OPUS-GO: UNLOCKING RESIDUE-LEVEL INSIGHTS FROM SEQUENCE-LEVEL ANNOTATIONS USING BIOLOGICAL LANGUAGE MODELS

Gang Xu[1,2,†], Ying Lv[2,†], Ruoxi Zhang[1,†], Xinyuan Xia[1], Qinghua Wang[3], & Jianpeng Ma[1,2,*]
[1]Fudan University
[2]Shanghai AI Laboratory
[3]Center for Biomolecular Innovation, Harcam Biomedicines
jpma@fudan.edu.cn

ABSTRACT

Accurate annotation of protein is crucial for understanding their structural and functional properties. Existing biological language model (BLM)-based methods, however, often prioritize sequence-level classification accuracy while neglecting residue-level interpretability, as sequence-level annotations are easier to obtain. To address this, we introduce OPUS-GO, a method that improves sequence-level predictions while also providing detailed residue-level insights by pinpointing critical residues associated with functional labels. By employing a modified Multiple Instance Learning (MIL) strategy with BLM representations, OPUS-GO outperforms baseline methods in both sequence-level and residue-level classification accuracy across various downstream tasks for protein sequences, including Gene Oncology (GO)-term prediction for proteins. Furthermore, the identified residues can serve as promising "prompts" for molecular design models, such as ESM-3, enabling the generation of sequences with the desired functionality.

1 INTRODUCTION

Recent advancements in large-scale biological language models (BLMs), particularly protein models like ESM-2 (Lin et al., 2023), have enhanced the detection of intricate structural and functional patterns in biological molecules, enabling broad applications such as Gene Oncology (GO) (Gu et al., 2023; Mi et al., 2024; Xu et al., 2023; Zhuo et al., 2024) term and Enzyme Commission (EC) number prediction (Song et al., 2024; Yu et al., 2023). Despite these advances, there is a growing demand for deeper insights into model predictions to ensure model robustness and biological relevance beyond accuracy improvements. To address this, interpretable machine learning techniques have been introduced, generally categorized into by-design workflows that integrate interpretability into the model architecture and post-hoc workflows that derive explanations after training.

However, several challenges persist in applying interpretive methods to leverage biological language models for relevant downstream tasks (Chen et al., 2024). To work with existing biological language models with limited built-in interpretability, in-design workflows often rely on attention analysis, whose effectiveness and trustworthiness are still debated (Bai et al., 2021). Post hoc methods such as Grad-CAM (Selvaraju et al., 2017) enable feature importance analysis of language model representations in downstream tasks but operate independently of efforts to enhance model performance.

To address these limitations, we introduce OPUS-GO, a multiple-instance learning (MIL) approach that combines the strengths of both by-design and post-hoc workflows. While not directly probing the language model's internal structure, its interpretability is inherently aligned with the decision-making process. OPUS-GO is broadly applicable to various functional annotation tasks, providing residue-level insights into model predictions while enhancing sequence-level performance. The evaluation results on a handful protein multi-class or multi-label classification downstream tasks demonstrate OPUS-GO's superior classification accuracy compared to its baseline methods. No-

[†]These authors contributed equally to this work.

tably, OPUS-GO satisfactorily identifies the residues associated with the corresponding sequence-level labels.

The main contributions of this work are as follows:

- We introduce OPUS-GO, a MIL-based method that not only improves sequence-level predictions while also providing detailed residue-level insights by pinpointing critical residues associated with functional labels.
- We show that OPUS-GO outperforms baseline methods in both sequence-level and residue-level classification accuracy across downstream tasks for protein sequences.
- OPUS-GO can be seamlessly integrated into any BLM, combining the strengths of both by-design and post-hoc model interpretability methods.

2 Related Work

Protein language models. Protein language models have emerged as powerful tools for predicting protein structures and various functional attributes. These models can be broadly categorized into three types: BERT-based models, such as the ESM-series (Lin et al., 2023; Zhang et al., 2024); GPT-based models, such as the ProGen-series (Madani et al., 2023; Nijkamp et al., 2023); and span-mask-based models, such as ProtT5 Elnaggar et al. (2021). By training on extensive datasets of protein sequences, protein language models acquire foundational evolutionary knowledge about protein structure and function, enabling them to perform a wide range of protein modeling and functional prediction tasks. Among these, ESM-2 stands out as one of the most prominent and widely utilized models in protein functional prediction tasks.

Interpretability for protein language models. By-design workflows use inherently interpretable models, with the most common ones being logistic regression and decision trees. In transformer-based architectures like BLMs, recent work apply sparse autoencoders to provide interpretable features Simon & Zou (2024); Adams et al. (2025). Attention weights also offer explanations to some extent; for instance, Geneformer (Theodoris et al., 2023) analyzes attention weights across different layers to investigate how the model encodes the hierarchy of gene regulatory networks. In contrast, post-hoc methods are usually employed subsequent to the design and training of a prediction model (Chen et al., 2024). For protein GO-term and EC number prediction, researchers often use post-hoc methods to demonstrate the success of their models by pinpointing residues within a protein structure that are pivotal for predicting specific functions. Among the most commonly used methods is Grad-CAM (Selvaraju et al., 2017; Mi et al., 2024), a class-discriminative localization technique that offers visual explanations for predictions made by convolutional neural network (CNN)-based models (Gligorijević et al., 2021).

3 Method

Overview. OPUS-GO employs a MIL strategy to effectively handle multi-class and multi-label classification tasks. Specifically, it conceptualizes an entire sequence as a *bag* (i.e., sequence-level), with each residue within it acting as an *instance* (i.e., residue-level). Since only sequence-level labels are available, OPUS-GO leverages these annotations to train a classifier that assigns labels to individual residues. This approach enables the identification of the most representative residues for each label. As illustrated in Figure 1, full sequence representations from the BLM are input into a residue-level classifier, which then predicts the likelihood of each residue being associated with its corresponding sequence-level labels.

Loss Function. To train the residue-level classifier, we implemented a modified MIL loss function. Since sequence-level annotations may contain multiple labels, we apply binary cross-entropy loss to each label individually. The modified MIL loss comprises two key components. As outlined in Algorithm A1, for each positive sequence-level label, the top n residues with the highest probabilities are selected and assigned positive residue-level labels, while an equal number of residues are randomly sampled from the remaining pool and assigned negative labels. Additionally, an auxiliary MIL loss, applied exclusively to negative samples, is computed (Algorithm A2). Both loss components are combined with equal weighting to update the parameters of the residue-level classifier.

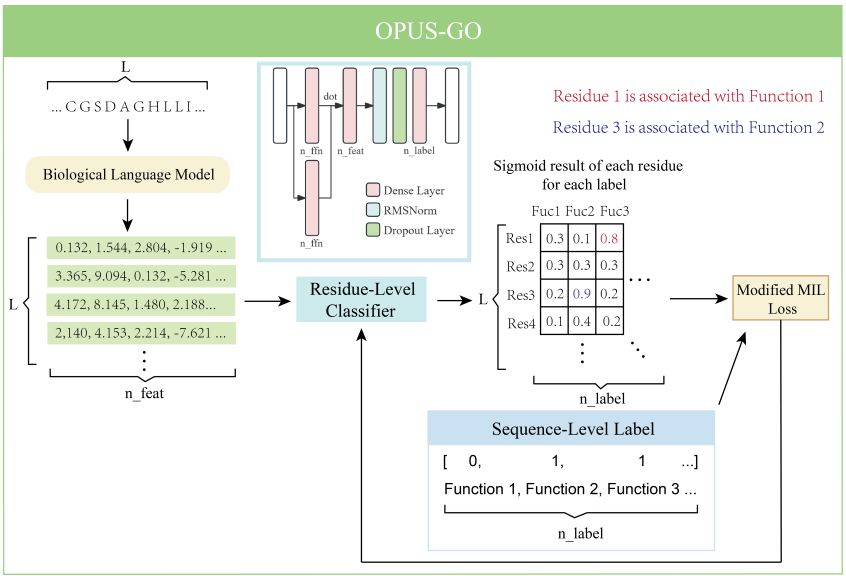

Figure 1: The workflow of OPUS-GO. "L" denotes the length of the protein sequence. "n_feat" denotes the number of features obtained from the biological language models. Specifically, the features utilized are derived from the last transformer block, such as the representations from the 33rd layer of ESM-2. "n_ffn" denotes the dimension of the hidden layers within the FeedForward module. "n_label" denotes the number of sequence-level labels, each treated as a binary classification task. The final output comprises the sigmoid values of each residue for each label. The sequence-level output for each label is determined by selecting the maximum sigmoid value across all residues within the corresponding column. Meanwhile, residues with a sigmoid value greater than 0.5 are considered to be associated with the respective label.

4 Experiments

Sequence-Level Prediction with Residue Insights. The protein function annotation task involves assigning multiple functional labels to a protein. To evaluate our method, we use three GO-term prediction benchmarks (Gligorijević et al., 2021; Zhang et al., 2022). As shown in Table 1, OPUS-GO demonstrates superior sequence-level classification accuracy in most cases compared to the baseline method ESM-2, which averages all residue features as the sequence-level representative. We also present the results of several other leading methods including HEAL (Gu et al., 2023) and GGN-GO (Mi et al., 2024), which incorporate structural features alongside sequence features from language models, and ProtST (Xu et al., 2023) and PROTLLM (Zhuo et al., 2024), which leverage additional protein-text data to enhance BLM representations. The results show that OPUS-GO outperforms these methods in most cases.

To further assess the significance of the residues identified by OPUS-GO, we compared sequence-level classification models trained on three selected residues. Specifically, we use the average ESM-2 features of the three highest-probability residues predicted by OPUS-GO and contrast them with models trained on three randomly chosen residues under the same experimental settings. The results indicate that using three residues selected by OPUS-GO maintains relatively stable accuracies compared to those based on randomly selected residues, thereby demonstrating OPUS-GO's efficacy in identifying residues associated with the corresponding label.

Example of annotations in Figure 2 demonstrate that OPUS-GO is capable of locating the residues associated with the corresponding labels with satisfactory accuracy. Recall that OPUS-GO only utilizes the sequence of the target and does not require any structural information or information about other molecules, such as DNA, chemical molecules, or other protein partners.

Residue-Level Experiments. We evaluate the residue-level interpretability of OPUS-GO, by comparing it with ESM-2 using Grad-CAM and GGN-GO (Mi et al., 2024), which also incorporates Grad-CAM for interpretability. For this, we construct three test sets focusing on DNA binding,

Table 1: The results of different methods on three protein GO term prediction datasets.

	GO-BP		GO-MF		GO-CC	
Method	AUPR	Fmax	AUPR	Fmax	AUPR	Fmax
ESM-2	0.365	0.510	0.659	0.660	**0.478**	0.574
OPUS-GO	**0.377**	**0.518**	**0.678**	**0.690**	**0.478**	0.577
ESM-2 (three residues)	0.224	0.365	0.465	0.459	0.319	0.452
OPUS-GO (three residues)	0.313	0.476	0.619	0.658	0.405	0.544
HEAL	0.323	0.484	0.623	0.634	0.416	0.543
GGN-GO	0.345	0.473	0.651	0.648	0.422	0.538
ProtST	0.342	0.482	0.647	0.668	0.364	0.487
PROTLLM	0.349	0.503	0.652	0.668	0.469	**0.596**

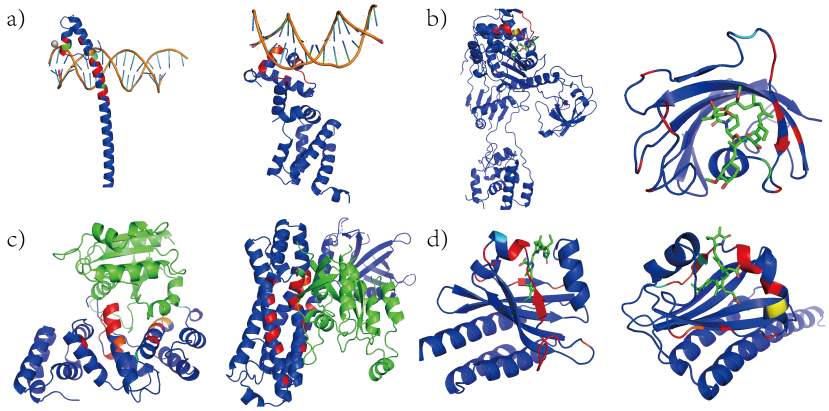

Figure 2: The interpretability results of OPUS-GO for protein GO term prediction. The results are visually represented using a color spectrum (ranging from blue to red) in PyMOL software, according to the predicted probability of corresponding label. It is important to note that, in each case, only the sequence information of the blue protein structure is utilized. The structures of these proteins, as well as the DNA in subfigure a), the chemical molecules in subfigures b) and d), and the green protein structure related to signal transduction, are not used in OPUS-GO and are only shown here for illustrative purposes. The probabilities of each residue for label a) "DNA binding", b) "drug binding", c) "regulation of signal transduction", d) "protein-chromophore linkage".

calcium ion binding, and heme binding, using GO-MF test set targets, with available residue-level labels from the BioLip database (Yang et al., 2012). As shown in Table 2, OPUS-GO consistently outperforms both methods. Representative examples from each method are illustrated in Figure 3.

Other Experiments. We also evaluated OPUS-GO for predicting protein EC numbers and found that it exceeds baseline methods in sequence-level classification accuracy. At the residue level, OPUS-GO successfully identifies active sites in certain cases, but there are also instances where it fails to accurately pinpoint them. Notably, OPUS-GO identifies residues it considers most closely related to the corresponding labels, which may not always correspond to specific sites of interest such as active sites in EC number prediction tasks. Consequently, OPUS-GO appears to be more effective at detecting consistent patterns within each enzyme category, rather than solely focusing on identifying specific active sites.

5 Discussion

In this study, we introduce OPUS-GO, a method that utilizes sequence-level annotations to provide both sequence-level and residue-level classification results, thereby enabling the identification of residues most critical for the sequence-level annotation. Our results demonstrate that, based on the features derived from BLMs and our modified MIL strategy, OPUS-GO shows superior performance

Table 2: The residue-level interpretability results of different methods on three binding-related datasets. F1-shift-1, F1-shift-2: if the binding site residue is located within 1 or 2 residues from the predicted residue, it is considered a true positive.

	AUC	F1-score	F1-shift-1	F1-shift-2
DNA binding				
OPUS-GO	**0.719**	**0.204**	**0.346**	**0.448**
ESM-2	0.613	0.151	0.288	0.369
GGN-GO	0.618	0.180	0.327	0.407
Calcium ion binding				
OPUS-GO	**0.826**	**0.327**	**0.479**	**0.544**
ESM-2	0.716	0.242	0.346	0.385
GGN-GO	0.470	0.090	0.227	0.293
Heme binding				
OPUS-GO	**0.689**	**0.124**	**0.344**	**0.433**
ESM-2	0.595	0.093	0.233	0.316
GGN-GO	0.544	0.117	0.268	0.350

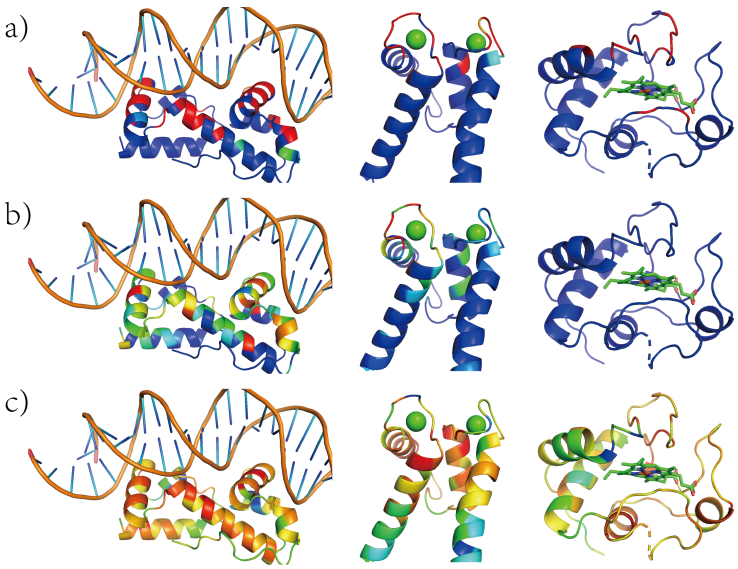

Figure 3: The interpretability results of each method. The results are visually represented using a color spectrum (ranging from blue to red) in PyMOL software, according to the predicted probability of corresponding label. a) The results of OPUS-GO. b) The results of ESM-2 with Grad-CAM. c) The results of GGN-GO.

in sequence-level classification compared to baseline methods for protein downstream tasks. Additionally, OPUS-GO offers strong interpretability by accurately pinpointing residues associated with respective labels. Furthermore, the consistently good performance of OPUS-GO on datasets comprising over 5,000 labels underscores its wide applicability across various contexts characterized by a large number of labels.

OPUS-GO stands as a promising tool for protein design, offering crucial patterns that can facilitate the design of proteins with specific functionalities. For example, the identified critical residues for a particular functional label can serve as "prompts" for protein design models, such as ESM-3, enabling them to generate sequences with desired function. This is particularly beneficial when the critical regions, such as the active site, binding site, or conserved regions, within the target protein that are crucial for the desired function have not been previously characterized or studied.

6 ACKNOWLEDGEMENTS

J.M. wants to thank the support from the National Key Research and Development Program of China (No. 2024YFA1307502), the Science and Technology Innovation Plan of Shanghai Science and Technology Commission (No. 23JS1400200), and the Research Fund for International Senior Scientists (No. W2431060). G.X. wants to thank the support from the National Natural Science Foundation of China (No. 32300535).

7 DATA AVAILABILITY

The training and evaluation codes and pre-trained models of OPUS-GO for each downstream task can be downloaded from `http://github.com/thuxugang/opus_go`.

REFERENCES

Etowah Adams, Liam Bai, Minji Lee, Yiyang Yu, and Mohammed AlQuraishi. From mechanistic interpretability to mechanistic biology: Training, evaluating, and interpreting sparse autoencoders on protein language models. *bioRxiv*, pp. 2025–02, 2025.

Bing Bai, Jian Liang, Guanhua Zhang, Hao Li, Kun Bai, and Fei Wang. Why attentions may not be interpretable? In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 25–34, 2021.

Valerie Chen, Muyu Yang, Wenbo Cui, Joon Sik Kim, Ameet Talwalkar, and Jian Ma. Applying interpretable machine learning in computational biology—pitfalls, recommendations and opportunities for new developments. *Nature methods*, 21(8):1454–1461, 2024.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

Vladimir Gligorijević, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):3168, 2021.

Zhonghui Gu, Xiao Luo, Jiaxiao Chen, Minghua Deng, and Luhua Lai. Hierarchical graph transformer with contrastive learning for protein function prediction. *Bioinformatics*, 39(7):btad410, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, 2023.

Jia Mi, Han Wang, Jing Li, Jinghong Sun, Chang Li, Jing Wan, Yuan Zeng, and Jingyang Gao. Ggngo: geometric graph networks for predicting protein function by multi-scale structure features. *Briefings in Bioinformatics*, 25(6):bbae559, 2024.

Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. Progen2: exploring the boundaries of protein language models. *Cell systems*, 14(11):968–978, 2023.

Theo Sanderson, Maxwell L Bileschi, David Belanger, and Lucy J Colwell. Proteinfer, deep neural networks for protein functional inference. *Elife*, 12:e80942, 2023.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Elana Simon and James Zou. Interplm: Discovering interpretable features in protein language models via sparse autoencoders. *bioRxiv*, pp. 2024–11, 2024.

Yidong Song, Qianmu Yuan, Sheng Chen, Yuansong Zeng, Huiying Zhao, and Yuedong Yang. Accurately predicting enzyme functions through geometric graph learning on esmfold-predicted structures. *Nature Communications*, 15(1):8180, 2024.

Christina V Theodoris, Ling Xiao, Anant Chopra, Mark D Chaffin, Zeina R Al Sayed, Matthew C Hill, Helene Mantineo, Elizabeth M Brydon, Zexian Zeng, X Shirley Liu, et al. Transfer learning enables predictions in network biology. *Nature*, 618(7965):616–624, 2023.

Minghao Xu, Xinyu Yuan, Santiago Miret, and Jian Tang. Protst: Multi-modality learning of protein sequences and biomedical texts. In *International Conference on Machine Learning*, pp. 38749–38767. PMLR, 2023.

Jianyi Yang, Ambrish Roy, and Yang Zhang. Biolip: a semi-manually curated database for biologically relevant ligand–protein interactions. *Nucleic acids research*, 41(D1):D1096–D1103, 2012.

Tianhao Yu, Haiyang Cui, Jianan Canal Li, Yunan Luo, Guangde Jiang, and Huimin Zhao. Enzyme function prediction using contrastive learning. *Science*, 379(6639):1358–1363, 2023.

Zhidian Zhang, Hannah K Wayment-Steele, Garyk Brixi, Haobo Wang, Dorothee Kern, and Sergey Ovchinnikov. Protein language models learn evolutionary statistics of interacting sequence motifs. *Proceedings of the National Academy of Sciences*, 121(45):e2406285121, 2024.

Zuobai Zhang, Minghao Xu, Arian R. Jamasb, Vijil Chenthamarakshan, Aurélie C. Lozano, Payel Das, and Jian Protein representation learning by geometric structure pretraining. abs/2203.06125, 2022.

Le Zhuo, Zewen Chi, Minghao Xu, Heyan Huang, Heqi Zheng, Conghui He, Xian-Ling Mao, and Wentao Zhang. Protllm: An interleaved protein-language llm with protein-as-word pre-training. *ArXiv*, abs/2403.07920, 2024.

A APPENDIX

A.1 RESULTS

Performance of OPUS-GO on three protein GO term prediction datasets.

Beside the results presented in main text, we also examine the results on the target for its various functional labels. Specifically, Figure A1a and Figure A1b present the predicted probabilities of each residue for the labels "transition metal ion binding" and "DNA binding", respectively, on the same target. The findings indicate that OPUS-GO effectively distinguishes between residues pertinent to each functional group. Additionally, Figure A1c-f show the predicted probabilities of each residue corresponding to the labels "heme binding", "tetrapyrrole binding", "quinone binding" and "organic acid binding" on the same target. Despite belonging to distinct functional categories, it is noteworthy that heme is a subclass of tetrapyrrole, and quinone falls within the category of organic acids. The results reveal a significant similarity in the residues with high probabilities for "heme binding" and "tetrapyrrole binding", as well as for "quinone binding" and "organic acid binding", thereby confirming OPUS-GO's ability to accurately pinpoint residues associated with their respective functions with good interpretability.

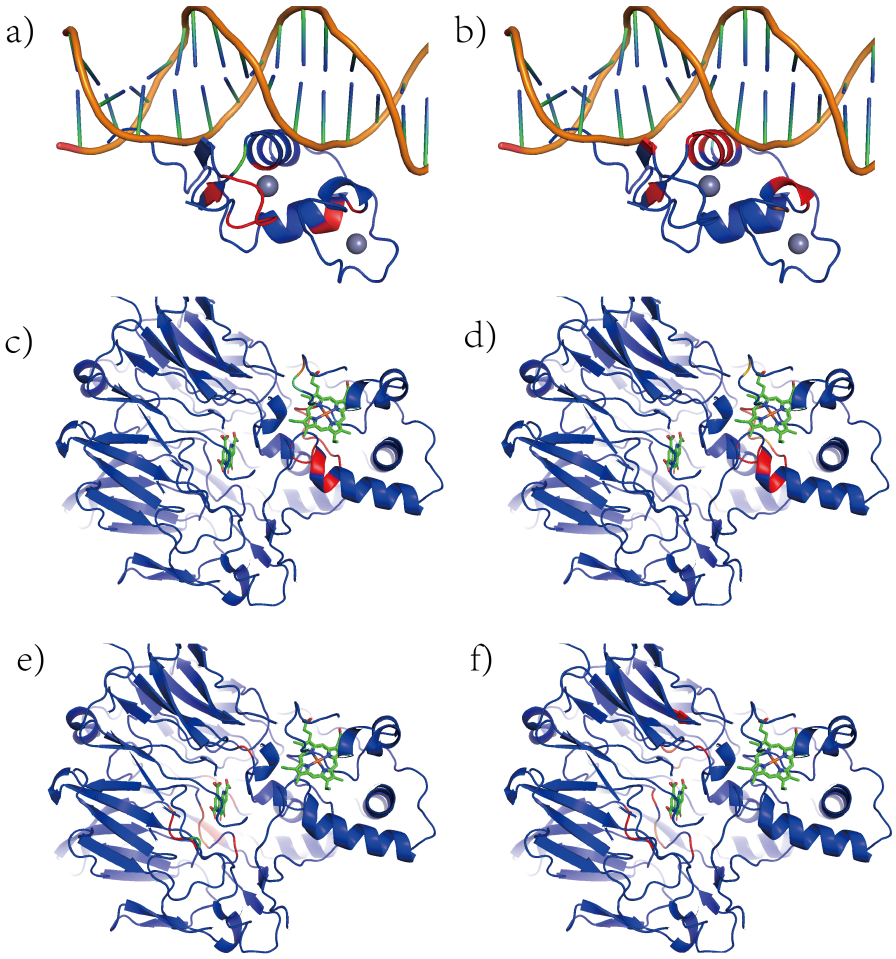

Figure A1: The interpretability results of OPUS-GO for protein GO term prediction. The results are visually represented using a color spectrum (ranging from blue to red) in PyMOL software, according to the predicted probability of corresponding label.The probabilities of each residue for label a) "transition metal ion binding", b) "DNA binding", c) "heme binding", d) "tetrapyrrole binding", e) "quinone binding", f) "organic acid binding".

Performance of OPUS-GO on two protein EC number prediction datasets.

The Enzyme Commission (EC) number is a widely employed system for categorizing protein enzyme functions through a standardized four-digit structure. This system provides a unified framework and accelerates the development in enzyme engineering. To assess the efficacy of our method on this task, we utilize two standard benchmarks: one proposed by DeepFRI, consisting of 538 labels (Gligorijević et al., 2021; Zhang et al., 2022), and another recently introduced by GraphEC, comprising 5,106 labels (Song et al., 2024). The former is denoted as EC in Table A1. The latter includes two test tests: the New-392 (Yu et al., 2023) and the Price-149 (Sanderson et al., 2023). We follow the official dataset splits for both benchmarks, allocating 10% of the training data, selected at random, to serve as the validation set for the second benchmark. Both benchmarks belong to the multi-label classification tasks, and we adopt Area Under the Precision-Recall Curve (AUPR) and Fmax as evaluation metrics for each method. Furthermore, for the second benchmark, we report the accuracy of the top-1 prediction for each method.

Table A1: The results of different methods on two protein EC number prediction datasets. Best results for each metric are shown in boldface.

	EC		NEW-392			Price-149		
Method	AUPR	Fmax	AUPR	Fmax	TOP1	AUPR	Fmax	TOP1
ESM-2	0.899	0.862	0.485	0.610	0.543	0.219	0.439	0.396
OPUS-GO	**0.902**	**0.881**	**0.487**	**0.641**	**0.566**	**0.244**	**0.448**	**0.423**
ESM-2 (three residues)	0.610	0.597	0.222	0.369	0.347	0.038	0.144	0.114
OPUS-GO (three residues)	0.872	0.863	0.454	0.570	0.561	0.184	0.393	0.396
GraphEC	-	-	0.276	0.458	0.402	0.121	0.258	0.195
CLEAN	-	-	-	-	0.541	-	-	0.403

As shown in Table A1, OPUS-GO demonstrates better sequence-level classification accuracy in all cases when compared to the baseline methods. For the second benchmark, we incorporate the results of GraphEC (Song et al., 2024) and CLEAN (Yu et al., 2023) for comparison, given that they utilized the same dataset in their studies. In GraphEC, for each residue, features derived from protein language model ProtTrans and structural predictions made by ESMFold are integrated through a geometric graph learning network to determine the final EC number. CLEAN (Contrastive Learning-Enabled Enzyme Annotation) is a machine learning algorithm designed to assign EC numbers to enzymes with enhanced accuracy, reliability, and sensitivity, leveraging features also obtained from protein language model ESM-1b. Since the probability of each label cannot be obtained from the output of CLEAN's algorithm, we only calculate the accuracy of its prediction using the output label with the highest probability. In addition, for the NEW-392 dataset, we utilize only 391 results from GraphEC for calculation as it is unable to produce results for the target "Q694C5". The results indicate that OPUS-GO outperforms both GraphEC and CLEAN. Consistent with the results observed in previous tasks, the performance of the sequence-level classification model trained using the average value of three residues selected by OPUS-GO (OPUS-GO (three residues) in Table A1) exceeds that of the model trained using the average of three randomly selected residues (ESM-2 (three residues) in Table A1), indicating the effectiveness of OPUS-GO in locating residues associated with the corresponding label.

The accuracy of the top-1 prediction for each method is presented in Figure A2a (NEW-392) and Figure A2b (Price-149) at four different levels. OPUS-GO consistently demonstrates superior performance compared to other methods. Additionally, Figure A2c-f displays the interpretability results of OPUS-GO for the protein EC number prediction task. Notably, although OPUS-GO solely utilizes the sequence information of the target protein, we have selected examples whose 3D structures have been released in the PDB for clearer presentation. The residues colored in red represent the active site of each target protein, while the yellow residues indicate positions that OPUS-GO predicts to be related to its enzymatic functional annotation. The results indicate that, in some examples (Figure A2c-e), the predicted positions are situated near the active sites. Conversely, there are also examples where OPUS-GO is unable to precisely identify the active sites (Figure A2f).

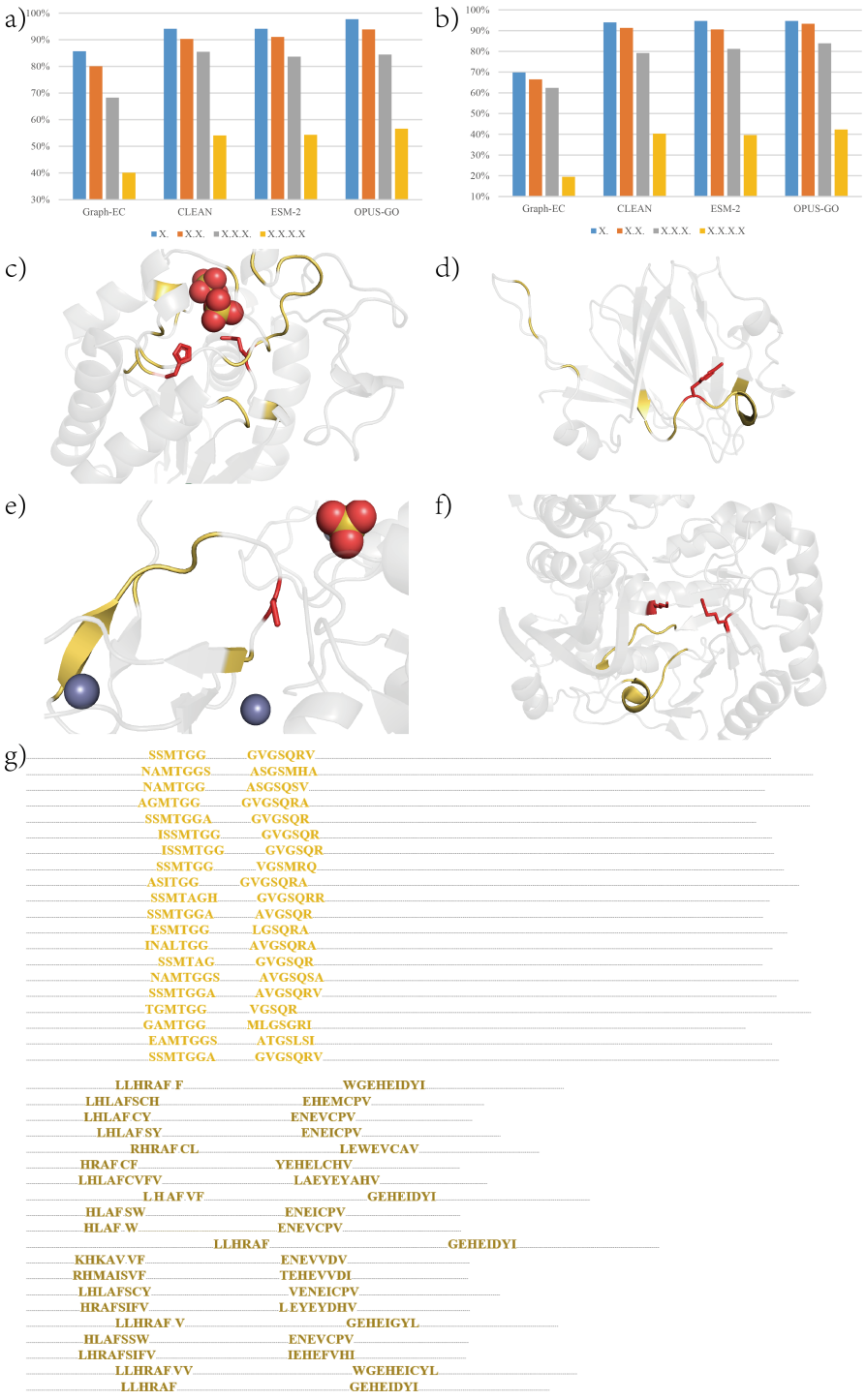

Figure A2: The results of OPUS-GO on protein EC number prediction task. a) The accuracy of the top-1 prediction for each method on NEW-392. b) The accuracy of the top-1 prediction for each method on Price-149. c-f) The interpretability results of OPUS-GO. Only the sequence information of the cyan protein structure is utilized in OPUS-GO. The residues with side chains illustrated and colored in red represent the active site of each target. The residues colored in yellow represent the positions that OPUS-GO predicts to be related to its enzymatic functional annotation. g) The residues that OPUS-GO identified as being related (with a probability greater than 0.5) to the label "5.3.3.2" (Isopentenyl-diphosphate Delta-isomerase) for the sequences within the training set.

However, it is important to note that the interpretability feature of OPUS-GO is not specifically tailored for identifying active sites. Instead, it identifies residues that it deems to be most directly associated with the corresponding labels, which may or may not align with the active sites. To verify this point, we conduct an analysis of the targets with the EC number "5.3.3.2" (Isopentenyl-diphosphate Delta-isomerase) within the training set, examining the residues that OPUS-GO identified as being most related to the label "5.3.3.2" for each sequence. As illustrated in Figure A2g, instead of identifying specific activate sites, OPUS-GO successfully identifies consistent patterns among these targets. These patterns can be categorized into two classes and are relatively conserved, and situated near the binding sites of each target. Considering the potential essentiality of these patterns for the function of the enzyme, OPUS-GO emerges as a valuable tool in protein design, offering crucial patterns that can facilitate the design of proteins with specific functionalities.

A.2 METHOD

The workflow of OPUS-GO is depicted in Figure 1. Initially, we extract features for each residue across each sequence target using relevant biological language models, such as ESM-2 (Lin et al., 2023). Subsequently, a residue-level classifier is employed to predict the likelihood of each residue being associated with the corresponding sequence-level labels. The architecture of this classifier comprises a FeedForward module, a RMSNorm layer, and a Dropout Layer with a dropout rate of 0.5. The Swish activation function is employed within this architecture. No further aggregation of information from other residues in the sequence is necessary, as it has already been incorporated through the transformer block during the pretraining phase of the biological language models.

Algorithm A1 Pseudocode for calculating the loss of OPUS-GO

Require: labels $[N_{\text{labels}}]$ ▷ Number of labels in the dataset
Require: logits $[L_{\text{seq}}, N_{\text{labels}}]$ ▷ Logit values of each residue for respective labels

function MIL_LOSS(labels, logits)
    loss_fn ← BinaryCrossEntropyLoss()
    $n_{\text{top}} \leftarrow \max(\min(0.1 \times L_{\text{seq}}, 10), 1)$ ▷ Set number of top residues per label
    positive_indices ← GetIndicesWhere(labels $= 1$)
    n_{sample} ← len(positive_indices) $\times n_{\text{top}}$
    logits_positive ← [ ] ▷ Store positive examples
    labels_positive ← ArrayOfSize(n_{sample}, Value=1)
    for each index in positive_indices **do**
        sorted_indices ← SortIndices(sigmoid[:, index], order=descending)
        top_indices ← sorted_indices[:n_{top}]
        Append(logits_positive, logits[top_indices])
    end for
    logits_negative ← RandomSample(IndicesNotIn(logits_positive), Size=n_{sample})
    labels_negative ← ArrayOfSize(n_{sample}, Value=0)
    labels_MIL ← Concatenate(labels_positive, labels_negative)
    logits_MIL ← Concatenate(logits_positive, logits_negative)
    loss ← loss_fn(labels_MIL, logits_MIL)
    return loss
end function

In this study, a modified MIL loss function is introduced to compute the loss and update the classifier. Since the sequence-level annotation may consists of multiple labels, the binary cross-entropy loss is applied to each label. The modified MIL loss can be further divided into two components. As shown in Algorithm A1, n_{top} is a hyperparameter that specifies the number of residues utilized

in the calculation for each label. In OPUS-GO, we set n_{top} to the lesser value between 10% of all residues in sequence and 10 residues. For each positive label, we select the top n_{top} residues based on their probabilities in descending order and assign these residues with positive labels for loss computation. Concurrently, we randomly sample an equivalent number of residues from those not previously chosen and assign their labels as negative for loss computation.

Additionally, we introduce an auxiliary MIL loss, as detailed in Algorithm A2. This loss term exclusively utilizes negative samples in its calculation. The hyperparameter $n_{top\ labels}$ specifies the number of labels considered for computation. In OPUS-GO, if the total number of labels exceeds 5,000, $n_{top\ labels}$ is set to 20. Otherwise, all labels are considered. Firstly, the top $n_{top\ labels}$ labels are selected based on their maximum probabilities across all residues, sorted in descending order. Subsequently, for each negative label within the $n_{top\ labels}$ labels, the top n_{top} residues are chosen based on their probabilities, sorted in descending order, and assigned negative labels for loss computation. Both the auxiliary MIL loss and the previously mentioned MIL loss are added with equal weight to update the parameters of the residue-level classifier.

The incorporation of the hyperparameter $n_{top\ labels}$ is essential in tasks with an extremely large number of labels, such as those exceeding 5,000. In auxiliary MIL loss, it is observed that negative labels typically exhibit low probabilities for each residue. However, the accurate penalization of negative samples that exhibit relatively high probabilities for negative labels is crucial. Utilizing the average value across all negative labels would undermine the significance of pertinent information. Therefore, we select the $n_{top\ labels}$ based on their maximum probabilities across all residues, thereby ensuring that the necessary penalizations are adequately applied.

In this study, we set the hyperparameter $n_{top\ labels}$ with values of 1, 10, 20, 50, 200, 500, and 5,106 on the EC number prediction benchmark proposed by GraphEC (Song et al., 2024). Subsequently, we train the models respectively. As shown in Table A2, the results suggest that the F1-Score attains its maximum value when $n_{top\ labels}$ is set to 20. Therefore, we recommend a default setting of 20 for tasks with an extremely large number of labels.

Table A2: The results of OPUS-GO with different hyperparameter settings for $n_{top\ labels}$ (denoted as N in the table) on the validation set of the EC number prediction benchmark proposed by GraphEC.

Method	Precision	Recall	F1-Score
OPUS-GO (N=1)	0.0203	0.7177	0.0342
OPUS-GO (N=10)	0.9190	0.9308	0.9217
OPUS-GO (N=20)	0.9180	0.9335	**0.9221**
OPUS-GO (N=50)	0.9122	0.9339	0.9182
OPUS-GO (N=200)	0.8950	0.9362	0.9063
OPUS-GO (N=500)	0.8556	0.9420	0.8782
OPUS-GO (N=5,106)	0.7582	0.9397	0.8010

During the training phase, the Adam optimizer (Kingma & Ba, 2015) is utilized. The model is trained for a maximum of 25 epochs, with an initial learning rate set to 1e-3. This learning rate is halved when a decrease in validation accuracy is observed after each epoch. Early stopping is implemented if the learning rate has been reduced four times. OPUS-GO is developed using TensorFlow version 2.4 and trained on four NVIDIA Tesla V100 GPUs. The total batch size is set to 16.

During the inference phase, the sequence-level prediction is aggregated by selecting the maximum value for each label across all residues, with residues possessing a sigmoid value greater than 0.5 being considered as associated with the corresponding label. For a given label, its probability is assigned as the maximum sigmoid value among all residues. Additionally, for multi-class classification tasks, where each sequence has only one label, the final output is determined by selecting the label with the highest sigmoid value.

Algorithm A2 Pseudocode for calculating the auxiliary loss of OPUS-GO

Require: labels [N_{labels}] ▷ Number of labels in the dataset
Require: logits [$L_{\text{seq}}, N_{\text{labels}}$] ▷ Logit values of each residue for respective labels
Require: $n_{\text{top labels}}$ ▷ Number of labels considered for computation

function MIL_LOSS_AUXILIARY(labels, logits, $n_{\text{top labels}} = 20$)
    loss_fn ← BinaryCrossEntropyLoss()
    $n_{\text{top}} \leftarrow \max(\min(0.1 \times L_{\text{seq}}, 10), 1)$
    if $N_{\text{labels}} > 5000$ **then**
        sigmoids ← Sigmoid(logits)
        sigmoids_max ← Max(sigmoids, axis=0) ▷ Get max probability for each label
        threshold ← Sort(sigmoids_max, order=descending)[$n_{\text{top labels}}$]
        top_labels ← sigmoids_max > threshold
        indices_0 ← GetIndicesWhere(labels == 0 & top_labels == 1)
        n_{sample} ← len(indices_0) $\times n_{\text{top}}$
        logits_0 ← [ ]
        labels_0 ← ArrayOfSize(n_{sample}, Value=0)
        for each index in indices_0 **do**
            sorted_indices ← SortIndices(sigmoid[:, index], order=descending)
            Append(logits_0, logits[sorted_indices[:n_{top}]])
        end for
    else
        indices_0 ← GetIndicesWhere(labels == 0)
        n_{sample} ← len(indices_0) $\times n_{\text{top}}$
        logits_0 ← [ ]
        labels_0 ← ArrayOfSize(n_{sample}, Value=0)
        for each index in indices_0 **do**
            sorted_indices ← SortIndices(sigmoid[:, index], order=descending)
            Append(logits_0, logits[sorted_indices[:n_{top}]])
        end for
    end if
    labels_MIL ← logits_0
    logits_MIL ← labels_0
    loss ← loss_fn(labels_MIL, logits_MIL)
    return loss
end function