. Tezde, amino asitler arasındaki mekansal ilişkiyi etkili bir şekilde öğrenerek, sonuçta allosterik alanların/ceplerin tanımlanmasını hedeflenmektedir. ProtBERT-BFD (ProtTrans), test veri kümesinde %61,54'lük bir F1 puanıyla allosterik kalıntıları tahmin eden protein dizilerinin Allosterik Veri Kümesine (AVK) göre ince ayar yapılmıştır. XGBoost, SVM, AutoML ve GNN'ler dahil olmak üzere çeşitli MÖ ve DÖ yaklaşımlarından yararlanılmıştır, İnce ayarlı pDM özelliklerinin dahil edilmesiyle, yukarıda belirtilen yaklaşımların tümü, allosterik bölgelerin tahmin performansını önceki çalışmalara göre önemli bir farkla artırdığı bulunmuştur. Bu çalışmada en yüksek performansa sahip model olan XGBoost, ince ayarlı ProtBERT'ten çıkarılan özellikleri FPocket tarafından çıkarılan cep özellikleriyle birleştirerek sonuçları iyileştiriyor ve allosterik cepler/bölgeler için %75,76'lık bir F1 puanı erişmektedir. Bilinen allosterik bölgelere sahip proteinler üzerinde örnek çalışmaların yanı sıra, farklı proteinler üzerindeki yeni allosterik bölgeleri tahmin etmek için de çalışmalar yapılmıştır.

AKNOWLEDGEMENTS

First and foremost, I would like to express my heartfelt and undying gratitude to my supervisors Prof. Attila Gürsoy and Prof. Özlem Keskin for their unwavering support and guidance throught my master studies and this research. Their insights and their way of guidance has been a motivating factor for me. Prof. Attila Gürsoy, with his background in computer science and his transformative journey into computational biology, has influenced me highly to put my efforts in this field of study.

I am privileged to have shared this journey with remarkable colleagues in the lab/office. While each of them has played a unique role in my academic development, I feel compelled to extend special gratitude and pay respects to Damla Övek and Zeynep Aydın. Damla's intelligence and Zeynep's dedication have consistently inspired me; their unwavering readiness to assist, no matter the circumstance, has been a backbone for my academic journey at Koç.

I would like to mention that I have leveraged ChatGPT to understand several concepts and complex theories given in the literature. Although, as the best of my knowledge and efforts, I have ensured the originality of my thesis, however, there might be instances where its influence is obsevable.

I would like to thank the GSSE department for giving me the opportunity to be a part of this exciting research. I am also deeply grateful to KUIS-AI for funding my studies and research at Koç University, providing me the opportunity to learn the field of AI, and last but not least providing me with the fellowship that allowed me to run huge DL models on AI nodes in KUACC cluster.

I thank my committee members for their valuable time and upcoming feedback.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ASD	AlloSteric Database
BERT	Bidirectional Encoder Representations from Transformers
BFD	Big Fantastic Database
DL	Deep Learning
GCNN	Graph Convolutional Neural Network
GNN	Graph Neural Network
HMM	Hidden Markov Model
LLM	Large Language Model
MD	Molecular Dynamics
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSA	Multiple Sequence Alignment
NMA	Normal Mode Analysis
NMR	Nuclear Magnetic Resonance
pLM	Protein Language Model
SCA	Statistical Coupling Analysis

Chapter 1:
INTRODUCTION

"There is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order of things."

— N. MACHIAVELLI

Since the dawn of time mankind hath sought to decode human biology and inner workings of a human building blocks i.e., cells. For this purpose, they have created several physical and computation tools that have been continuously helping them in decoding and understanding the processes in an organism's cell. One of the most important processes in cells is *"Allostery"* that is also referred to as the "second secret of life" [1] [2]. According to Fenton [1], although allostery does not have an agreed-upon definition, however, all definitions agree on common features of energetic coupling and protein structural changes. On a very literal level, it is depicted as "*energetic coupling between two binding events*" [3] [4] [5]. Allostery, however, commonly being referred to as regulation at distant sites [6]; this effect takes place when a certain perturbation at a distant site of a protein that is structurally distant from the orthosteric functional site of the protein and consequently modulates the activity of that protein [7] [8]. This perturbation can either be external (ligand binding) or internal (mutation) [9]. Allosteric regulation is perceived to have the possibility to control gene expression, as well as regulate the synthesis of hormones, neurotransmitters, and various other molecules [10] [11] [12] [13] [14]. It has been suggested by some researchers that every protein possesses allosteric behavior. Even if a certain protein has not yet exhibited allosteric behavior, it could be because of the absence of right conditions such as allosteric effectors or certain mutations [15] [16]. If this proposition holds, it opens opportunities to focus attention on understanding complexity of protein behavior and allosteric effectors. As it is aimed to gain more insight into the mechanisms of life and treatment of diseases, it becomes pivotal to perform quantitative analysis of the complex phenomenon of allostery [17] [18] [19].

Recent advances in experimental methods such as X-Ray Crystallography and Nuclear Magnetic Resonance (NMR) Spectroscopy [20] have enabled structural studies of proteins at atomic resolution however, these methods are expensive and time-consuming. To overcome this problem, several computational methods have been

developed that leverage Molecular Dynamics (MD) [21] simulation, Normal Mode Analysis (NMA) [22], and Machine Learning (ML) [23] [24] [25] techniques. These methods have enabled prediction of allosteric sites in proteins with significant accuracy. Out of these methods, ML models have been the most successful ones and recent studies have implemented ML models for the purpose of predicting allosteric sites in proteins. Although, these methods provide significant results, they have not leveraged the features of pre-trained Protein Language Models (pLMs) or Protein Large Language Models (pLLMs).

This work leverages the pLM from the family of ProtTrans [26] based on the fact that the pLMs effectively capture the spatial relationship among residues. ProtTrans provides state of the art pre-trained language models for proteins. These models have been trained on different sizes of chunks of protein sequences from UniProt [27] database. These protein databases were clustered into Uniref50 [28], Uniref100 [28], and Big Fantastic Database (BFD) [29] [30]. Most of the models in this family are based on Bidirectional Encoder Representations from Transformers (BERT) architecture, hence named ProtBERT.

In this work, ProtBERT model was finetuned on protein sequences obtained from the ASD database [31]. It was finetuned for the task of token (residue) classification. It classifies whether a residue is allosteric (positive) or non-allosteric (negative). This model generates a 1024-D feature vector for each residue and these features were further used in combination with pocket features (19-D vector) extracted from FPocket [32] in order to predict whether a pocket is allosteric or non-allosteric. These features were fed into different ML and DL models including XGBoost [33], SVM [34], a collection of AutoML [35] models, Multi-layer Perceptrons (MLPs) [36], and Graph Neural Network (GNN) [37]. Training these models by including the pLM features, increased the prediction performance of allosteric pockets and surpassed the previous prediction results. These features were also used to investigate prediction performance of the techniques in the previous studies [38] [39] [40] [41], whether the pLM features increase the prediction performance or decrease it. Several hyperparameters were tried on different models. All models have the finetuned pLM as the backbone for feature extraction; they perform binary classification on pockets whether a pocket is allosteric or not by combining the powerful features of the pLM. These models outperformed the same previous techniques, and our highest performing model is XGBoost which gives an F1 score of 75.76%.

Chapter 2 gives the literature review of studies focusing on predicting allosteric sites in proteins. Tools and techniques mentioned in the literature review cover the NMA, MD, ML, and DL approaches.

In chapter 3, methodology of the approach is given with detailed explanation of the algorithm with implementation details. Finetuned pLM is used in this section to extract sequence features which are then combined with pocket features (in the case of GNN, pocket features were not used), and fed into several ML and DL models. These models are trained on these features and predict whether a given pocket is allosteric or non-allosteric.

Chapter 4 discusses the results and comparison of different techniques under different hyperparameter conditions. To further evaluate the model, its performance was evaluated by ranking prediction scores, resulting in predicting an allosteric site in top 3 positions with 95% confidence. Case studies show that the highest performing model can predict novel allosteric sites.

Finally, chapter 5 concludes this study by providing lessons learnt and provides insights into probable future work.

Chapter 2:
LITERATURE REVIEW

"If I have seen further [than others], it is by standing on the shoulders of giants."

— I. NEWTON

Nobuhiro Gō's model [42] of protein folding is stated as base and relating factor to allostery by Wang et. al. [43]. This model, as summarized by Wang et al., is based on the proposal that in order to find the folding pathway of a protein, distance and closeness of nearby residues/amino acids is an important factor; leading to a perception that native protein structure and amino acid sequence has important role in folding pathway. They add that this concept is linked with allostery as perturbation at one site results in changes at another site in the same protein. In recent years, there has been a huge amount of research addressing various aspects of protein allostery including but not limited to detection of allosteric sites in proteins as evidenced by the literature available and discussed in this chapter. One of the studies conducted by Haliloglu et al. showed the utilization of Hidden Markov Models (HMMs) to predict pathways through which allosteric communication may be performed [9]. This approach might as well be used to identify and consequently verify and validate allosteric residues predicted by computational methods. Understanding the existing research articles on protein allostery is essential to grasp the current state of knowledge and identify research gaps.

To predict allosteric sites, several methods and techniques have been developed ranging from Normal Mode Analysis (NMA) [22], Molecular Dynamics (MD) simulations [21], Machine Learning (ML) models [23] [24] [25], and combination of these [44] [45]. Some of the methods, based on previous mentioned techniques, are available as webservers or open-source packages. Allosite [25], SPACER [46], PARS [47], AlloPred [44], AllositePro [45], PASSer (and with its different variants) [41] are some of the examples of these webservers. The aforementioned models demonstrate the potential of combining pocket features and protein dynamics in predicting allosteric sites. According to the summary provided by Lu et al. [48], the studies in question can be categorized into sequence-based, structure-based, dynamics-based, NMA-based, or combined prediction approaches. Recently, ML has been heavily utilized in chemistry and biology [49] [50] [51] [52]; and it has been proved to be better than anachronistic

non-ML approaches for the classification of allosteric sites in proteins. Some of these approaches are discussed in the following subsections.

2.1 Sequence-based Prediction Approaches

2.1.1 Statistical Coupling Analysis

Statistical coupling analysis (SCA) [53] is a technique that utilizes Multiple Sequence Alignment (MSA) to measure and quantify coupling energy. For example, it checks how much an amino acid's distribution is changed on perturbation from another amino acid at another position. It discovers the sections of evolving amino acids and gives an energy value that shows the dependence of amino acids. Communication between an allosteric site and active site is highly dependent on these sections. Hence, allosteric sites can be predicted by pinpointing these active sites that are interacting with the sections mentioned above. This theory is reinforced by the experiments of Ranganathan et. al on PDZ domain family, verified by mutational analysis [54] [55].

2.2 Network-based Prediction Approach

2.2.1 Ohm

Ohm [43] was developed to analyze allostery in proteins which identifies allosteric sites, residues, pathways, and correlation between pairs of residues. Ohm was built and tested solely on known protein structures that compares its predictions with experimental data. The verified results, as assumed, effectively map the dynamic interactions within proteins without using any expensive experimental methods. The team designed a perturbation propagation algorithm to calculate the changes. The algorithm builds a probability matrix by calculating the likelihood of contact between pairs of residues. A certain threshold value is used to decide whether to propagate or not. With multiple iterations, allosteric sites and pathways are discovered hence providing an understanding on how changes at one site of a protein influence changes in other sites of protein.

2.3 Normal Mode Analysis-based Prediction Approaches

NMA is another technique which can help in investigation of protein dynamics and function. NMA not only helps in understanding functional dynamics but can also help in identifying certain areas that are responsible for transferring signals in the protein. Some of the methods employing NMA are discussed in this section.

2.3.1 PARS

Panjkovich and Daura have developed a web server named Protein Allosteric and Regulatory Sites (PARS) [47]. PARS makes allosteric sites predictions that are based on how a protein flexibility changes when a ligand binds to the protein. The authors provide the steps of their algorithm that starts by uploading a PDB file and choosing of chains by the user. Next, LIGSITE^{cs} identifies ligand-binding site in the protein. Subsequently, NMA is performed on the protein without a bound ligand (also called apo protein) and for potential ligand-binding site in the protein-ligand complex. After this, the algorithm compares the B-factor derived by NMA by comparing difference between unbound and the ligand-bound states. If the difference crosses a threshold (set by the authors), the ligand-binding site is considered as allosteric site. Results from this model were generated by a benchmark dataset of 102 allosteric proteins and overall, 73% of the sites could be found within the top three positions in the PARS ranking.

2.4 Dynamics-based Prediction Approaches

2.4.1 Molecular Dynamics Simulations

Molecular dynamics (MD) simulations are considered to provide accurate conformational changes in explicit solvent environments [56]. As stated by Shan et al. [57], a detailed process of drug binding can be identified by the MD simulations without having any prior knowledge about the protein binding site. Consequently, rendering this approach valuable for identifying protein allosteric sites. Studies by Dror et al. [58] and Shukla et al. [59] show the usage of MD simulations to detect allosteric sites. In the study by Dror et al., MD simulations were able to locate allosteric sites in M2 receptor by placing a single allosteric modulator away from the receptor, allowing the modulator to bind to naturally to the receptor without any external factors. Same for the Shukla et al.

that they were able to predict allosteric sites by running MD simulations in order to analyze the pathway of conformational transition in the c-Src tyrosine kinase. They were able to detect inactive and active states of the kinase pathway that would suggest that MD simulations can detect allosteric sites in a certain state of the aforementioned kinase.

2.4.2 Two-State Gō Model

Qi and team [60] developed a methodology, known as the coarse-grained two-state Gō model, to predict allosteric sites on proteins. As obvious by the name of the model, authors created an ensemble of two functional states of protein and slanted the energy towards one state. After this, they introduced perturbation at a binding site and observed the effect at another site. The authors were able to identify new allosteric sites in the proteins from the test dataset. Moreover, using the proposed technique they were able to identify allosteric sites on E. coli phosphoglycerate dehydrogenase (PGDH). This is one of the old methods introduced in 2012 that utilized MD simulations.

2.5 Combined Dynamics and NMA-based Prediction Approaches

2.5.1 SPACER

SPACER is a web-based tool written in Python and developed by Mitternacht and Berezovsky [46] that identifies allosteric sites in proteins. It is based on two features that are “local closeness” and “binding leverage” [61]. Local closeness deals with the static and geometric features while binding leverage is based on protein dynamics. To identify potentials binding sites, SPACER utilizes Monte Carlo simulations to scan the protein’s surface. After this step, for each predicted site, the algorithm evaluates and measures the strain on the ligand-protein contacts; a high strain is considered to have more “binding leverage” on the protein structure.

2.6 Combined NMA and Machine Learning-based Prediction Approaches

This section lists the approaches that use ML techniques or combine ML techniques with NMA. Most of these tools’ flow starts from an input PDB file that contains the structure of the protein. A geometry-based algorithm, FPocket [32], is used to extract pockets from the 3D structure of the given protein. Subsequently, these pockets are fed

into ML or ensemble (combination of ML and NMA) pipeline, which results in classifying pockets into allosteric or non-allosteric.

2.6.1 Allosite

Allosite [25] is a webserver that was designed to predict allosteric sites in proteins. Allosite leverages techniques such as pocket-based analysis and a support vector machine (SVM) classifier [34], trained on a selected high-quality dataset (filtered proteins better than 3 Å resolution). It identifies pockets and trains the features by an SVM classifier to predict allosteric sites. Allosite, on available test dataset at that time, has shown a high accuracy of 95%, and proved to be useful in predicting novel allosteric sites in proteins. Having performed further evaluations, it showed 92% accuracy in 5-fold cross-validation for all three training sets and mutations were detected in the allosteric sites that affected the orthosteric sites in proteins.

2.6.2 AlloPred

AlloPred [44] was developed to leverage NMA for predicting the allosteric pockets in a protein. This method surpasses standard methods by incorporating NMA and how a modulator at a site can introduce change in protein's dynamics.

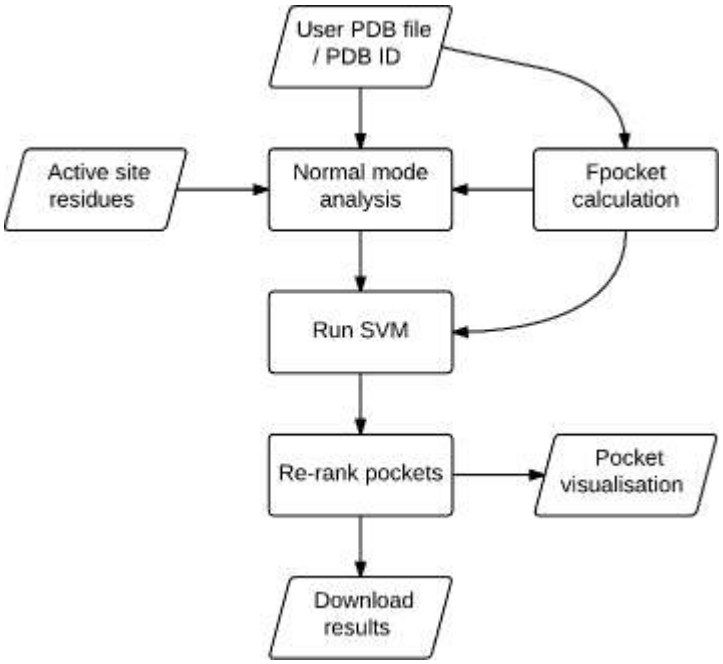

Figure 2.1: AlloPred Pipeline [44]

The AlloPred method utilizes the Fpocket algorithm, which identifies potential binding pockets, in combination with an elastic network model to understand the normal modes of a protein. Furthermore, containing in a certain pocket, the algorithm tries to tweak a spring constant of any atom pair and measures the effect of this perturbation at the active site. These results are then combined with the Fpocket output in an SVM to predict allosteric pockets on proteins. Figure 2.1 gives the pipeline of AlloPred that starts with a PDB file which is fed into FPocket to extract pockets. These pockets are used by NMA and SVM modules which are trained to predict pockets as allosteric or non-allosteric. The data used for this research was sourced from ASBench, a benchmarking set for allosteric sites prediction.

2.6.3 Random Forest Model for Allosteric Sites Prediction

Chen and team [62] used Random Forest (RF) [63] in order to build a three-way model to predict allosteric sites in proteins by using available crystal data in the PDB. The research leverages non-cognate bound ligand molecules that are often found in protein crystal structures in order to generate descriptors for the RF model. Objective here was to differentiate between an allosteric site and a binding site. With their analysis, they collected the data with three kinds of sites based on function: active sites, allosteric sites, and other protein clefts. Random Forest, being an ensemble of stochastically built decision trees, takes advantage of "out-of-bag" (OOB) data, was used for classification. The resulting model was then used to predict the types of sites to identify potential allosteric sites.

2.6.4 AllositePro

AllositePro [45], that combines structural features and NMA, was developed to predict the of allosteric sites in proteins. Results from the authors show that it can predict 52% and 63% known allosteric sites in training and testing datasets, respectively. Moreover, to validate their model models, authors identified (previously undiscovered allosteric sites) allosteric sites in the Cyclin Dependent Kinase 2 (CDK2) protein.

ASBench [64] was used to train and develop AllositePro. Fpocket [32] was utilized to detect binding pockets on protein surfaces. These detected pockets were then analyzed by a logistic regression model. The evaluation was performed using the Matthews

Correlation Coefficient (MCC). The performance of AllositePro outperformed other methods, showing a 28.2% improvement in prediction compared to Allosite [25] and PARS [47].

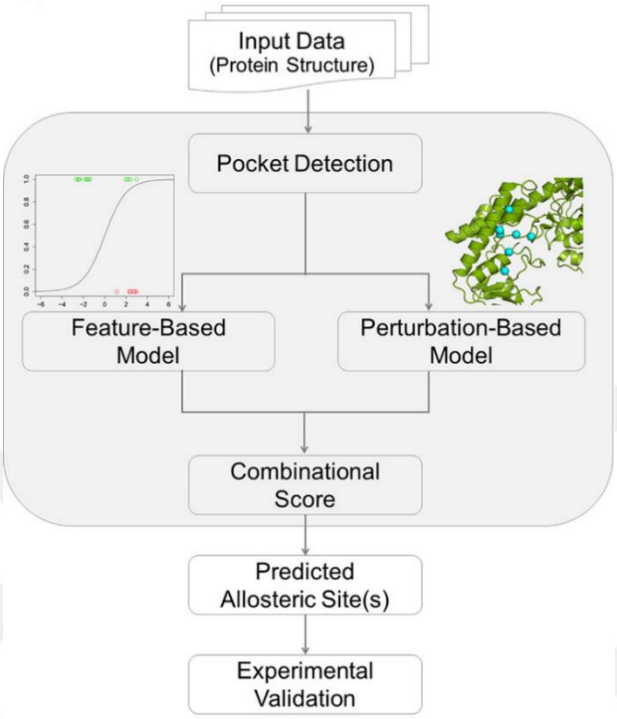

Figure 2.2: AllositePro Pipeline [45]

2.6.5 PASSer

PASSer [38] provides an ensemble learning method to predict allosteric sites in proteins by combining XGBoost [33] and GCNN [37]. It leverages physical properties and topology information of pockets. The resultant ensemble model shows improved performance over individual XGBoost and GCNN models. ASD dataset was used for training. After performing preprocessing steps, 90 proteins were selected. The algorithm flow starts by inputting a PDB file to PASSer. FPocket was used to extract pockets from the provided protein structure, with pockets labeled as positive or negative. This labeling was decided by the presence of at least one residue binding to allosteric modulators. The ensemble model, utilizing GCNN and XGBoost, leveraged topological features by the GCNN and physical properties by the XGBoost. The resultant ensemble model increased the F1 score of pockets' prediction over previous studies. The authors mentioned that a correct allosteric pocket could be placed in top 3 detected pockets in a protein with 84.9%

probability. These prediction results are comparable and better than those or other studies, however, size of the dataset should be considered.

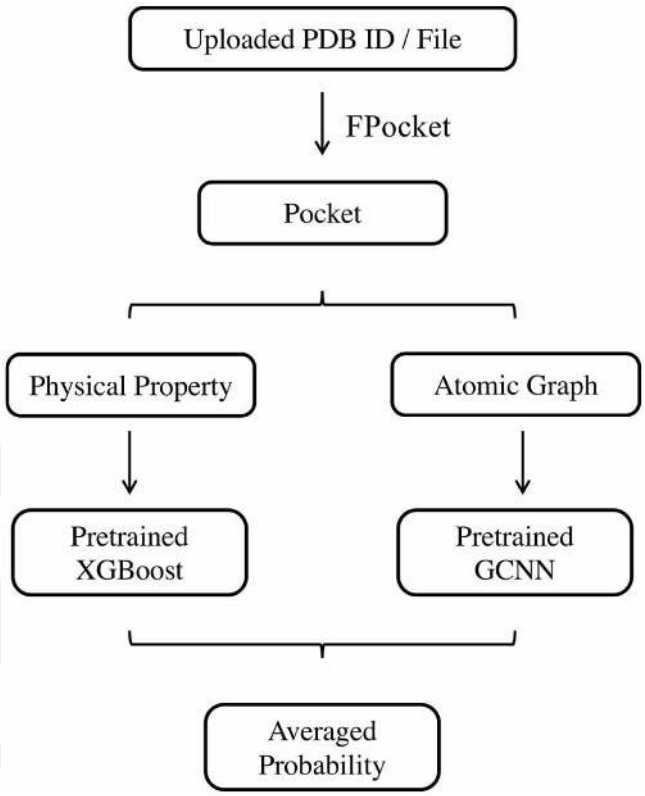

Figure 2.3: PASSer Ensemble Flow [38]

2.6.6 PASSer 2.0

This tool [39] is the next version of PASSer [38] that utilizes AutoML approach focusing on enhancing the prediction of protein allosteric sites. Their baseline model gave 0.624 F1 score. They made further improvements using two AutoML frameworks, AutoKeras [65] and AutoGluon [66], resulting in precision, recall, and F1 score values of 0.850, 0.616, and 0.701 respectively. Additionally, this updated model ranked 82.7% of allosteric sites in the top three positions for test data and proved to be promising for novel allosteric sites predictions. The data used in this study was ASD and the ASBench selection of ASD data, comprising of 204 proteins. Again, FPocket was used to identify protein pockets and calculate 19 numerical features for each. For each protein, pockets were labeled as either positive (allosteric) or negative (non-allosteric). Automated machine learning (AutoML) using Keras and AutoGluon was applied. AutoKeras based on Keras, applied Bayesian optimization for efficient neural architecture search, while

AutoGluon from Amazon Web Services (AWS) automated ML tasks for best performance. Finally, the authors also utilized PocketFinder [67] for allosteric site prediction to compare results with pockets extracted by FPocket.

2.6.7 PASSerRank

As outlined by the authors, previous studies attempted to build universal models that make definitive and binary predictions for all detected pockets in proteins. PASSerRank [40] utilizes the Learning to Rank (LTR) method that was originally applied in information retrieval [68]. Rather than making absolute or binary predictions, LTR models make relative predictions by ranking the samples (pockets) according to the relevance to the target output. LambdaMART implements the LTR by combining gradient boosting trees and LambdaRank as loss function. According to their study, LambdaMART can give better results than XGBoost, SVM, RF models by giving high F1 scores. The authors utilized the ASD and CASBench [69] database to train and validate different models. Preprocessing steps were same as those of used in PASSer and PASSer2.0. These steps ensured high data quality and eliminated low resolution and redundant structures. FPocket was used to identify potential pockets in each protein and calculate physical and chemical features for each detected pocket. Following the approach of PASSer, PASSerRank fed these pockets into the LTR model to rank allosteric pockets. The results, although are inferior to ensemble model [38], suggest that ranking pockets in relation to one another, rather than building a universal model to perform absolute predictions, is a more effective approach to allosteric site prediction.

Chapter 3:
METHODOLOGY

"What we observe is not nature in itself, but nature exposed to our method of questioning."

— W. HEISENBERG

This chapter is devoted to a comprehensive explanation of the research design and methodology, providing detailed methodology followed during the data collection and analysis stages of the study. Aforementioned approaches in the Literature Review section employ several techniques, of which, ML and DL approaches are highly significant and provide better results compared to non-ML approaches. Although, these approaches can predict allosteric sites, they do not leverage the pLM features. pLMs have been of great interest in recent years and several pLMs, trained on protein sequences, have been leveraged in predicting protein-protein interactions [70] [71] in combination with GCNNs [37] and/or GAT [72] models.

The focus of this research is to incorporate pLM's features into ML/DL approaches and investigate the impact of inclusion of these features on the prediction performance of allosteric sites. As mentioned in previous studies [41], ASD dataset was used to collect allosteric proteins. Furthermore, ProtBERT pLM – based on BERT architecture [73] – from the family of ProtTrans [26] was finetuned on proteins sequences from the ASD dataset. The pLM gives a 1024-d vector for each residue in a protein sequence and these feature vectors, combined with 19-d pocket feature vectors extracted by FPocket, were fed into ML/DL models for training and prediction of allosteric sites in proteins.

3.1 Dataset

AlloSteric Database (ASD)[1] is a collection of allosteric proteins, allosteric interactions, allosteric potential sites, and allosteric modulators. This database is updated annually and freely available for researchers to investigate allosteric regulation and allosteric drug discovery. At the time of this research, ASD contained 1944 allosteric protein entries which were downloaded and preprocessed. Same preprocessing steps were applied as mentioned in the previous studies [41] and there is a Python script[2] available

[1] https://mdl.shsmu.edu.cn/ASD/
[2] https://github.com/smu-tao-group/PASSerRank

to automate the preprocessing stage, that is believed to be the first of its kind to automate preprocessing stage [40]. However, some changes were made in the available script in order to cater for data acceptance by the pLM. Moreover, the filtration criteria were made stricter for the alignment of protein sequence to be more stringent. Following steps mention preprocessing steps in detail:

1. Proteins were downloaded in the PDB format from Protein Data Bank.
2. PDBs were dropped that had modulators in multiple distinct chains.
3. PDBs were dropped that had different modulators.
4. Only those residues were selected that were in the same chain of modulator.
5. Structure was dropped if the resolution value was higher than 3 Å.
6. Pockets were extracted by FPocket.
7. Each pocket's distance was calculated by the modulator in the HETATM section of the PDB file. A 10 Å threshold was used such that a certain protein was dropped if the distance was more than 10 Å between the center of mass of modulator and center of mass of the nearest pocket.

$$\text{keep protein if dist}(Modulator, NearestPocket) < 10 \text{ Å}$$

8. In previous studies, 30% sequence identity criterion was used. However, in this research 25% sequence identity criterion was used. If a sequence was more than 25% similar, it was dropped.
9. To mark a pocket as allosteric following two criteria were used:
    - Nearest pocket to the modulator was marked as allosteric.
    - As provided in ASD dataset, if a pocket contained at least one allosteric residue, it was marked as allosteric.

Aforementioned steps were performed to build a clean base dataset which resulted in 603 proteins. Moreover, pockets extracted by FPocket sum to 12,486, collectively. On average, ~21 pockets were detected in each protein. These pockets were saved in the PDB file format where each file/pocket contained relative residues with their X, Y, Z coordinates. 603 proteins sequences dataset was used to train, validate, and test the pLM while the pockets were used to train, validate, and test the ML/DL models. Protein sequences and pockets were split into 80:12:8 splits for train:validation:test datasets. Table 3.1 gives the breakdown of train, validation, test dataset splits.

Table 3.1: Dataset Split

DATA SPLIT	PROTEINS	POCKETS
Train	482	10,008
Validation	72	1373
Test	49	1105

There was a huge class imbalance: positive samples accounting to only 6.61% of the dataset. Figure 3.1 gives the counts of positive and negative pocket samples.

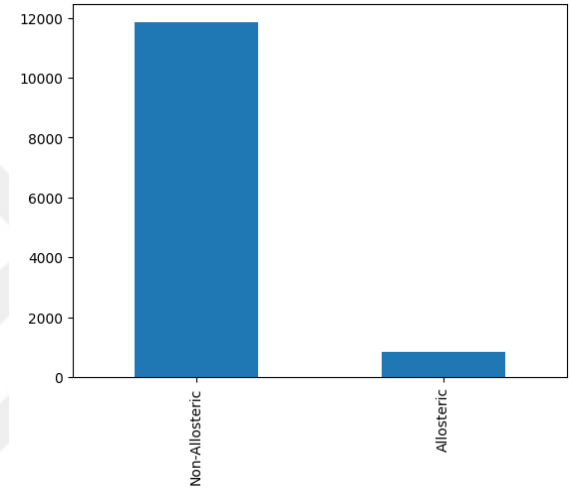

Figure 3.1: Counts of Allosteric and Non-Allosteric Pockets

Further, preprocessing steps were applied on the dataset for different approaches mentioned later in this chapter.

3.2 ProtBERT-BFD pLM

ProtBERT-BFD is a pretrained LM from the family of ProtTrans [26] language models. It is based on the BERT model/architecture and trained on large corpus of protein sequences in a self-supervised manner. ProtBERT has an important difference from the original BERT model that it deals sequences as separate documents. It means that the "next sentence" prediction was not used as each sequence is considered as a complete document. However, masking process follows the original BERT masking process i.e., to mask 15% of amino acids randomly, while training. ProtBERT-BFD was trained on BFD[3] database that is the biggest database comprised of 2.1 billion protein sequences, hence

[3] https://bfd.mmseqs.com/

postfixed with BFD. This model's layers and attention heads were increased to 30 and 16 by the authors, respectively. This model gives a 1024-d feature vector for each residue in a protein.

The authors of this model mentioned that the model was able to accurately represent important physical features of proteins by the LM-embeddings. Although, proteins' dataset was unlabeled, the model learnt the structure of the proteins just like as a natural language model can understand grammatical structure of a sentence in a natural language by learning a lot of sentences or protein sequences.

This model was finetuned on ASD dataset in order to be used for pocket prediction tasks by other ML/DL models.

3.2.1 Dataset Preparation

The same processed dataset was used that is mentioned in the Dataset section. However, some further preprocessing steps were performed to fine-tune the LM. These steps are as follows:

1. Each residue in an allosteric pocket was labelled as allosteric. Hence, each residue in a protein sequence was labelled as either positive or negative referring to being allosteric or non-allosteric, respectively.
2. Each residue in each sequence was separated by *space* character, as the expected input of ProtBERT model.

3.2.2 Finetuning of ProtBERT-BFD

Each sequence was truncated to a maximum of 1024 length. Flow of finetuning starts by inputting a protein sequence, as shown in Figure 3.2. Model was finetuned as token (residue) classification task.

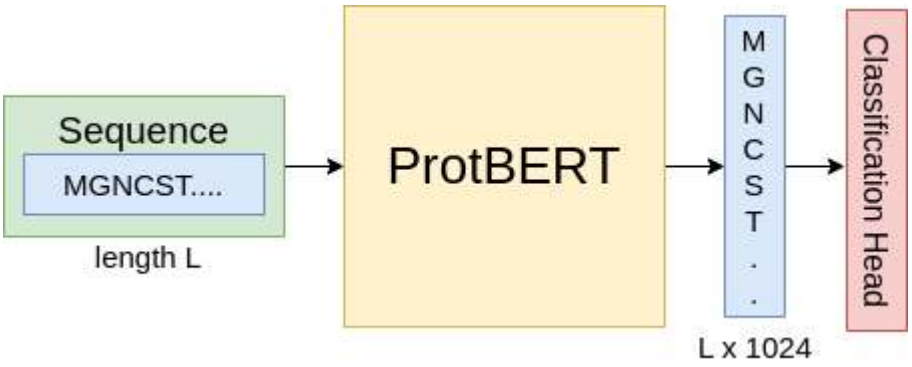

Figure 3.2: ProtBERT Finetuning Flow

It was finetuned on Tesla v100 GPU against the training dataset mentioned in the dataset section and evaluated on validation dataset against F1-score. As the dataset has high class imbalance, weighted Cross Entropy loss function (given below) was used by giving more weight to positive class.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

The model was trained on the following parameters:

Table 3.2: ProtBERT Training Parameters

PARAMETER	VALUE
Epochs	16
Batch Size	4
Learning Rate	$2e^{-04}$
Weight Decay	$1e^{-1}$
Evaluation Metric	F1 Score
Optimizer	AdamW

The pretrained model was downloaded from Huggingface[4]. Huggingface's TokenClassification and Trainer classes were used to create token classifier and train the model, respectively.

As in the data preparation step, all residues in a pocket were labelled as positive, a threshold of 8 residues was used. Having said that, at inference time, from the model's predicted residues; these residues were mapped to the pockets by the residue indices. If a

[4] https://huggingface.co/Rostlab/prot_bert_bfd

pocket contained at least 8 (model's predicted output) allosteric residues, this pocket was considered as allosteric. This threshold was chosen by maximizing the performance of validation dataset by considering *number of allosteric residues in a pocket* as a hyperparameter. Values ranging from 4 to 12 were tried and the value of 8 gave the highest performance.

This finetuned model gives 1024-d feature vector for each residue in a sequence. These feature vectors were further used for training of ML/DL models that directly trained on pockets to predict whether a pocket was allosteric or non-allosteric.

3.3 XGBoost

XGBoost stands for Extreme Gradient Boosting is a supervised ML method that uses an ensemble of decision trees and gradient boosting to make predictions. A single tree in XGBoost performs prediction on a sample (x_i, y_i) as:

$$g(x_i) = w.x_i$$

where w is the weight matrix or vector of a leaf. More information on XGBoost is given in Appendix A.1.

XGBoost accumulates and combines new trees iteratively and calculates the objective function. It works as Newton-Raphson[5] method with first and second order derivatives in loss function by applying Taylor expansion on objective function.

This model was trained on pockets dataset. Each pocket's feature was concatenated with the LM features such that only those LM residue features were used for a certain pocket that were part of that pocket. All residue features of a pocket were aggregated into one 1024-d feature vector and concatenated with pocket feature vector of size 19 resulting in a 1043-d vector.

For XGBoost, no normalization was performed as it does not require normal distribution. Although our dataset is highly imbalanced, XGBoost can perform quite well even if the dataset is imbalanced.

XGBoost was trained by using binary logistic function as objective function with a regularization $\lambda = 0.2$. To overcome the class imbalance problem, *"scale_pos_weight"* was set to the ratio of negative samples to the positive samples. This model was trained

[5] https://en.wikipedia.org/wiki/Newton%27s_method

for 60 iterations with 10-fold cross validation strategy, hence the *scale_pos_weight* parameter was calculated dynamically for each fold.

3.4 Automated Machine Learning (AutoML)

While developing ML models, it takes a lot of time and expertise. This process starts by preparing and preprocessing the dataset. Next, it is often required to perform feature analysis, selection, and extraction from the data. This process does not stop here and continue to further evaluation (including k-fold or other cross validation techniques) and selection of the model with best hyperparameters. All these tasks are time consuming and challenging. AutoML makes all these steps easier for researchers in order to find the best model for a task. It performs these steps behind the hood and tries several models from its library with different hyperparameters for a given task.

AutoGluon [66] is one of the AutoML libraries that was developed by Amazon Web Services (AWS). It automates the ML tasks and provides an easy way to train models for a problem. In order to reduce variance while training, AutoGluon utilizes multi-layer stacking with k-fold bagging. These layers and value of k is automatically selected by the framework while training. Moreover, it provides fault tolerance facility by saving models at different checkpoints. It does not require any feature engineering or data preprocessing. It automatically selects the best features from the data.

Same dataset was used as mentioned in XGBoost section. Dataset was comprised of LM features combined with pocket features and the AutoML framework was trained to perform binary classification on pockets. 14 models were trained including Random Forest, XGBoost, LightGBM, KNN, Neural Network, and Weighted Ensemble. These models were trained with default parameters and the models' performance was significantly good.

3.5 Support Vector Machine (SVM)

SVM is a supervised ML algorithm that is primarily used for classification but also for regression problems. Allosite [25], AlloPred [44], and AllositePro [45] have utilized SVM (solely or in combination with NMA) to predict allosteric sites in proteins.

In classification, SVM works by finding a hyperplane in an N-dimensional space (where N is the number of features) that distinctly classifies the data points. It finds the

decision boundary that maximizes the distance (margin) from the nearest data points of all classes. These data points are known as support vectors, as they help determine the hyperplane or decision boundary. In its simplest from, a hyperplane can be written as:

$$w^T x - b = 0$$

where w is a normalized vector to the hyperplane and b is bias parameter.

For non-linearly separable data, SVM uses *kernel-trick* to transform the input space into a higher dimensional space where the data can be separated linearly. Some of the functions that are used in SVM are linear, polynomial, radial basis function (RBF), and sigmoid.

SVM was used to predict pockets of the same dataset as mentioned in earlier approaches by combining LM features. Data was normalized by gaussian distribution and RBF kernel was used. The distribution of RBF kernel is as a bell-shaped graph hence it works better if the data has gaussian distribution. RBF kernel is defined as:

$$K(x, y) = \exp(-\gamma|x - y|^2)$$

where x and y are data points.

Finetuned and original LM model (without finetuning) features were used to train SVM however, it performed better with the finetuned LM. SVM from the Scikit's library was used which provides two hyperparameters for RBF kernel; γ and C. γ was set to auto and $C = 3$ was used.

3.6 Graph Neural Network (GNN)

GNN is a message passing algorithm that have been utilized in supervised and semi-supervised learning tasks, specifically in the field of structural biology where the proteins are made up of geometrical structure and cannot be easily handled by conventional convolutional neural networks (CNNs). GNNs, being more complex than CNNs, provide better handling of graph-like protein structures. Comparatively, GNNs have smaller number of parameters than that of CNNs, hence consume lower amount of memory than CNNs. A protein structure can be represented by a graph $G = (V, E)$ where V denotes the nodes (residues) in the graph (protein structure) with $|V| = n$ and E denotes edges. Each node has a feature vector, and m nodes can be represented by the feature matrix $X \in \mathbb{R}^{n \times m}$. Normally a graph is represented by an adjacency matrix to train a GNN and the adjacency matrix is given by $A = [a_{i,j}] \in \mathbb{R}^{n \times n}$. In this research, Graph Transformer [74]

was used that implements the same multi-head attention transformer mechanism which was proposed by Vaswani et al. [75].

As the LM features were used, it made sense to use Graph Transformer for feature propagation. Conventional GCN aggregates the nodes by taking average or weighted average, however, in the Graph Transformer self-attention mechanism is used to weight the contributions of different neighboring nodes. This can allow the model to focus more on important neighbors and less on less important neighbors.

For this approach, straightforward features were not used, rather each pocket was converted to a graph using NetworkX [76]. Following are the steps followed to build a pocket graph:

1. Each pocket has a PDB format file which contains pocket residues with their 3D coordinates. For each residue, C_α atom's coordinates were used to build graphs.
2. To connect atoms, a threshold of 9 Å was used that was selected by performing tests on validation data. If the distance between two C_α atoms was less than or equal to 9 Å, they were connected by an edge.
3. Each node in the graph represents a residue in the pocket.
4. These graphs were built using NetworkX and each graph had a label either 1 or 0 representing allosteric or non-allosteric pocket, respectively.
5. Most importantly, each node in the graph had its corresponding feature vector of size 1024 that was obtained from the ProtBERT pLM.
6. Graphs did not have pocket features that were extracted from FPocket, however the pockets 3d information was used from FPocket. Hence, the GNN was trained solely on features obtained from the pLM.
7. The GNN model was trained to perform binary classification on graphs.

Figure 3.3 gives the architecture of the GNN model (inspired by DeepFindr[6]) that was used to train on pockets data. The model takes pocket graph as input where each node in the graph has feature vector extracted from the pLM. It goes through the Graph Transformer [74] layer comprised of 5 attention heads and embedding size of 64. After that it passes through a FC layer of size 320 x 64. The 320-dimension size comes from 5 heads x 64 = 320. Further ReLU and Batch Norm were applied. Moreover, Top-K pooling

[6] https://deepfindr.github.io/

with ratio of 0.3 was applied which is then followed by global pooling. Global pooling performs Global Average and Global Max pooling, and concatenates them, resulting in dimension of size 128. This block is repeated 5 times however, global pooling does not take part in this block; rather it is summed 5 times and fed into next FC layer of the network.

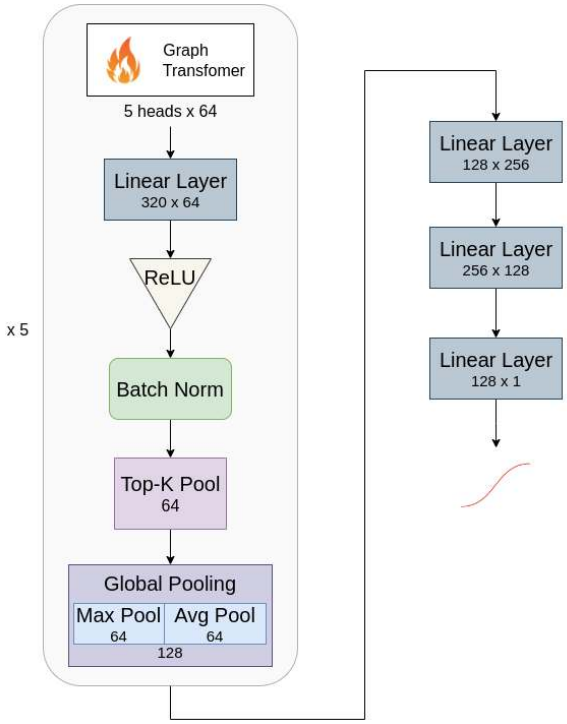

Figure 3.3: GNN Architecture

After this block, 3 FC layers are applied of size 256, 128, 1. A sigmoid function is applied at the end which gives the probability of a pocket.

The model was trained with the following hyperparameters:

Table 3.3: GNN Hyperparameters

PARAMETER	VALUE
Epochs	16
Batch Size	64
Learning Rate	$5e^{-5}$
Weight Decay	$2e^{-3}$
Evaluation Metric	F1 Score
Optimizer	Adam
Scheduler	CosineAnnealingLR

Binary Cross Entropy loss function was used by giving higher weights to the positive class. Positive class weight was calculated by the ratio between negative samples and the positive samples in training dataset.

3.7 Architecture of the Approach

All above mentioned models have the same backbone; pLM features and FPocket pockets. Figure 3.4 gives the architecture of the approach followed in this study.

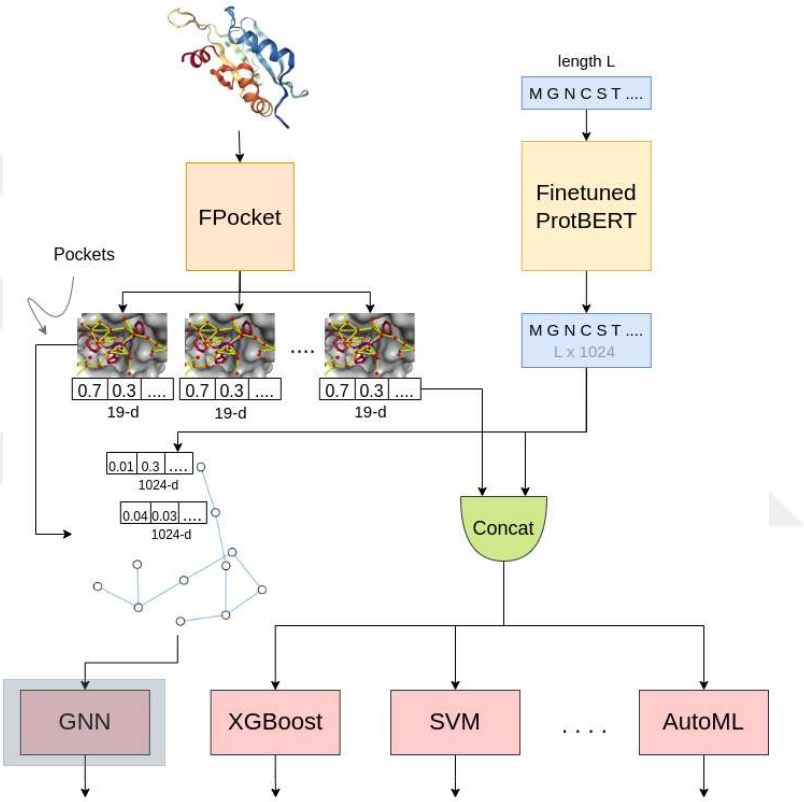

Figure 3.4: Thesis Architecture and Approach

All models, shown in pink color, were trained separately and they don't have any interdependence on any parallel models. For the sake of clarity and succinctness, all models used in the approach are presented in a single diagram (Figure 3.4). Note that while these models share the same dataset and ProtBERT backbone, they operate independently.

For each model, a structure and sequence of a protein is fed into FPocket and ProtBERT, respectively. FPocket extracts pockets, where each pocket has PDB format like coordinates and a 19-d feature vector. The pLM also produces features where each feature vector is of size 1024 and represents one residue in the sequence. For the GNN model, combination of pLM features and 3d structure is used (no pocket features are

used). For other models, 3d structure of pockets is not used, rather the pocket features are concatenated with the pLM features and fed into the models either for training or inference.

3.8 Lesser but Noteworthy (Base) Models

In the initial stages of this study, small MLPs were also trained which did not perform well. However, they are worthy of being mentioned here. A small MLP, shown in Figure 3.5, was trained on pocket features only. Each linear layer was followed by batch normalization and ReLU activation. A total of 6 layers were used and the network was constructed in PyTorch. For three layers, the number of neurons were doubled at each layer and for the remaining 3 layers the number of neurons was decreased by 2. Moreover, a Multihead Self Attention (MHSA) layer was applied at the middle of the network so that features could learn how they correlated with each other.

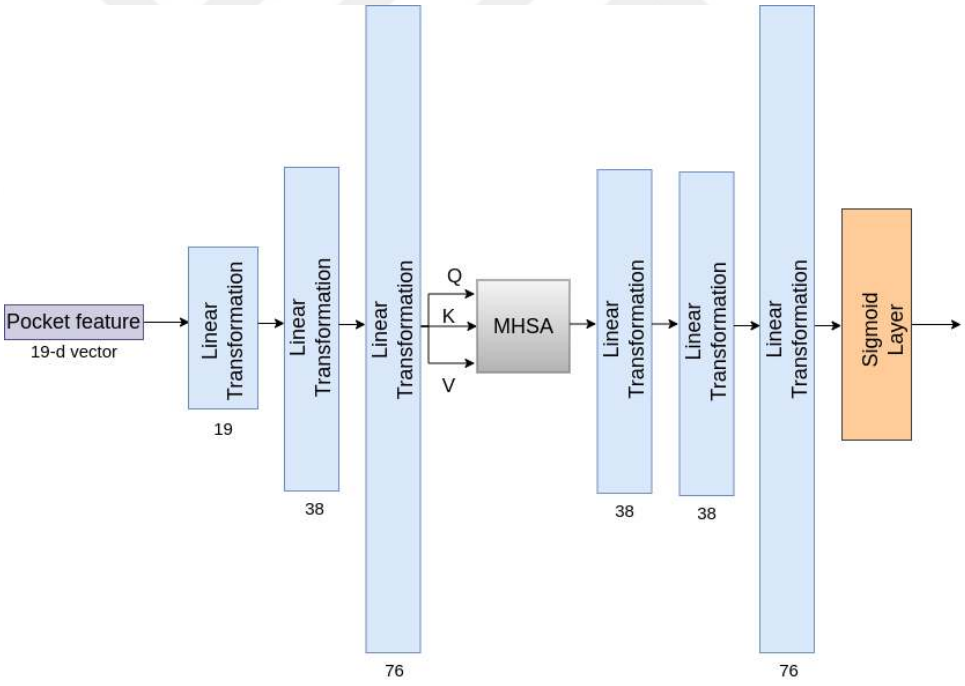

Figure 3.5: MLP of Pockets

However, this MLP does not utilize structural or language features, hence performed poorly.

In addition to this, ProtBERT (without finetuning) was added with a variant of MLP as depicted in Figure 3.6. Each "Linear Transformations" layer represents a combination of linear transformations. A sequence is input to the ProtBERT-BFD pretrained model which gives a 1024-d vector that represents the features of a sequence. This vector is then

combined with the pocket of that sequence in order to increase the performance of the overall model. Multihead Cross Attention (MHCA) was added between the 1024-d vector and the pocket features vector so that pocket features and sequence features can correlate with each other. In the MHCA, sequence representation was considered as Key and Value, while the pocket features vector was considered as Query. For MHSA and MHCA, 2 heads were used.

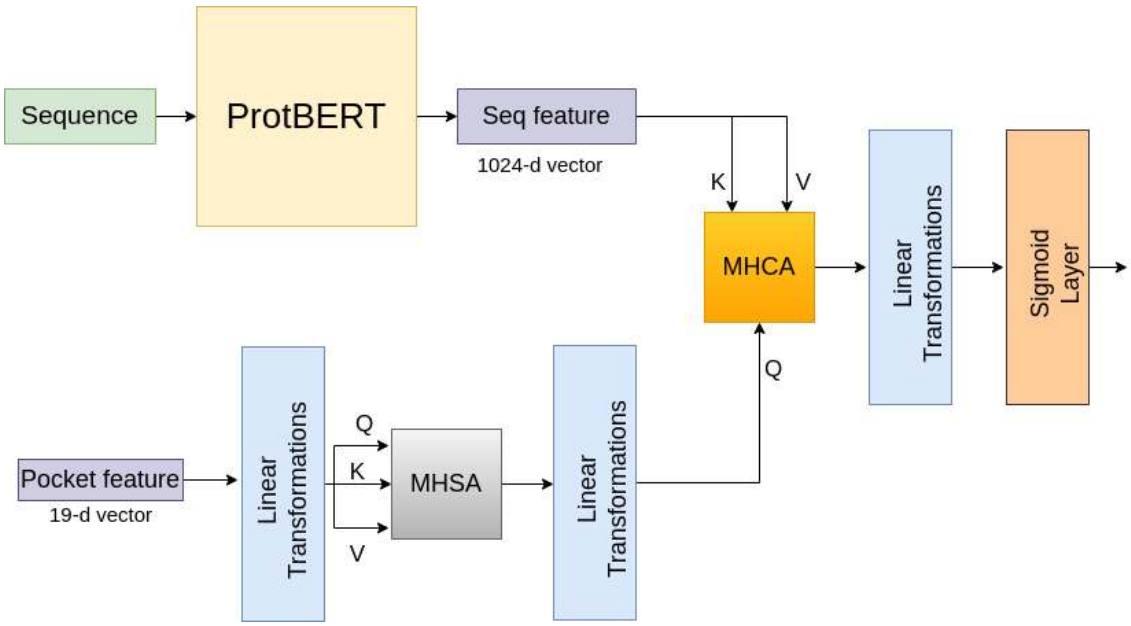

Figure 3.6: MLP with Multihead Attention and LLM

This model was different in a sense that it took whole sequence feature vector and combined the vector with each pocket in a certain protein, hence a pLM feature vector was repeated several times with each pocket input feature vector, introducing redundancy. Moreover, incorporating MHCA did not benefit as expected.

These two models do not capture the 3d structure information and do not offer any significant advantage over other techniques, hence they were not further pursued. However, small MLPs can be used as aggregators of modules in aforementioned models.

Chapter 4:
RESULTS AND DISCUSSION

"An experiment is a question which science poses to Nature and a measurement is the recording of Nature's answer."

— M. PLANCK

"In questions of science, the authority of a thousand is not worth the humble reasoning of a single individual."

— G. GALILEI

This chapter explores and analyzes the results of different methodologies and offers a deeper understanding of protein allosteric pockets prediction by the amalgamation of machine learning/deep learning techniques and pLM features. Results discussed in this section are believed to play the role of advocate for the utilization of pLM-incorporated approaches for allosteric sites prediction and consequently for allosteric drug discovery. Moreover, these results are compared with the available literature in allosteric sites prediction.

4.1 Features' Analysis

Feature exploration and analysis is an important step in machine learning. In this study, two types of features were used: FPocket features (representing pockets) and ProtBERT features (representing sequence residues). FPocket extracts pockets from a protein and generates 19 features corresponding to each pocket. These features are listed below:

- Score
- Druggability Score
- Number of Alpha Spheres
- Total SASA
- Polar SASA
- Apolar SASA
- Volume
- Mean local hydrophobic density
- Mean alpha sphere radius
- Apolar alpha sphere proportion
- Hydrophobicity score
- Volume score
- Polarity score
- Charge score
- Proportion of polar atoms
- Alpha sphere density
- Cent. of mass – Alpha Sphere max distance

- Mean alpha sphere solvent access
- Flexibility

Considering only the pocket features, *score* feature (or the pocket probability score), based on the SHAP values calculated by Tian et al. [40], was the most important feature and contributed ~70% among remaining 18 features. Moreover, *Alpha sphere density* and *Cent. of mass – alpha sphere max distance* were highly correlated among the pockets [38].

Furthermore, to analyze the class distribution, pockets features were concatenated with residue features (making up 1043 feature vector corresponding to each pocket) and t-SNE (t-Distributed Stochastic Neighbor Embedding) [77] was performed to reduce dimensions from 1043 (1024+19) to 2 dimensions. Here *X* and *Y* are simply the names given to the reduced dimensions.

Visually, pockets do not cluster into distinct positive or negative partitions. However, it can be seen in Figure 4.1 that most positive samples are shown in the top right corner of the plot. This suggests that they share some similarities in high-dimensional space. This could mean that they have similar properties or features that make them distinct from many of the negative samples.

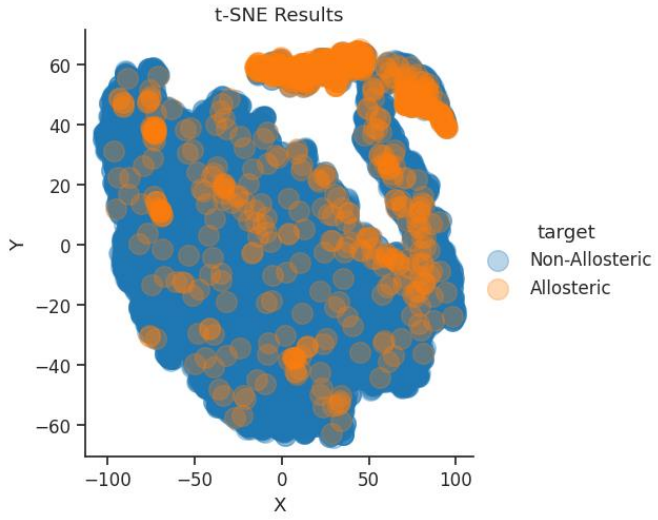

Figure 4.1: Allosteric and Non-Allosteric Sites' t-SNE Features Distribution

The overlap between positive and negative samples indicates that these samples are not easily separable based on their features. This could be the reason that aforementioned MLP could not make distinction between two classes, accurately. Also, linear SVM performed poorly on this dataset, as feature distribution is not separable linearly. Moreover, negative samples are scattered all over the plot that suggests a greater diversity within the negative class.

4.2 Evaluation Metrics

The main evaluation metric was F1-Score or sometimes referred to as Dice Coefficient. As the dataset was highly imbalanced; F1 score is not only a measure of how many positives are found, but it also penalizes for the false positives that the method finds. F1 score is given as:

$$F_1 = \frac{2}{\mathrm{Recall}^{-1} + \mathrm{Precision}^{-1}} = 2\frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

where

$$\mathrm{Precision} = \frac{TP}{TP + FP}$$

$$\mathrm{Recall} = \frac{TP}{TP + FN}$$

As the negative class samples heavily outnumber the other, models achieved a high accuracy simply by predicting the majority class all the time, thus making the accuracy metric misleading. F1 score is more suitable in this case as it takes into account both precision (how many of the predicted positive instances are actually positive) and recall (how many of the actual positive instances are predicted positive). A model with a high F1 score is both good at avoiding false positives and false negatives, making it a more balanced measure than accuracy in imbalanced datasets.

4.3 ProtBERT-BFD pLM

To train the ProtBERT pLM, dataset was split randomly into train, validation, and test sets with the partition ratios of 0.8, 0.12, and 0.08, respectively. The dataset comprised of protein sequences with each residue labelled as either positive (allosteric) or negative (non-allosteric). *BertTokenizer* from *HuggingFace* was used to tokenize the sequences with sequence length restricted to 1024 size. Model was trained to classify a token (residue) in the sequence (protein) either to be positive (allosteric) or negative (non-allosteric). Hence, this model does not *directly* predict pockets, rather it classifies residues.

This model was trained for 16 epochs, and after 16 epochs this model started to overfit. *Learning rate* was set to 0.0002 with *weight decay* of 0.1. Figure 4.2 gives the train/val loss curve.

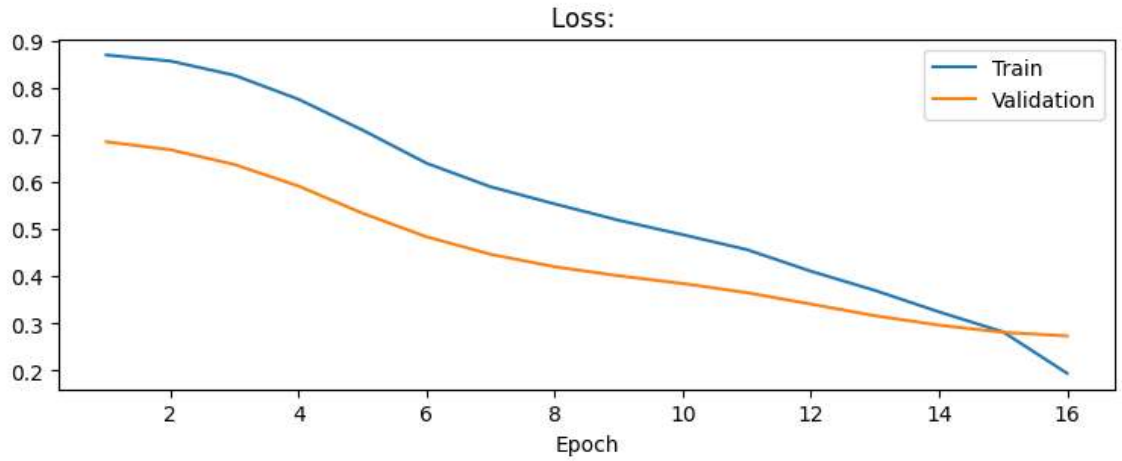

Figure 4.2: ProtBERT-BFD Loss Plot

Loss plot shows that after epoch number 15, the model started to overfit hence this the training was stopped and according to the loss curve, best model was at epoch 16. Moreover, tradeoff between precision and recall was also considered. Even if an F1 score is high it does not mean that model would always perform well. In this case, a predicted positive pocket must be positive (that means we are looking for accurate true positives); precision score needs to be higher. A low precision and high recall would give high F1 score but that also means that there are a lot of false positives in the predictions. Hence, the model was trained such that, on the validation data, there must be approximately a 50/50 tradeoff (or precision had to be a little higher). We want higher precision, but we also don't want to ignore recall, as low recall would mean high false negatives in the prediction.

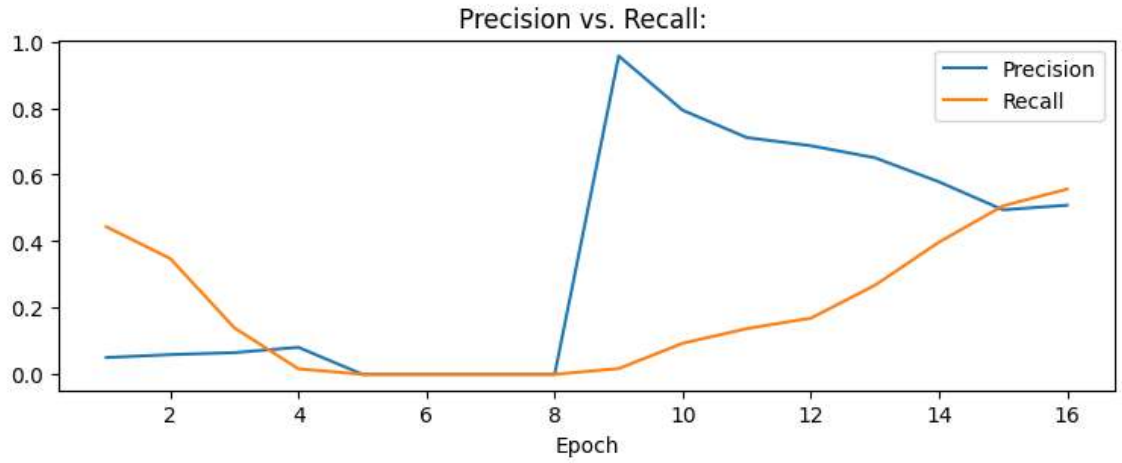

Figure 4.3: ProtBERT-BFD Precision vs. Recall Curve on Validation Data

In Figure 4.3, Precision versus Recall has been shown. It can be seen that at epoch number 15, model performs well on the validation data, after which recall and precision

start to increase and decrease, respectively. Getting information from loss curve and this curve, best model was saved at epoch 16, trained on the above-mentioned parameters.

Table 4.1: ProtBERT-BFD Residue Classification Metrics

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9646	0.6884	0.6012	0.8054
VALIDATION	0.9530	0.5313	0.5082	0.5565
TEST	0.9552	0.5956	0.5661	0.6283

Moreover, the model was tested to classify pockets, as well. As mentioned in the Methodology chapter that, to fine-tune the pLM all residues in a pocket were considered allosteric if that pocket is allosteric; however, the pLM cannot simply classify a pocket, as pockets do not contain consecutive sequence residues. Hence, evaluated on validation data, if a pocket contains at least 8 *predicted* allosteric residues, this pocket was classified as allosteric.

Table 4.2: ProtBERT-BFD Pocket Classification Metrics

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9614	0.6765	0.7505	0.6158
VALIDATION	0.9470	0.5680	0.6957	0.4800
TEST	0.9548	0.6154	0.6667	0.5714

Some other parameters were also tried to check the performance. Following subsections give performance on different hyperparameters.

4.3.1 ProtBERT Residue Classification Metrics – lr=0.001 weight_decay=0.02

Table 4.3: ProtBERT-BFD Residue Classification Metrics - lr=0.001 weight_decay=0.02

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9592	0.6710	0.5506	0.8587
VALIDATION	0.9428	0.5007	0.4301	0.5989
TEST	0.9467	0.5687	0.4940	0.6701

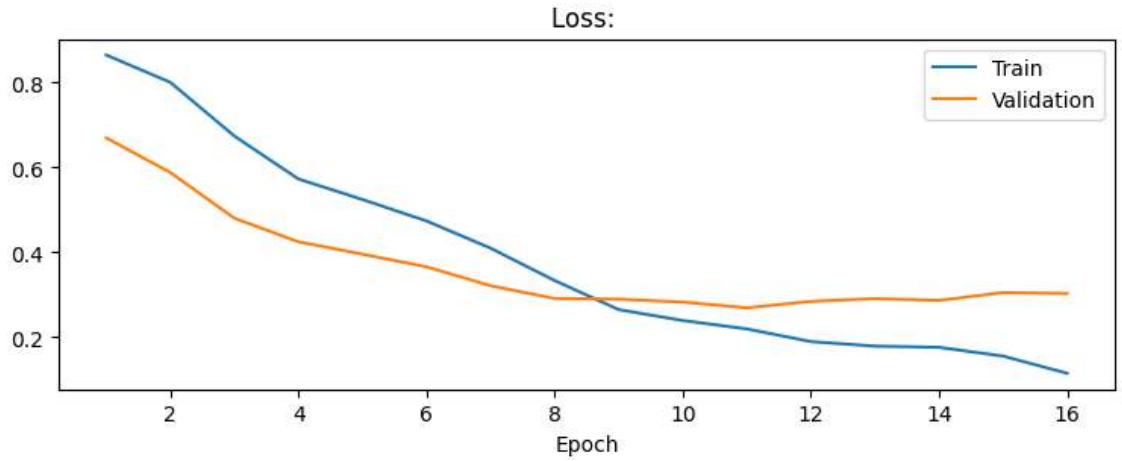

Figure 4.4: ProtBERT-BFD Loss Plot - lr=0.001 weight_decay=0.02

With these parameters as shown in Figure 4.4 and Figure 4.5, the model started to overfit. After arriving at ~0.5 F1 score, the loss started to increase and decrease. It could be the issue of high learning rate that it was jumping around the minima. It was clearly worse than the results given in Table 4.1. As the *weight_decay* was decreased, it could also be the issue of model being overfitted.

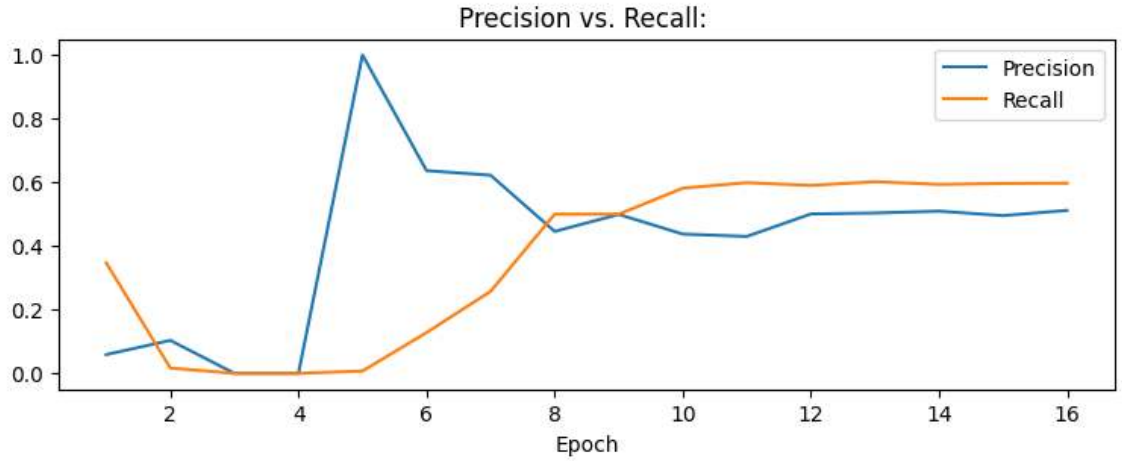

Figure 4.5: ProtBERT-BFD Precision vs. Recall Plot - lr=0.001 weight_decay=0.02

4.3.2 ProtBERT Residue Classification Metrics – lr=0.0002 weight_decay=0.1

Table 4.4: ProtBERT-BFD Residue Classification Metrics - lr=0.0002 weight_decay=0.1 positive_weight=8

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9484	0.6243	0.4825	0.8842
VALIDATION	0.9381	0.4885	0.4043	0.6170
TEST	0.9443	0.5538	0.4772	0.6599

In this run, above mentioned parameters (the best *lr* and *weight_decay* parameters) were used however, the weight ratio of negative to positive was set at 1:8 in the cross-entropy loss function.

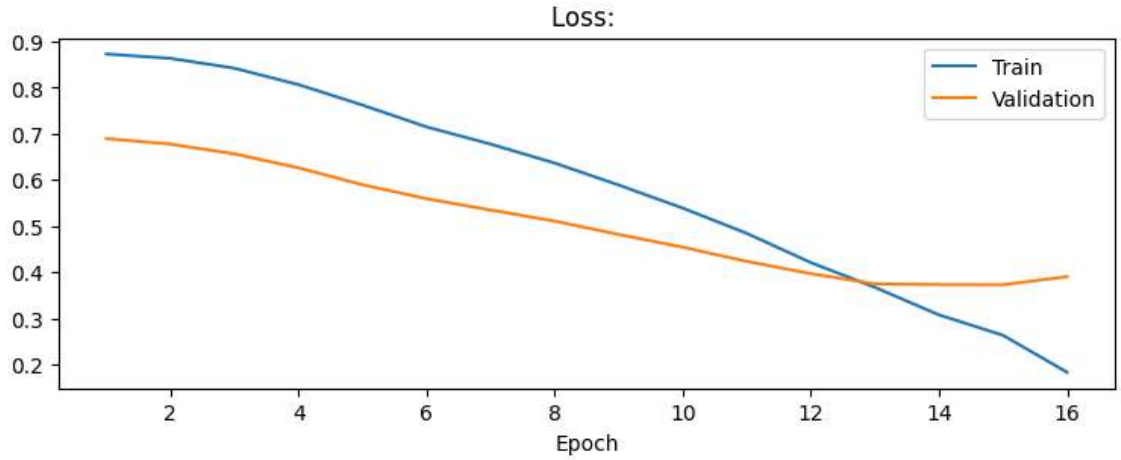

Figure 4.6: ProtBERT-BFD Loss Plot - lr=0.0002 weight_decay=0.1 positive_weight=8

Although by decreasing the *lr* and increasing the *weight_decay*, F1 score increases, as shown in Table 4.4, but at the cost of decrease in precision. By giving more weight to positive class, recall value increased, and precision decreased, hence the number of false positives increased compared to the best recall of 0.6283 given in Table 4.1. Moreover, with these training parameters, model again starts of overfit after epoch 13 and there was no decrease in validation loss depicted by in Figure 4.6.

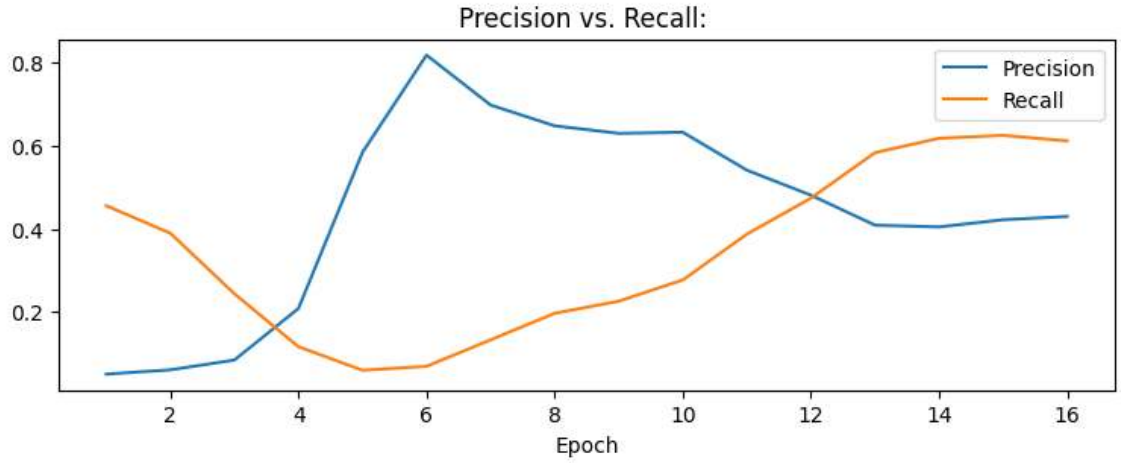

Figure 4.7: ProtBERT-BFD Precision vs. Recall Plot - lr=0.0001 weight_decay=0.1 positive_weight=8

4.3.3 ProtBERT Results Discussion

It can be seen, in Table 4.2 that finetuned ProtBERT is pretty good (better than XGBoost and GCNN models by Tao et al., [38] [40]) at classifying pockets. However, a

point is to be noted that, in case of pockets' classification, it does not give probability scores, rather a threshold of *at least 8 positive predicted residues in a pocket* was used, for it to be considered being an allosteric pocket.

Another model named ProtBERT (without BFD) was also finetuned on the same best hyperparameters that were used to finetune ProtBERT-BFD. Originally, ProtBERT was trained on Uniref100 database (smaller than BFD) by Elnagger et al. [26].

Table 4.5: ProtBERT Residue Classification Metrics - lr=0.0002 weight_decay=0.1 positive_weight=8

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9601	0.6668	0.5606	0.8226
VALIDATION	0.9515	0.5138	0.4938	0.5353
TEST	0.9511	0.5483	0.5315	0.5662

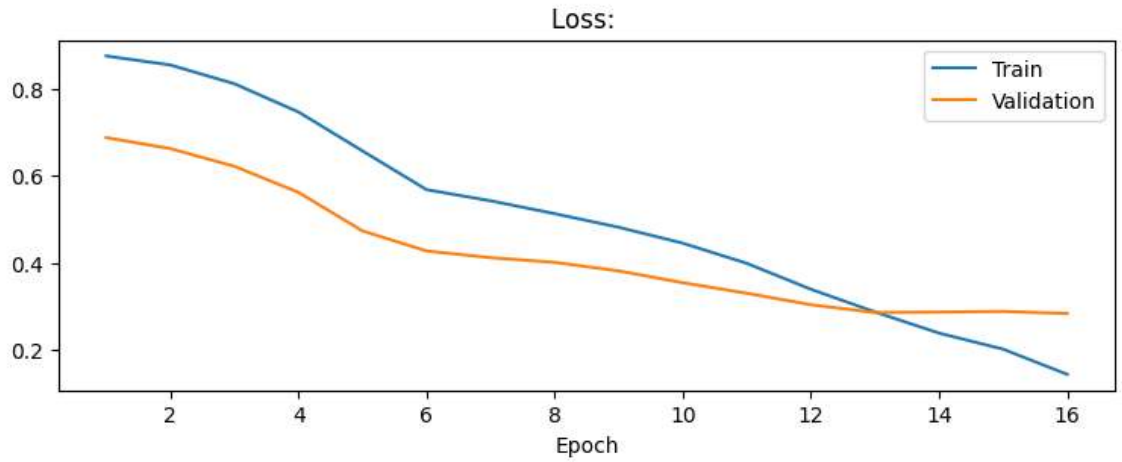

Figure 4.8: ProtBERT Loss Plot - lr=0.0002 weight_decay=0.1 positive_weight=8

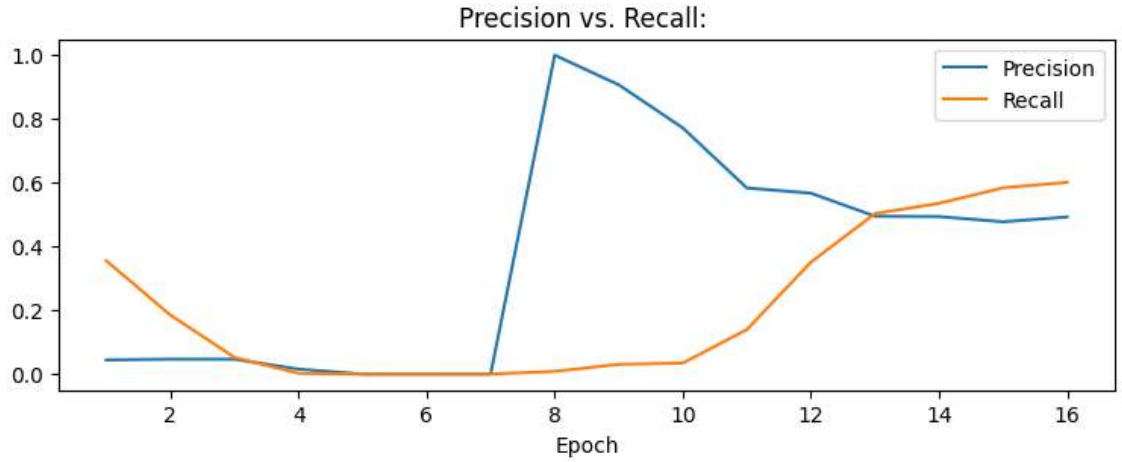

Figure 4.9: ProtBERT Precision vs. Recall Plot - lr=0.0001 weight_decay=0.1 positive_weight=8

Although both models share the same architecture however, as expected, the performance of ProtBERT was inferior to that of ProtBERT-BFD. ProtBERT gave an F1 score of 0.5483, as shown in Table 4.5, which is 8.63% lesser than that (Table 4.1) of ProtBERT-BFD. As ProtBERT has the same architecture as that of ProtBERT-BFD, model's training curves follow the same pattern, as show in Figure 4.8 and Figure 4.9, with a small decrease in performance.

Vig et al. [78] interpreted attention in the ProtTrans models and they concluded that in the deeper layers of the LLM, heads give more attention to the binding sites and contacting residues, relatively to the earlier layers. Binding sites in proteins play an important role in interacting with other molecules that influence the protein's functionality. Even with evolution, binding sites are conserved and remain unchanged as they have important role in protein [79]. This is the reason that attention targets binding sites.

Like the main binding sites, it can be conjectured that allosteric sites are also conserved to some extent. Hence, the model can give more attention to the conserved regions (allosteric sites) even when other parts of the sequence may vary.

4.4 Support Vector Machine

SVM, from scikit-learn, was used to train and classify pockets. A pocket's 19 feature vector was concatenated with the pLM's 1024-d feature vector, and the SVM was trained on the dataset. However, it was tested with and without pocket features vector to check how effective the pLM features were. As mentioned in the Methodology chapter, RBF kernel was used with different values of C.

4.4.1 Evaluation by Combining the pLM and FPocket Features

Table 4.6: SVM - C=1 pLM + FPocket Features

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9830	0.8674	0.9011	0.8361
VALIDATION	0.9563	0.6341	0.8000	0.5253
TEST	0.9665	0.6838	0.8511	0.5714

Table 4.7: SVM - C=3 pLM + FPocket Features

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9907	0.9283	0.9525	0.9053
VALIDATION	0.9578	0.6628	0.7808	0.5758
TEST	0.9674	0.6842	0.8864	0.5571

Table 4.8: SVM - C=5 pLM + FPocket Features

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9930	0.9467	0.9584	0.9353
VALIDATION	0.9592	0.6744	0.7945	0.5859
TEST	0.9656	0.6667	0.8636	0.5429

4.4.2 Evaluation on the pLM Features

Following metrics show the results obtained by using only the pLM features. These features correspond to the residues that are in a certain pocket. Feature vectors of residues (in a pocket) were averaged-out to get a single 1024-d feature vector representing a pocket.

Table 4.9: SVM - C=1 pLM

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9790	0.8385	0.8583	0.8195
VALIDATION	0.9519	0.5976	0.7538	0.4949
TEST	0.9656	0.6833	0.8200	0.5857

Table 4.10: SVM - C=3 pLM

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9878	0.9053	0.9358	0.8767
VALIDATION	0.9570	0.6467	0.7941	0.5455
TEST	0.9665	0.6838	0.8511	0.5714

Table 4.11: SVM - C=5 pLM

DATASET	ACCURACY	F1	PRECISION	RECALL

TRAIN	0.9907	0.9287	0.9469	0.9113
VALIDATION	0.9578	0.6588	0.7887	0.5657
TEST	0.9674	0.6842	0.8864	0.5571

4.4.3 SVM Results Discussion

Best model was with parameters C=3 and pLM + FPocket features. However, with different values of C, SVM performed better with FPocket features. Hence, pLM features positively influenced classification of allosteric pockets.

4.5 XGBoost

Same as SVM's dataset, XGBoost was also trained and tested on features making up of (a) combination of pLM and FPocket and (b) only pLM features. Several parameters were tried including tree depth, regularization lambda, and fold size. This model was trained and validated with K-Fold cross validation technique. A weight was given to positive class while training, which was calculated at runtime by ratio of negative samples to the positive samples in that fold.

4.5.1 Evaluation by Combining the pLM and FPocket Features

Table 4.12: XGBoost (pLM + FPocket Features) - Max_Depth=3 Lambda=0.1

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9625	0.7791	0.6448	0.9840
VALIDATION	0.9464	0.6904	0.5620	0.8947
TEST	0.9548	0.7024	0.6020	0.8429

Table 4.13: XGBoost (pLM + FPocket Features) - Max_Depth=6 Lambda=0.1

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9972	0.9795	0.9598	1.0000
VALIDATION	0.9736	0.8000	0.7692	0.8333
TEST	0.9683	0.7287	0.7966	0.6714

Table 4.14: XGBoost (pLM + FPocket Features) - Max_Depth=3 Lambda=0.2

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9750	0.8433	0.7314	0.9957
VALIDATION	0.9596	0.7356	0.6275	0.8889
TEST	0.9584	0.7051	0.6395	0.7857

Table 4.15: XGBoost (pLM + FPocket Features) - Max_Depth=6 Lambda=0.2

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9998	0.9986	0.9971	1.0000
VALIDATION	0.9798	0.8435	0.8267	0.8611
TEST	0.9710	0.7576	0.8065	0.7143

4.5.2 Evaluation on the pLM Features

This model was evaluated on dataset comprised only of the pLM features with the same parameters mentioned in the previous section.

Table 4.16: XGBoost (pLM Features) - Max_Depth=3 Lambda=0.1

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9651	0.7926	0.6596	0.9927
VALIDATION	0.9561	0.7396	0.6174	0.9221
TEST	0.9484	0.6369	0.5747	0.7143

Table 4.17: XGBoost (pLM Features) - Max_Depth=6 Lambda=0.1

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9980	0.9857	0.9718	1.0000
VALIDATION	0.9640	0.7285	0.7237	0.7333
TEST	0.9629	0.6822	0.7458	0.6286

Table 4.18: XGBoost (pLM Features) - Max_Depth=3 Lambda=0.2

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9640	0.7871	0.6520	0.9927

VALIDATION	0.9561	0.7449	0.6134	0.9481
TEST	0.9448	0.6347	0.5464	0.7571

Table 4.19: XGBoost (pLM Features) - Max_Depth=6 Lambda=0.2

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9998	0.9985	0.9971	1.0000
VALIDATION	0.9640	0.7453	0.7059	0.7895
TEST	0.9638	0.6774	0.7778	0.6000

4.5.3 XGBoost Results Discussion

XGBoost gave the highest F1 score of 75.76% with *lambda* = 0.2 and *max_depth* of 6 while other parameters were default. This score was achieved with the combination of pLM and FPocket features. Without FPocket features, XGBoost gave an *average* F1 score of 65.78% while with FPocket and pLM features, the average F1 was 72.35%. It can be concluded that, pLM features positively impacted the F1 score.

4.6 Automated Machine Learning (AutoML)

For AutoML, AutoGluon framework was utilized. This framework automatically tries several models and tries to increase their performance on different hyperparameter. It can also perform stacking, where ensemble of different models is tried, and the prediction performance is increased. It tried the following 14 models:

- WeightedEnsemble_L2
- NeuralNetTorch_BAG_L1
- LightGBMXT_BAG_L1
- NeuralNetFastAI_BAG_L1
- LightGBM_BAG_L1
- XGBoost_BAG_L1
- CatBoost_BAG_L1
- LightGBMLarge_BAG_L1
- RandomForestEntr_BAG_L1
- RandomForestGini_BAG_L1
- ExtraTreesEntr_BAG_L1
- ExtraTreesGini_BAG_L1
- KNeighborsDist_BAG_L1
- KNeighborsUnif_BAG_L1

To train the framework, stacking and non-stacking options were used; and as expected, models performed better with stacking. It trained models on 8-fold cross

validation and models' performance was evaluated on F1 score. Moreover, as it is an automated framework, it gave only the F1 score for validation data. However, for test data, several metrics were measured. Following subsections give these metrics.

4.6.1 Evaluation by Combining the pLM and FPocket Features

Table 4.20: AutoGluon (pLM + FPocket Features) - Auto_Stack=True

DATASET	ACCURACY	F1	PRECISION	RECALL
TEST	0.9701	0.7360	0.8364	0.6571

Table 4.20 gives the evaluation metrics on the test data. pLM and FPocket features were combined, and stacking was set to true. This setting performs the best compared to other AutoML parameters tried. Best model in this case was WeightedEnsemble_L2, which gave an F1 score of 0.8408 on validation data; hence picked for evaluation on the test data.

Table 4.21: AutoGluon (pLM + FPocket Features) - Auto_Stack=False

DATASET	ACCURACY	F1	PRECISION	RECALL
TEST	0.9665	0.7132	0.7797	0.6571

In Table 4.21, stacking was set to *False* and the best model was again WeightedEnsemble_L2 that gave an F1 score of 0.8707 on validation.

4.6.2 Evaluation on the pLM Features

Table 4.22: AutoGluon (pLM Features) - Auto_Stack=True

DATASET	ACCURACY	F1	PRECISION	RECALL
TEST	0.9629	0.6667	0.7736	0.5857

Pockets were trained only on the pLM features and results are shown in Table 4.22 with stacking. Model selected as best by evaluating on validation data was WeightedEnsemble_L2, giving an F1 score of 0.7907.

Table 4.23: AutoGluon (pLM Features) - Auto_Stack=False

DATASET	ACCURACY	F1	PRECISION	RECALL
TEST	0.9575	0.6412	0.6885	0.6000

Without stacking, results given in Table 4.23, best model was WeightedEnsemble_L2 that gave an F1 score of 0.7925 on validation data.

4.6.3 AutoGluon Results Discussion

AutoGluon was trained on both the pLM features and combination of pLM + FPocket features. Moreover, it was trained with and without stacking option. The best model was WeightedEnsemble_L2 with stacking and pLM + FPocket features that gave an F1 score of 0.7360 on test data. No normalization was performed on the training features dataset as AutoGluon handles it itself. Moreover, it also selects the best features for enhanced performance. As solely pLM features performed very well, giving an average F1 of ~65%; it can be reasoned that the pLM features do increase prediction performance of allosteric sites. A cherry on the top with the FPocket features, added an average increase of ~7% in the F1 score.

4.7 Graph Neural Network (GNN)

As the main focus is the utilization of the pLM features and the pLM architecture is based on the BERT [73] architecture that is based on Transformer architecture and uses attention mechanism [75]; Graph Transformer [74] was used to build the GNN to fully leverage and scrutinize the ProtBERT-BFD pLM features.

The GNN built in this study is completely different from the aforementioned models and approaches. It does not utilize FPocket features, rather it solely depends on the features extracted from the finetuned ProtBERT-BFD pLM. A pocket was first converted into a graph using a threshold of 9 Å, each node (residue) in the graph (pocket) has its feature vector, that was extracted from the pLM. The model was trained to classify whether a graph (pocket) was allosteric or non-allosteric.

Several hyperparameters were tried to train the model and the following subsections give the results against these hyperparameters.

4.7.1 GNN Evaluation – Embed=64 Heads=5 Dense=256 LR=5e-5 Reg=2e-3

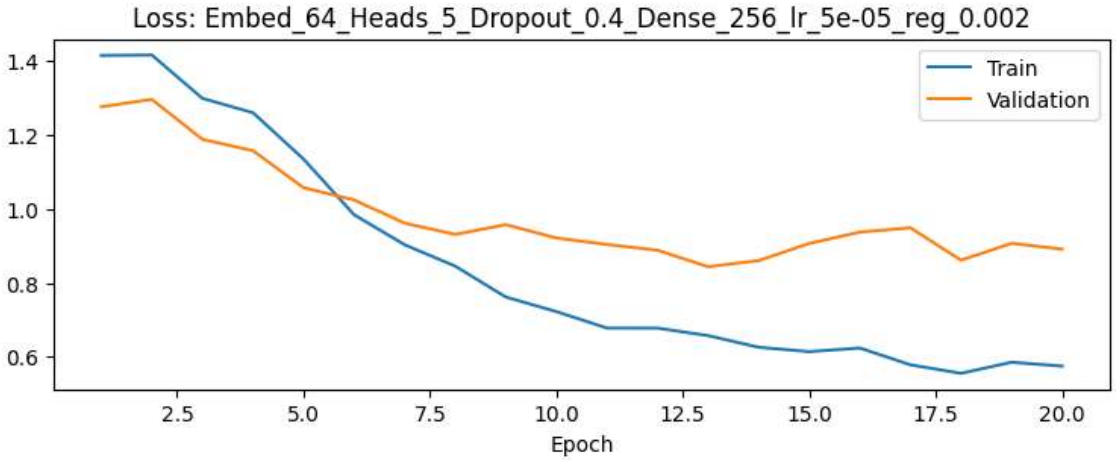

Figure 4.10: GNN Loss - Embed=64 Heads=5 LR=5e-5 Reg=2e-3

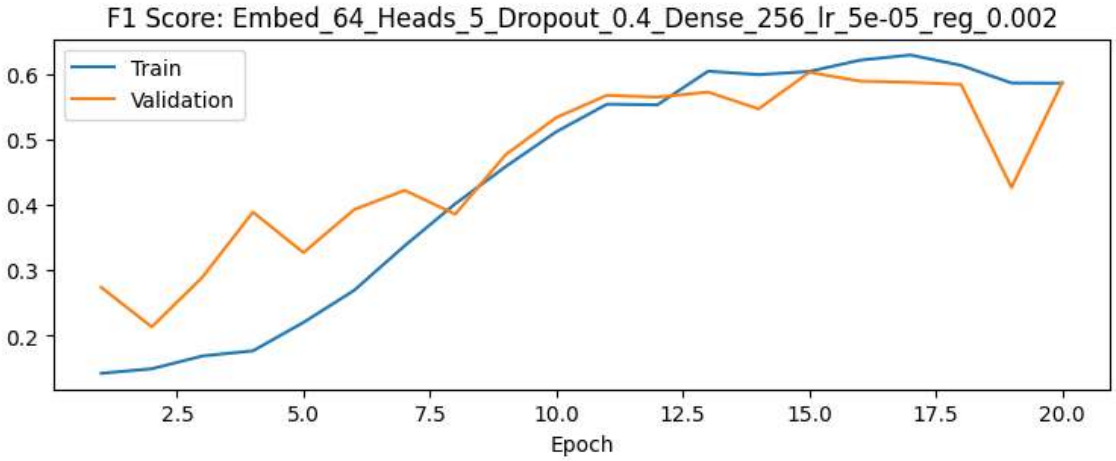

Figure 4.11: GNN F1 - Embed=64 Heads=5 LR=5e-5 Reg=2e-3

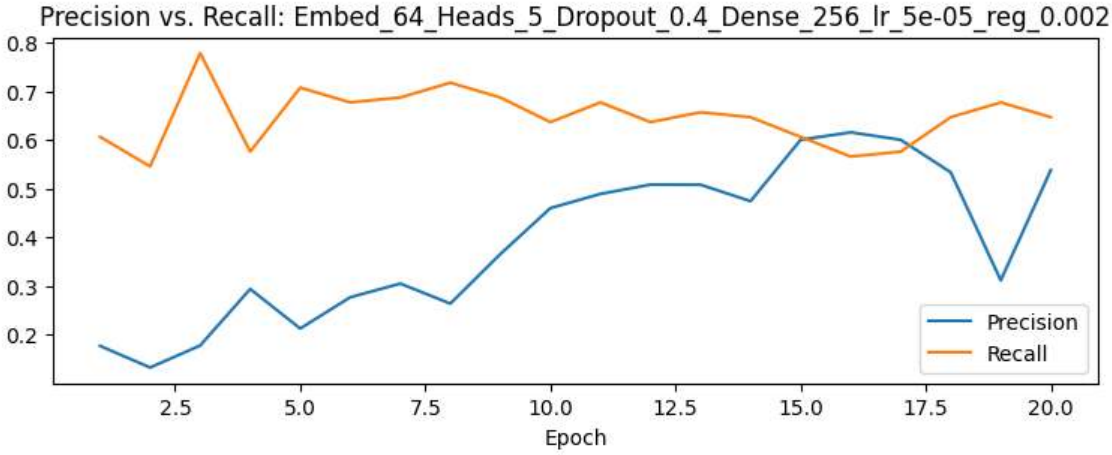

Figure 4.12: GNN Precision vs. Recall - Embed=64 Heads=5 LR=8e-5 Reg=1e-3

The model gave the highest F1 score on test dataset with the above-mentioned parameters.

As seen in loss and Precision vs. Recall curves, best model was at the epoch 15 on validation dataset. Hence, the model at epoch 15 was picked with these parameters which gave the following metrics:

Table 4.24: GNN Metrics - Embed=64 Heads=5 LR=5e-5 Reg=2e-3

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9127	0.6042	0.4661	0.8586
VALIDATION	0.9371	0.6030	0.6000	0.6061
TEST	0.9584	0.6933	0.6500	0.7429

For this run, higher regularization was used as compared to upcoming runs; this could be the reason that the model performs slightly poor on the train dataset. Moreover, it can be seen that precision is very low on train dataset, that indicates the high false positives. However, to select the best model, equal tradeoff between precision and recall was used, shown at epoch 15 in Figure 4.12. Model performs very good on test dataset with the F1 score of 0.6933 with very good precision of 0.65; that means model does not have high false positives and that is what we are aiming for. Another important factor is the dropout rate of 0.4, which helped decrease model overfitting. With these parameters, model was able to generalize better on the validation dataset and consequently on the test dataset.

4.7.2 GNN Evaluation – Embed=128 Heads=3 Dense=256 LR=6e-5 Reg=1e-3

In this run, transformer embedding size was increased to 128 and numbers of heads was decreased to 3. Moreover, it was observed that higher learning rates make the model jump around the minima, hence the *lr* was also decreased.

It can be seen from the plots in Figure 4.13, Figure 4.14, and Figure 4.15 that the best model is at epoch 15 as the loss was decreasing until epoch 15, and Precision and Loss are fairly equal at epoch 15.

Table 4.25: GNN Metrics - Embed=128 Heads=3 LR=6e-5 Reg=1e-3

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9252	0.6503	0.5103	0.8962
VALIDATION	0.9386	0.6091	0.6122	0.6061

TEST	0.9439	0.6173	0.5435	0.7143

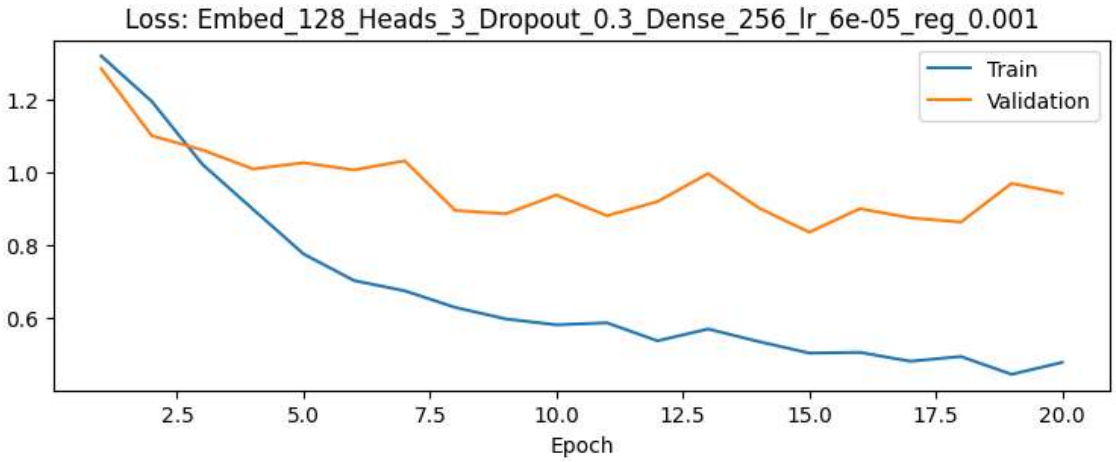

Figure 4.13: GNN Loss - Embed=128 Heads=3 LR=6e-5 Reg=1e-3

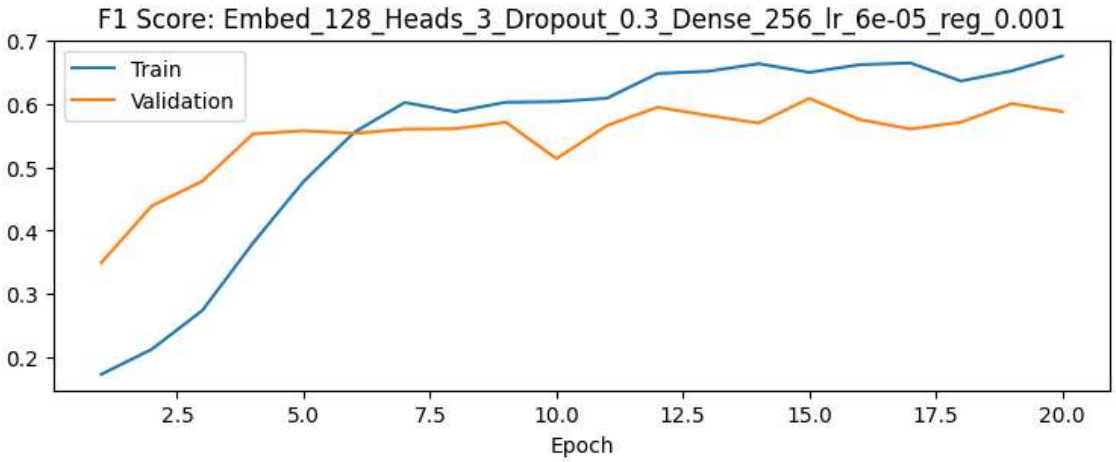

Figure 4.14: GNN F1 - Embed=128 Heads=3 LR=6e-5 Reg=1e-3

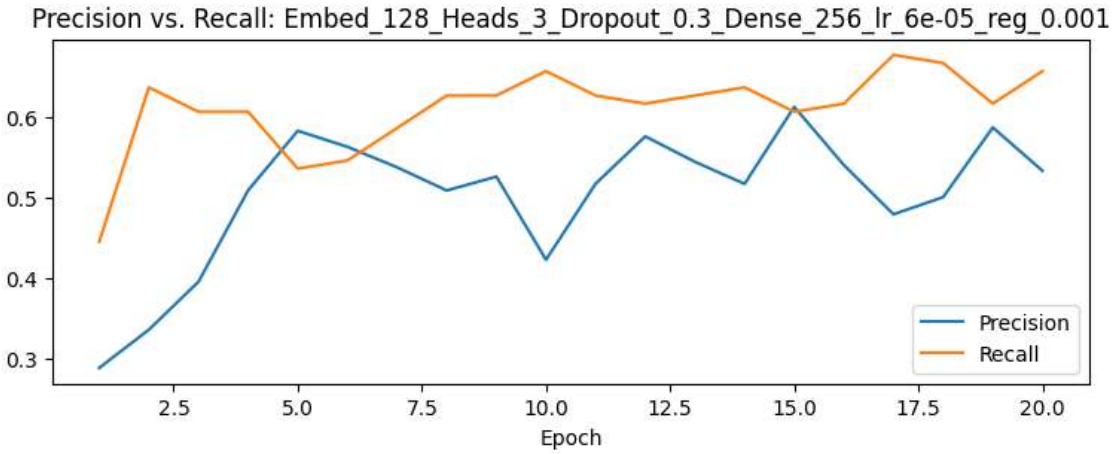

Figure 4.15: GNN Precision vs. Recall - Embed=128 Heads=3 LR=6e-5 Reg=1e-3

4.7.3 GNN Evaluation – Embed=32 Heads=2 Dense=512 LR=6e-5 Reg=1e-2

In this run, transformer embedding size was decreased to 32, number of heads was decreased to 2, and number of dense neurons was increased to 512. It was observed with these parameters that higher number of dense neurons do not add well to the performance rather they decrease it.

Table 4.26: GNN Metrics - Embed=32 Heads=2 Dense=512 LR=6e-5 Reg=1e-3

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9330	0.6786	0.5406	0.8932
VALIDATION	0.9076	0.5323	0.4430	0.6667
TEST	0.9339	0.5967	0.4865	0.7714

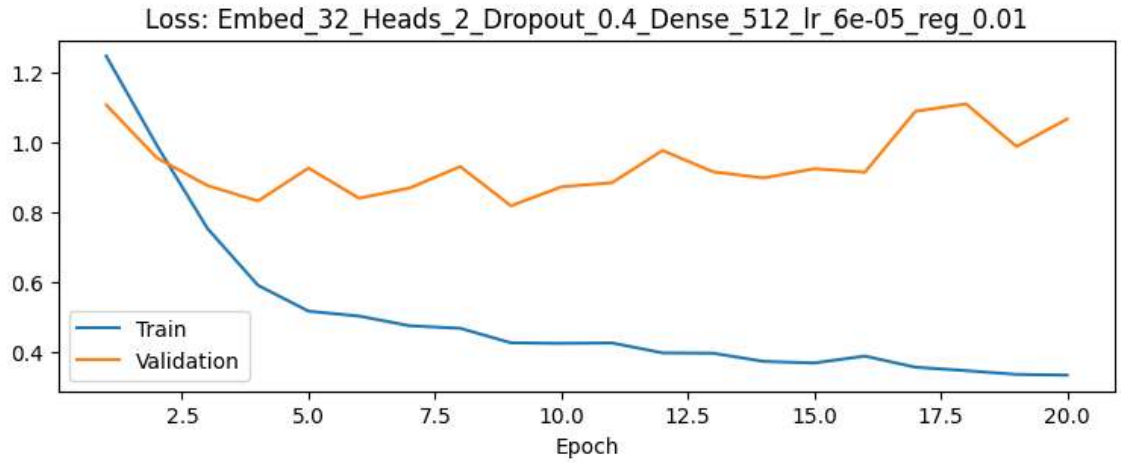

Figure 4.16: GNN Loss - Embed=32 Heads=2 Dense=512 LR=6e-5 Reg=1e-2

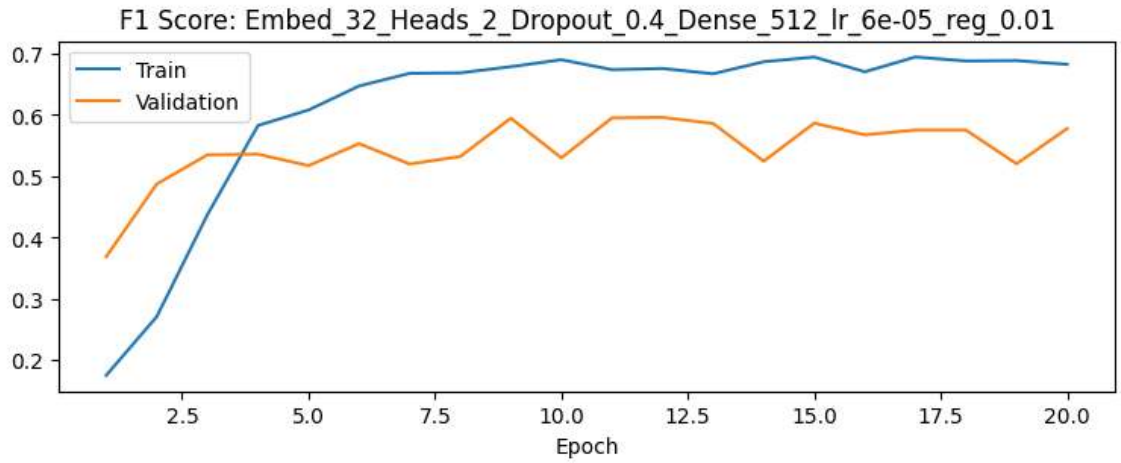

Figure 4.17: GNN F1 - Embed=32 Heads=2 Dense=512 LR=6e-5 Reg=1e-2

As the number of dense neurons was increased, it did not give any benefit to the performance rather resulted in model overfitting just after 9 epochs; however, decreasing the number of heads and transformer embedding size decreased the performance.

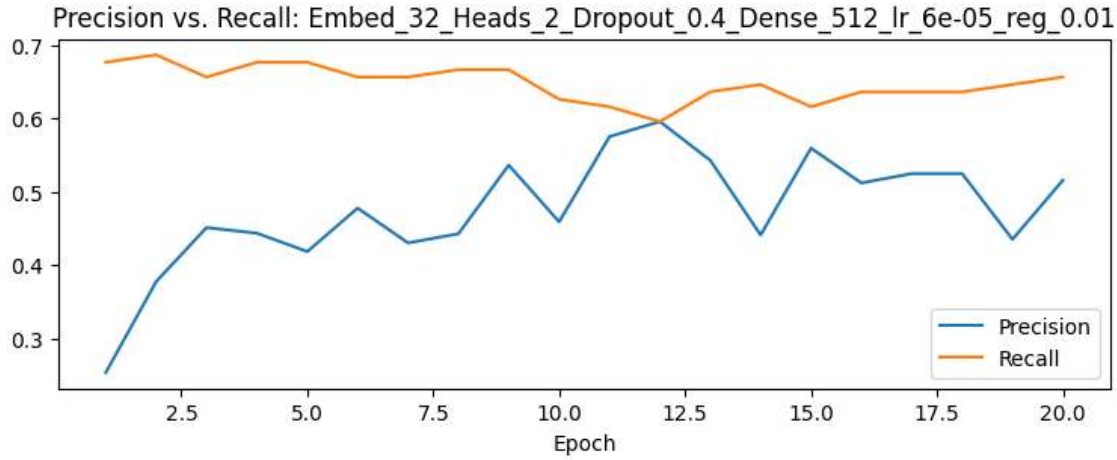

Figure 4.18: GNN Precision vs. Recall - Embed=32 Heads=2 Dense=512 LR=6e-5 Reg=1e-2

4.7.4 GNN Evaluation – Embed=32 Heads=3 Dense=256 LR=5e-5 Reg=2e-2

In this run, number of heads was increased to 3 and number of dense neurons was decreased to 256 to show that attention heads and transformer embeddings can get more meaning from the input structure. With these parameters, the model performed better than the previous run where the number of dense neurons was 512.

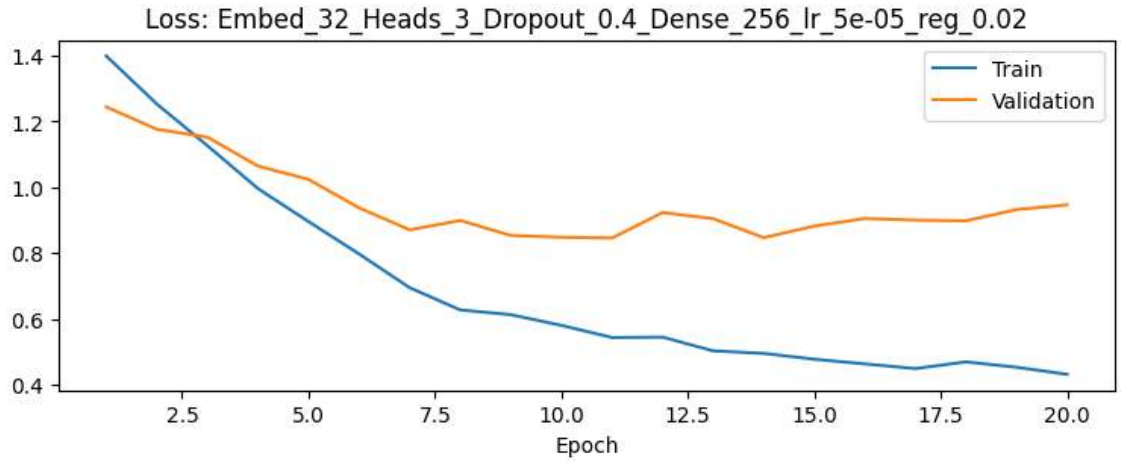

Figure 4.19: GNN Loss - Embed=32 Heads=3 Dense=256 LR=5e-5 Reg=2e-2

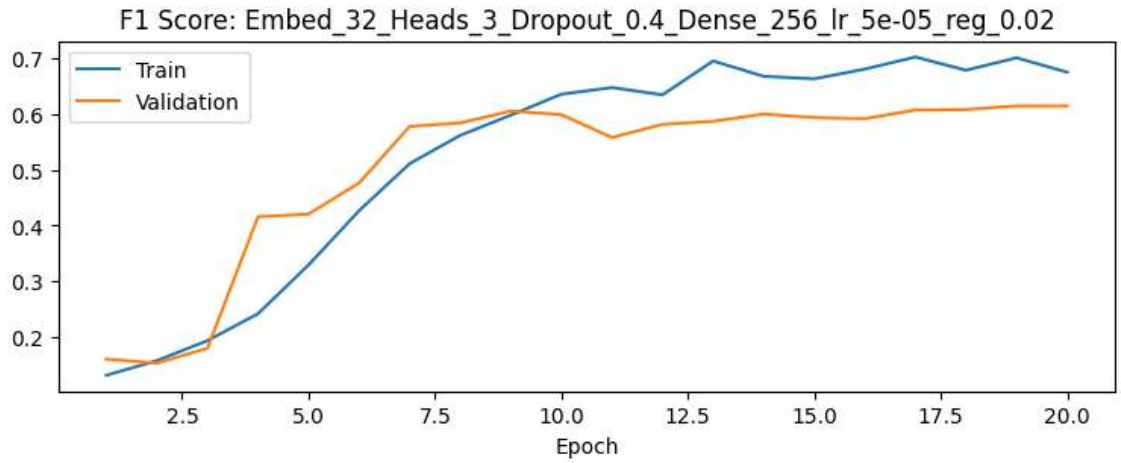

Figure 4.20: GNN F1 - Embed=32 Heads=3 Dense=256 LR=5e-5 Reg=2e-2

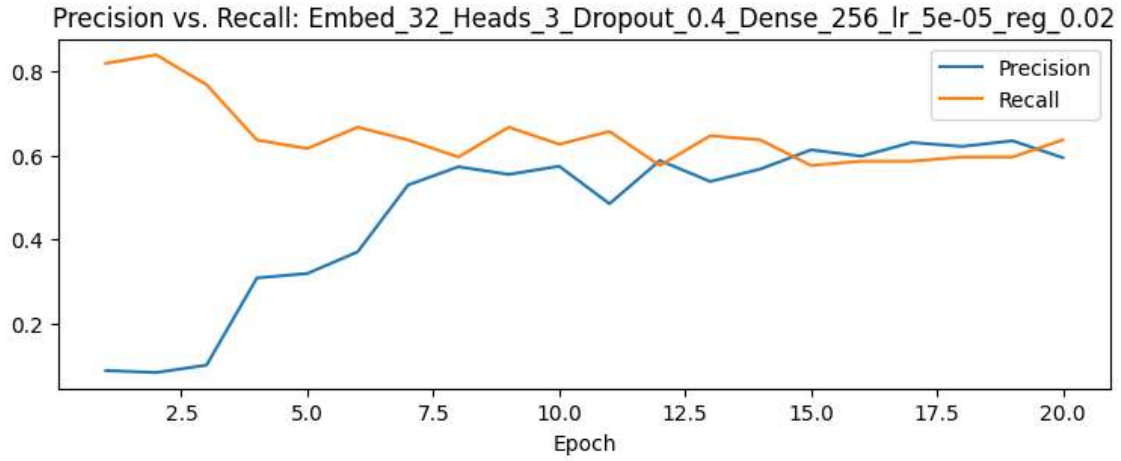

Figure 4.21: GNN Precision vs. Recall - Embed=32 Heads=3 Dense=256 LR=5e-5 Reg=2e-2

Table 4.27: GNN Metrics - Embed=32 Heads=3 Dense=256 LR=5e-5 Reg=2e-3

DATASET	ACCURACY	F1	PRECISION	RECALL
TRAIN	0.9296	0.6633	0.5275	0.8932
VALIDATION	0.9378	0.5938	0.6129	0.5758
TEST	0.9484	0.6275	0.5783	0.6857

4.7.5 GNN Evaluation on Different Distances

The GNN model with best parameters was fine-tuned on different distance thresholds. The distance threshold is a parameter to build the graph i.e. the maximum distance cutoff between two residues to be considered as connected by an edge in a graph.

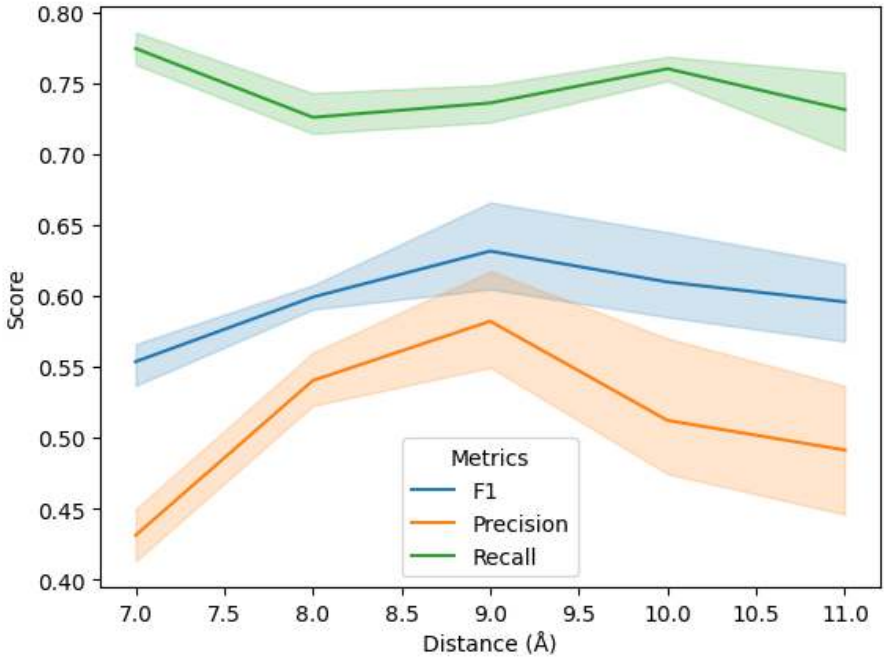

Figure 4.22: Metrics' Scores on Different Distance Thresholds

Distance thresholds from 7 to 11 were tested and the best F1 score was given on 9 Å threshold as shown in Figure 4.22. After 9 Å, there was no increase in the F1 score; hence it was selected as the distance cutoff. These values were selected and averaged over 5 independent runs of the model.

4.8 Results Comparison and Discussion

The techniques discussed above, used LM features and/or pocket features. Both results have been given with and without pocket features in order to show how pLM features affect the prediction performance of allosteric sites.

Figure 4.23 gives models' performance only using LM features (without pocket features). GNN model outperform other models based on F1 score followed by the SVM and XGBoost models. SVM and GNN show significantly higher scores in case of precision and recall, respectively. This shows that GNN is better able to capture structural attributes of a pocket by using LM features.

By combining pocket features with LM features, models' performance increased as shown in Figure 4.24 given by F1 score. Evidently, XGBoost performs better than other models followed by AutoML. It also helped XGBoost in increasing the precision score.

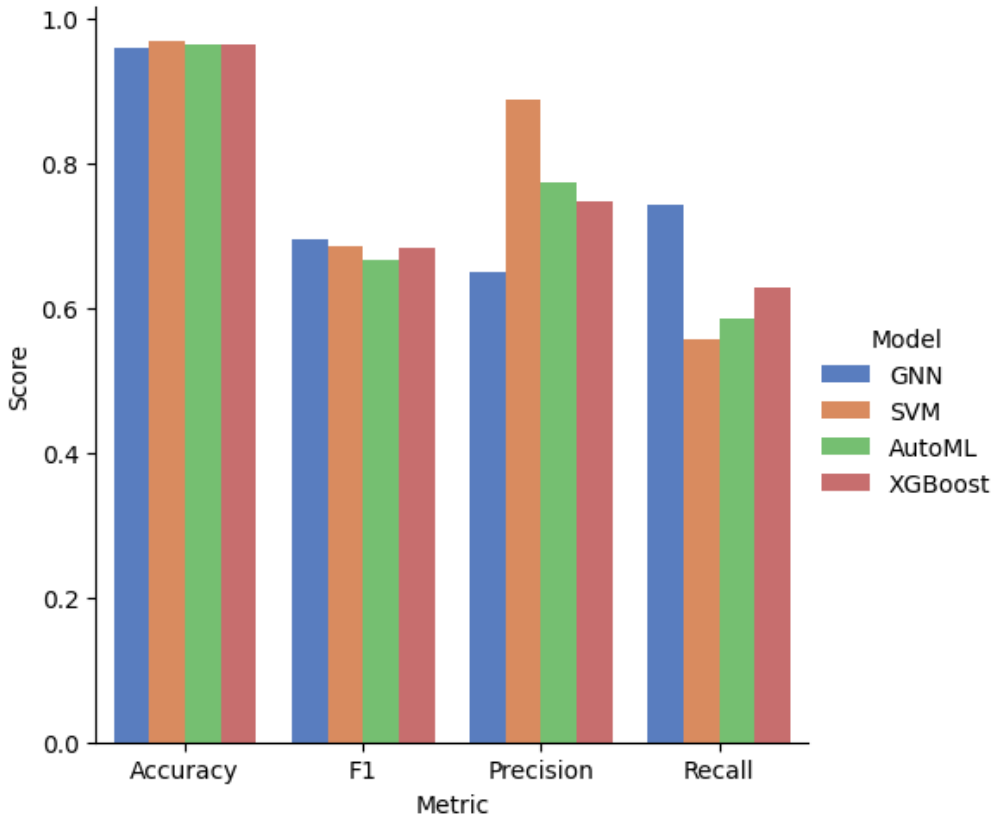

Figure 4.23: Models's Metrics Comparison with LM Features Only

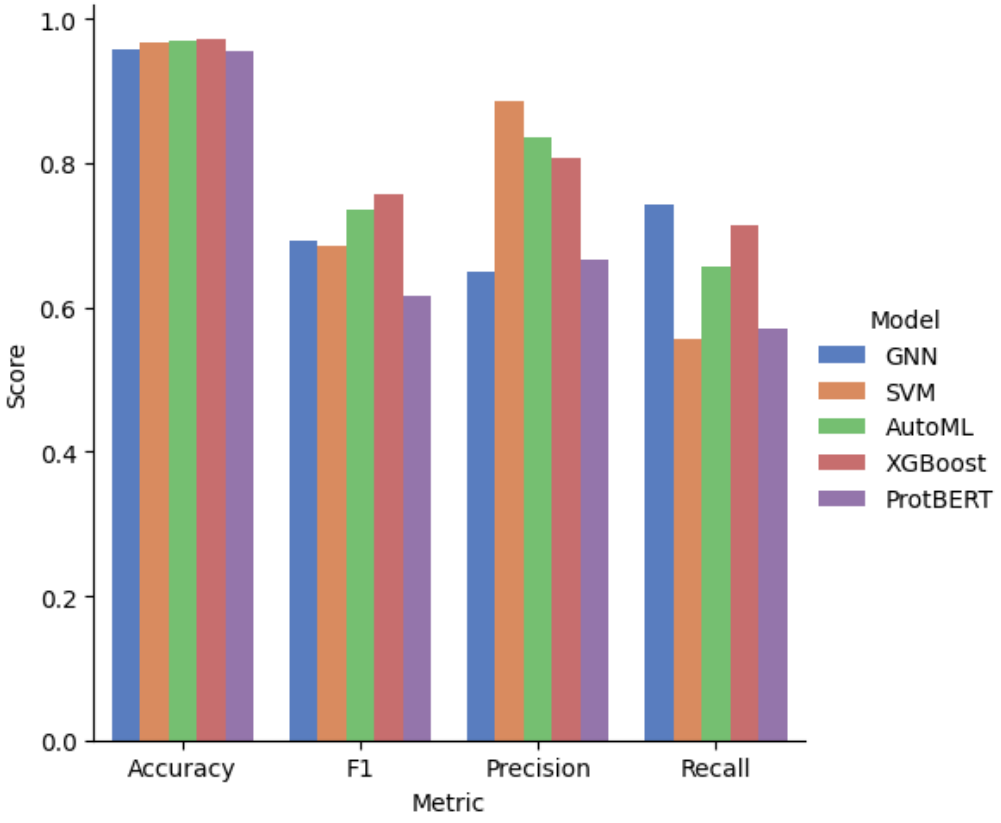

Figure 4.24: Models's Metrics Comparison with LM Features and Pocket Features

The GNN, however, does not utilize pocket features and still performs better than the SVM model on F1 score. Other models perform well on precision score, i.e., they can eliminate false positives while the GNN outperforms other models on recall score i.e., it is able to identify false negatives better than other models.

Additionally, it also compares the prediction performance of fine-tuned ProtBERT-BFD model against other models. The pLM surprisingly achieved a very good F1 score (above 0.6) and can be used as a standalone model to predict pockets. Since its performance is relatively lower than other models, it is being used to extract fine-tuned features to train other listed models.

Allosteric sites were ranked in 5 different top positions namely, Top 1%, Top 3%, Top 5%, Top 10%, and Top 20%. Figure 4.25 shows that XGBoost performs the best in all top positions however, there is not a huge difference in top 1%. The difference becomes significant at top 5% as the XGBoost and AutoML models outperform GNN and SVM models. At and after the Top 10% position, all models (XGBoost, AutoML, and GNN) outperform the SVM model with an average significant difference of ~19%.

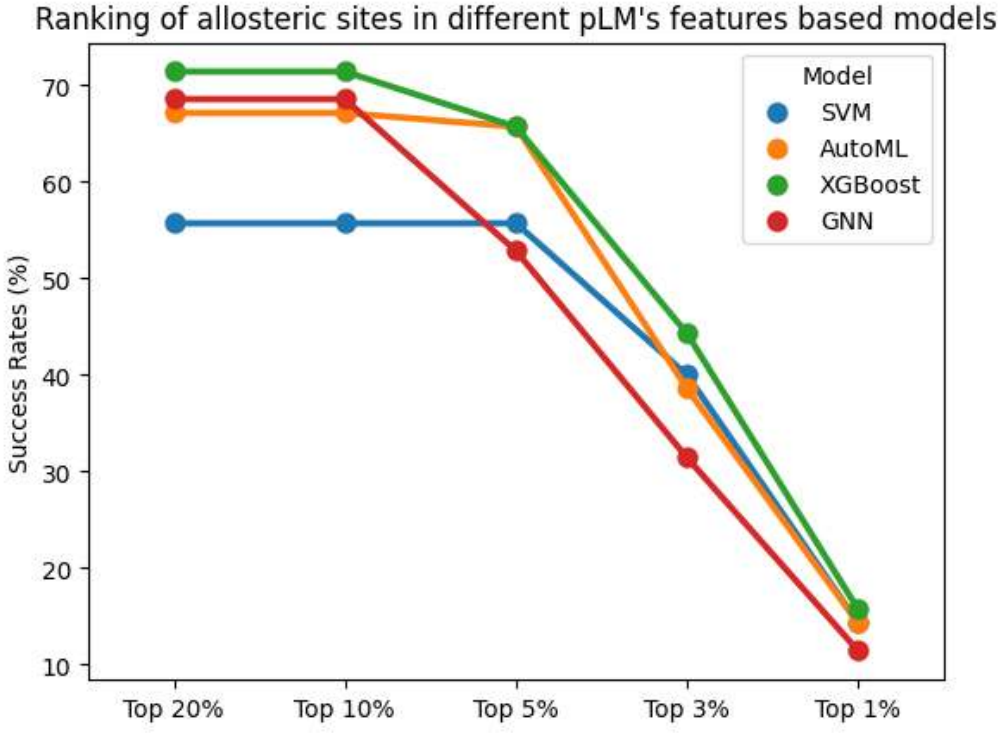

Figure 4.25: Ranking of Allosteric Sites in Different Models based on pLM Features

Although, the GNN model outperforms the SVM model as overall, the SVM model, nevertheless, beats the GNN in the Top 5%. It could mean that SVM can perform better and rank better than GNN while predicting allosteric pockets with higher probability.

The AutoML and XGBoost models perform similarly with slight differences at Top 1% and Top 3% with XGBoost being at advantage. However, overall and in Top 10% and Top 20% XGboost beats AutoML and all other models. Moreover, after Top 10% all models give the same result being said that all positive or predicted allosteric sites are ranked in Top 10% of the results.

Table 4.28: Ranking of Allosteric Sites in Different Models based on pLM Features

Models	Top 1%	Top 3%	Top 5%	Top 10%	Top 20%
SVM	14.29%	40.0%	55.71%	55.71%	55.71%
GNN	11.43%	31.43%	52.86%	68.57%	68.57%
AutoML	14.29%	38.57%	65.71%	67.14%	67.14%
XGBoost	15.71%	44.29%	65.71%	71.43%	71.43%

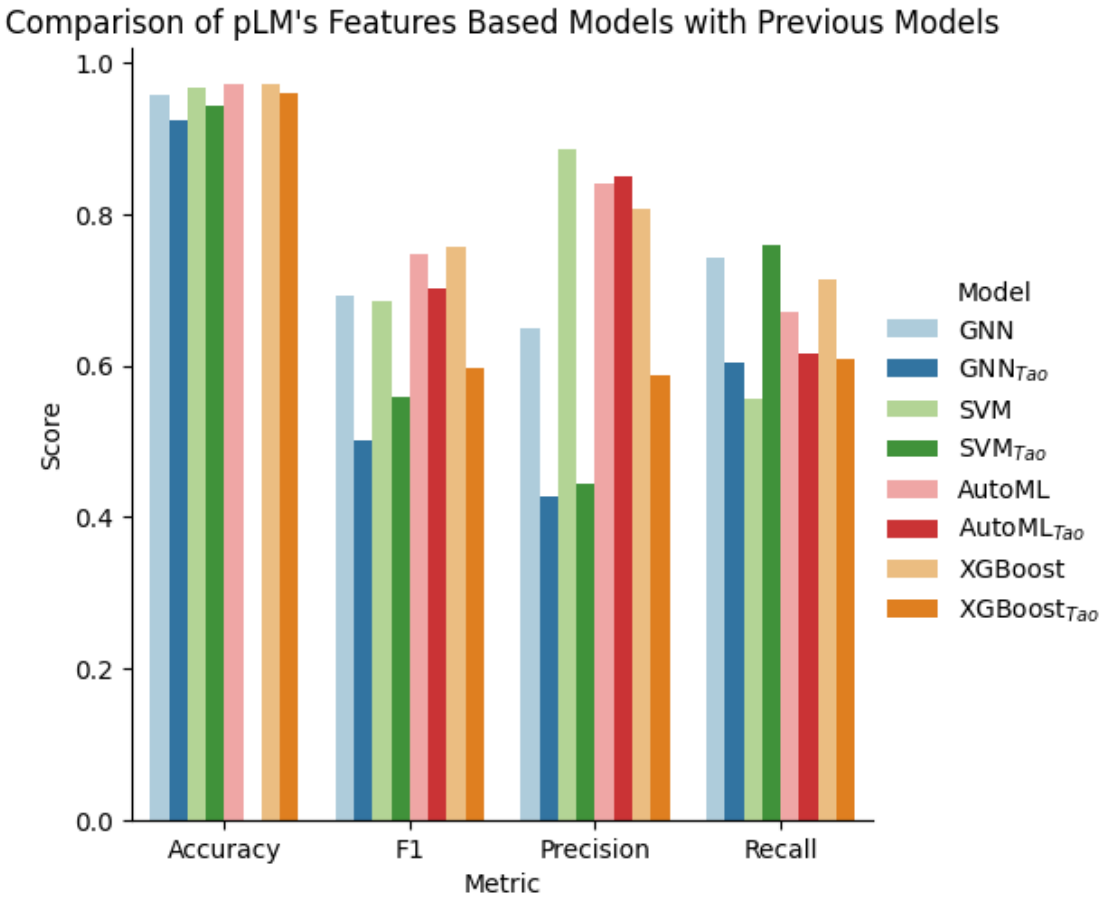

Figure 4.26: Comparison of pLM's Features Based Models with Previous Models

Previous studies have used only pocket features to predict allosteric sites while the models in this research have utilized the pLM features. As the models perform better with

the combination of pLM and pocket features; the result are compared for the models trained on pLM and pocket features with one exception of the GNN that was trained only on the pLM features.

Figure 4.26 gives the pairwise comparison between pLM based models trained in this thesis and previous models from Peng Tao group [41] trained only on pocket features. Models are shown in pair of colors where light colors show the models from this study while the dark colors show the models from Tao group. Models from Tao group have "*Tao"* subscript in the legend.

All models from this study outperform previous models with GNN, XGBoost, and SVM showing an average of ~20% increase in the F1 score while the AutoML's performance slightly increased. Table 4.29 gives the numeric comparison of models based on pLM features with models from previous studies. The table is divided into groups of two rows such that each two-row group compares model from previous studies with the model from the current research. In each row group, higher scores are in bold for each metric.

Table 4.29: Overall Comparison of pLM based Models with Previous Studies

Models	Average	F1	Precision	Recall
SVM_{Tao}	0.944	0.559	0.444	**0.758**
SVM	**0.9674**	**0.6842**	**0.8864**	0.5571
$AutoML_{Tao}$	–	0.701	**0.850**	0.616
AutoML	**0.9710**	**0.7460**	0.8393	**0.6715**
$XGBoost_{Tao}$	0.961	0.596	0.586	0.609
XGBoost	**0.9710**	**0.7576**	**0.8065**	**0.7143**
GNN_{Tao}	0.923	0.5	0.427	0.604
GNN	**0.9584**	**0.6933**	**0.6500**	**0.7429**
$Ensemble_{Tao}$	0.974	0.782	0.726	0.847
ProtBERT	0.9548	0.6154	0.6667	0.5714

As highlighted in the earlier sections that dataset was highly imbalanced, F1 score was used to evaluate models in combination with precision and recall scores. Models from this study outperform models from previous studies.

In the second last row of the Table 4.29, Ensemble model from Tao group [41] is listed which shows the highest F1 score of 0.782. This ensemble model is based on XGBoost and GCN from their paper [38] in 2021, where the XGBoost gave the 0.764 F1 score and GCN gave 0.5 F1 score. However, in one of the recent studies [40] from the same group in 2023, XGBoost's new F1 score decreased to 0.596 from the original score of 0.764. As the ensemble model was based on the older XGBoost model from 2021, the newer results from 2023 might have also impacted the performance of ensemble model; however, the Ensemble model's results based on the latest XGBoost model are not available and only the original results were reported in their latest paper [41]. The source code of the ensemble model is not available; manual evaluation was tried; however, it does not give complete information using which the newer results could be updated.

In the last row, ProtBERT model is given which was used to predict pockets using some heuristics made on validation data. After fine-tuning the pLM to classify residues, each predicted residue is then mapped to its relative pocket. If a pocket has at least 8 allosteric resides, this pocket is marked as allosteric. F1 score is generated by this methodology and this score is surprisingly good, as prediction is being performed using directly the pLM.

As XGBoost is the highest performing method in this study, its performance was evaluated to check how it ranks the pockets in each protein. For each protein, pockets were ranked in descending order of probabilities. 67.35% of allosteric pockets were predicted as the first position, 91.84% were in the second position, and 95.92% were among the top three positions. It means that there is 95.92% probability that a predicted allosteric pocket will be in the top 3 positions among all detected pockets in a certain protein. Top 3 score in this study surpassed the top 3 score reported by Tao group.

Table 4.30: Probabilities of Predicting Allosteric Sites in Top 3 Positions

	Top 1	Top 2	Top 3
LTR [40]	59.5%	73.9%	83.6%
Ensemble [41]	60.7%	81.6%	84.9%
XGBoost	**67.35%**	**91.84%**	**95.92%**

4.9 Case Study

4.9.1 Known Allosteric Sites Prediction

For this case study, five proteins were selected that were not in any of the datasets, however, these were put aside from the source dataset having known allosteric sites. These proteins have PDB IDs as: 1Q5O, 2FPL, 3BCR, 3PEE, and 4HO6. XGBoost model was used to predict the allosteric pockets. As the model's predicted pockets would be in top 3 positions; top 3 pockets were selected by ordering predicted pockets in the descending order of probabilities.

In Figure 4.27, predicted pockets are marked in red, orange, and purple color with probability shown next to each pocket in the same color. Red color shows the pocket with the highest probability, orange with the next highest probability, and the purple shows the third highest probability of an allosteric pocket. Moreover, the modulators are shown in green color. PDB IDs of these proteins are: (A) 1Q5O, (B) 2FPL, (C) 3BCR, (D) 3PEE, and (E) 4HO6. Figure 4.27A, Figure 4.27B, and Figure 4.27E show the allosteric pockets predicted with ~99% in red color and remaining 2, in each protein, have less than 0.02% probability.

In Figure 4.27D, highest probability is 27.39% in red color. As the model was evaluated on 50% threshold i.e., a pocket must have at least 0.5 probability to be predicted as allosteric, this pocket would be considered as negative. However, it is a true allosteric pocket and as one of the top 3 pockets is being chosen as allosteric, this is an allosteric pocket. 3PEE shown in Figure 4.27C, gives top 3 pockets with ~99%, ~98%, and ~89% probabilities. Although all three pockets have high probabilities, pocket with the highest probability will be selected as allosteric pocket, which is a true allosteric pocket.

It can be observed in Figure 4.27 that all predicted true allosteric pockets are near to the modulator although one of these has a low probability; hence the pocket with the highest probability was selected as the allosteric pocket. Moreover, third pocket (purple color) is extremely far from the modulator, which is a proof that the model learns to predict correct allosteric pockets and takes into consideration the distance between the modulator and the allosteric pocket. It can be conjectured that the pLM features provide geometrical aspects of a protein and consequently help a model to differentiate between residues near the modulator and residues far from the modulator.

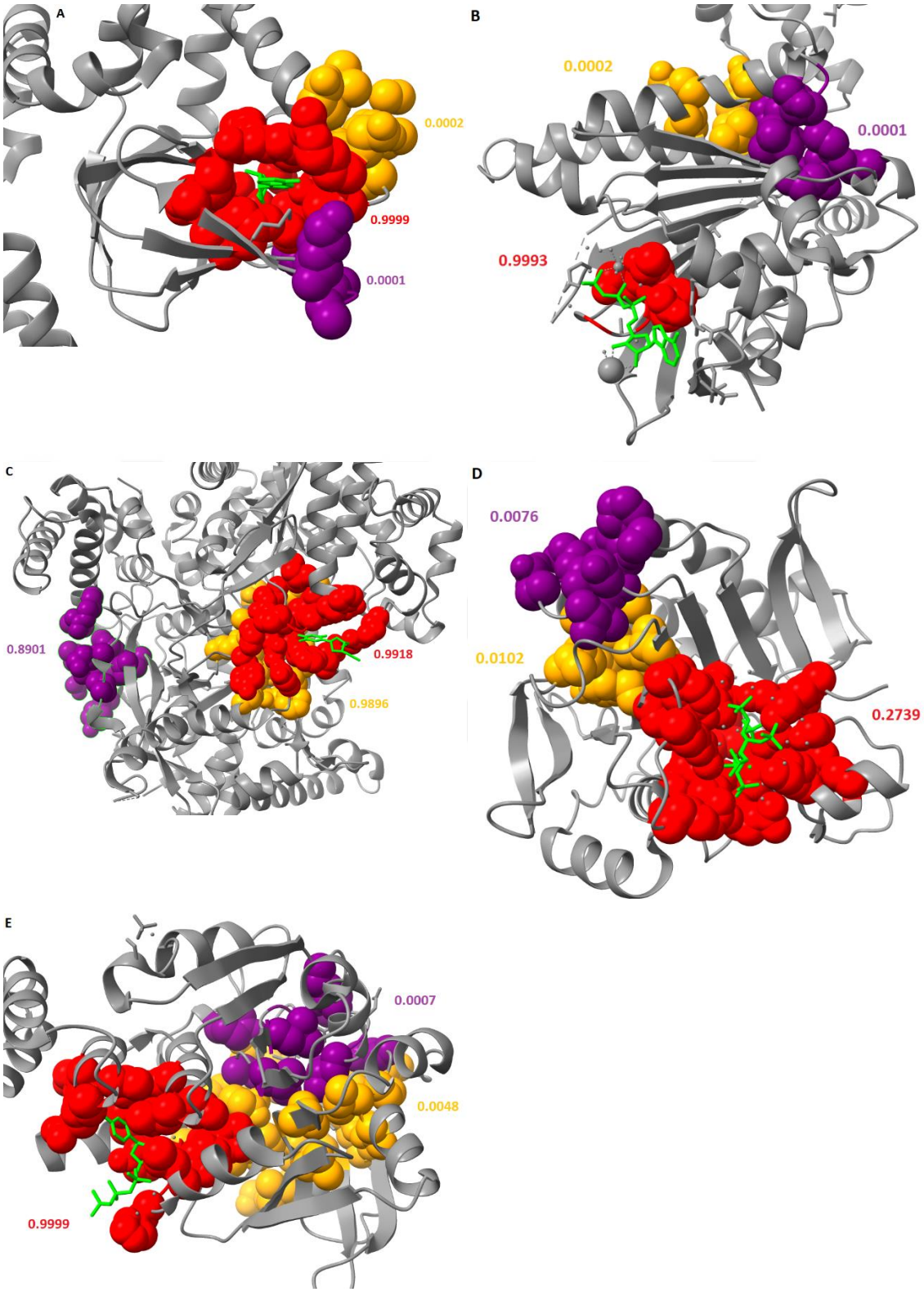

Figure 4.27: Known Allosteric Sites Prediction with Probabilities

4.9.2 Novel Allosteric Sites Prediction

For novel allosteric sites predictions, two proteins were selected (as chosen by Tian et al. [38]) that were not in any of the datasets having PDB IDs as 5DKK and 3LNY.

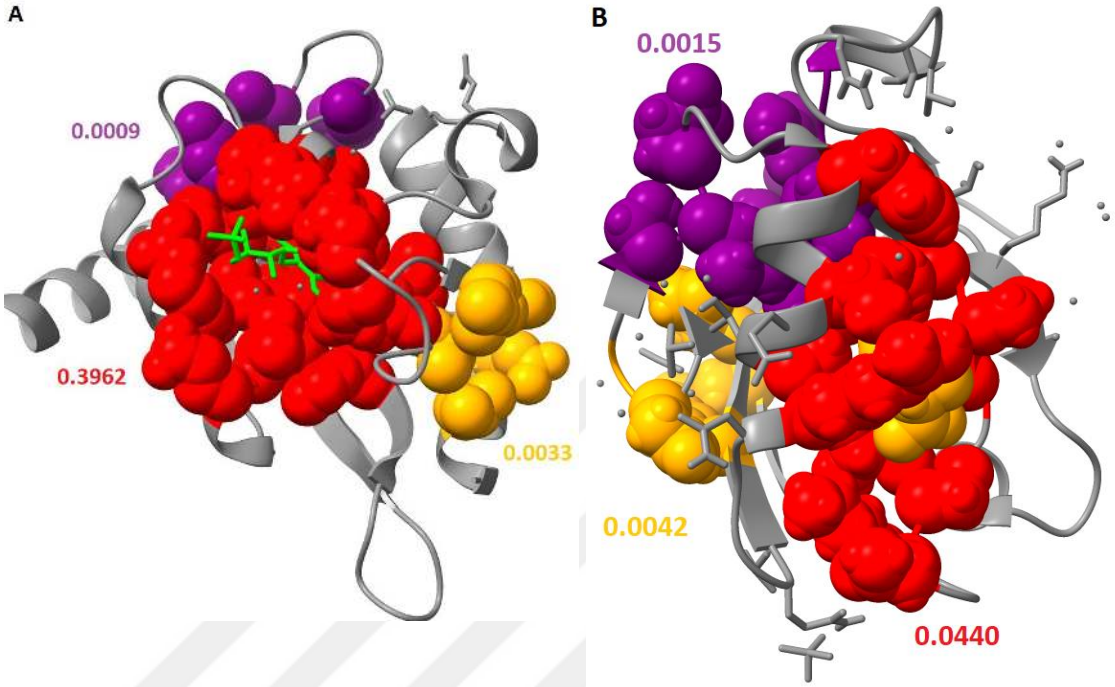

Figure 4.28: Novel Allosteric Sites Prediction with Probabilities

These proteins represent two different types of allosteric proteins. In the case of conformational-driven allosteric protein shown in Figure 4.28A, the model predicts the correct allosteric site and ranks the allosteric site as the top 1 with the 39.62% probability. The predicted allosteric site is the nearest to the modulator of the protein. In the case of dynamics-driven allosteric protein (3LNY), top 1 pocket has the 4.4% probability that is not a correct allosteric pocket. Moreover, the second ranked predicted sites' molecules are mixed with the first pocket. This could be coming from the FPocket, as the model runs on top of predicted pockets through FPocket. Moreover, from this protein, it may be assumed that the model does not perform well in case of dynamics-driven proteins.

4.9.3 ProtBERT-BFD Explanation and Visualization

ProtBERT has 30 layers and 16 heads, resulting in a total of $30 \times 16 = 480$ distinct attention mechanisms. ProtBERT's attention at different layers was also visualized in order to understand which head at which layer attends and which attention mechanism is followed. The visualization was performed for 5DKK protein. In all figures, pocket

residues were postfixed by an “underscore” for visualization purposes. Attention is being visualized as lines connecting the residue being updated (left) with the residue being attended to. Color intensity reflects the attention weight; weights close to one are shown as dark lines, whereas weights close to zero appear as faint lines or not visible at all.

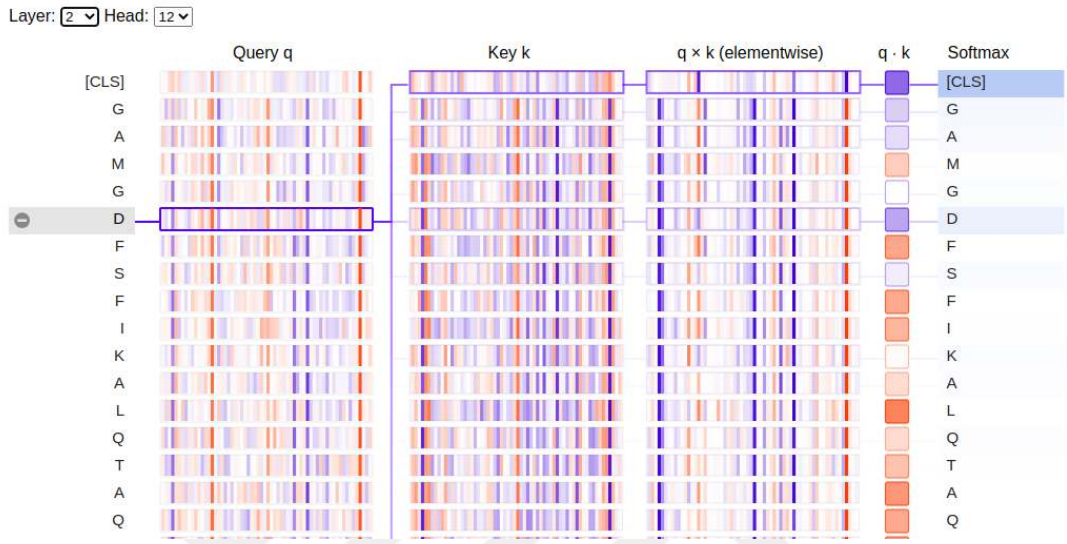

Figure 4.29: ProtBERT Neuron View

Figure 4.29 shows an example of selecting a layer 2 and head 12 (layers and heads are 0-indexed). It shows D is attending to *[CLS]* token as q.k column has the most intense blue color. Here, query vector q encodes the residue on the left that is paying attention and the key vector k encodes the residue on the right to which the attention is being paid. qxk is elementwise product between query and key whereas q.k is scaled dot product. Orange color represents negative value while blue color represents positive value.

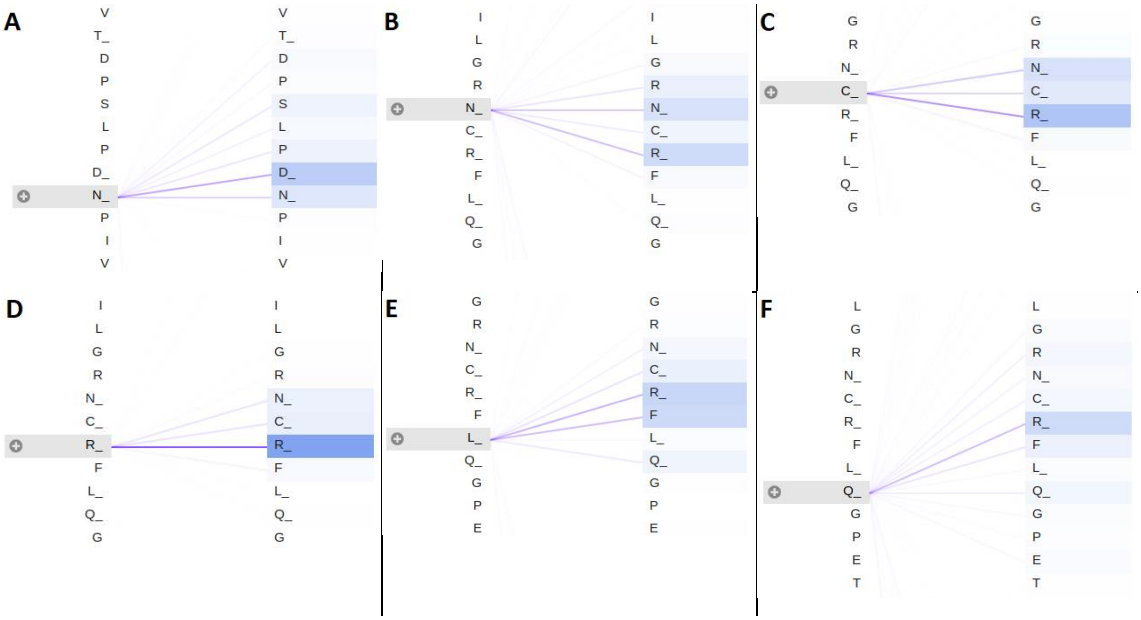

Figure 4.30: ProtBERT Attention at Layer 29 Head 12

In Figure 4.30, attention to related/identical words (residues) and site residues is being paid. This mechanism at layer 29 and head 12, can identify pocket residues (postfixed with _). Figure 4.30A shows that N and D residues (words) attend to each other the most (visualized by higher intensity color). It can be gathered that they are next to each other and biologically they are augmented together as "B" by Asparagine or Aspartic acid; if found in a pocket, can be identified by the model. Moreover, Figure 4.30 (A-F) show that although amino acids have different structures and molecular formulas, they attend to pocket residues rendering the pLM to identify pocket residues. Cysteine (C) attends to N, C, and R in Figure 4.30C, could be the indicator of high resulting probability by being adjacent in a sequence consequently being identified as pocket residue. In Figure 4.30F it can be seen that, Glutamine (Q) attends to Arginine (R) even though it is far behind in the sequence from Q; indicating that the pLM can capture long-range dependencies (relationships) among residues in a certain pocket.

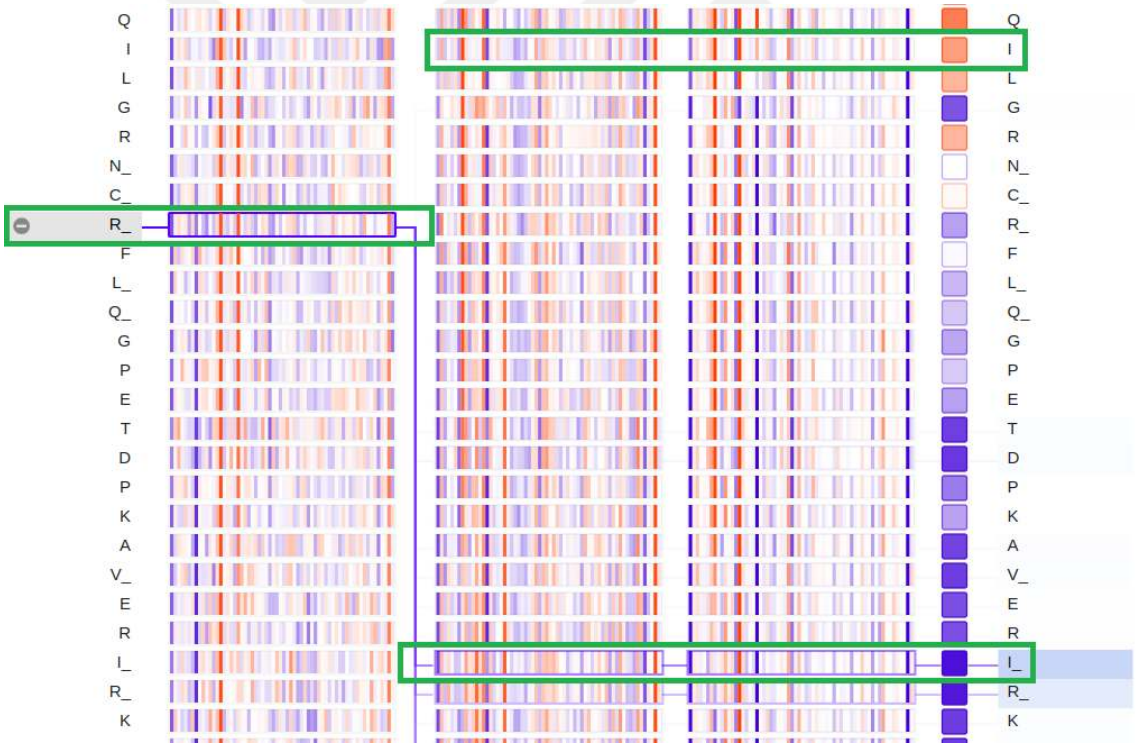

Figure 4.31: ProtBERT Neuron View - Layer 7 Head 3

Figure 4.31 gives the neuron view at layer 7 and head 3. Residues in question are highlighted with green rectangles. Residue *R_* has same Euclidean distance of 10.5 Å from both residues *I_* and *I*, however, it gives attention to *I_* residue and does not attend to residue *I* (non-allosteric residue). Although, the geometrical distance is same, it can

capture the allosteric pocket residues even though, in the sequence, *I_* is at a long distance from *R_* compared to the distance from *I*.

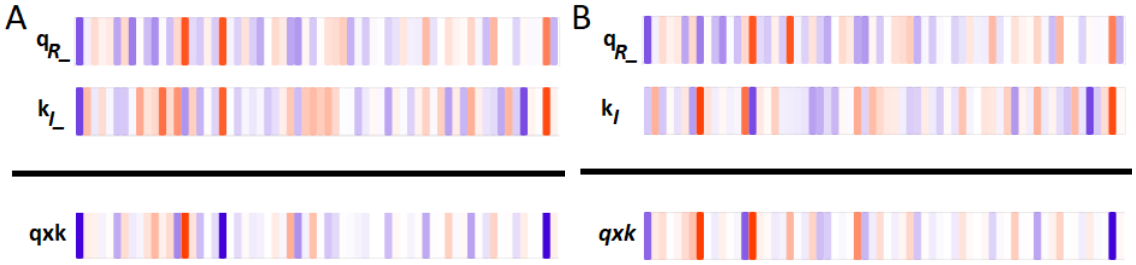

Figure 4.32: q-k product tends to be positive when a residue is an allosteric residue, negative otherwise

Figure 4.32A shows the query-key product between *R_* and *I_* while Figure 4.32B gives the query-key product between *R_* and *I*. When the *R* and *I* are both from the same pocket (and that pocket is allosteric), the product is positive. If the residue belongs to allosteric pocket, and *I* residue is outside the pocket region, the product is negative. A small number of neurons dominate the calculation of q and k in this pattern. As can be seen that, neurons 4 to 6 and a neuron just before the middle are different in both products, hence they have deciding role in this product.

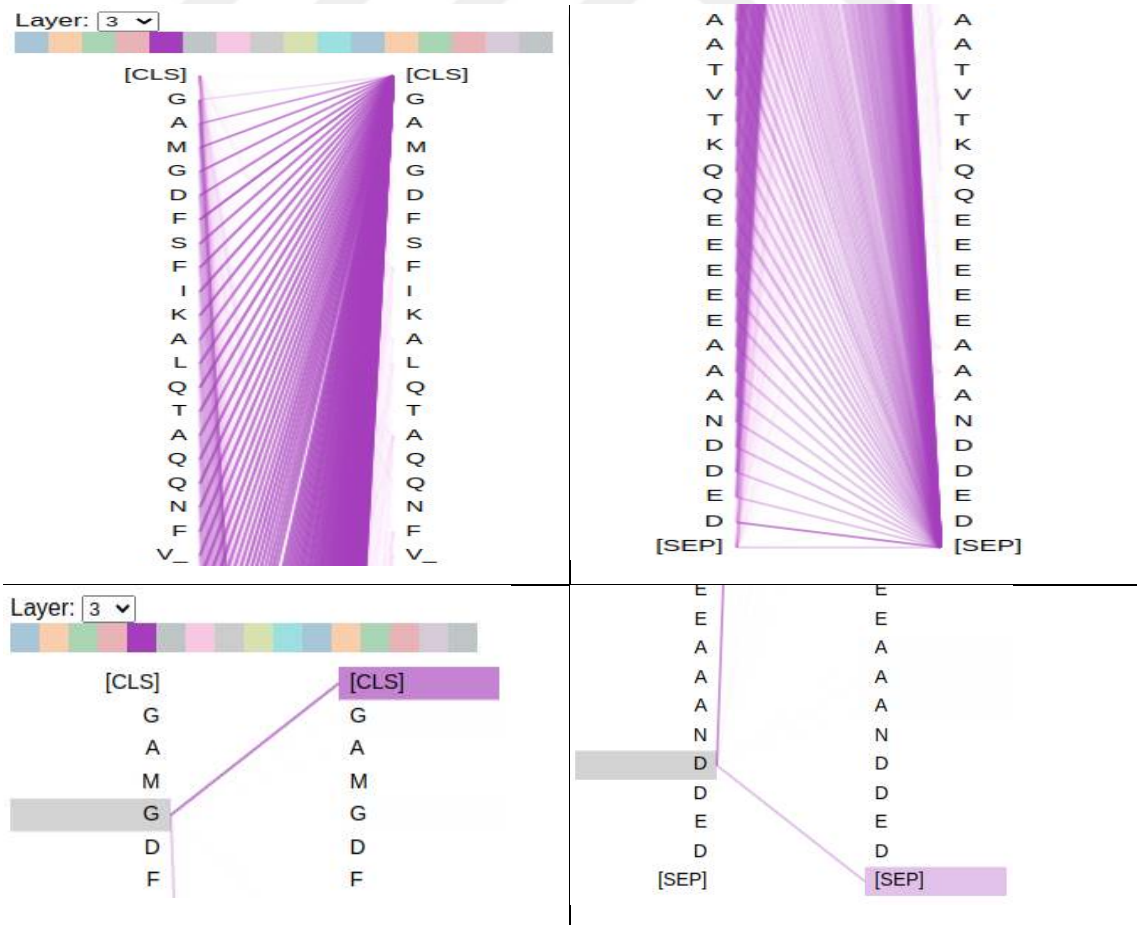

*Figure 4.33: Attention to delimiter tokens. **Top**: attention weights for all tokens **Bottom**: attention weights for selected tokens*

In the pattern shown in Figure 4.33, most of the attention is directed towards delimiter tokens i.e. *[CLS]* and *[SEP]*. An attention head focuses on *[SEP]* token when it cannot find anything meaningful in the input sequence to focus on [80].

To conclude, the pLM mostly focuses on delimiter tokens in the initial layers and as the input passes through deep layers, the model can make better connections among the residues (tokens). Several other patterns also make their presence known in the model such as attending to previous residue in the sequence and attending to next residue in the sequence. Interpreting from visualizations is a little bit like Rorschach tests [81]: *"our interpretations may be colored by our own beliefs and expectations"*. Patterns discussed above show distinct information, however, some patterns' interpretation could be subjective.

Chapter 5:
CONCLUSION

"The important thing is not to stop questioning. Curiosity has its own reason for existing. One cannot help but be in awe when he contemplates the mysteries of eternity, of life, of the marvelous structure of reality."

— A. EINSTEIN

After diving deep into protein allosteric sites prediction, this study aimed to shed light on leveraging pLMs and combining their features with ML and DL approaches to improve performance of these models to predict allosteric sites. To conclude, it is time to summarize the key findings and their significance for future work. As in the literature, ML and NMA approaches have been utilized to predict allosteric sites in proteins; pLMs have never been utilized in this domain of study, hence posed a challenge for their utilization and with the availability of datasets gave an opportunity to employ them to scrutinize contemporary methods to bring probable improvement.

ProtBERT-BFD pLM was finetuned on the allosteric proteins dataset which would be used as feature extractor for several ML and DL methods. FPocket was used to extract pockets from allosteric proteins that would be used as input for ML and DL models. pLM features combined with FPocket features were fed into SVM, AutoML, GNN, and XGBoost models. As in the literature, these models were significant (based only on the FPocket features); utilization of pLM features and amalgamation with FPocket features improved the overall prediction performance of allosteric sites.

As the dataset was highly imbalanced, aforementioned models were evaluated on F1 score and XGBoost model gave the highest prediction performance by achieving an F1 score of 75.76%. Collectively, all methods employed in this study improved over previous methods and gained ~20% advantage in F1 score. XGBoost, being the supreme model in this study, can predict allosteric sites in top 3 positions with ~95% confidence.

Additionally, case studies were performed on 5 proteins with known allosteric sites and 2 proteins with unknown allosteric sites. 4 out of 5 proteins' top 1 allosteric sites were predicted with 99% confidence. In case of protein with unknown allosteric sites, correct allosteric site was predicted proving that the pLM feature do improve prediction performance and would help researchers aiming to extend and/or investigate prediction of allosteric sites in proteins.

Interpretation and demystification of a DL model is an important aspect of any research employing such models. An effort was made to explain the pLM and how it captures the allosteric residues. Several attention mechanisms were identified including but not limited to *attention towards identical/similar residues* and *attention towards structurally different but molecularly similar residues*. Visual interpretation from these patterns shows distinct information, however, the perceived information could be subjective.

In subsequent research, diving into more sophisticated pLM models such as ProtT5 is expected to further improve the prediction performance of allosteric sites. DL models are data-hungry; with a current opportunity to build synthetic datasets and availability of experimental datasets in the near future, DL models are envisaged to improve over the methods studied in this research.

BIBLIOGRAPHY

[1] A. W. Fenton, "Allostery: an illustrated definition for the 'second secret of life'," *Trends in biochemical sciences,* vol. 33, pp. 420-425, 9 2008.

[2] J. Monod, Chance and necessity an essay on the natural philosophy of modern biology, 1971.

[3] J. Monod, J.-P. Changeux and F. Jacob, "Allosteric proteins and cellular control systems," *Journal of molecular biology,* vol. 6, no. 4, pp. 306--329, 1963.

[4] G. Weber, "Ligand binding and internal equilibiums in proteins," *Biochemistry,* vol. 11, no. 5, pp. 864--878, 1972.

[5] G. D. Reinhart, "Quantitative analysis and interpretation of allosteric behavior," *Methods in enzymology,* vol. 380, pp. 187-203, 2004.

[6] R. Nussinov and J. Liu, "Allostery: An Overview of Its History, Concepts, Methods, and Applications," *PLOS Computational Biology,* vol. 12, pp. 1-5, 6 2016.

[7] J. Zha, M. Li, R. Kong, S. Lu and J. Zhang, "Explaining and predicting allostery with allosteric database and modern analytical techniques," *Journal of Molecular Biology,* p. 167481, 2022.

[8] S. Lu, Q. Shen and J. Zhang, "Allosteric methods and their applications: facilitating the discovery of allosteric drugs and the investigation of allosteric mechanisms," *Accounts of Chemical Research,* vol. 52, no. 2, pp. 492-500, 2019.

[9] T. Haliloglu, A. Hacisuleyman and B. Erman, "Prediction of allosteric communication pathways in proteins," *Bioinformatics,* vol. 38, no. 14, pp. 3590-3599, 2022.

[10] J. Monod, J. Wyman and J. Changeux, "On the structure of allosteric transitions: a plausible model," *Journal of Molecular Biology,* vol. 12, pp. 88-118, 1965.

[11] D. E. Koshland, "Conformational changes: how small is big enough?," *Nature medicine,* vol. 4, no. 10, pp. 1112-1114, 1998.

[12] J.-P. Changeux, "Allostery and the Monod-Wyman-Changeux model after 50 years," *Annual review of biophysics,* vol. 41, pp. 103-133, 2012.

[13] J.-P. Changeux and S. J. Edelstein, "Allosteric mechanisms of signal transduction," *Science,* vol. 308, no. 5727, pp. 1424-1428, 2005.

[14] N. Popovych, S. Sun, R. H. Ebright and C. G. Kalodimos, "Dynamically driven protein allostery," *Nature structural & molecular biology,* vol. 13, no. 9, pp. 831-838, 2006.

[15] K. Gunasekaran, B. Ma and R. Nussinov, "Is allostery an intrinsic property of all dynamic proteins?," *Proteins: Structure, Function, and Bioinformatics,* vol. 57, no. 3, pp. 433-443, 2004.

[16] C.-J. Tsai, A. Del Sol and R. Nussinov, "Allostery: absence of a change in shape does not imply that allostery is not at play," *Journal of molecular biology,* vol. 378, no. 1, pp. 1-11, 2008.

[17] H. N. Motlagh, J. O. Wrabl, J. Li and V. J. Hilser, "The ensemble nature of allostery," *Nature,* vol. 508, no. 7496, pp. 331-339, 2014.

[18] L. A. Freiburger, O. M. Baettig, T. Sprules, A. M. Berghuis, K. Auclair and A. K. Mittermaier, "Competing allosteric mechanisms modulate substrate binding in a dimeric enzyme," *Nature structural & molecular biology,* vol. 18, no. 3, pp. 288-294, 2011.

[19] R. Nussinov, C.-J. Tsai and B. Ma, "The underappreciated role of allostery in the cellular network," *Annual review of biophysics,* vol. 42, pp. 169-189, 2013.

[20] S. Grutsch, S. Brüschweiler and M. Tollinger, "NMR methods to study dynamic allostery," *PLoS computational biology,* vol. 12, no. 3, p. e1004620, 2016.

[21] E. Laine, C. Goncalves, J. C. Karst, A. Lesnard, S. Rault, W.-J. Tang, T. Malliavin, D. Ladant and A. Blondel, "Use of allostery to identify inhibitors of calmodulin-induced activation of Bacillus anthracis edema factor," *Proceedings of the National Academy of Sciences,* vol. 107, no. 25, pp. 11277-11282, 2010.

[22] A. Panjkovich and X. Daura, "Exploiting protein flexibility to predict the location of allosteric sites," *BMC bioinformatics,* vol. 13, pp. 1-12, 2012.

[23] B. R. Amor, M. T. Schaub, S. N. Yaliraki and M. Barahona, "Prediction of allosteric sites and mediating interactions through bond-to-bond propensities," *Nature communications,* vol. 7, no. 1, p. 12477, 2016.

[24] Y. Bian, Y. Jing, L. Wang, S. Ma, J. J. Jun and X.-Q. Xie, "Prediction of orthosteric and allosteric regulations on cannabinoid receptors using supervised machine learning classifiers," *Molecular pharmaceutics,* vol. 16, no. 6, pp. 2605-2615, 2019.

[25] W. Huang, S. Lu, Z. Huang, X. Liu, L. Mou, Y. Luo, Y. Zhao, Y. Liu, Z. Chen and T. Hou, "Allosite: a method for predicting allosteric sites," *Bioinformatics,* vol. 29, no. 18, pp. 2357-2359, 2013.

[26] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, W. Yu, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik and B. Rost, "ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 1-1, 2021.

[27] T. U. Consortium, "UniProt: a worldwide hub of protein knowledge," *Nucleic Acids Research,* vol. 47, no. D1, pp. D506-D515, 10.1093/nar/gky1049 2018.

[28] B. E. Suzek, Y. Wang, H. Huang, P. B. McGarvey, C. H. Wu and T. U. Consortium, "UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches," *Bioinformatics,* vol. 31, no. 6, pp. 926-932, 11 2014.

[29] M. Steinegger, M. Mirdita and J. Soding, "Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold," *Nature methods,* vol. 16, no. 7, pp. 603-606, 2019.

[30] M. Steinegger and J. Soding, "Clustering huge protein sequence sets in linear time," *Nature communications,* vol. 9, no. 1, p. 2542, 2018.

[31] Z. Huang, L. Zhu, Y. Cao, G. Wu, X. Liu, Y. Chen, Q. Wang, T. Shi, Y. Zhao and Y. Wang, "ASD: a comprehensive database of allosteric proteins and modulators," *Nucleic acids research,* vol. 39, pp. D663--D669, 2011.

[32] V. Le Guilloux, P. Schmidtke and P. Tuffery, "Fpocket: an open source platform for ligand pocket detection," *BMC bioinformatics,* vol. 10, no. 1, pp. 1-11, 2009.

[33] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016.

[34] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters,* pp. 293-300, 1999.

[35] F. Hutter, L. Kotthoff and J. Vanschoren, "Automated machine learning: methods, systems, challenges," 2019.

[36] S. Haykin, "Neural networks: a comprehensive foundation," 1994.

[37] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907,* 2016.

[38] H. Tian, X. Jiang and P. Tao, "PASSer: Prediction of allosteric sites server," *Machine learning: science and technology,* vol. 2, no. 3, p. 035015, 2021.

[39] S. Xiao, H. Tian and P. Tao, "PASSer2.0: accurate prediction of protein allosteric sites through automated machine learning," *Frontiers in Molecular Biosciences,* vol. 9, p. 879251, 2022.

[40] H. Tian, S. Xiao, X. Jiang and P. Tao, "PASSerRank: prediction of allosteric sites with learning to rank," *arXiv preprint arXiv:2302.01117,* 2023.

[41] H. Tian, S. Xiao, X. Jiang and P. Tao, "PASSer: fast and accurate prediction of protein allosteric sites," *Nucleic Acids Research,* p. gkad303, 2023.

[42] H. Taketomi, Y. Ueda and N. Gō, "STUDIES ON PROTEIN FOLDING, UNFOLDING AND FLUCTUATIONS BY COMPUTER SIMULATION: I. The effect of specific amino acid sequence represented by specific inter-unit interactions," *International journal of peptide and protein research,* vol. 7, no. 6, pp. 445-459, 1975.

[43] J. Wang, A. Jain, L. R. McDonald, C. Gambogi, A. L. Lee and N. V. Dokholyan, "Mapping allosteric communications within individual proteins," *Nature communications,* vol. 11, no. 1, p. 3862, 2020.

[44] J. G. Greener and M. J. Sternberg, "AlloPred: prediction of allosteric pockets on proteins using normal mode perturbation analysis," *BMC bioinformatics,* vol. 16, no. 1, pp. 1-7, 2015.

[45] K. Song, X. Liu, W. Huang, S. Lu, Q. Shen, L. Zhang and J. Zhang, "Improved method for the identification and validation of allosteric sites," *Journal of Chemical Information and Modeling,* vol. 57, no. 9, pp. 2358-2363, 2017.

[46] A. Goncearenco, S. Mitternacht, T. Yong, B. Eisenhaber, F. Eisenhaber and I. N. Berezovsky, "SPACER: server for predicting allosteric communication and effects of regulation," *Nucleic acids research,* vol. 41, no. W1, pp. W266-W272, 2013.

[47] A. Panjkovich and X. Daura, "PARS: a web server for the prediction of protein allosteric and regulatory sites," *Bioinformatics,* vol. 30, no. 9, pp. 1314-1315, 2014.

[48] S. Lu, W. Huang and J. Zhang, "Recent computational advances in the identification of allosteric sites in proteins," *Drug discovery today,* vol. 19, no. 10, pp. 1595-1600, 2014.

[49] Q. Zhang, C. D. Heldermon and C. Toler-Franklin, "Multiscale detection of cancerous tissue in high resolution slide scans," *International Symposium on Visual Computing,* pp. 139-153, 2020.

[50] L. Chen, Y. Lu, C.-T. Wu, R. Clarke, G. Yu, J. E. Van Eyk, D. M. Herrington and Y. Wang, "Data-driven detection of subtype-specific differentially expressed genes," *Scientific reports,* vol. 11, no. 1, p. 332, 2021.

[51] H. Tian, F. Trozzi, B. D. Zoltowski and P. Tao, "Deciphering the allosteric process of the Phaeodactylum tricornutum Aureochrome 1a LOV domain," *The Journal of Physical Chemistry B,* vol. 124, no. 41, pp. 8960-8972, 2020.

[52] H. Tian, X. Jiang, F. Trozzi, S. Xiao, E. C. Larson and P. Tao, "Explore protein conformational space with variational autoencoder," *Frontiers in molecular biosciences,* vol. 8, p. 781635, 2021.

[53] S. W. Lockless and R. Ranganathan, "Evolutionarily conserved pathways of energetic connectivity in protein families," *Science,* vol. 286, no. 5438, pp. 295-299, 1999.

[54] K. A. Reynolds, R. N. McLaughlin and R. Ranganathan, "Hot spots for allosteric regulation on protein surfaces," *Cell,* vol. 147, no. 7, pp. 1564-1575, 2011.

[55] R. N. McLaughlin Jr, F. J. Poelwijk, A. Raman, W. S. Gosal and R. Ranganathan, "The spatial architecture of protein function and adaptation," *Nature,* vol. 491, no. 7422, pp. 138-142, 2012.

[56] M. Sahún-Roncero, B. Rubio-Ruiz, G. Saladino, A. Conejo García, A. Espinosa, A. Velázquez-Campoy, F. L. Gervasio, A. Entrena and R. Hurtado-Guerrero, "The mechanism of allosteric coupling in choline kinase α1 revealed by the action of a rationally designed inhibitor," *Angewandte Chemie International Edition,* vol. 52, no. 17, pp. 4582-4586, 2013.

[57] Y. Shan, E. T. Kim, M. P. Eastwood, R. O. Dror, M. A. Seeliger and D. E. Shaw, "How does a drug molecule find its target binding site?," *Journal of the American Chemical Society,* vol. 133, no. 24, pp. 9181-9183, 2011.

[58] R. O. Dror, H. F. Green, C. Valant, D. W. Borhani, J. R. Valcourt, A. C. Pan, D. H. Arlow, M. Canals, J. R. Lane, R. Rahmani, J. B. Baell, P. M. Sexton, A. Christopoulos and D. E. Shaw, "Structural basis for modulation of a G-protein-coupled receptor by allosteric drugs," *Nature,* vol. 503, no. 7475, pp. 295-299, 2013.

[59] D. Shukla, Y. Meng, B. Roux and V. S. Pande, "Activation pathway of Src kinase reveals intermediate states as targets for drug design," *Nature communications,* vol. 5, no. 1, p. 3397, 2014.

[60] Y. Qi, Q. Wang, B. Tang and L. Lai, "Identifying allosteric binding sites in proteins with a two-state Go model for novel allosteric effector discovery," *Journal of chemical theory and computation,* vol. 8, no. 8, pp. 2962-2971, 2012.

[61] S. Mitternacht and I. N. Berezovsky, "Binding leverage as a molecular basis for allosteric regulation," *PLoS computational biology,* vol. 7, no. 9, p. e1002148, 2011.

[62] A. S.-Y. Chen, N. J. Westwood, P. Brear, G. W. Rogers, L. Mavridis and J. B. Mitchell, "A random forest model for predicting allosteric and functional sites on proteins," *Molecular informatics,* vol. 35, no. 3-4, pp. 125-135, 2016.

[63] A. Liaw and M. Wiener, "Classification and regression by RandomForest," *R news,* vol. 2, no. 3, pp. 18-22, 2002.

[64] W. Huang, G. Wang, Q. Shen, X. Liu, S. Lu, L. Geng, Z. Huang and J. Zhang, "ASBench: benchmarking sets for allosteric discovery," *Bioinformatics,* vol. 31, no. 15, pp. 2598-2600, 2015.

[65] H. Jin, Q. Song and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 1946-1956.

[66] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li and A. Smola, "Autogluon-tabular: Robust and accurate automl for structured data," *arXiv preprint arXiv:2003.06505,* 2020.

[67] J. An, M. Totrov and R. Abagyan, "Pocketome via comprehensive identification and classification of ligand binding envelopes," *Molecular & Cellular Proteomics,* vol. 4, no. 6, pp. 752-761, 2005.

[68] A. Trotman, "Learning to rank," *Information Retrieval,* vol. 8, pp. 359-381, 2005.

[69] A. S. Zlobin, D. Suplatov, K. Kopylov and V. Švedas, "CASBench: a benchmarking set of proteins with annotated catalytic and allosteric sites in their structures," *Acta Naturae,* vol. 11, no. 1 (40), pp. 74-80, 2019.

[70] S. Wang, W. Chen, P. Han, X. Li and T. Song, "RGN: Residue-Based Graph Attention and Convolutional Network for Protein--Protein Interaction Site Prediction," *Journal of Chemical Information and Modeling,* vol. 62, no. 23, pp. 5961-5974, 2022.

[71] K. Jha, S. Saha and H. Singh, "Prediction of protein--protein interaction using graph neural networks," *Scientific Reports,* vol. 12, no. 1, p. 8360, 2022.

[72] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903,* 2017.

[73] 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.

[74] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang and Y. Sun, "Masked Label Prediction: Unified Message Passing Model for Semi-Supervised," *arXiv preprint arXiv:2009.03509,* 2020.

[75] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention Is All You Need," *Advances in neural information processing systems,* vol. 30, 2017.

[76] A. Hagberg, D. A. Schult and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," *Proceedings of the 7th Python in Science Conference (SciPy2008),* pp. 11-15, 2008.

[77] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research,* vol. 9, no. 86, pp. 2579-2605, 2008.

[78] J. Vig, A. Madani, L. R. Varshney, C. Xiong, R. Socher and N. F. Rajani, "BERTology Meets Biology: Interpreting Attention in Protein Language Models," *International Conference on Learning Representations,* 2021.

[79] A. R. Kinjo and H. Nakamura, "Comprehensive structural classification of ligand-binding motifs in proteins," *Structure,* vol. 17, no. 2, pp. 234-246, 2009.

[80] K. Clark, U. Khandelwal, O. Levy and C. D. Manning, "What Does BERT Look At? An Analysis of BERT's Attention," *arXiv preprint arXiv:1906.04341,* 2019.

[81] H. Rorschach, Psychodiagnostics, a diagnostic test based on perception, including Rorschach's paper, The application of the form interpretation test (published posthumously by Dr. Emil Oberholzer), Berne, Switzerland : New York, N. Y. :H. Huber; Grune & Stratton inc., 1942.

Appendix A: Models' Mathematics and Formulas

This section gives a brief description of models and the mathematics behind them.

A.1 XGBoost

Starting with XGBoost, a prediction on a sample (x_i, y_i) is performed by a decision tree *j* as follows:

$$g_j(x_i) = w_q(x_i)$$

where w_q is the weight matrix or vector of a leaf. XGBoost accumulates prediction of all M decision trees and makes the final prediction by the formula:

$$\widehat{y_i} = \sum_{j=1}^{M} g_j(x_i)$$

Moreover, XGBoost uses loss function l with regularization Ω to overcome overfitting. Objective function is calculated as:

$$\text{obj}(\theta) = \sum_{i=1}^{N} l(y_i, \widehat{y_i}) + \sum_{j=1}^{M} \Omega(f_i)$$

where $\Omega(f) = \gamma T + \frac{\lambda}{2}\sum_{l=1}^{T} \omega_l^2$ and T is total number of leaves and regularization parameters are represented by γ, λ.

Appendix B: Source Code Sample

The following code snippet gives the whole pipeline that performs the following steps:

1. Download the PDB file with given PDB ID
2. Extract pockets and their features from the downloaded structure by using FPocket.
3. Extract sequence from the PDB file.
4. Extract pLM features using the sequence from the fine-tuned ProtBERT model.
5. Prepare and concatenate the pocket and pLM features.
6. Predict pockets by using a specified trained model such as XGBoost.

```
import gc
import torch
import requests
import glob, os, math
import numpy as np
from transformers import BertModel, BertTokenizer
import xgboost as xgb

from utils.extract_sequence import extract_sequence
from utils.pocket_feature import pocket_feature
from utils.sequence_indices import sequence_indices
from utils.pocket_coordinates import pocket_coordinates

N_ATOMS = 9
MODEL_PATH = "../prot_bert_allosteric"
base_url = "https://files.rcsb.org/download"
pdb_dir = "../data/pdbs/"
pocket_dir = "../data/pockets/"
pdb_id = "3BCR"
chain_id = "A"

pdb_path = os.path.join(pdb_dir, f"{pdb_id}.pdb")
pocket_path = os.path.join(pocket_dir, f"{pdb_id}_out")
```

```
#### Test - begin ####
ASD_path = "../data/source_data/ASD_Release_201909_AS.txt"

asd = None
with open(ASD_path, "r") as f:
    asd = f.readlines()

mod_id, modulator, residues = None, None, None
for line in asd[1:]:
    line = line.strip().split("\t")
    pdb, modulator, chain_id, mod_id = line[4], line[6], line[7], line[11]

    if pdb != pdb_id:
        continue

    if len(set(chain_id.split(";"))) != 1:
        continue
    chain_id = chain_id[0]

    if len(set(modulator.split(";"))) != 1:
        continue
    modulator = modulator.split(";")[0]

    # extract residues
    res_raw = [
        res.replace(":", ",").split(",") for res in line[-1].split("; ")
    ]
    # residue_clean format: chain id + residue type + residue number
    residues = [
        [res[0][-1], ch[:3], ch[3:]] for res in res_raw for ch in res[1:]
    ]
    # select only residues in the same chain of modulator
    residues = [res for res in residues if res[0] == chain_id]

    break
#### Test - end ####


if not os.path.exists(pdb_path):
```

```
    response = requests.get(f"{base_url}/{pdb_id}.pdb")
    if response.status_code == 200:  # Check if the request was successful
        with open(pdb_path, 'wb') as file:
            file.write(response.content)
        print(f"PDB file {pdb_id}.pdb downloaded successfully.")
    else:
        raise Exception(f"Failed to download {pdb_id}.pdb. Check if the PDB
ID is correct.")

sequence = extract_sequence(pdb_path, chain_id)

if len(sequence) <= 10:
    raise Exception("Sequence is too short.")

if not os.path.exists(pocket_path):
    os.system(f"fpocket -f {pdb_path} -k {chain_id}")
    os.system(f"mv {os.path.join(pdb_dir, pdb_id)}_out {pocket_dir}")

#### Test - begin ####
protein = None
lig_x, lig_y, lig_z, lig_cnt = 0, 0, 0, 0

with open(pdb_path, "r") as f:
    protein = f.readlines()

for line in protein:
    if (
        line[:6] == "HETATM" and modulator == line[17:20].strip()
        and line[21] == chain_id and mod_id == line[22:26].strip()
    ):
        lig_x += float(line[30:38])
        lig_y += float(line[38:46])
        lig_z += float(line[46:54])
        lig_cnt += 1

lig_x /= lig_cnt
lig_y /= lig_cnt
lig_z /= lig_cnt
#### Test - end ####
```

```
pocket_names = glob.glob(f"{pocket_path}/pockets/*.pdb")
pocket_names = sorted(
    pocket_names,
    key=lambda x: int(x.split("pocket")[-1].split("_")[0])
)

pockets_feats = pocket_feature(f"{pocket_path}/{pdb_id}_info.txt")
selected_idxs = []
pocket_residue_indices = []

#### Test - begin ####
atomTarget = {}
for res in residues:
    atomTarget[f'{res[1]}{res[2]}'] = res[0]

dists = []
countsPockets = [] # for atom count
#### Test - end ####

for idx, pocket_name in enumerate(pocket_names):
    pocket = None
    with open(pocket_name, "r") as f:
        pocket = f.readlines()

#### Test - begin ####
    poc_x, poc_y, poc_z = 0, 0, 0
    pocketAtomCount = 0
#### Test - end ####

    poc_cnt = 0
    residue_indices = set()

    for line in pocket:
        if line[:4] == "ATOM":
            poc_cnt += 1
            residue_index = line[22:26].strip()
            atom = line[17:20] + residue_index
            residue_indices.add(residue_index)
```

```
#### Test - begin ####
            poc_x += float(line[30:38])
            poc_y += float(line[38:46])
            poc_z += float(line[46:54])
            chainID = line[21]
            if atom in atomTarget and atomTarget[atom] == chainID:
                pocketAtomCount += 1
#### Test - end ####

    if poc_cnt == 0:
        continue

#### Test - begin ####
    poc_x /= poc_cnt
    poc_y /= poc_cnt
    poc_z /= poc_cnt
    dist = math.sqrt(
        (poc_x - lig_x) ** 2 + (poc_y - lig_y) ** 2 +
        (poc_z - lig_z) ** 2
    )

    dists.append(dist)
    countsPockets.append(pocketAtomCount)
#### Test - end ####

    selected_idxs.append(idx)
    pocket_residue_indices.append(list(residue_indices))

if len(selected_idxs) <= 2:
    raise Exception("Too few pockets extracted.")

pocket_features = [pockets_feats[idx] for idx in selected_idxs]

seq_indices = sequence_indices(pdb_id, chain_id)

#### Test - begin ####
dist_min_idx = np.argmin(dists)
```

```
labels = [1 if item >= N_ATOMS else 0 for item in countsPockets] # for atom
count
labels[dist_min_idx] = 1

seq_labels = ['N'] * len(sequence)
for i in range(len(labels)):
        if labels[i] == 1:
            for residue_index in pocket_residue_indices[i]:
                if residue_index in seq_indices and
seq_indices[residue_index] < len(sequence):
                    seq_labels[seq_indices[residue_index]] = 'Y'
#### Test - end ####

pocket_coord = pocket_coordinates(pdb_path, f"{pocket_path}/pockets/",
pdb_id, chain_id, pocket_residue_indices)

tokenizer = BertTokenizer.from_pretrained(MODEL_PATH, do_lower_case=False )
model = BertModel.from_pretrained(MODEL_PATH)
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
model = model.to(device)
model = model.eval()

seq_emb = None
poc_res_emb = []

#### Test - begin ####
poc_labels = []
#### Test - end ####

with torch.no_grad():
    seq = " ".join(sequence)
    encoding = tokenizer.batch_encode_plus(
        [seq],
        add_special_tokens=True,
        padding='max_length'
    )
    input_ids = torch.tensor(encoding['input_ids']).to(device)
    attention_mask = torch.tensor(encoding['attention_mask']).to(device)
```

```
    embedding = model(input_ids=input_ids,
attention_mask=attention_mask)[0].cpu().numpy()

    seq_len = (attention_mask[0] == 1).sum()
    token_emb = embedding[0][1:seq_len-1]
    seq_emb = token_emb.mean(axis=0)

    for i in range(len(pocket_residue_indices)):
        add_pocket = True
        cur_poc_emb = []

#### Test - begin ####
        poc_labels.append(labels[i])
#### Test - end ####

        for idx in pocket_residue_indices[i]:
            try:
                token = token_emb[seq_indices[idx]]
                cur_poc_emb.append(token)
            except Exception as e:
                add_pocket = False
#### Test - begin ####
                poc_labels.pop()
#### Test - end ####
                break

        if add_pocket:
            poc_res_emb.append(cur_poc_emb)

del model
torch.cuda.empty_cache()
gc.collect()

def get_res_data(poc_res_emb, pocket_coord):
    X = []
    Y = []

    for i in range(min(len(poc_res_emb), len(pocket_coord))):
        seq_emb = []
```

```
        for res_idx in range(min(len(poc_res_emb[i]), len(pocket_coord[i]))):
            seq_emb.append(poc_res_emb[i][res_idx])
        seq_emb = np.array(seq_emb).mean(axis=0)
        poc = pocket_features[i]
        X.append(np.concatenate((seq_emb, poc)))
#### Test - begin ####
        Y.append(labels[i])
#### Test - end ####

    return X, Y

X_Test, Y_Test = get_res_data(poc_res_emb, pocket_coord)
dtest = xgb.DMatrix(X_Test, label=Y_Test)
bst = xgb.Booster()
bst.load_model('./xgboost.model')
y_pred = bst.predict(dtest)

paired = list(zip(y_pred, Y_Test, pocket_residue_indices))
paired_sorted = sorted(paired, key=lambda x: x[0], reverse=True)

top3 = [paired_sorted[i] for i in range(min(len(paired_sorted), 3))]

for top in top3:
    print(top)
```