

PSF: A Web Application Tool for Protein Scaffold Filling

Kushal Badal*, Binhai Zhu[†], Xiaowen Liu[‡], Letu Qingge*

*Department of Computer Science, North Carolina Agricultural and Technical State University
Greensboro, NC, USA.

Email: kbadal@aggies.ncat.edu, lqingge@ncat.edu

[†]Gianforte School of Computing, Montana State University
Bozeman, MT, USA.

Email: bhz@montana.edu

[‡]John W. Deming Dept. of Medicine, Tulane University
New Orleans, LA, USA.

Email: xwliu@tulane.edu

Abstract—The protein scaffold filling problem remains a significant challenge in computational proteomics, which is critical for accurate protein function prediction and drug design. Despite recent advancements, current sequencing methods often yield incomplete protein sequences, referred to as scaffolds, which require precise filling for further analysis. This paper presents a web-based application, implemented using the Django framework, adopting our previously developed machine learning and deep learning techniques for protein scaffold filling. The platform allows users to try our pre-trained models or train models on their datasets for new scaffolds. This system provides a versatile tool for researchers in computational proteomics, enhancing the efficiency of protein sequence prediction. The developed web application can be accessed through <https://psf.ncat.edu/>.

Index Terms—Protein Scaffold Filling, Deep Learning, Generative AI, Django, Web Application

I. INTRODUCTION

Proteins are the fundamental building blocks of life, playing crucial roles in various biological processes, from catalysis and signaling to providing structural support [1]. Determining the amino acid sequence of a protein is essential for understanding its function, structure, and interactions within the cellular environment. This knowledge has wide-ranging applications in fields such as drug design, protein engineering, and deciphering the molecular mechanisms underlying diseases [2].

Mass spectrometry-based de novo protein sequencing has revolutionized the field of proteomics by enabling the rapid and sensitive determination of protein sequences [3]. However, despite the advancements in sequencing technologies, the resulting protein sequences often contain gaps, resulting in incomplete structures known as protein scaffolds [4]. Filling these gaps to obtain complete protein sequences is a critical challenge in bioinformatics.

Recent developments in artificial intelligence, particularly deep learning, have opened up new avenues for addressing the

protein scaffold filling problem. Deep learning models, such as convolutional neural networks (CNNs) and long short-term memory (LSTM) networks, have shown promise in capturing complex patterns in protein sequences and predicting missing information [5]. Moreover, generative AI models, such as convolutional denoising autoencoder (CDA) have demonstrated potential in predicting missing amino acids and correcting errors in protein scaffolds [6].

To harness the power of these AI techniques and make them accessible to researchers, we have developed PSF (Protein Scaffold Filler), a web-based application that assists in completing protein sequences by filling gaps in scaffolds. PSF is built using the Django web framework [7] and provides a user-friendly interface for researchers to utilize state-of-the-art AI models for protein scaffold filling.

The PSF application offers two main functionalities: (1) Users can test our pre-trained models and (2) users can train models and fill gaps in new protein scaffolds. By incorporating a range of AI techniques, PSF enables the prediction of missing amino acids in gaps and the correction of errors in non-gap regions of the scaffold, which caused by bottom-up or top-down mass spectrometry analysis.

The primary focus of this paper is to present PSF as a comprehensive and user-friendly web-based tool that empowers researchers to leverage artificial intelligence for protein scaffold filling. We aim to showcase the integration of various AI models into a unified platform, providing a seamless experience for users to complete protein sequences and advance their research in proteomics and related fields.

The rest of the paper is organized as follows: Section II describes the background and related work on protein sequencing and scaffold filling. Section III provides an overview of the PSF web application, including its architecture and implementation details. Section IV discusses results and analysis. Finally, section V concludes the paper and outlines potential future directions for enhancing the PSF application.

*This work is supported by the U.S. National Science Foundation under Awards 2307571, 2307572 and 2307573, as well as partially supported by the U.S. National Institutes of Health U24 HG013013. Corresponding author: Dr. Letu Qingge.

II. BACKGROUND AND RELATED WORK

A. Protein Sequencing and Scaffold Filling

Protein sequencing is the process of determining the linear order of amino acids in a protein molecule. This information is crucial for understanding the structure, function, and interactions of proteins in biological systems. Mass spectrometry (MS) has emerged as a powerful tool for protein sequencing, offering high sensitivity, speed, and the ability to handle complex protein mixtures [8]. However, MS-based protein sequencing often results in incomplete protein sequences with gaps, known as protein scaffolds [4].

Protein scaffold filling is a critical challenge in proteomics, which is crucial for accurate protein function prediction and other bioinformatics applications. Traditional methods for scaffold filling often rely on combinatorial algorithms and homology-based techniques. Qingge et al. [9] proposed polynomial-time algorithms using dynamic programming and local search to infer missing amino acids by aligning incomplete sequences with homologous references. These methods are effective for small gaps but suffer from computational inefficiency and reliance on high-quality reference data.

Homology-based methods involve aligning the protein scaffold with known protein sequences in databases to infer the missing amino acids [10]. While effective, their performance is limited when homologous sequences are unavailable or of low quality [2]. De novo sequencing algorithms, on the other hand, attempt to directly predict the missing amino acids based on the mass spectra of the protein fragments [11], [12].

B. AI in Protein Sequencing and Scaffold Filling

Artificial intelligence (AI) has revolutionized various fields, including bioinformatics and proteomics research. Machine learning and deep learning has transformed the approach to protein scaffold filling by enabling models to learn complex sequence patterns from large datasets. Sturtz et al. [5] developed a hybrid model combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. Their CNN-LSTM model achieves 100% gap filling accuracy on the MabCampath data. This approach significantly improved over traditional methods, especially for sequences with complex patterns and large gaps.

Jordan et al. [6] introduced a Convolutional Denoising Autoencoder (CDA) for protein scaffold filling. Their model used a denoising strategy to predict and correct missing amino acids, achieving 95.327% full sequence accuracy and 100% gap filling accuracy on the MabCampath data. The CDA model reconstructs the full sequence by learning to denoise corrupted inputs, providing a robust approach to scaffold completion that addresses some of the limitations of sequential prediction models. Furthermore, Qingge et al. (2024) also developed transformer and GPT-2 models for protein sequencing problems.

C. Web-based Tools for Protein Analysis

Web-based tools have become increasingly popular in the bioinformatics community, providing researchers with easy

access to sophisticated computational methods and resources. These tools offer user-friendly interfaces, eliminate the need for local software installation, and facilitate collaboration among researchers. Examples of web-based tools for protein analysis include BLAST [13] for comparing protein or nucleotide sequences against sequence databases, UniProt [14] for searching and retrieving protein data, I-TASSER [15] for protein structure and function prediction, and Phyre2 [16] for protein structure prediction using homology modeling and fold recognition techniques.

Despite the availability of various web-based tools for protein analysis, there is a lack of tools specifically designed for protein scaffold filling that integrate state-of-the-art AI techniques. While the authors of [5], [6] provide codebases for their deep learning models, these codebases lack user-friendly interfaces and require manual steps for data preprocessing and training data collection, making them cumbersome for researchers.

PSF aims to fill this gap by providing a user-friendly web application that leverages advanced AI models for protein scaffold filling, enabling researchers to obtain complete protein sequences efficiently. Our work addresses the usability issues by integrating different machine learning and deep learning models into a web-based platform that automates data collection, preprocessing, and model training, enhancing the accessibility and efficiency of scaffold filling tools for bioinformatics research.

III. PSF: A WEB-BASED TOOL FOR PROTEIN SCAFFOLD FILLING

A. Problem Statement and Overview

Our web-based system, Protein Scaffold Filler (PSF), addresses the critical challenge of completing partial protein sequences, specifically tackling the Protein Scaffold Filling Problem. This problem can be formally defined as follows: Given a complete target protein sequence S and an incomplete scaffold T , the objective is to fill the missing amino acids in T such that $\text{Score}(S, T)$ is maximized. Here, Score represents the total number of one-to-one matches of amino acids between S and T .

The PSF is a web-based application that integrates state-of-the-art AI techniques to assist researchers in completing protein sequences by filling gaps in scaffolds. The primary goal of PSF is to provide a user-friendly and accessible platform for researchers to leverage advanced AI models for protein scaffold filling without the need for extensive computational expertise or local software installation.

PSF is developed using the Django framework [7] which follows the Model View Controller (MVC) architectural pattern and is deployed using Nginx [17] and Unicorn [18]. PSF offers a user-friendly interface for leveraging advanced AI models in protein scaffold filling tasks. This section details the system architecture of the protein scaffold filler web application.

B. System Architecture

The Protein Scaffold Filler (PSF) web application presents a novel approach to protein sequence analysis, integrating advanced AI models with a user-friendly interface. Our system architecture, illustrated in Figure 1, is specifically designed to handle the unique challenges of protein scaffold filling efficiently.

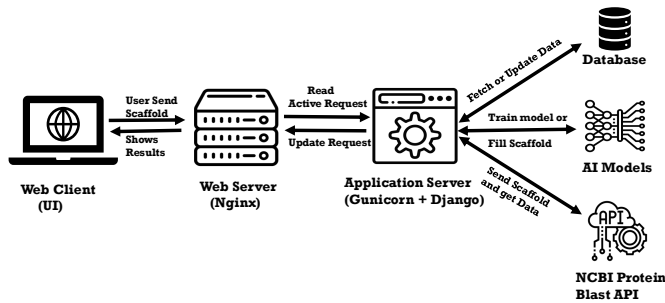


Fig. 1. PSF System Architecture

1) **Frontend Interface:** The frontend of PSF is engineered using a synergistic combination of HTML, CSS, and JavaScript, resulting in a highly responsive and intuitive user interface. This interface is meticulously crafted to support two primary functionalities:

- Utilization of pre-trained models on the Mabcampath dataset
- Training models on user-provided scaffolds

Users can use the PSF frontend user interface to input protein scaffolds in FASTA format, select from a range of AI models, and upload homologous sequences. Real-time JavaScript validation ensures data integrity before submission. For users engaging in custom model training, the interface provides streamlined options for uploading homologous sequences or generating them using our integrated NCBI Protein BLAST API [19].

2) **Web Server Infrastructure:** The forefront of PSF's server-side architecture is Nginx [17], a high-performance web server renowned for its efficiency and scalability, which acts as the initial point of contact for all HTTP requests. In our implementation, Nginx serves several crucial functions:

- Optimized static file serving, dramatically reducing load times for CSS, JavaScript, and image resources
- Sophisticated load balancing across multiple application server instances, ensuring consistent performance under varying user loads
- Advanced reverse proxy capabilities, seamlessly directing dynamic requests to our Gunicorn application server
- Robust SSL/TLS encryption management, guaranteeing secure communications between clients and our servers

3) **Application Server and Django Framework:** The application server of PSF is built upon Gunicorn (Green Unicorn) [18], a production-grade Web Server Gateway Interface (WSGI) HTTP Server specifically designed for Unix

environments. Gunicorn serves as a crucial intermediary between our Nginx web server and the Django application, efficiently managing and processing HTTP requests [7], [17], [18]. Gunicorn's architecture is based on the pre-fork worker model, which is particularly well-suited for the computational demands of PSF. In this model, Gunicorn creates a set of worker processes in advance, each capable of handling incoming requests independently. The WSGI specification, which Gunicorn implements, provides a standard interface between web servers and Python web applications. This standardization is crucial for PSF as it allows our Django application to communicate seamlessly with the Gunicorn server.

In the context of our PSF web application, this setup enables efficient handling of various request types. When a user submits a protein scaffold or initiates a training job, the request is first received by Nginx, which then passes it to Gunicorn. Gunicorn, in turn, delegates the request to an available worker process, which executes the appropriate Django view function to process the request.

We have developed PSF using the Django framework [7] which utilizes MVC architecture. Our implementation of Django's MVC architecture is tailored specifically to the unique requirements of protein scaffold filler:

- **Models:** We have designed custom Django models that provide a robust representation of AI models, and training jobs.
- **Views** Our view layer implements sophisticated logic for processing user requests, managing data flow between the frontend and backend systems, and organizing the execution of AI models.
- **Templates:** Utilizing Django's advanced template system, we generate dynamic HTML pages that adapt to user inputs and model outputs, providing a seamless and interactive user experience.
- **URL Configuration:** We employ Django's URL dispatcher to create a logical and intuitive mapping between URL patterns and view functions, facilitating easy navigation.

4) **Data Storage and Task Queue System:** PSF employs a cutting-edge JSON-based NoSQL approach for data storage and management. Our data storage system efficiently handles:

- User-submitted scaffolds and homologous sequences
- Model training information

Figure 2 provides a detailed visualization of our JSON-based database structure, highlighting the sophisticated organization of model training information.

We implemented a custom task queue system to handle the computational demands of model training. This system uses training information from the database, allowing for reliable task management even in the event of system restarts. A dedicated background worker continuously monitors this queue, processing training jobs sequentially. This approach enables PSF to manage multiple concurrent training requests without compromising system responsiveness.

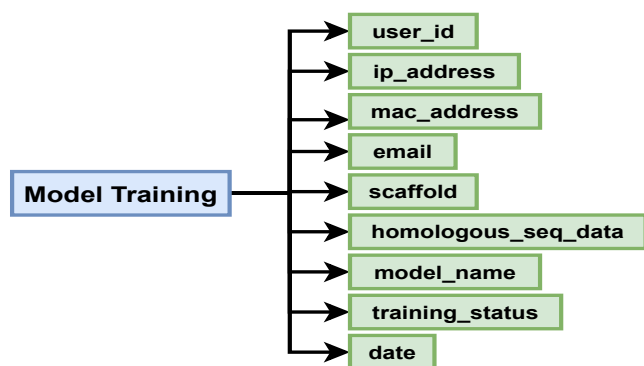


Fig. 2. Model Training Database Table

5) **NCBI Protein BLAST API Integration:** A key feature of PSF is its integration with the NCBI Protein BLAST API [19]. This integration significantly enhances the platform's capabilities by allowing users to generate homologous sequences directly within the application environment. The process involves a series of sophisticated steps:

- Secure transmission of user-provided scaffold sequences to the NCBI BLAST server
- Implementation of an intelligent polling mechanism to monitor job completion status
- Efficient retrieval and parsing of BLAST results

Our system sends requests to the NCBI API, processes the returned data, and seamlessly incorporates it into the model training pipeline. This functionality significantly enhances the usability of PSF.

6) **Email Notification System:** To enhance user experience, particularly for long-running tasks, we implemented an email notification system. Upon completion of a training task, the system automatically sends an email to the user containing the results. This includes the filled scaffold sequence, allowing researchers to submit jobs and receive results without continuous monitoring of the web interface.

7) **AI Model Integration and Core Algorithm:** PSF incorporates a diverse range of AI models, including traditional machine learning algorithms (KNN, Decision Tree, Random Forest) and advanced deep learning architectures (FCNN, CNN-LSTM [5], Convolutional Denoising AutoEncoder (CDA) [6], Transformer and GPT-2). Using those AI models, we are particularly focusing on two protein scaffold filling problems: one is for protein gap filling and another is for full protein sequence prediction. Machine learning algorithms and FCNN, CNN-LSTM models are used for protein gap filling, in which we only need to fill the gaps in the given scaffold. While other models, such as CDA, Transformer and GPT-2 models are applied for full protein sequence prediction, which also can correct the amino acids in the given scaffolds data generated from bottom-up and top-down analysis.

These models are implemented using TensorFlow and PyTorch. The core functionality of PSF is encapsulated in a

generalized algorithm that handles both model training and scaffold filling:

Algorithm 1 PSF Core Algorithm for Training and Prediction

Require: scaffold, homologous_sequences, model_name

Ensure: Trained model M or completed sequence S'

```

1:  $S \leftarrow \text{preprocess}(\text{scaffold})$ 
2:  $H \leftarrow \text{preprocess}(\text{homologous\_sequences})$ 
3: if mode == "train" then
4:    $M \leftarrow \text{initialize\_model}(\text{model\_name})$ 
5:   Train_model()
6:   save_model( $M$ )
7:   return  $M$ 
8: else
9:    $M \leftarrow \text{load\_model}(\text{model\_type})$ 
10:   $S'_n \leftarrow M.\text{predict}(S)$ 
11:   $S' \leftarrow \text{postprocess}(S'_n)$ 
12:  return  $S'$ 
13: end if
  
```

Algorithm 1 unifies the training and prediction processes:

- 1) **Preprocessing:** Converts input sequences into a numerical representation suitable for the chosen model.
- 2) **Model Initialization/Loading:** Either initializes a new model for training or loads a pre-trained model for prediction.
- 3) **Training Process:** For training mode, it creates a dataset, splits it into training and validation sets, and iteratively trains the model, saving the best performing version.
- 4) **Prediction Process:** For prediction mode, it uses the loaded model to predict missing amino acids in the scaffold.
- 5) **Postprocessing:** Converts the model's numerical output back into standard amino acid notation.

By implementing this specialized architecture, PSF offers a scalable, efficient, and user-friendly platform for protein scaffold filling research. Our system empowers researchers to leverage advanced AI techniques for protein sequence analysis without requiring extensive computational expertise, potentially accelerating discoveries in this critical field of study.

IV. RESULTS AND DISCUSSION

The PSF web application showcases its capabilities through two primary functionalities: pre-trained model inference and custom model training. This section provides an in-depth analysis of these features, supported by detailed explanations of user interactions and system processes.

A. System Interaction Overview

Figure 3 presents a comprehensive sequence diagram that illustrates the intricate interactions between users, the PSF interface, and various backend components during typical scaffold filling operations.

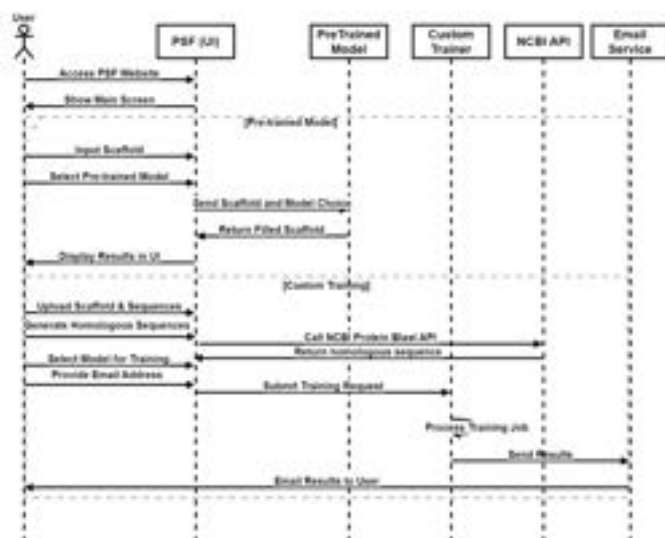


Fig. 3. Sequence Diagram of PSF Operation

This diagram offers a step-by-step visualization of the system's workflow, from initial user input to final result delivery. It demonstrates how PSF efficiently manages both pre-trained model inference and custom model training processes, highlighting the seamless integration of multiple components including the user interface, AI models, NCBI API, and email service.

B. Pre-trained Model Inference

The pre-trained model inference feature allows researchers to utilize models trained on the Mabcampath dataset for immediate scaffold filling tasks and test our model performance. This functionality demonstrates the system's ability to quickly and accurately fill gaps in protein scaffolds.

As shown in Figure 4, the user interface for pre-trained model inference is designed for simplicity and efficiency. Users can input a Mabcampath protein scaffold in FASTA format through a text area or file upload option. A dropdown menu allows selection from a range of pre-trained models, including CDA [6], CNN-LSTM [5], DNN, transformer, GPT-2 and various machine learning algorithms such as KNN, Decision Trees, and Random Forests. Upon submission, the system processes the input using the selected model and returns results directly on the web interface.

C. Custom Model Training and Scaffold Filling

The custom model training functionality extends PSF's utility beyond pre-existing datasets, enabling researchers to train new models on novel protein scaffolds. This feature is particularly valuable for studying proteins not well-represented in standard datasets. Figure 5 displays the user interface for custom model training. This interface guides users through a multi-step process:

- **Scaffold Upload:** Users can upload their protein scaffold sequence in FASTA format.

Enter Scaffold Formatted as FASTA:

```

...MTQPSISASVGVDTTC...ADKYNWYQKPGKAPKQWYTWNGTQVPSF...Q...FTPT...
YQCHSRPTTGGTGVKRSAPSVIRFSDQKSGSASVQWNVYRLAQPRKVDNAKSGSGSVTQDSKSTYSEF
ITSAZYHSHVACVTHQGSPTKSN...
  
```

Select data:
Mabcampath

Select Filling Type:
Gap Filling Only

Select Model:
KNN

Submit

See Result in Table below:

Result
Given Scaffold ...MTQPSISASVGVDTTC...ADKYNWYQKPGKAPKQWYTWNGTQVPSF...Q...FTPT...
Predicted/Filled Sequence ...MTQPSISASVGVDTTC...ADKYNWYQKPGKAPKQWYTWNGTQVPSF...Q...FTPT...

Fig. 4. Pre-trained Model Interface

Try Our Trained Model Train with Pre-Trained Scaffold

Enter Scaffold Formatted as FASTA:

```

...MTQPSISASVGVDTTC...ADKYNWYQKPGKAPKQWYTWNGTQVPSF...Q...FTPT...
YQCHSRPTTGGTGVKRSAPSVIRFSDQKSGSASVQWNVYRLAQPRKVDNAKSGSGSVTQDSKSTYSEF
ITSAZYHSHVACVTHQGSPTKSN...
  
```

Upload Homologous Sequence File if any in FASTA Format: Choose File null_training_sequences

Generate Homologous Sequence from NCBI Database: Generate

Select Model to Train:
KNN

Enter Email to get Result:
example@gmail.com

Train

Fig. 5. Custom Model Training Interface

- **Homologous Sequence Input:** The system provides two options for obtaining homologous sequences:
 - 1) Direct upload of known homologous sequences
 - 2) Generation of homologous sequences using the integrated NCBI Protein BLAST API [19]
- **Model Selection:** Users choose an AI model for training from options including CNN-LSTM [5], CDA [6], DNN, and various machine learning algorithms such as KNN, Decision Trees, and Random Forests.
- **Email Notification:** Users provide an email address for result delivery.

Upon submission, the application server validates the input and adds the training request to a task queue system. This asynchronous processing approach allows PSF to handle multiple long-running training jobs without compromising the

responsiveness of the web interface. The custom training process involves several backend steps:

- **Data Preprocessing:** The system prepares the input data for the selected model, including sequence encoding and splitting into training and validation sets.
- **Model Initialization and Training:** The chosen AI model is initialized with appropriate hyperparameters and trained on the provided data. Model Evaluation: The system evaluates the trained model's performance on a held-out validation set.
- **Result Compilation:** The system generates a comprehensive report including the trained model's performance metrics, and filled scaffold sequences.
- **Email Notification:** Upon completion of training, the system automatically emails the results to the user, facilitating efficient offline analysis.

This feature significantly enhances PSF's flexibility, allowing researchers to adapt the tool to their specific protein families or research contexts, potentially improving scaffold filling accuracy for specialized scenarios. The PSF web application is now live and accessible at <https://psf.ncat.edu/>.

V. COMPARATIVE ANALYSIS AND MODEL PERFORMANCE

To provide a comprehensive evaluation of PSF web application capabilities, we conducted extensive comparative analyses of the various AI models integrated into our platform. This section presents the results of these analyses, highlighting the strengths and specificities of each model for different aspects of protein scaffold filling.

A. Experimental Setup

We evaluated the performance of our models on two primary tasks:

- **Gap Filling:** This task involves predicting missing amino acids in gap regions of a protein scaffold. For testing our model on the basis of gap filling we calculate the gap filling accuracy. This metric specifically measures the model's ability to correctly predict amino acids in the gap regions of a scaffold. It is calculated as:

$$GFA = \frac{C_g}{T_g} \quad (1)$$

where GFA is Gap Filling Accuracy, C_g is the number of correctly predicted amino acids in gaps, and T_g is the total number of amino acids in gaps.

Gap filling accuracy metric provides a focused assessment of how well the model performs in filling the missing regions of the scaffold.

- **Complete Sequence Generation:** This task involves generating the entire protein sequence from a protein scaffold. We calculate full sequence accuracy for testing our model on the basis of complete sequence generation. Full sequence accuracy metric evaluates the overall accuracy of the predicted sequence compared to the true target sequence. It is calculated as:

- 1) **Full Sequence Accuracy:** This metric evaluates the overall accuracy of the predicted sequence:

$$FSA = \frac{C_t}{T_t} \quad (2)$$

where FSA is Full Sequence Accuracy, C_t is the number of correctly predicted amino acids in the entire sequence, and T_t is the total number of amino acids in the sequence.

We used the Mabcampath scaffold [2] as in figure 6 to ensure robust evaluation of all models in our web application. Performance metrics included gap filling accuracy and full sequence accuracy. Table I summarizes the performance of different models across both tasks.

```
Mabcampath target sequence:
DIQMTQSPSSLSASVGDRTITCKASQNIIDKYLNNHYQQKPGKAPKLLIYN
TNNLTQGVPSRFSGSGSGTDFTFTISSLPQEDIAITYYCLQHISRPTFGQ
GTVKEIKRTVAAPSVFIFPPSDEQLKSGTASVVCLLNNFYPREAKVQWKV
DNALQSGNSQESVTEQDSKSTYSLSSTLTLSKADYEKHKVYACEVTHQG
LSSPVTKSFNRGEC
Mabcampath scaffold sequence:
---MTQSPSSLSASVGDRTITCK---NIDKYINNNHYQQKPGKAPKIIINYN
TNNLTQGVPSRF---G---FTFTI-----YCIQHSRPTFGQ
GTVKEIKRSIAAPSVFIFPPSDEQIKSGTASVVCIIINNFYPREAPRRKV
DNALQSGNSQESVTEQDSKSTYSISSITISKADYEKHKVYACEVTHQG
ISSPVTKSFN----
```

Fig. 6. Mabcampath Scaffold Data

TABLE I
MODEL PERFORMANCE COMPARISON

Model	Gap Filling Accuracy	Full Sequence Accuracy
KNN	89.28	89.25
Decision Tree	57.14	85.04
Random Forest	100.00	90.65
DNN	71.42	86.91
CNN-LSTM	92.85	89.70
CDA	100.00	95.33
Transformer	96.42	96.72
GPT-2 Model	100.00	100.00

B. Gap Filling Performance

For the task of gap filling, we observe that random forest, CDA, and GPT-2 models achieved the highest 100% gap filling accuracy on Mabcampath scaffold. Traditional machine learning models, while less accurate, offer the advantage of faster inference times and may be suitable for quick preliminary analysis.

C. Complete Sequence Generation

In the task of complete sequence generation, the GPT-2 model demonstrated the best performance, with the highest 100% full sequence accuracy. The Transformer model was a close second, confirming the effectiveness for protein sequence tasks. The CDA models showed competitive performance, indicating their potential for sequence generation tasks. However, the CDA model might be more suitable for scenarios where computational resources are limited or faster inference is required.

D. Model Selection Guidelines

Based on our analysis, we recommend the following guidelines for model selection in PSF web application:

- For high-accuracy gap filling, especially in complex proteins: GPT-2 or Transformer
- For balanced performance in both gap filling and sequence generation: CDA
- For quick preliminary analysis or resource-constrained environments: Random Forest or CNN-LSTM
- For scaffolds with fewer gaps and errors, or for quick preliminary analysis: Random Forest, DNN, KNN, or Decision Tree
- For complete sequence generation of novel protein scaffolds: GPT-2 or Transformer

In summary, our comprehensive analysis of model performance in PSF web application reveals a clear advantage of deep learning approaches over traditional machine learning methods for protein scaffold filling tasks. Deep learning models, particularly GPT-2 and Transformer architectures, demonstrate superior accuracy in both gap filling and complete sequence generation, especially for complex scaffolds with numerous or large gaps. This superiority stems from their ability to capture long range dependencies and intricate sequence patterns in protein structures. However, machine learning models retain their utility for scaffolds with fewer gaps and errors, offering quick and reasonably accurate results in resource constrained scenarios or for initial assessments.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented PSF, a comprehensive web-based tool for protein scaffold filling that leverages state-of-the-art AI techniques. PSF addresses the critical challenge of completing protein sequences with missing regions, which is essential for advancing research in proteomics and related fields. By integrating a diverse set of AI models, including CNN-LSTM, machine learning algorithms, and generative AI models, PSF provides researchers with a powerful and user-friendly platform for accurate and efficient protein scaffold filling.

The development of PSF was motivated by the need for an accessible and versatile tool that can handle the complexities of protein sequences and facilitate the completion of partial sequences. Through the integration of cutting-edge AI models and a streamlined web interface, PSF simplifies the process of scaffold filling, enabling researchers to obtain complete protein sequences without the need for extensive computational expertise or local software installation.

The general algorithm incorporated in PSF serves as a unifying framework for the various AI models, ensuring modularity and extensibility. This algorithm abstracts the common steps involved in scaffold filling, such as preprocessing, model inference, and postprocessing, while allowing for model-specific computations. The flexibility of the general algorithm enables seamless integration of new AI models into the PSF platform, accommodating future advancements in the field.

While PSF represents a significant step forward in protein scaffold filling, there are several avenues for future work and improvements. One potential direction is the incorporation of additional AI models and algorithms into the platform. As new techniques emerge in the field of deep learning and computational biology, integrating them into PSF would further enhance its capabilities and provide researchers with a broader range of options for scaffold filling. Furthermore, the scalability and performance of PSF can be optimized to handle larger datasets and more complex protein sequences. Implementing distributed computing techniques, such as parallel processing and GPU acceleration, would enable PSF to efficiently process massive amounts of data and accommodate the growing needs of the research community.

ACKNOWLEDGMENT

This work is supported by the U.S. National Science Foundation under Awards 2307571, 2307572 and 2307573, as well as partially supported by the U.S. National Institutes of Health U24 HG013013. We thank anonymous reviewers for their insightful comments and inputs.

REFERENCES

- [1] N. Saraswathy and P. Ramalingam, *Concepts and Techniques in Genomics and Proteomics*. Elsevier, 2011.
- [2] X. Liu, L. J. M. Dekker, S. Wu, M. M. Vanduijn, T. M. Luiders, N. Tolić, Q. Kou, M. Dvorkin, S. Alexandrova, K. Vyatkina, et al., "De novo protein sequencing by combining top-down and bottom-up tandem mass spectra," *J. Proteome Res.*, vol. 13, no. 7, pp. 3241–3248, 2014.
- [3] M. Kinter and N. E. Sherman, *Protein Sequencing and Identification Using Tandem Mass Spectrometry*. John Wiley & Sons, 2005.
- [4] N. H. Tran, M. Z. Rahman, L. He, L. Xin, B. Shan, and M. Li, "Complete de novo assembly of monoclonal antibody sequences," *Sci. Rep.*, vol. 6, no. 1, pp. 1–10, 2016.
- [5] J. Sturtz, B. Zhu, X. Liu, X. Fu, X. Yuan, and L. Qingge, "Deep learning approaches for the protein scaffold filling problem," in *Proc. 2022 IEEE 34th Int. Conf. Tools Artif. Intell. (ICTAI)*, 2022, pp. 1055–1061.
- [6] J. Sturtz, R. Annan, B. Zhu, X. Liu, and L. Qingge, "A convolutional denoising autoencoder for protein scaffold filling," in *Bioinformatics Research and Applications*, X. Guo, S. Mangul, M. Patterson, and A. Zelikovsky, Eds., Springer, 2023, pp. 518–529.
- [7] "Django: The Web framework for perfectionists with deadlines," Django Software Foundation, 2024. [Online]. Available: <https://www.djangoproject.com/>. [Accessed: 27-Mar-2024].
- [8] R. Aebersold and M. Mann, "Mass spectrometry-based proteomics," *Nature*, vol. 422, no. 6928, pp. 198–207, 2003.
- [9] L. Qingge, X. Liu, F. Zhong, and B. Zhu, "Filling a protein scaffold with a reference," *IEEE Trans. Nanobiosci.*, vol. 16, no. 2, pp. 123–130, 2017.
- [10] B. Ma, K. Zhang, C. Hendrie, C. Liang, M. Li, A. Doherty-Kirby, and G. Lajoie, "PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry," *Rapid Commun. Mass Spectrom.*, vol. 17, no. 20, pp. 2337–2342, 2003.
- [11] B. Ma, "Novor: real-time peptide de novo sequencing software," *J. Am. Soc. Mass Spectrom.*, vol. 26, no. 11, pp. 1885–1894, 2015.
- [12] A. Frank and P. Pevzner, "PepNovo: de novo peptide sequencing via probabilistic network modeling," *Anal. Chem.*, vol. 77, no. 4, pp. 964–973, 2005.
- [13] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, 1990.
- [14] "UniProt: the universal protein knowledgebase in 2021," *Nucleic Acids Res.*, vol. 49, no. D1, pp. D480–D489, 2021.
- [15] J. Yang, R. Yan, A. Roy, D. Xu, J. Poisson, and Y. Zhang, "The I-TASSER Suite: protein structure and function prediction," *Nat. Methods*, vol. 12, no. 1, pp. 7–8, 2015.

- [16] L. A. Kelley, S. Mezulis, C. M. Yates, M. N. Wass, and M. J. E. Sternberg, "The Phyre2 web portal for protein modeling, prediction and analysis," *Nat. Protoc.*, vol. 10, no. 6, pp. 845–858, 2015.
- [17] "NGINX," NGINX, Inc., 2024. [Online]. Available: <https://www.nginx.com/>. [Accessed: 27-Mar-2024].
- [18] "Gunicorn: Python WSGI HTTP Server for UNIX," Gunicorn, 2024. [Online]. Available: <https://gunicorn.org/>. [Accessed: 29-Mar-2024].
- [19] "Protein Blast API," NCBI, 2024. [Online]. Available: <https://ncbi.github.io/blast-cloud/dev/api.html>. [Accessed: 8-Mar-2024].



Dr. Letu Qingge PhD, is an Assistant Professor in the Department of Computer Science at North Carolina A&T State University, NC. He received his PhD degree in Computer Science from Montana State University, MT, USA. His research interests include algorithms, machine learning, deep learning, bioinformatics and computational biology, computer vision and neural network control. He has published over 50 peer-reviewed papers. His research has been funded by NSF and NIH.



Mr. Kushal Badal is currently a Master student at North Carolina A&T State University with a strong academic record and a passion for AI and data science. He gained practical experience in data analysis, machine learning, and research, complemented by leadership roles and a commitment to continuous learning. His goal is to leverage his technical skills and enthusiasm for problem-solving to make meaningful contributions in the field of Artificial Intelligence.



Dr. Binhai Zhu obtained his PhD in computer science at McGill University in 1994. Following a 2-year post-doc at Los Alamos National Laboratory, he has been teaching at City University of Hong Kong and Montana State University since Aug, 1996. He is currently a professor at Gianforte School of Computing at Montana State University. His research interests are Algorithms, Computational Biology, Computational Geometry, and Combinatorial Optimization. He has published over 220 papers in these areas. He is currently on the editorial board

of *Acta Informatica* and *Algorithms*.



Dr. Xiaowen Liu received his bachelor's (1996) and master's (2004) degrees in computer science from Shandong University, China and obtained his Ph.D. degree (2008) in computer science from City University of Hong Kong, Hong Kong. After 4-year postdoc training at the University of Western Ontario, the University of Waterloo, and the University of California, San Diego, Dr. Liu took positions as an Assistant Professor and Associate Professor at the Department of BioHealth Informatics, Indiana University-Purdue University Indianapolis from

2012 to 2021. Dr. Liu is currently a professor of bioinformatics in the Division of Biomedical Informatics and Genomics, John W. Deming Department of Medicine, Tulane University School of Medicine. Dr. Liu's research focuses on computational proteomics, especially mass spectrometry-based top-down proteomics, which can identify various proteoforms with alterations in biological samples. Dr. Liu's lab is supported by several NIH grants, and his work has resulted in more than 80 peer-reviewed journal papers.